Database and DBMS Introduction

Database:
Database is the collection of interrelated data. or collection of related information stored so that it is available to many users for different purposes. The content of database is obtained by combining data from all the different sources in an organisation.

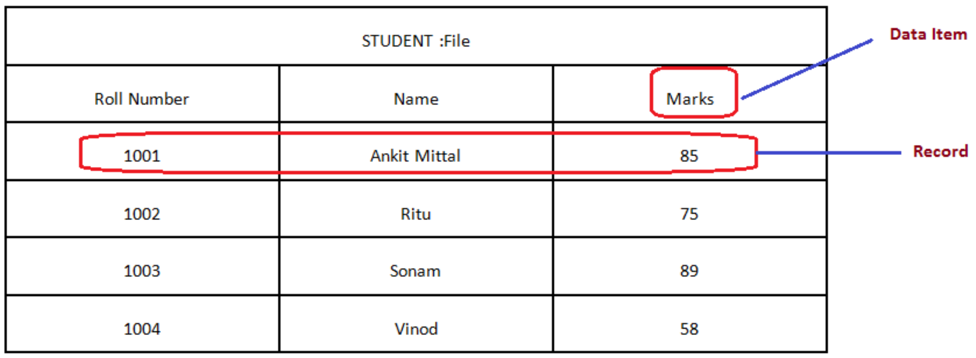
Database Management System (DBMS)

DBMS is a software used to manage and access the database in efficient way. It manages the data in a database. There are various DBMS available - For example –

INGERS
ORACLE
Sybase
Foxbase
MS Access
dBASE.

Some related terms used in DBMS -

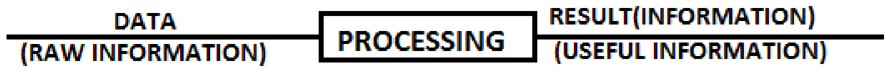
Data : is the raw facts and figures. It is represented with the help of characters like alphabets(A-Z), digits(0-9) and special characters (+,-,/,*,<,>,= etc.) **Data Item (Field) :** Set of characters which are used to represent a specific data element. For example, Name of student in a class is represented by a data item, say name. **Record :** A collection of related items. Example a Student record containing name, age, DOB, Fathers name, mothers name, rollnumber, marks, etc. **File :** is a collection of related records. For example, Student file .


Data Item
Record

STUDENT :File		
Roll Number	Name	Marks
1001	Ankit Mittal	85
1002	Ritu	75
1003	Sonam	89
1004	Vinod	58

Information : is organised or classified data so that it has some meaningful values. It is the processed data, which has the following characteristics :
Timely
Accurate
Complete and
Given to the right persons

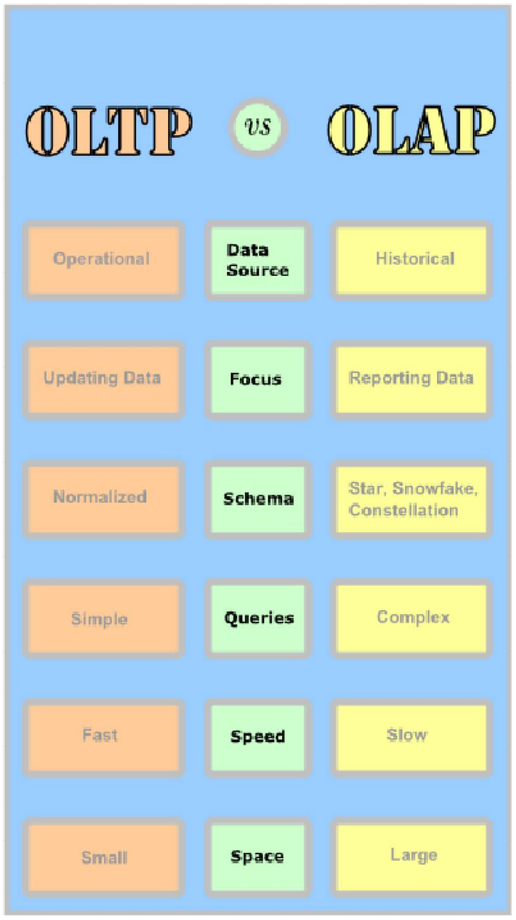
Data Processing and Data Manipulation : Data Processing is the manipulation of raw data to make it more useful. Data Manipulation consists of such operations as classification, sorting calculations and summarisation.

Basis	**Data**	**Information**
Meaning	Raw facts and Figures	Processed data so that it has some meaningful values.
Example	Test Score of each student	Average Score of a class
 DATA (RAW INFORMATION) PROCESSING RESULT(INFORMATION) (USEFUL INFORMATION)		

Database Design Steps

Requirements Analysis
Conceptual Modelling (ER Model)
Logical Modelling (Relational Model)
Schema Refinement (Normalization)
Once the construction is over, we cannot change it. In a single user application, we may not think of concurrency and transaction management. But in multi-user applications, they comes into account.

Types of Databases
OLTP - Online Transaction Processing -
The main focus for OLTP systems is updating data to put on very fast query processing, maintaining the integrity of data in multi-access environments.
The effectiveness of OLTP systems is measured by number of transactions per second.
It is simply called as Database.
The main operations performed on OLTP are Read and Write.
Use of simple SQL
OLAP - Online Analytical Processing -
The main focus of OLAP systems is reporting data.
The effectiveness of OLAP systems is measured by response time.
OLAP applications are widely used by Data Mining Techniques.
It is also called as Data Warehousing.
The main operations performed on OLTP are Read only.
Use of Complex SQL (Joins etc.)
Difference between OLTP and OLAP systems



OLTP	vs	OLAP
Operational	Data Source	Historical
Updating Data	Focus	Reporting Data
Normalized	Schema	Star, Snowfake, Constellation
Simple	Queries	Complex
Fast	Speed	Slow
Small	Space	Large

Another Classification of Databases

Commercial Databases : Inventory Control System, Library, Student Information System
Multimedia Databases : Voice, Video Clips, Images
Temporal Databases : Attaching time aspect to data. For example, Railway Reservation.
GIS (Geographical Information System) Databases : To give ways to reach destination. For example, Route Map, Google Earth
Distributed Databases : Web Environment (Consistency is very high) ⇒ Concurrency and Transactions
Deductive Databases : Rule Based : (A=B, B=C ⇒A=C), If we buy one thing, then there is chance of buying the other.
Document Databases : Google, Content Information System
Active Databases : Flooded with Triggers
Web Databases : Any database can become a web database, if it is available in web environment

Different Types of DBMS

Relational DBMS
OR DBMS (Object Relational DBMS) : example oracle i. For RDBMS, we added object oriented principles
OD DBMS (Object Oriented DBMS) : For object oriented, we added relational principles
For multimedia, GIS, Document and Web : OODBMS. For others except above four : ORDBMS

Components of DBMS -

A DBMS software is partitioned into several modules or components and each module or component is assigned a specific operation to perform. While designing a DBMS, its interface must be taken into account with the OS as some of the functions of DBMS are supported by OS to provide basic services and DBMS is built on top of it. The functional components of a database system can be broadly divided into –

Query Processor Component: It simplify and facilitate access to data (convenient and efficient)
Storage Manager Component: It minimize the need to move data between disk and main memory
Transaction Manager Component: It handle atomicity and concurrency of transactions and consistency and durability of the databases

Query Processor Component –

Query Processor- Query processor is used to interpret the online user query and converts it into an efficient series of operation in a form capable of being send to the data manager for execution. The query processor use the data dictionary to find the structure of the relevant portion of the data base and use this information in modifying the query and prepare an optimal plan to access the database. It is a program module that provides the interface between the database and the application programs/queries. The Query Processor Components include –

Data Definition Language(DDL) Compler- DDL compiler takes the data definition statement that is the source form & convert them into the object form (or) interprets DDL commands and records them in the data dictionary

Data Modelling Language (DML) compiler - translates DML commands into query evaluation plans
Query evaluation engine - executes queries according to the plans.

Storage Manager Component –

A Storage Manager is a component or program module that provides the interface between the low-level data stored in the database and the application programs/queries submitted to the system.

The Storage Manager Components include -

File Manager- File manager manages the file space and it takes care of the structure of the file. It manages the allocation space on disk storage and the data structures used to represent info stored on other media.

Buffer Manager - It transfers blocks between disk (or other devices) and Main Memory. A DMA (Direct Memory Access) is a form of Input/Output that controls the exchange of blocks process. When a processor receives a request for a transfer of a block, it sends it to the DMA Controller which transfers the block uninterrupted.

Authorization and Integrity Manager - This Component of storage manager checks for the authority of the users to access and modify information, as well as integrity constraints (keys, etc).

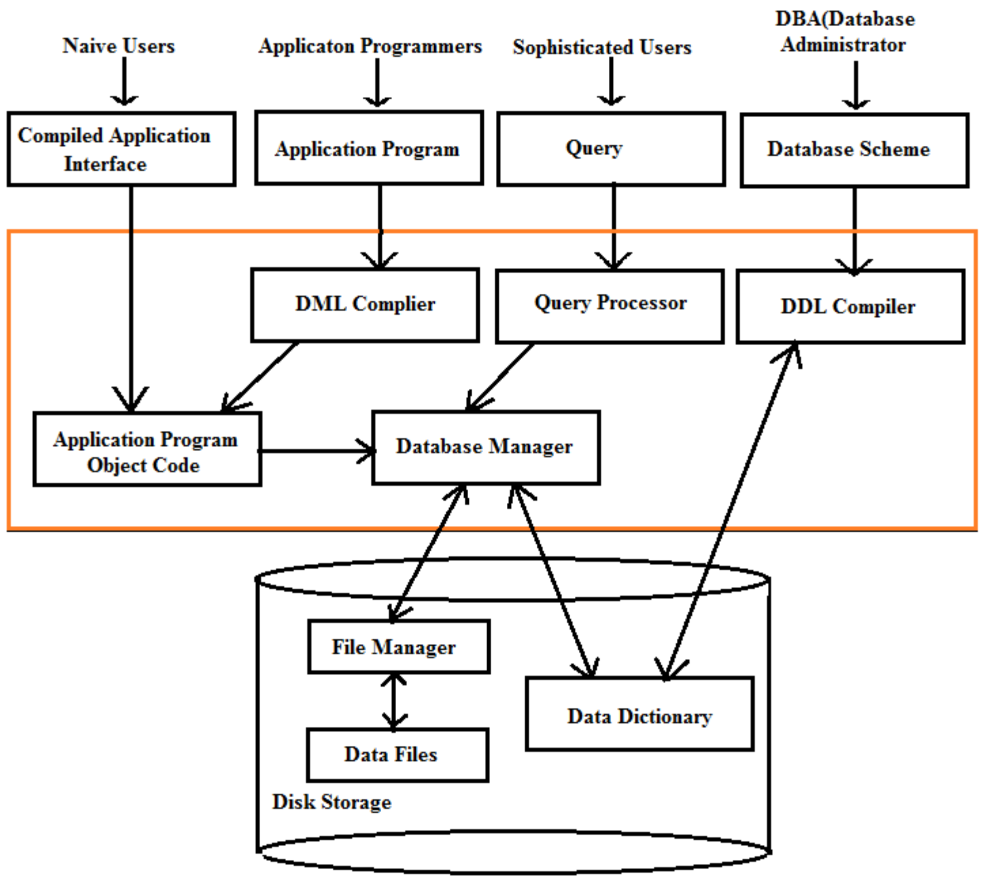
Disk Manager- The block requested by the file manager is transferred by the Disk Manager.

The Structures maintained by Storage manager are-

Data Files- Data files contains the data portion of the data base.

Data Dictionary- DBMS must a data dictionary function. The dictionary contains the data about the data. Rather than just raw data. The information about attributes, entity, mapping & cross reference information is contained in the data dictionary.

Indices or Indexing and Access Aids - An index is a small table having two columns in which the first column contains a copy of the primary or candidate key of a table and the second column contains a set of pointers holding the address of the disk block where that particular key value can be found. The advantage of using indices is that index makes search operation perform very fast. In a data base system, a set of access aids in the form of indexes are usually provided to improve the performance of a database system.



Transaction Manager Component/Data Manager -
A Transaction is a collection of operations that performs as a single logical function in a database application. All Transactions must follow transction properties, which are called ACID Properties. Data Manager converts the user queries from the user logical view to a physical file system.
The ACID Properties are -
Atomicity: either all operations succeed or all of them fail
Consistency: the database is changed from one consistent state to another consistent state
Isolation: no transaction interfere other transactions in the middle
Durability: operations of successful transactions must persist
The Components included by Transaction Manager is -
Transaction Manager - Transaction Manager controls the execution of transactions.
Lock Manager - Access of items in DBMS is controlled by LOCKS. And the part of DBMS that keeps a record of locks issued to transactions, is done by the Lock Manager. It maintains a LOCK table which is a hash table, with data object identifier as the key
Recovery Manager - Recovery manager is responsible for atomicity and durability. It allows DBMS to restore the database to a consistent state following a failure.
Tele-Communication System- Online user of the computer system whether remote or local communicate with it by sending and receving message over communication line. These messages are routed by communication line.

DBMS vs File System

To understand the difference between DBMS vs File System, we should know the disadvantages of file processing system first.
Disadvantages of File Processing System -
Duplicate Data - As all files are independent of each other, so, duplicate data may be present in more than one files.
Inconsistency - Inconsistency means different copies of same data that are not matching. Data may be inconsistent in file processing system.
Poor Data Integrity - Data integrity refers to the overall completeness, accuracy and consistency of data. But in Processing system, Poor data integrity often develops.
Data is isolated and separated - Data are separated in various files. SO, if it is needed to extract data from 2 different files, it will be require to determine which parts of each of the files are needed and how files are related to one another.
Application Programs are dependent on file formats - In processing suystem, the physical formats of file are entered in application program that process the files. Change in the file format result in program updates and a change which is time consuming and error prone.
Poor Data Security- Data is stored in different files causing the security problems
Difficult to represent complex object - Some data objects may be of variable length which might produce difficulty in representation in files.

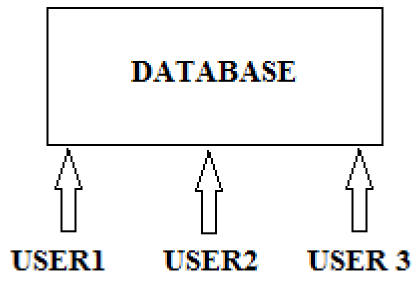
Comparison of File System vs Database Systems (Limitations to File System)–
If data is too large, it will create so many problems.
1. Too complex to access the data using Physical Details.
Suppose a university has 500 GB of data having files F1,F2,….,Fn. If we want to retrieve the student details who scored more than 80%,then the Program will require the low level details or physical details.
The physical details will be:
- Location
- Name of the file
- Format of the file

To avoid this problem DBMS provides Data Independency. User can access the data from database without knowing any physical details

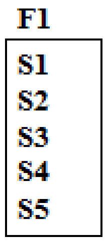
```
Select *
From table
Where condition
```

2. If data is too huge, more IO cost is required to access the data.
Suppose that one of the bank officers needs to find out the names of all customers who live within a particular postal-code area. The officer 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 customers. The bank officer has now two choices: either obtain the list of all customers and extract the needed information manually or ask a system programmer to write the necessary application program. Both alternatives are obviously unsatisfactory. Suppose that such a program is written, and

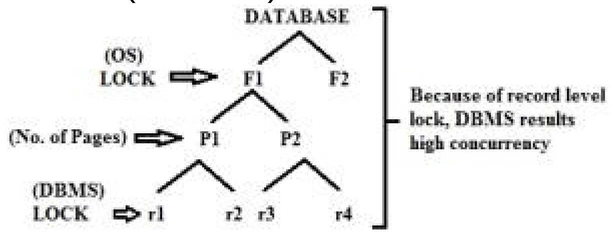
that, several days later, the same officer needs to trim that list to include only those customers who have an account balance of $10,000 or more. As expected, a program to generate such a list does not exist. Again, the officer has the preceding two options, neither of which is satisfactory.
The point 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.
3. Concurrent Execution - Concurrent execution means when multiple users access the same data at same time.



Concurrent Operations restrict by OS
•R<->W (If one user is reading, then other user cannot be write).
•W<->R/W (If one user is writing, then other user cannot be read or write).



U1: update student S1
U2: update student S3
OS controls concurrency at file levels. i.e. OS does not allow both the users to update simultaneously even both the users update different data's i.e. S1 & S3.
To avoid this problem –
DBMS allows/controls at record level.
U1: update S1
LOCK (S1 Record)
U2: update S3
LOCK (S3 record)

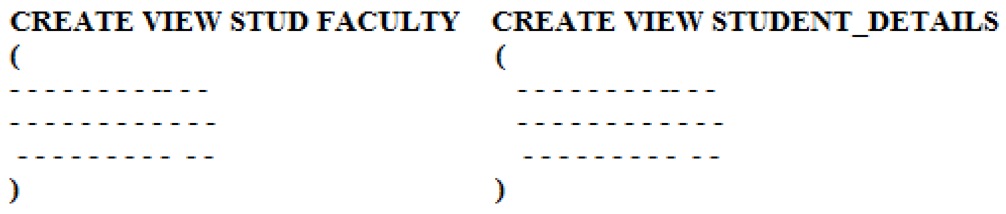


4. Security - Operating System gives file level security. For example, Student is a single file which contains various fields like SID, Sname, Marks, Address .. A password can be set to Student file but not to its fields. So here, Os level security is managed by password. If password matches, then the user will see the entire data.
DBMS provides different level of security. For instance, in the student file security will be decided according to

- Faculty can see SID, Sname, Marks only
- Admin can see SID, Sname, Marks and Address.

if two file are made for faculty as well as for admin, then there will be problem of data redundancy.

DBMS provides security by VIRTUAL TABLE **(VIEWS)** – View is a virtual table that refers to physical table.



```
CREATE VIEW STUD FACULTY      CREATE VIEW STUDENT_DETAILS
(                             (
- - - - - - - - -- - -          - - - - - - - - -- - -
- - - - - - - - - - - -         - - - - - - - - - - - -
 - - - - - - - - -  - -          - - - - - - - - -  - -
)                             )
```

Advantages and Disadvantage of DBMS

Advantages of DBMS -

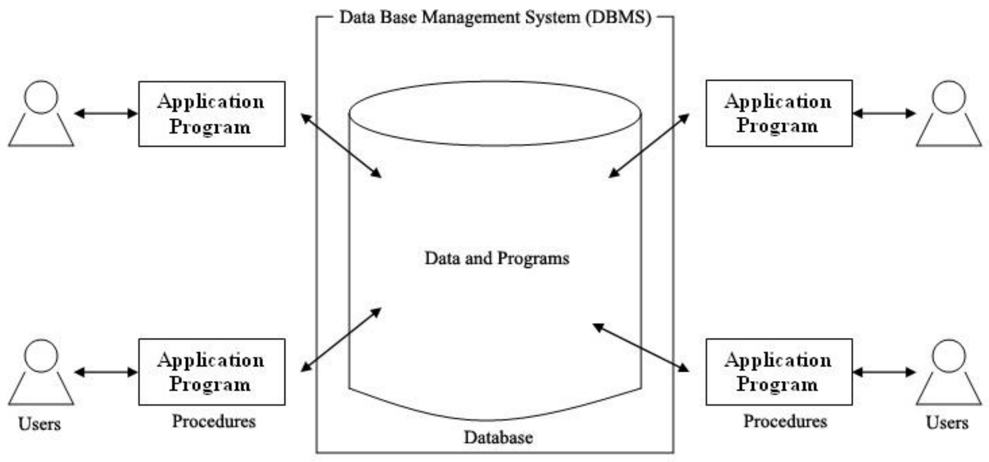
Minimal Redundancy
Inconsistency can be avoided
Sharing of Data
Standards can be enforced
Search Capability
Integrity
Privacy and Security
The interface with the Past
The interface with the future
Physical Data Independent
Logical Data Independence
Tunability
Data Migration
Simplicity
Powerful User Languages
Conflicting requirements can be balanced
Faster Development of New Application
More Control over Concurrency

Disadvantages of DBMS -

High Cost of DBMS - More costly licensed version (40 to 50 lacs)
High Hardware Cost- DBMS will not allow to add the data by user directly. It will take command from user.
High Conversion Cost-
Higher Programming Cost
Complex
Backup and Recovery are more difficult
Integration

Components of Database -

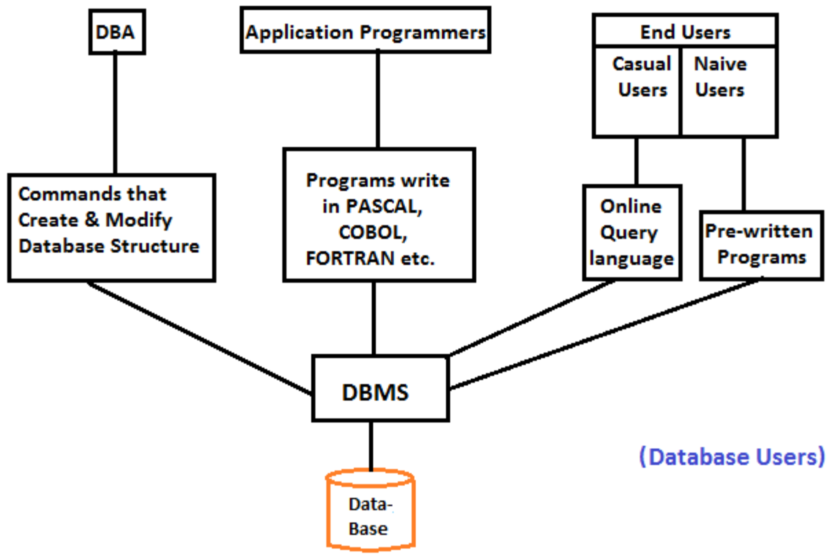
A database system involves four major components -
Data
Hardware
Software
Users



(Components of Database)

Data -
Data stored in database system is partitioned into one or more database. A database is both integrated and shared.
By Integrated, means that database can be thought as unification of several distinct data files with redundancy partially or wholly eliminated.
and By Shared means that, same piece of data in database may be shared among several users for different purposes.

Hardware -
Hardware consists of secondary storage volumes like disks, drums, etc on which database resides, together with associated devices, control units etc.
Software -
A software layer is in between the hardware and the users, and the software is usually called database management system (DBMS). It also includes operating system, network software(if necessary), application programs. All requests from users for accessing the data from database are handled by softwares.
Database Users -
There are 3 classes of user-



(Database Users)

Application Programmer - (is responsible for writing application programs typically in a language such as COBOL or PL/1 that use the database for retrieving,creating, deleting or modifying information)
End Users - access the database from a terminal. There are two types of end users-
Casual users - These users are trained in the use of the on-line query language and access data by entering queries at terminals.
Naive Users - These users access database through application programs. They do not need to know the structure or language of database systems.
Database Administrator (DBA) - is a person or group of persons responsible for overall control of database.
Functions Of DBA (Database Administrator) -
Deciding the information contents of database system
Deciding Hardware device to be used
Deciding the users and the data to be used by them.
Deciding the backup and recovery method
deciding the validation checks on the data
Responsible for Monitoring Performance and making the appropriate adjustments as requirements change.

Architecture of DBMS and Concept of Data Independence

Architecture of DBMS –

The architecture of DBMS is divided into 3 general levels :
Internal view/Physical Schema
Conceptual Schema
External Schema

Concept of Data Independence –

First of all, we should know about Data Independency. Data Independency is the ability to use the database without knowing the representation details. To provide data independency, there should be atleast two levels of abstraction:

Logical Data Independence - means that the overall logical structure of data may be changed without changing the application programs.

Physical Data Independence - means that the physical layout and the organisation of data may be changed without changing the overall logic structure of data or the application programs

Why there is requirement of data independency –

To allow the DBA to make changes in the content, location, representation and organisation of a database without causing the reprogramming of application programs which use the database.

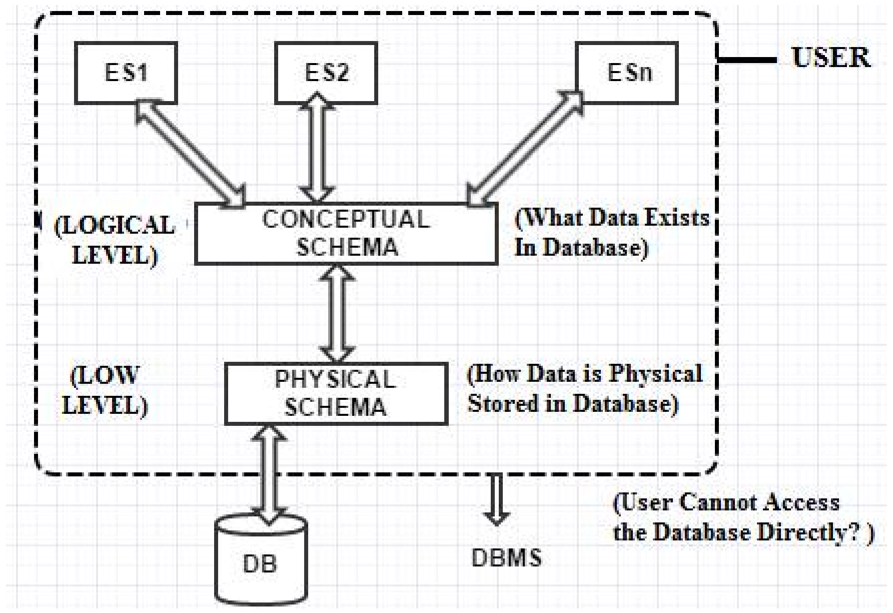
To allow the supplier of data processing equipment and software to introduce new technologies without causing reprogramming of customer's application.

To facilitate data sharing by allowing the same data to appear to be organised differently for different application programs.

To simplify application program development in particular, to facilitate the development of programs for interactive database processing.

To provide the centralization of control needed by the database administrator(DBA) to ensure the security and integrity of the database.

Levels Of Abstraction is defined as



ES1
ES2
ESn
USER
(LOGICAL LEVEL)
CONCEPTUAL SCHEMA
(What Data Exists In Database)
(LOW LEVEL)
PHYSICAL SCHEMA
(How Data is Physical Stored in Database)
(User Cannot Access the Database Directly?)
DB
DBMS

Physical schema:

Metadata about the physical data in DB.

```
CREATE TABLE STUDENT
(
   ----------
   ---------
   ----------
);
```

STUDENT FILENAME

Details of Database Files stored in Physical Scheme.

Physical schema is maintained by s/w & the user is not allowed to interface. This level is closest to physical storage, which is concerned with - in which the data is actually stored.

External Schema :

This level is closed to the users, and is concerned with - in which the data is viewed by individual viewers.

Conceptual Schema:

It is the representation of entire contents of database

If we access the data through conceptual scheme, then we can access all the data. To restrict the data we provide security through external scheme.

Hide the physical details.

Student (SID, SNAME, ……)

We don't know how the data exists.

DBMS is an interface b/w user & database.

Conceptual/Internal Mapping :
The conceptual/internal mapping defines the correspondance between the conceptual view and store database. It specifies how conceptual records and fields are represented at the internal level.

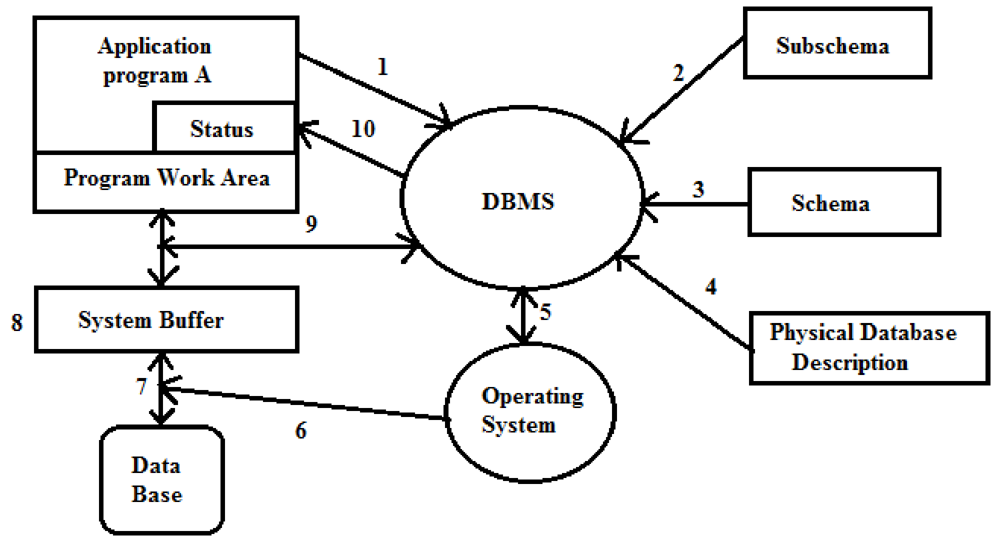
External/Conceptual Mapping :
External/Conceptual Mapping defines the correspondence between a particular external view and the conceptual view.

Execution Steps of DBMS - Sequence of events When DBMS Reads A Record -

Execution Steps of DBMS -

The events that occur when an application program reads a record by means of DBMS is shown in fig.

1. Application Program A issues a request to the DBMS to read a record.
2. The DBMS obtains the subschema and look up the description of the data in question.
3. The DBMS obtains the schema and determines which logical data types are needed.
4. The DBMS contains the physical data base description and determines which physical record(s) to be read.
5. A command is issued by the DBMS to the operating system to read the required record(s).
6. The operating system interacts which the physical storage where the data are kept.
7. The required data are transferred between the storage an the system buffers.
8. Comparing the schema and the subschema, the DBMS derives from the data, the logical record needed by the application program.
9. The data is transferred by the DBMS from the system buffers to the program work area.
10. The status information is given by the DBMS to the application program, included any error indication.
11. The application program then operates with the data in a work area.



ER -Model

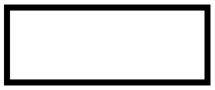
Entity and its Types - Strong Entity, Weak Entity, Composite Entity

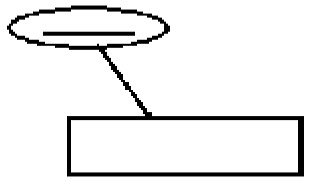
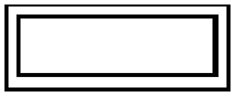
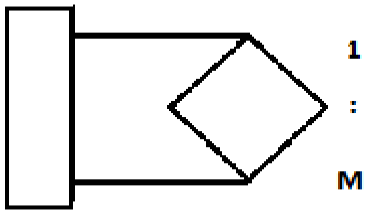
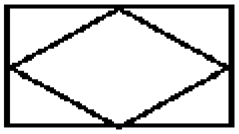
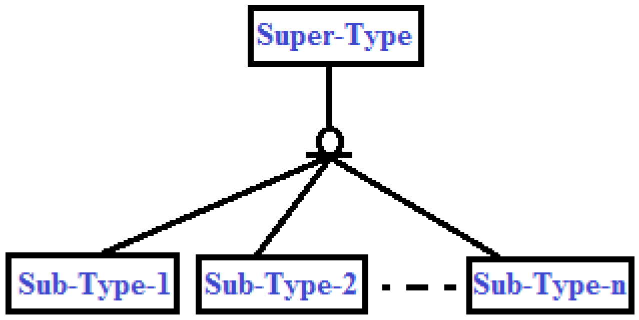
Entity -
An entity is an object that are represented in the database. For example Mohit, Vasu, CSE306 etc. An entity is represented or defined by set of **attributes. Attributes** are the properties used to describe an entity. For example, a STUDENT entity may have a Name, Roll number, Class, Marks etc. where STUDENT is the entity and name roll number class marks are the attributes.

Types of Entity -

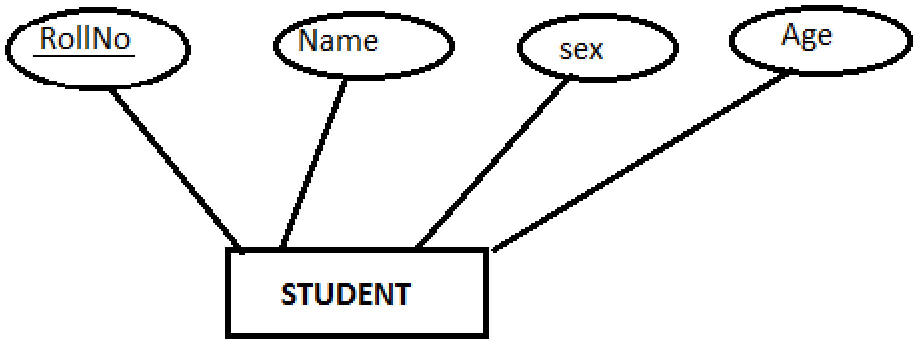
1. **Strong Entity Types**
2. **Recursive Entity Types**
3. **Weak Entity Types**
4. **Composite Entity Types or Associative Entity Types**
5. **SuperType and SubType Entities**

Notations Of different Entity Type in ER Diagram

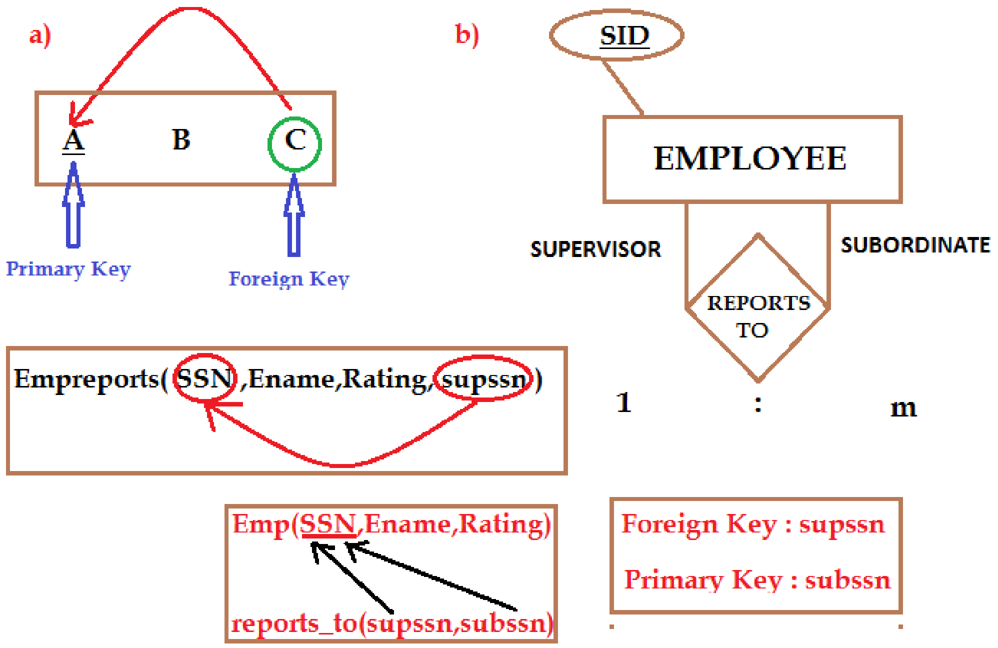
 -	**Entity**

	**Strong Entity Type**
	**Weak Entity Type**
	**Recursive Entity Type**
	**Composite Entity Type (or) Associative Entity**
	**Subtypes and Supertypes**

Strong Entity Type - are the entities which has a key attribute in its attribute list or a set that has a primary key. The strong entity type is also called regular entity type. For Example,



The Student's unique RollNo will identify the students. So, RollNo is set to be the Primary Key of the STUDENT entity, & Hence STUDENT is a strong entity type because of its key attribute. **Recursive Entity Type -** It is also called Self Referential Relationship Entity Type. It is an entity type with foreign key referencing to same table or itself. Recursive Entity Type occurs in a unary relationship. For example, a supervisor and subordinate relationship sets - One Supervisor can supervise multiple subordinates but each subordinate reporting to atmost one supervisor.



In example b) An employee supervises another employee. Let the employee who supervises another one is manager. So, Manager supervises employees. But a manager is also an employee, whose details are in the employee entity. So, to implement this, a foreign key of the employee's manager number would be held in each employee record i.e "supssn". So, the employee entity will contain its attributes as -

- Employee ID (**subssn**)
- Employee FirstName
- Employee Lastname
- Employee DateofBirth
- and Manager Number (i.e. employee number of the employee's manager) - (**supssn**)

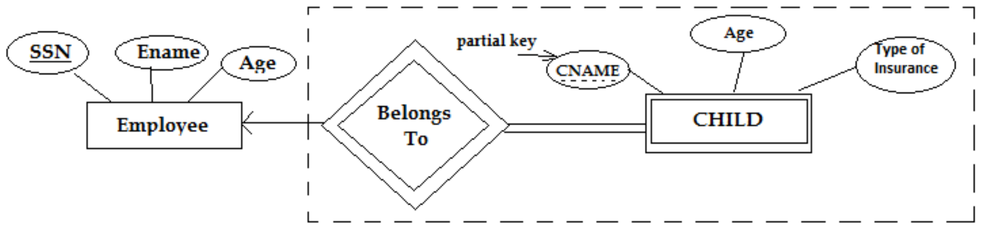
The next figure will clear you the employee entity as a recursive entity

subssn	FirstName	LastName	DateofBirth	supssn
101	Ankit	Mittal	16-7-90	
102	Vishal	Kamboj	17-8-90	101
103	Pooja	Mittal	31-12-90	101
104	Neeraj	Kedia	18-5-90	101
105	Neeraj	Kamboj	18-5-90	103

-

If M:M ==> Reports_To(supssn,subssn)
then 2 tables are made.

Weak Entity Type - Entity Type with no key or Primary Key are called weak entity Type. The Tuples of weak entity type may not be possible to differentiate using one attribute of weak entity.For every weak entity, there should be unique OWNER entity type. In the below example, CHILD is a WEAK entity type and Employee is the OWNER entity type. **Example of weak entity type is -**



SSN	Ename	Age
E1		
E2		
E3		
E4		

SSN	CNAME
E1	A
E2	A
E1	B
E3	B

CNAME	Age	Gender
A	8	M
A	10	M
B	10	F
B	10	F

Children Entity is depending upon Strong Entity Employee(as it has a unique ID named SSN). The relationship is established to associate children with their parents for insurance coverage. **Attributes of CHILDREN entity are-**

- CNAME (name of Child)
- Age of Child
- Type of Insurance

So, None of the attributes of CHILDREN does not give a unique ID to the entity. And the CHILDREN Entity has to depend on EMPLOYEE entity for identification. The next table will clear you the child entity as a weak entity and so is represented with employee id(SSN) –

SSN	CNAME	AGE	GENDER
E1	A	8	M
E2	A	10	M
E1	B	10	F
E3	B	10	F

CHILD_BELONGS_TO(SSN,CNAME,AGE,GENDER**)** Maximum number of tables = 2 Minimum number of tables = 2 Some Points about weak entity Type -
1) Weak entity Type should always combines with weak relationship set for database schema relation.

2) Relationship between OWNER ENTITY Type and Weak Entity Type is also partial.

3) Participation between weak entity Type and relationship is always TOTAL

4) Cardinality can be Many to Many or One to Many.

Composite Entities - If a Many to Many relationship exist then we must eliminate it by using composite entities. Composite Entities are the entities which exists as a relationship as well as an entity. The many to many relationship will be converted to 1 to many relationship. Composite Entities are also called Bridge Entities, because they acts like a bridge between the two entities which have many to many relationship. Bridge or Composite entity composed of the primary keys of each of the entities to be connected. A composite entity is represented by a diamond shape with in a rectangle in an ER Diagram.
Supertypes and Subtypes Entities - A subtype is a subprouping of the entities in an entity type that is meaningful to the organisation. For example, In a University, a STUDENT is an entity type. Two subtypes of STUDENT entity are

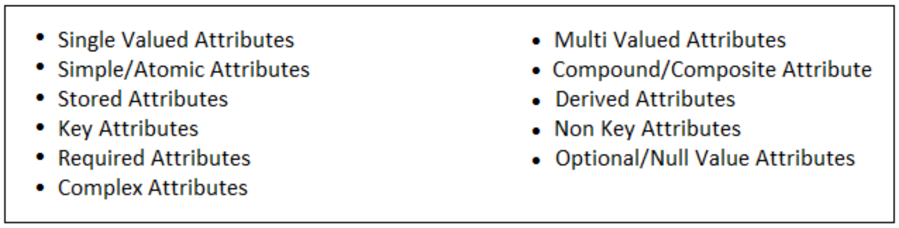
- GRADUATE STUDENT
- UNDERGRADUATE STUDENT

A supertype is a generic entity type that has a relationship with one or more sub-types. In the above example, STUDENT entity is a supertype

Types of Attributes in DBMS with Example (ER MODEL - Part-2)

Attribute : It is the name of the column. An attribute gives the characteristics of the entity. For example, A customer of bank may be described by : name, address, customer ID number. It is also called as data element, data field, a field, a data item, or an elementary item.

Type of Attributes in DBMS –



- Single Valued Attributes
- Simple/Atomic Attributes
- Stored Attributes
- Key Attributes
- Required Attributes
- Complex Attributes
- Multi Valued Attributes
- Compound/Composite Attribute
- Derived Attributes
- Non Key Attributes
- Optional/Null Value Attributes

Single valued Attributes :
An attribute, that has a single value for a particular entity is known as single valued attributes. For example, age of a employee entity.
Multi valued Attributes : An attributes that may have multiple values for the same entity is known as multi valued attributes. For example colors of a car entity.
Compound Attribute/Composite Attribute : Attribute can be subdivided into two or more other Attribute. For Example, Name can be divided into First name, Middle name and Last name.
Simple Attributes/Atomic Attributes : The attributes which cannot be divided into smaller subparts are called simple or atomic attributes. For example, age of employee entity
Stored Attribute : An attribute, which cannot be derived from other attribute, is known as stored attribute. For example, BirthDate of employee.
Derived Attribute : Attributes derived from other stored attribute. For example age from Date of Birth and Today's date.

- An attribute can be derived from a single attribute. Example age from DOB and current date.
- An attribute can be derived from multiple attribute.
- An entity can be derived from a separate table. Example,

RNO	Name	DeptNO
1	A	1
2	B	1
3	C	2
4	D	3
5	E	1

$\Rightarrow$

DNO	Dname
1	CSE
2	IT
3	ECE

Complex Attributes : If an attribute fr an entity, is built using composite and multivalued attributes, then these attributes are called complex attributes. For example, a person can have more than one residence and each residence can have multiple phones, an addressphone for a person entity can be specified as –

{Addressphone (phone {(Area Code, Phone Number)}, Address(Sector Address (Sector Number,House Number), City, State, Pin))}

Here {} are used to enclose multivalued attributes and () are used to enclose composite attributes with comma separating individual attributes.

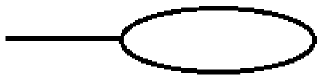
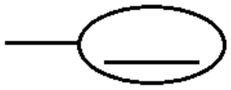
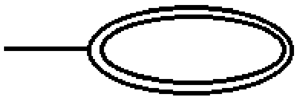
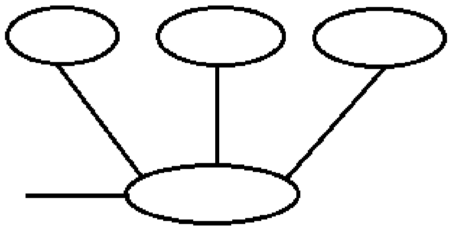
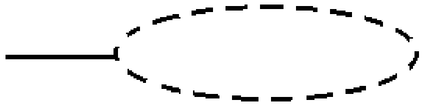
Key Attribute : represents primary key. (main characteristics of an entity). It is an attribute, that has distinct value for each entity/element in an entity set. For example, Roll number in a Student Entity Type.

Non Key Attributes : These are attributes other than candidate key attributes in a table. For example Firstname is a non key attribute as it does not represent the main characteristics of the entity.

Required Attribute : A required attribute is an attribute that must have a data value.These attributes are required because they describe what is important in the entity. For example, In a STUDENT entity, firstname and lastname is a required attribute.

Optional Attribute/Null Value Attribute - An optional attribute may not have a value in it and can be left blank. For example, In a STUDENT entity, Middlename or email address is an optional attribute. as some students may not have middlename or email address.

Notations Of Attributes in ER Diagram

	**Attribute**
	**Key Attribute**
	**Multivalued Attribute**
	**Compound/Composite Attribute**
	**Derived Attribute**

Prime and Non Prime Attributes in DBMS with Example -

- **Prime Attributes -** Attribute set that belongs to any candidate key are called Prime Attributes. (union of all the candidate key attribute) {CK1 ∪ CK2 ∪ CK3 ∪ ......} If Prime attribute determined by other attribute set, then more than one candidate key is possible. For example, If A is Candidate Key, and $X \to A$, then, X is also Candidate Key .
- **Non Prime Attribute -** Attribute set does not belongs to any candidate key are called Non Prime Attributes.

Some Questions Based on Prime Attributes and Non Prime Attributes -

Question 1 :

Given a relation R(ABCDEF) having FDs

{AB→C, C→D, D→E, F→B, E→F}

Identify the prime attributes and non prime attributes.

Solution :

$(AB)^+$: {ABCDEF} ⇒ Super Key

$(A)^+$: {A} ⇒ Not Super Key

$(B)^+$: {B} ⇒ Not Super Key

Prime Attributes : {A,B}

(AB) → Candidate Key

↓ (as F → B)

$(AF)^+$: {AFBCDE}

$(A)^+$: {A} ⇒ Not Super key

$(F)^+$: {FB} ⇒ Not Super Key

(AF) → Candidate Key

↓

$(AE)^+$: {AEFBCD}

$(A)^+$: {A} ⇒ Not Super key

$(E)^+$: {EFB} ⇒ Not Super key

(AE) → Candidate Key

↓

$(AD)^+$: {ADEFBC}

$(A)^+$: {A} ⇒ Not Super key

$(D)^+$: {DEFB} ⇒ Not Super key

(AD) → Candidate Key

↓

$(AC)^+$: {ACDEFB}

$(A)^+$: {A} ⇒ Not Super Key

$(C)^+$: {DCEFB} ⇒ Not Super Key

⇒ Candidate Keys {AB, AF, AE, AD, AC}

⇒ Prime Attributes {A,B,C,D,E,F}

⇒ Non Prime Attributes {}

Question 2:
Given a relation R(ABCDEF) having FDs
{AB → C, C → DE, E → F, C → B}
Identify the prime attributes and non prime attributes.
Solution :
$(AB)^+$: {A B C D E F}
$(A)^+$: {A}
$(B)^+$: {B}
(AB) ⇒ (AC), $(AC)^+$: {ABCDEF}
$(C)^+$: {DECBF}
⇒ Candidate Keys {AB, AC}
⇒ Prime Attributes {A,B,C}
⇒ Non Prime Attributes {D,E,F}
Question 3:
Given a relation R(ABCDEFGHIJ) having FDs
{AB → C, A → DE, B → F, F → GH, D → IJ}
Identify the prime attributes and non prime attributes.
Solution :
$(AB)^+$: {ABCDEFGHIJ}
$(A)^+$: {DEIJA}
$(B)^+$: {FGHB}
⇒ Candidate Keys {AB}
⇒ Prime Attributes {A,B}
⇒ Non Prime Attributes {C,D,E,F,G,H,I,J}
Question 4:
Given a relation R(ABDLPT) having FDs
{B → PT, A → D, T → L}
Identify the prime attributes and non prime attributes.
Solution :
$(AB)^+$: {ABPTDL}
$(A)^+$: {DA}
$(B)^+$: {BPTL}
⇒ Candidate Keys {AB}
⇒ Prime Attributes {A,B}
⇒ Non Prime Attributes {D,L,P,T}

Question 5:
Given a relation R(ABCDEFGH) having FDs
$\{E \rightarrow G, AB \rightarrow C, AC \rightarrow B, AD \rightarrow E, B \rightarrow D, BC \rightarrow A\}$
Identify the prime attributes and non prime attributes.
Solution :
$(ABFH)^+$: {ABCDEFGH}
$(A)^+$: {A}
$(B)^+$: {BD}
$(F)^+$: {F}
$(H)^+$: {H}
(AB) : {ABCDEG}
(AF) : {AF}
(AH) : {AH}
(BF) : {BFD}
(BH) : {BHD}
(FH) : {FH}
(ABF) : {ABFCDEG}
(ABH) : {ABHCDEG}
(AFH) : {AFH}
(BFH) : {BDFH}
None of the proper sets of {ABFH} will determine all the attributes.
So {ABFH} ⇒ minimal super key or candidate key

(A B FH)
↓
(A (AC) FH) ⇒ (ACFH)

(A BFH)
↓
((BC) BFH) ⇒ (BCFH)
⇒ Candidate Keys {ABFH, ACFH, BCFH}
⇒ Prime Attributes {A,B,C,F,H}
⇒ Non Prime Attributes {D,E,G}
Question 6 :
Given a relation R(ABCDE) having FDs
$\{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$
Identify the prime attributes and non prime attributes.
Solution :
$(A)^+$: {ABCDE} ⇒ (Candidate Key)
$(E)^+$: {ABCDE} ⇒ (Candidate Key)
⇒ Candidate Keys {A,E}
⇒ Prime Attributes {A,E}
⇒ Non Prime Attributes {B,C,D}

Question 7 :
Consider a relation scheme R(ABCDEH) on which the following Functional Dependencies hold:
$\{A \rightarrow B, BC \rightarrow D, E \rightarrow C, D \rightarrow A\}$.
What are the candidate keys of R and identify the prime attributes and non prime attributes.
Solution :
$(BEH)^+$: {BEHCDA} ⇒ Super Key
$(B)^+$: {B}
$(E)^+$: {EC}
$(H)^+$: {H}
$(BE)^+$: {BECDA}
$(BH)^+$: {BHD}
$(EH)^+$: {EHC}

None of the proper sets of {BEH} will determine all the attributes.
So, {BEH} ⇒ minimal super key or candidate key.

(B EH)
↑
(A EH)
↑
(D EH)
⇒ Candidate Keys {BEH, AEH, DEH}
⇒ Prime Attributes {A,B,D,E,H}
⇒ Non Prime Attributes {C}

Relationship Constraints in DBMS -

There are three Types of Relationship Constraints-

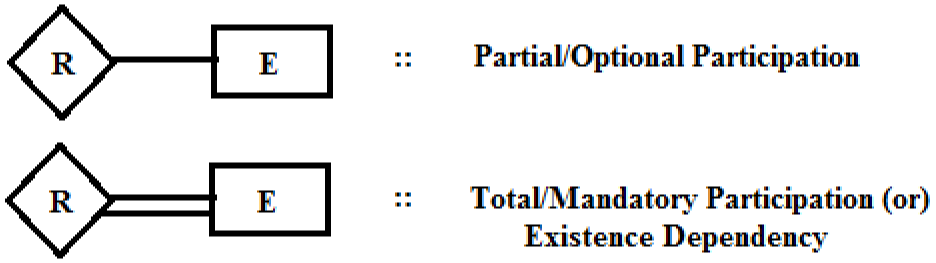
1. **Structural Constraints**
   - **Participation Constraints**
   - **Cardinality Ratio**
2. **Overlap Constraints**
3. **Covering Constraints**

Structural Constraints are applicable for binary relationships and **Overlap and Covering Constraints** are applicable for EERD(Extended ER Diagrams).

Participation (or) Optionality Constraints-
Participation concerns with the involvement of entities in a relationship. It specifies whether the existence of an entity depends on another entity. There are two types of Participation Constraints -

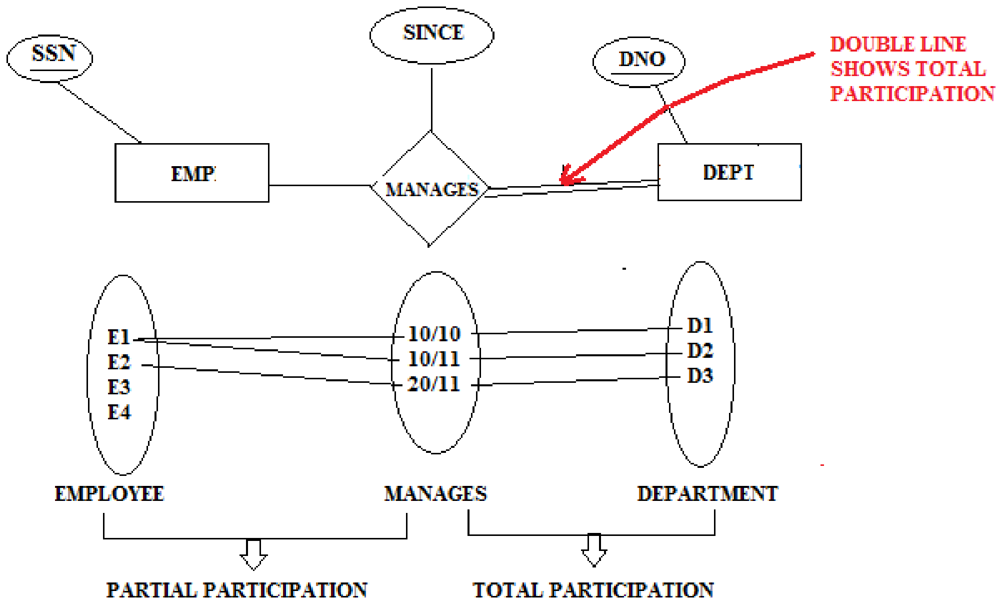
1. **Total/Mandatory Participation**
2. **Partial/Optional Participation**

Notations of Different Types of Participation In ER Diagram -


R
E
:: Partial/Optional Participation
R
E
:: Total/Mandatory Participation (or) Existence Dependency

Total/Mandatory Participation or Existence Dependency-
Participation is said to be total if every entity in E participates in atleast one relationship in R (or) every entity in entity set must depend on another entity. For example, every department will have a startup date as a department was created on that startup date (SINCE ATTRIBUTE of MANAGE relationship). and that department is being handled from that date through a Manager. So, the participation of DEPARTMENT entity in the "MANAGE" relationship type is total. Total Participation is also known as **Existence Dependency**. In ER Diagram, it is represented as a Double Line, connecting the participating entity to the relationship.

Partial/Optional Participation-
Participation is said to be partial if only some entities in E participate in relationships in R (or) some entities in entity set are depend on some another entities in entity set. For example, It is not necessary that all employees manage some department Because all employees may not be the Manager . So the participation of "EMPLOYEE" entity in the "MANAGES" relationship type is partial.



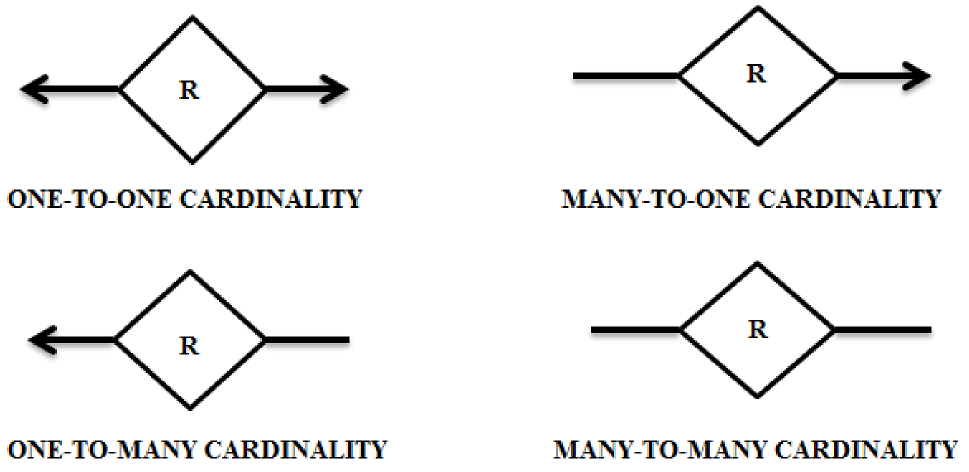
(Figure-1)

Cardinality/Mapping Cardinality-

Cardinality expresses the number of entities to which another entity can be associated via a relationship set (or) It specifies the number of relationship instances that an entity can participate in a relation set. There are 4 types of Cardinality Ratios :

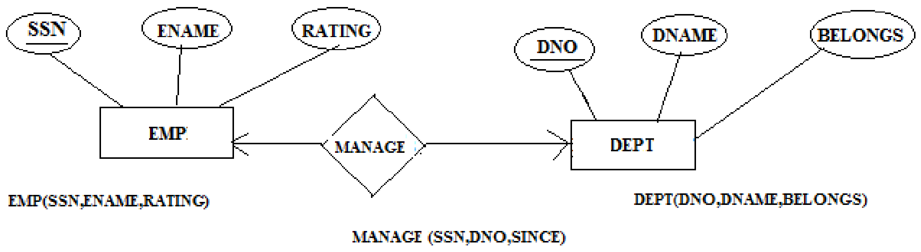
- **One-to-One Cardinality (1:1)**
- **One-to-Many Cardinality (1:m)**
- **Many-to-One Cardinality (m:1)**
- **Many-to-Many Cardinality (m:n)**

Notations of Different Types of Cardinality In ER Diagram -



One-to-One Cardinality (1:1) -

An entity in set A is associated with atmost one entity in B, and an entity in B is associated with atmost one entity in A. This type of cardinality is referred as one to one Cardinality. For example, an Employee as a Manager manage only one Department and the vice versa is also true as a department have only one Manager

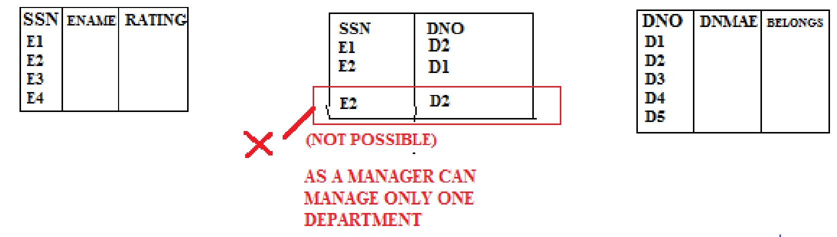


EMP(SSN,ENAME,RATING)

DEPT(DNO,DNAME,BELONGS)

MANAGE (SSN,DNO,SINCE)

Relational Tables -



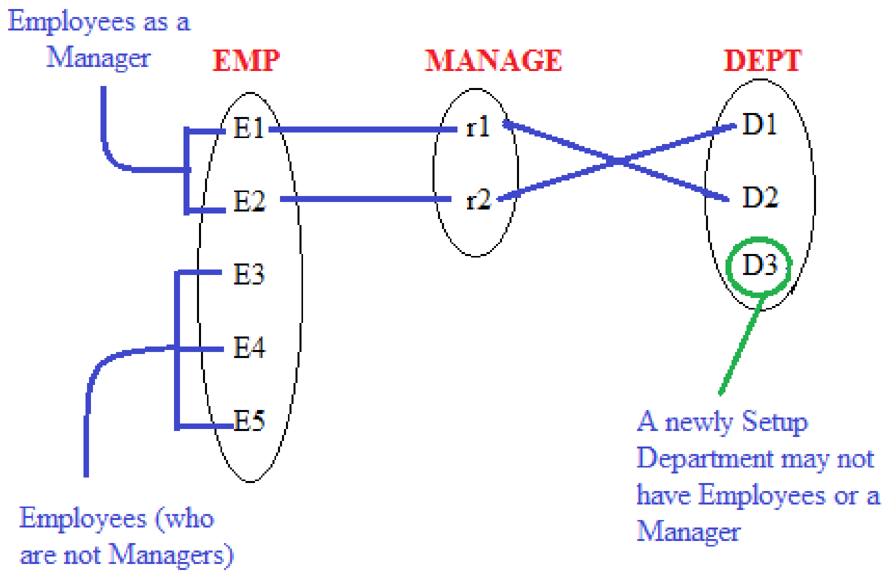
SSN	ENAME	RATING
E1		
E2		
E3		
E4		

SSN	DNO
E1	D2
E2	D1
E2	D2

(NOT POSSIBLE)

AS A MANAGER CAN MANAGE ONLY ONE DEPARTMENT

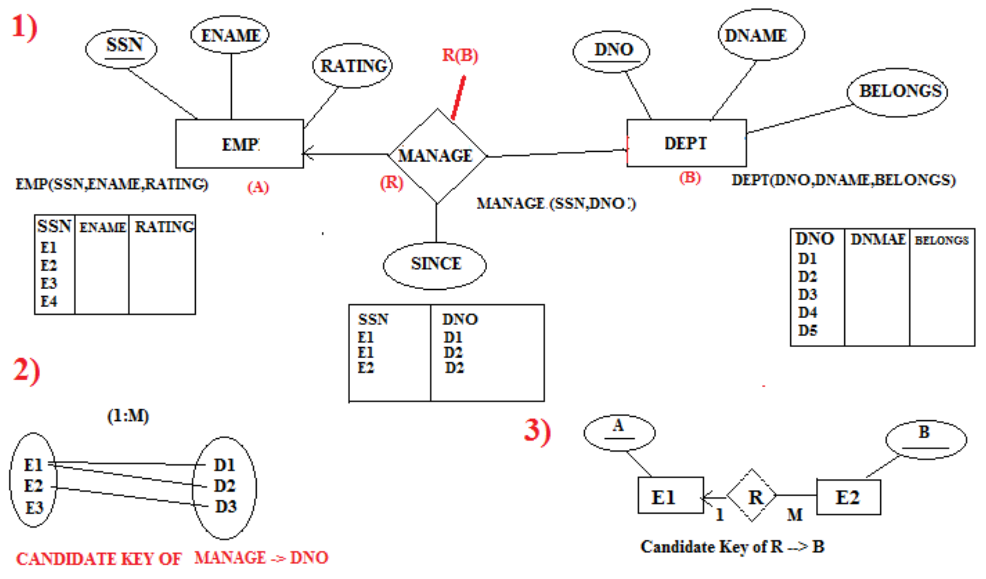
DNO	DNMAE	BELONGS
D1		
D2		
D3		
D4		
D5		


Employees as a Manager
EMP
MANAGE
DEPT
E1
E2
E3
E4
E5
r1
r2
D1
D2
D3
A newly Setup Department may not have Employees or a Manager
Employees (who are not Managers)

```
Create Table MANAGE
( SSN varchar(10),
  DNO varchar(15),
  Primary key(SSN),
  Foreign Key(SSN) .........,
  Foreign Key(DNO) .........,
);
```

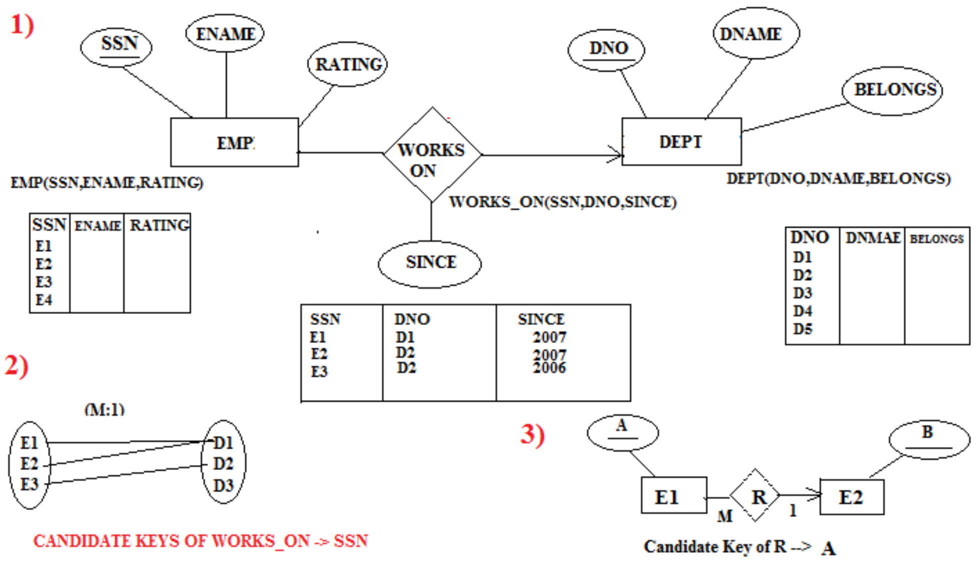
Candidate Keys of "MANAGE" relationship - SSN,DNO

One to Many Cardinality (1:M) -
An entity in A is associated with any number (0 or more) with an entity B, but a entity in B, however can be associated with atmost one entity in A. For example, An employee as a Manager can manage more than one Department.



Create Table MANAGE
(SSN varchar(10),
  DNO varchar(15),
  SINCE date,
  Primary key(DNO),
  Foreign Key(SSN) ....,
  Foreign Key(DNO) ....,
);
Candidate Keys of "MANAGE" relationship - DNO

Many to One Cardinality (M:1) -
An entity in A is associated with atmost one entity in B. An entity in B, however, can be associated with any number (0 or more) of entities in A. For example, An Employee can work only for one Department, But each Department can have 0 or more employees.



SSN	ENAME	RATING
E1		
E2		
E3		
E4		

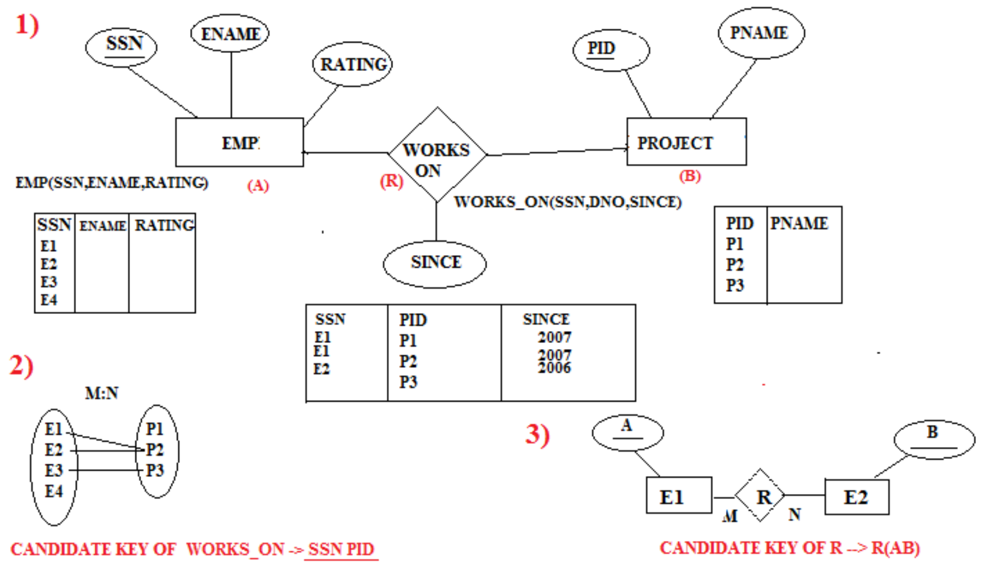
SSN	DNO	SINCE
E1	D1	2007
E2	D2	2007
E3	D2	2006

DNO	DNMAE	BELONGS
D1		
D2		
D3		
D4		
D5		

```
Create Table WORKS_ON
( SSN varchar(10),
  DNO varchar(15),
  SINCE date,
  Primary key(SSN),
  Foreign Key(SSN) .........,
  Foreign Key(DNO) .........,
);
```

Candidate Keys of "WORKS_ON" relationship - SSN

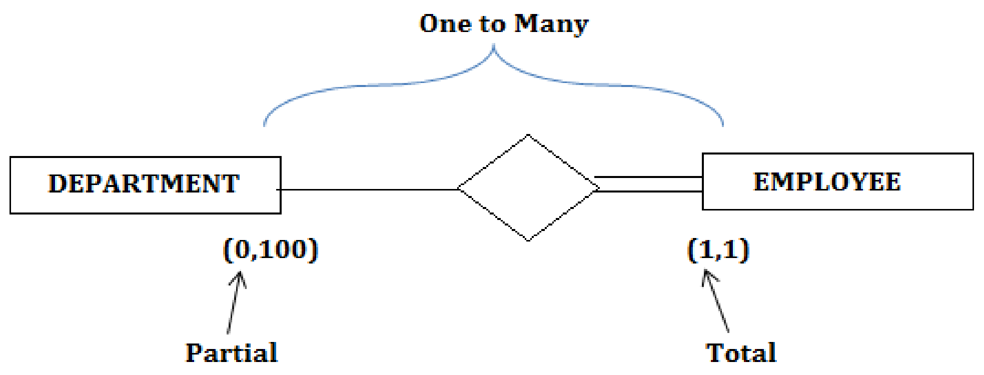
Many to Many Cardinality (M:N) -
An entity in A is associated with any number (0 or more) of entities in B, and an entity in B is associated with any number(0 or more) of entities in A. For example, An Employee can works on several Projects and a Project may have several Employees.



Create Table WORKS_ON
(SSN varchar(10),
  DNO varchar(15),
  SINCE date,
  Primary key(SSN,DNO),
  Foreign Key(SSN) ....,
  Foreign Key(DNO) ....,
);
Candidate Keys of "WORKS_ON" relationship - <u>SSN PID</u>

Some Points regarding Cardinality and Participation -

1. Minimum Cardinality = 0, for partial participation and equal to 1 for total participation.
2. Maximum Cardinality = n, if one entity occurrence relates to multiple entity occurrences.
3. Minimum Cardinality give partial or total participation.
4. Maximum Cardinality give the maximum number of entities that is related to.



Relation Between Cardinality and Joins

Cardinality means it is a relation between tables using joins which tells that how many rows of one table will match with rows in other tables when these tables are joined. In other words Relationships between the tables define Cardinality while explaining how each table links to the other.

Example :

Consider the two tables Consumer(custID) and Bill(BID, FK_custID). Each consumer pays the electricity bill every month. So, a customer will be associated with 12 bills after one year. It means there is one to many relationship between consumer and the bills he paid. In this case, the cardinality is defined from primary key of the Customer table(PK_Custid) to the foreign key (FK_Custid) of the Bill table. The number of same rows in both the tables depends upon which join is used.

Types of Relationships in ER Diagram

Different types of relationships in er diagram are - 1) Relationships based on degree, 2) Recursive Relationship. Let us discuss these one by one.

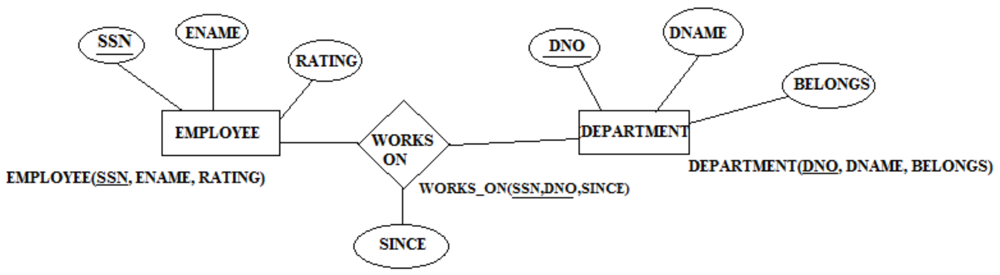
Relationship and Relationship Set :

Relationships connect the entities and represent meaningful dependencies between them.It represents an association among several entities. **Relationships sets** is a set of relationships of the same type. It is a mathematical relation on entity sets ($n>=2$). Relationship set R is a subset of -

$\{(r1,r2,r3,....r_n) \mid r1 \in E1,\ r2 \in E2,\ r_n \in E_n\}$

where $r1,r2,....r_n$ are called relationships and $E1,E2,....E_n$ are entity sets. The way in which two or more entity types are related is called **relation type.** For example, consider a relationship type **WORKS_FOR** between the two entity types EMPLOYEE and DEPARTMENT, which associates or links each employee with the department the employee works for. The WORKS_FOR relation type is shown as –

ER DIAGRAM -

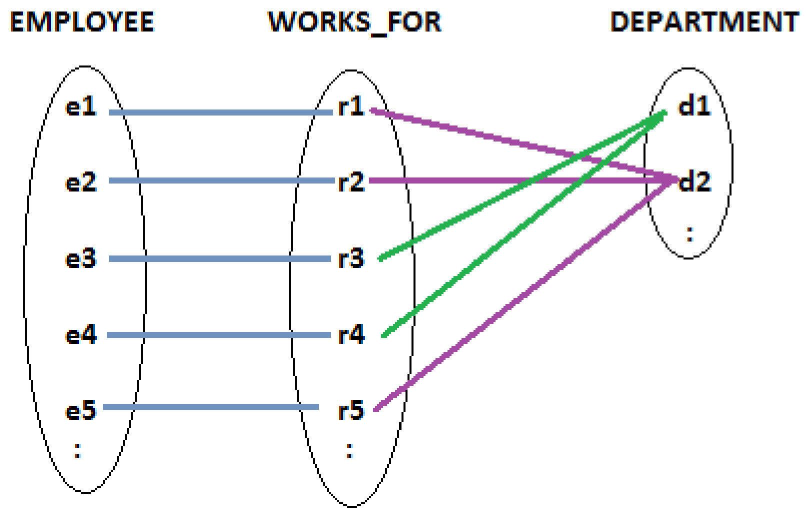


RELATIONAL DBMS OF ABOVE ER DIAGRAM -

SSN	ENAME	RATING
E1		
E2		
E3		
E4		
E5		

SSN	DNO	SINCE
E1	D2	2008
E2	D2	2008
E3	D1	2004
E4	D1	2006
E5	D2	2004
E9	D10	2005

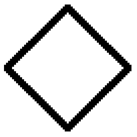
DNO	DNAME	BELONGS
D1		
D2		



In the above figure, each instance of relation type WORKS_FOR i.e.(r1, r2,...,r5) is connected to instances of employee and department entities. Employee e1, e2 and e5 work for department d2 and employee e3 and e4 work for department d1.

Notation to Represent Relation Type in ER Diagram-

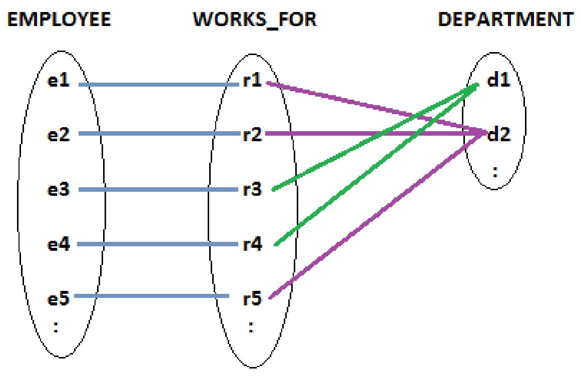
Relation types are represented as diamond shaped boxes.



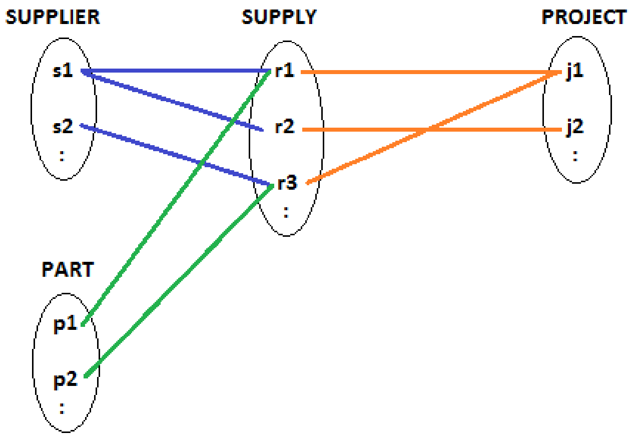
Degree of a Relationship Type-
The number of participating entity types is known as the degree of relationship type.

Types of Relationship Type Based on Degree -

- Binary Relationship - A relationship type of degree two is called binary relationship. The WORKS_FOR in above figure is a binary relationship as there are two participating entities-employee and department.



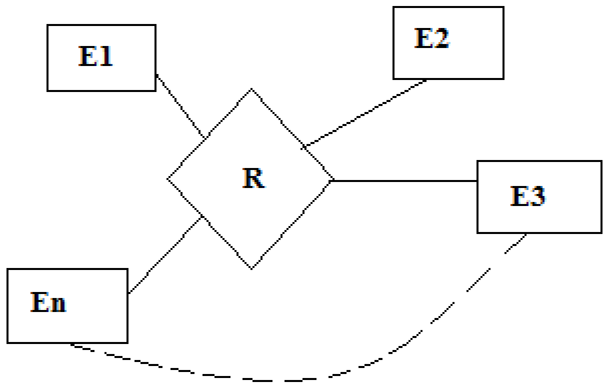
- Ternary Relationship- A relationship type of degree three is a ternary relationship for example, in the below figure **supply** relationship connects three entities



SUPPLIER, PART AND PROJECT.

The above diagram can be read as - a supplier supplies the parts to projects

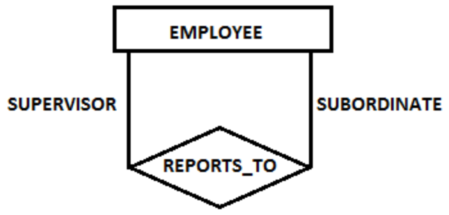
- **N-ary Relationship Set -** A relationship type of degree n is called n ary relationship . For example

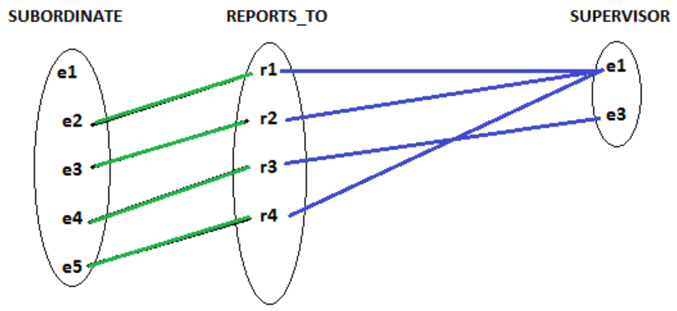


Attributes of R --> Primary Key of all Entities

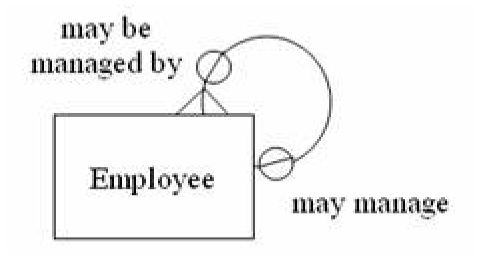
Role Names-
A relationship type has a name which signifies what role a participating entity plays in that relationship instance. The role names helps to explain what the relationship means. In the first example WORKS_FOR relationship type, employee plays the role of worker and department plays the role of employee(because a department consists of a number of employees.

Recursive Relationship
If the same entity type participate more than once in a relationship type in different roles then such relationship types are called recursive relationship. For example, in the below figure REPORTS_TO is a recursive relationship as the Employee entity type plays two roles - 1) Supervisor and 2) Subordinate.





You can also define the above example of recursive relationship as the relationship between a manager and an employee. An employee is a manager as well as employee.

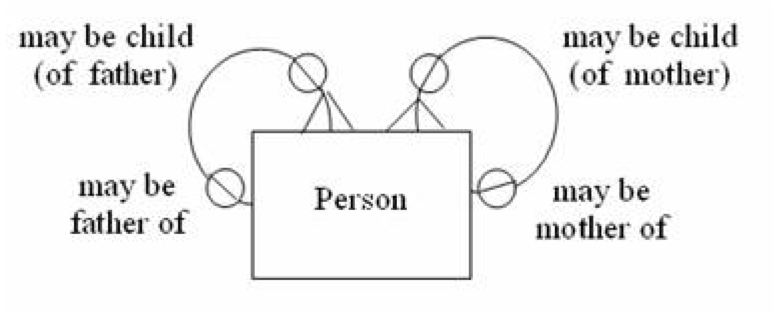


This is commonly know as a 'pig's ear'.

To implement recursive relationship, a foreign key of the employee's manager number would be held in each employee record.
Emp_entity(Emp_no,Emp_Fname, Emp_Lname, Emp_DOB, Emp_NI_Number, Manager _no);

Manager no - (this is the employee no of the employee's manager)

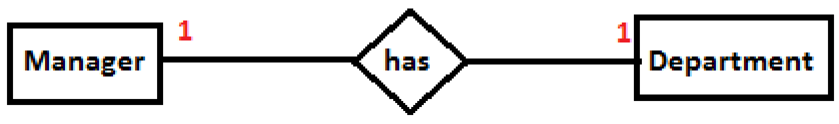
Another more Complicated Example of Recursive Relationship :
Relationship between a person and their parents :

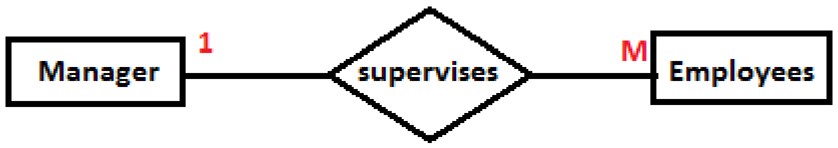


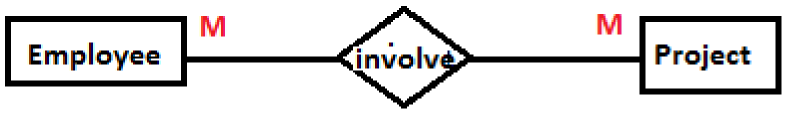
Composite Entity Example -

Composite Entity-
As earlier said in previous post (entity and its types), Composite Entities are the entities which exists as a relationship as well as an entity. Consider the three types of connectivity -

1. one to one cardinality
2. one to many cardinality
3. many to many cardinality

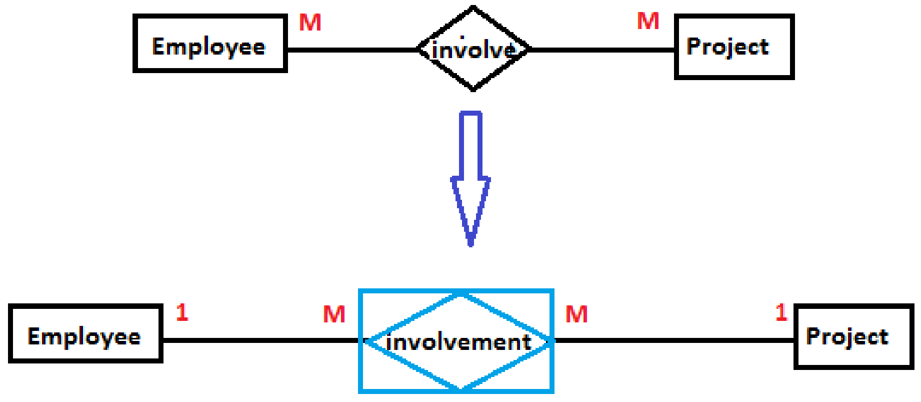






A many to many relationship does not exist during finalizing ERD (ER Diagram) as many to many relationship will cause problems to ERD in converting into Relational model. So, elimination of many to many relationship is necessary and is done by including the composite entities in ER Diagram.

Composite Entity Example :
Consider the above example, EMPLOYEE and PROJECT is many to many relationship. because each employee could be involved in a number of different projects and a number of employees could be working in a given project. So, the relationship notation will be replaced by the composite entity notation, which is shown as -



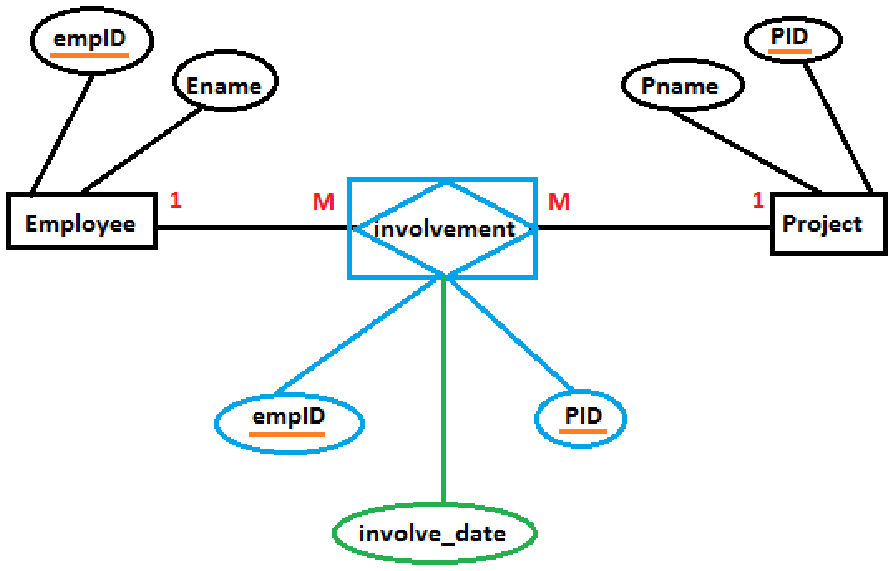
Changes in the Above Diagram are:

1) Relationship symbol is converted into composite entity symbol. The composite entity is represented by combining the relationship symbol (diamond) with the entity symbol (rectangle).

2) Entity names are usually described with a noun name(objects-employee and projects) and relationship are usually described with a verb (involve). Since the composite entity is the combination of an entity and a relationship, the name of the many-to-many relationship should be changed(involvement).

3) Redo the relationship connectivity- the employee and the project are no longer directly related to each other. The new relationship is now between the employee and involvement, and the project and involvement. So reconsidering the connectivities among the new entities-
⇒ each employee has involvement in many projects(one-to-many relationship) and
⇒ in each project, there is involvement of many employees(many-to-one relationship).

Key of the Composite Entity -
The key of the composite Entity is the combination of other two table's key. In the above example, if the key of the EMPLOYEE entity is empID and the key of the PROJECT entity is PID then the key of the composite entity INVOLVEMENT will be the combination of

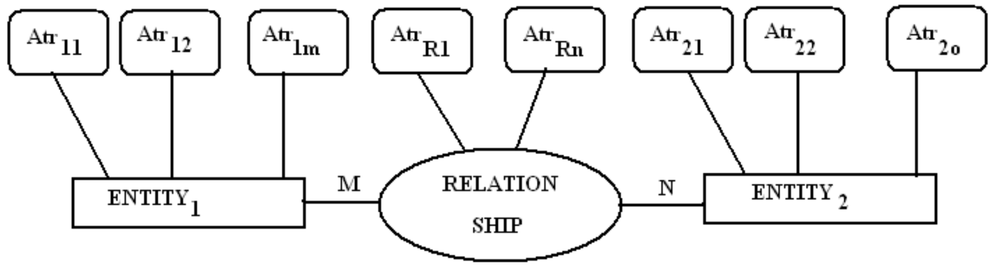


(empID, PID). In case of composite entity, the key will always be a composite key i.e. the combination of the other two table's keys.

Attribute of the Composite Entity -
We can give extra attributes to the composite entity. in the above example, involve_date will be the extra attribute given to the composite key.

How to Draw ER Diagram ??

We have read all the basic terms of ER Diagram. Now, let's understand how to draw er diagram? In ER Model, objects of similar structures are collected into an **entity set**. The relationships between an entity sets is represented by a named **E-R relationship**, which may be **(one-to-one, one-to-many, many-to-one, many-to-many)**, which maps one entity set to another entity set. A General ER Diagram is shown as-



In Figure, there are two entities **ENTITY-1** and **ENTITY-2** having attributes **(Atr_{11}, Atr_{12}, ... Atr_{1m})** and **(Atr_{21}, Atr_{22}, ... Atr_{2n})** respectively, connected via many to many relationship **(M:N).** The attributes of **RELATIONSHIP** are **(Atr_{R1}, Atr_{R2}, ... Atr_{RO}).**

Steps - How to Draw ER Diagram -

1. Identify all the entities of the given problem
2. Identify all the attributes of the entities identified in step 1.
3. Identify the Primary Keys of entities identified in Step 1.
4. Identify the Attribute Types of attributes identified in step 2
5. Identify relationship between the entities and constraints on the entities and implement them.

Need of ER Diagram -

The ER Diagrams are useful in representing the relationship among entities. It helps to show basic data structures in a way that different people can understand. Many types of people are involved in the database environment, including programmers, designers, managers and end users. But not all of these people work with database and might not be as skilled as others to understand the making of a software or a program etc, so, a conceptual model like the ERD helps show the design to many different people in a way they can all understand.

Example of drawing ER Diagram -

How to draw er diagram of a company database if the following requirements are given : Question : Make an ER Diagram for the company database with the following description :

1. The company is organised into departments. Each department has a unique name and a unique number. A department may have several locations.
2. A department controls a number of projects, each of which has a unique name, a unique number and a single location.

3. We store each employee's name, social security number, address and salary. An employee is assigned to one department but may work on several projects, which are not necessarily controlled by the same departments.
4. We want to keep track of the departments of each employee for insurance purposes. We keep each dependent's name, age and relationship to the employee.

Answer :
Step 1 : Identifies Entities of the given problem.
Entities :

1. DEPARTMENT (From 1st Point)
2. PROJECT (From 2nd Point)
3. EMPLOYEE (From 3rd Point)
4. DEPENDENT (From 4th Point)

Step 2 : Identify the attributes of the above entities.
Attributes :

1. DEPARTMENT : Name, Number, Location;
2. PROJECT : Name, Number, Location;
3. EMPLOYEE : SSN, Name, Address, Salary
4. DEPENDENT : Name, Age, Relationship

Step 3 : Identify the Primary Keys of all entities identified in Step 1.
Primary Keys :

1. DEPARTMENT : Name, Number, Location; (Unique Name and Unique Number)
2. PROJECT : Name, Number, Location; (Unique Name and Unique Number)
3. EMPLOYEE : SSN, Name, Address, Salary (Since, Social Security Number will be Unique, so SSN is selected as primary Key)
4. DEPENDENT : Name, Age, Relationship (No Unique Attribute to identify DEPENDENT entity. So, It is referred as a Weak Entity)

Step 4: Identify the Attribute Types -
Attributes Types :

1. Location Attribute of DEPARTMENT entity : Multivalued Attribute (Since there are several locations)
2. Name Attribute of EMPLOYEE entity : Composite Attribute (since a name consists of first name, middle name and last name)
3. Adderess Attribute of EMPLOYEE entity : Composite Attribute (Address consists of H.no, Street, City, State, Country)

Step 5 : Identify the Relationships and relationships attributes between the entities.
Relationships and their Attributes :

Relationship Name	Entities Name having relationships among them	Attributes

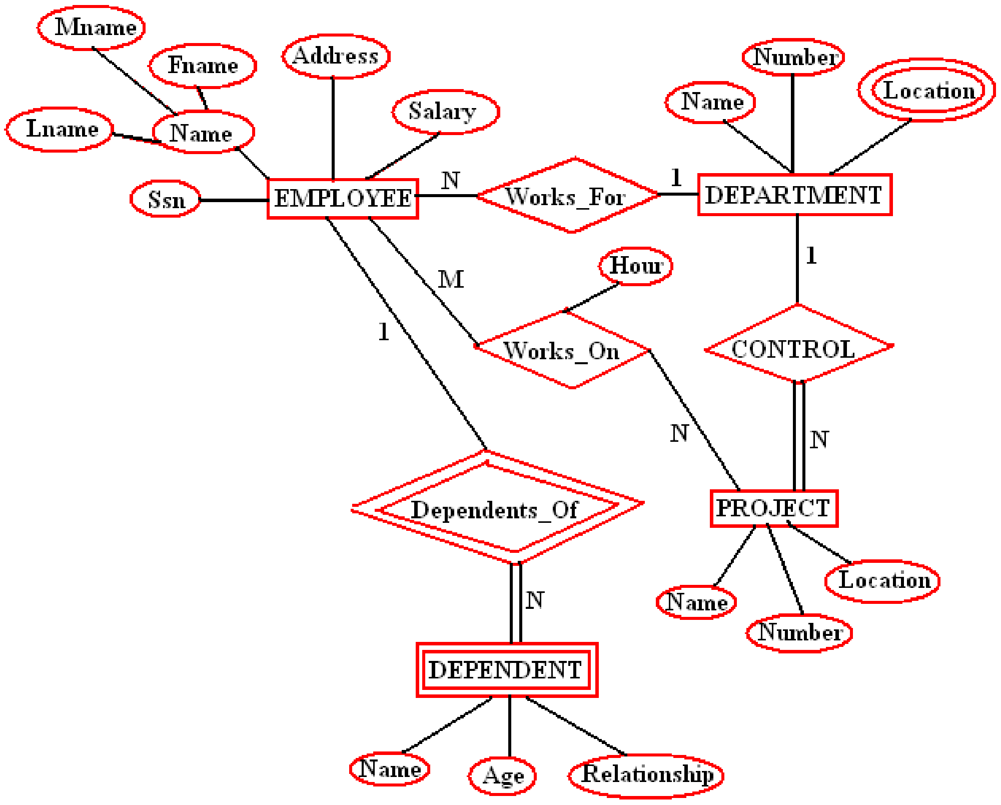
1. Works_For	EMPLOYEE and DEPARTMENT	-
2. Control	DEPARTMENT and PROJECT	-
3. Works_On	EMPLOYEE and PROJECT	Hours
4. Dependents_Of	EMPLOYEE and DEPENDENT	-

Step 6: Identify the constraints on the entities.

Cardinality Constraints :

Relationship	Cardinality	Reason
1. Works_For	N:1	Since N employees Works_For a Department.
2. Works_On	M:N	Since different employees works on different projects.
3. Control	1:N	Since a department Controls a number of projects.
4. Dependents_Of	1:N	Since each Dependents has name, age and relationship to the employee.

Implementation of ER Diagram of the given problem on the basis of above steps :



-

Identifying and Non Identifying Relationships -

Relationships among the entities may be classified as being either

- Identifying Relationships
- Non Identifying Relationships

Identifying Relationships -

Definition 1 :

Weak entity are the entities, which does not have a key attribute. So, To identify it, what we do is to relate such type of entity with some other entity type. The weak entity type relate to another entity type in combination with some of their attribute values. We call this entity type the identifying or Owner Entity Type and the relationship type which a weak entity type creates with Owner Entity Type is identifying relationship of weak entity type.

Definition 2:

If we consider the weak entity type as a child object and the owner entity type as a parent object, an Identifying relationship specifies that a child object cannot exist without the parent object and child object cannot be uniquely identified without the parent. If the parent entity is deleted, then the child entity must be deleted.

Definition 3:

Identifying relationships exist when the primary key of the parent entity is included in the primary key of the child entity. This means that foreign key is a primary key too. So, put the primary key column from the parent into the primary key column of the child (To provide uniqueness, other columns can be included in the child entity if needed).

Example :

```
CREATE TABLE AuthoredBook (
  author_id INT NOT NULL,
  book_id INT NOT NULL,
  PRIMARY KEY (author_id, book_id),
  FOREIGN KEY (author_id) REFERENCES Authors(author_id),
  FOREIGN KEY (book_id) REFERENCES Books(book_id)
);
```

author_id is a foreign key, but it's also one of the columns in the primary key. So there is an identifying relationship of the AuthoredBook with the referenced table Authors. Likewise it has an identifying relationship with Books as author_id is a foreign key, but it's also one of the columns in the primary key. So there is an identifying relationship of this table with the referenced table Books.

Some more Examples-
1) A driving licence entity cannot exist unless it is related to person entity.
2) The account of any person cannot exist without that person.
3) Account (AccountID, AccountNum, AccountTypeID)
PersonAccount (AccountID, PersonID, Balance)
Person(PersonID, Name)

The Account to PersonAccount relationship and the Person to PersonAccount relationship are identifying because the child row (PersonAccount) cannot exist without having been defined in the parent
(Account or Person).
In other words: there is no personaccount when the Person or account does not exists.

Non Identifying Relationships-

Definition 1:

A **non-identifying relationship** is when the primary key attributes of the parent entity must not become primary key attributes of the child entity.So, put the primary key column of the parent into the table of the child, but not in the primary key.

Definition 2: A non-identifying relationship means that a child entity is related to parent entity but it can be identified independently of the parent entity. The child item should be kept even though the parent is deleted.

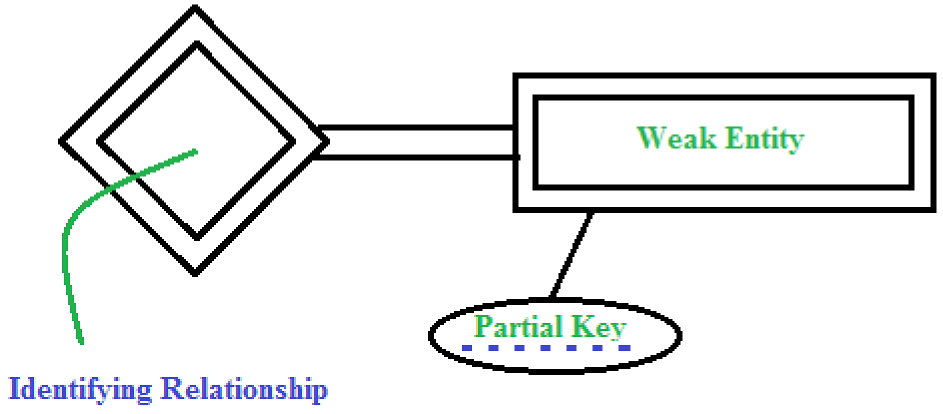
Types of Non Identifying Relationship -
Non Identifying Relationships can be further classified into -

- **Mandatory Non Identifying Relationship-** It exists when the foreign key value in the child table cannot be null or where a parent is required by setting the parent table cardinality.
- **Optional (or) Non Mandatory Non Identifying Relationship -** An optional (or) non-mandatory non-identifying relationship exists when the value in the child table can be null or where a parent is not required.

Example of Non-Identifying Relationship -
Account(AccountID, AccountNum, AccountTypeID)
AccountType(AccountTypeID, Code, Name, Description)

The relationship between the Account and AccountType is <u>non-identifying</u> because each AccountType can be identified without having to exist in the parent table.

Notation of Identifying Relationship -
In the below figure, both a weak entity types and its identifying relationship are distinguished by surrounding their boxes and diamonds with double lines as shown in the figure. Partial key attributes(e.g. Name) is underlined with a dashed or dotted line.



Generalization Specialization and Aggregation in DBMS

Generalization Specialization and Aggregation in DBMS are abstraction mechanisms used to model information. The abstraction is the mechanism used to hide the superfluous details of a set of objects. For example, vehicle is a abstraction, that includes the types car, jeep and bus. So, the two abstraction mechanisms used to model information :

- Generalization (Specialization is the reverse process of Generalization)
- Aggregation

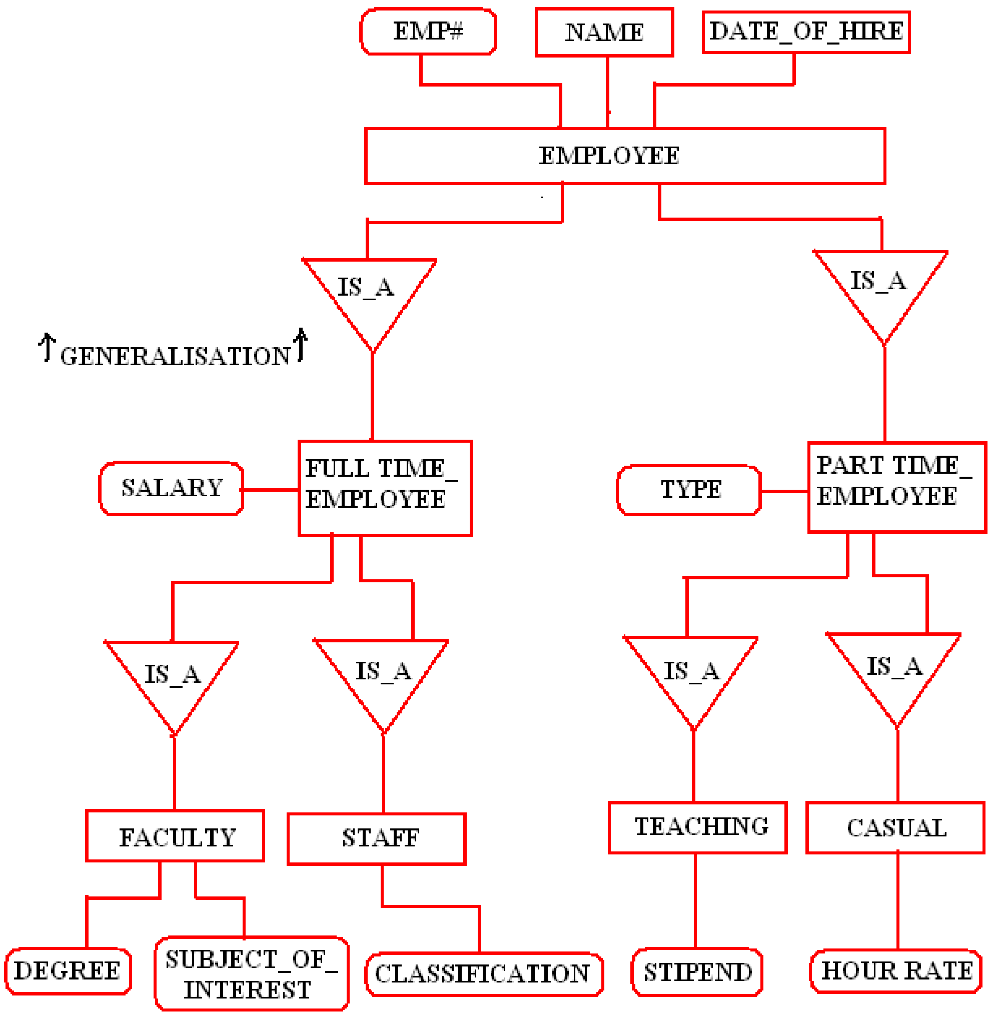
Generalization in DBMS -
Generalization is a abstracting process of viewing sets of objects as a single general class by concentrating on the general characteristics of the constituent sets while suppressing or ignoring their differences. In simple terms, Generalization is a process of extracting common characteristics from two or more classes and combining them into a generalized superclass. So, it is a bottom up approach as two or more lower lever entities are combined to form a higher level entity. The common characteristics means here attributes or methods.

⇧

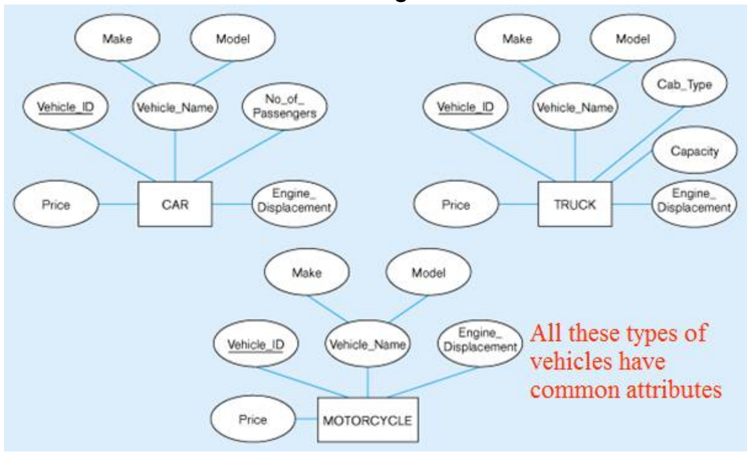
Notation of Generalization -Generalization is represented by a triangle with a line. Generalization is an IS_A relationship

Example of Generalization :
For example, a Part_Time Employee and Full_Time Employee entity sets can be generalized as EMPLOYEE entity sets; The Full_Time_Employee is a generalization of the entity set Faculty and Staff; The Part_Time_Employee is a generalization of the entity set

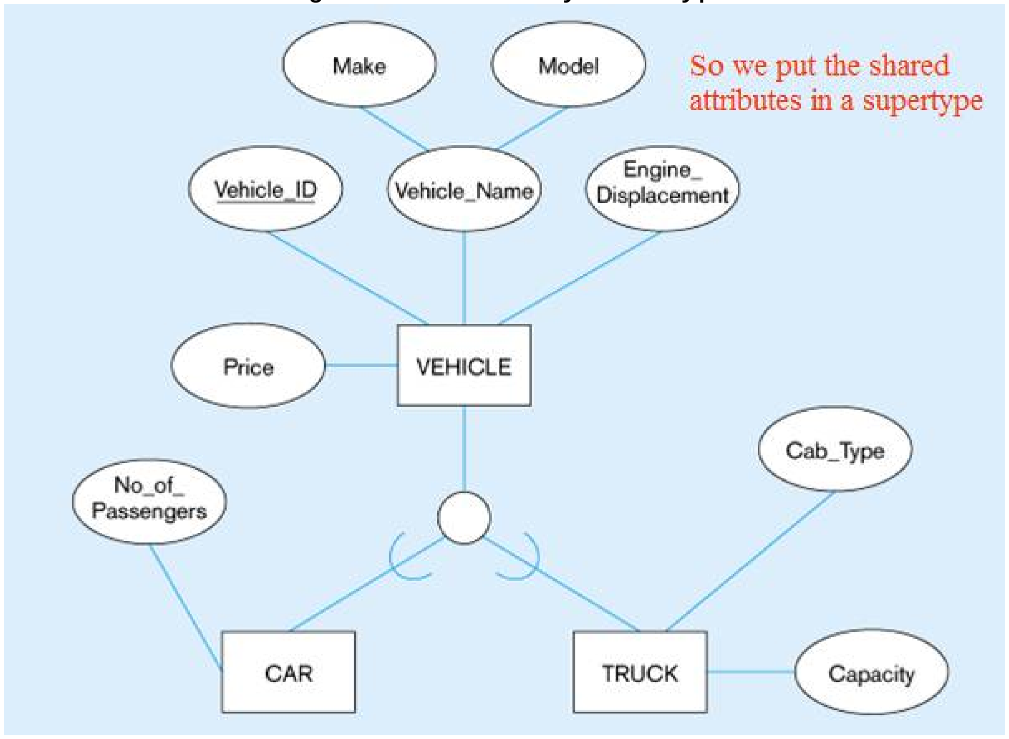
Teaching and Casual.



Another Example : Following Figures shows 3 entities: CAR, TRUCK and MOTORCYCLE In second Fig.



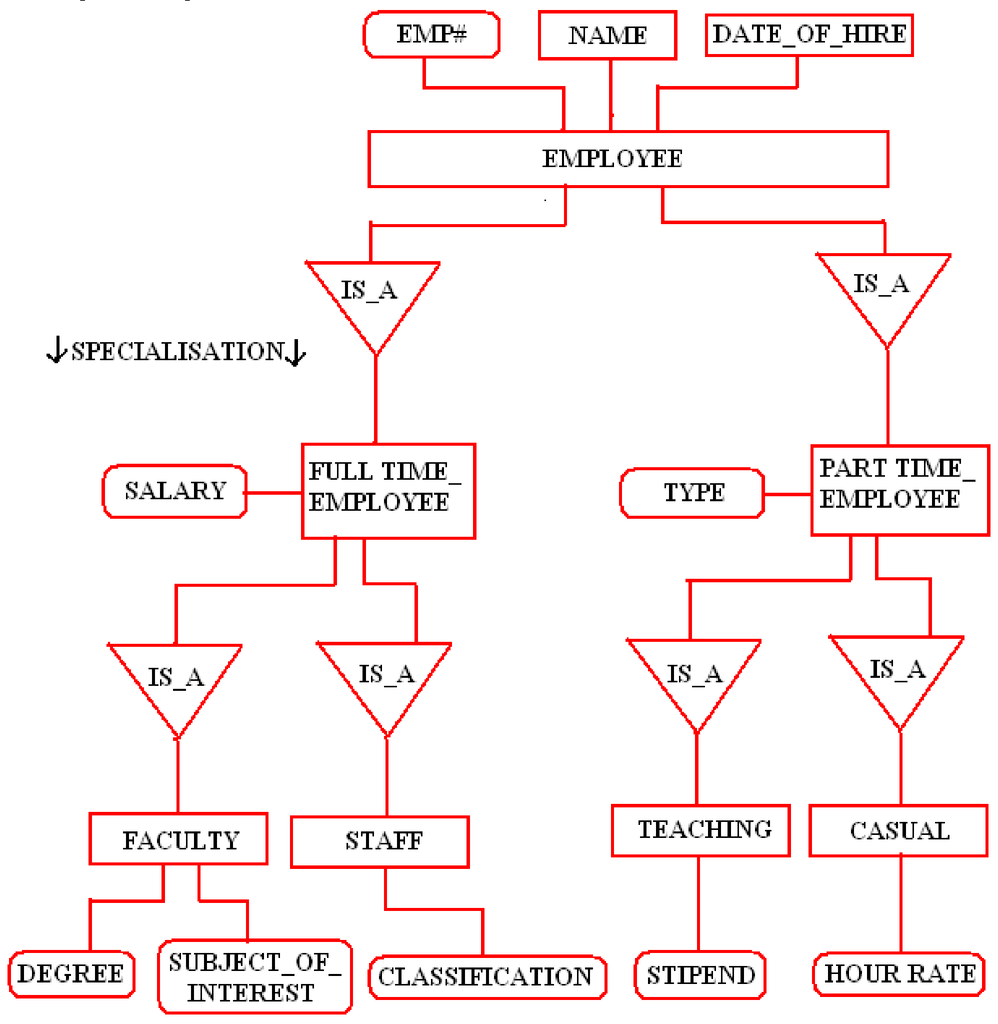
The more general entity type VEHICLE is shown.



MOTORCYCLE entity is not shown as it does not satisfy conditions for a subtype because it has all the attributes common to all vehicles. So, there are no attributes specific to motorcycles. Further, MOTORCYCLE does not have a relationship to another entity type

Specialization in DBMS -
Specialization may be seen as the reverse process of Generalization. Specialization is the abstracting process of introducing new characteristics to an existing class of objects to create one or more new classes of objects. In simple terms, a group of entities in specialization can be categorized into sub-groups based on their characteristics. So it is a top-down approach in which one higher level entity can be broken down into two lower level entity. It defines one or more subtypes of the supertypes and forming supertype/subtype relationships.

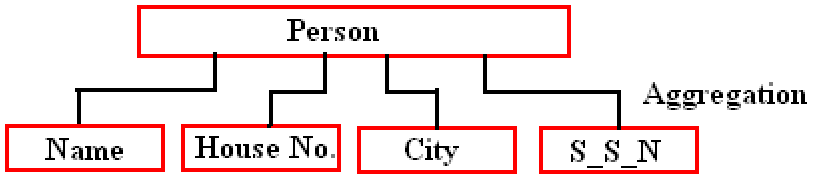
Example of Specialization :



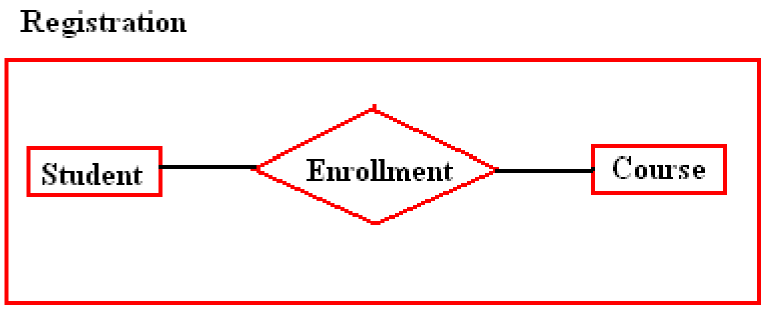
In the above example, Faculty and Staff inherits the attribute salary of the entity set Full_Time_Employee and latter, in turn inherits the attributes of Employee. So, Faculty and Staff is a specialization of Full_Time_Employee and Full_Time_Employee is a specialization of Employee. Similarly, Teaching and Casual inherits the attribute Wage of the entity set Part_Time_Employee and latter, in turn inherits the attributes of Employee. So, Teaching and Casual is a specialization of Part_Time_Employee and Part_Time_Employee is a specialization of Employee.

Aggregation in DBMS -

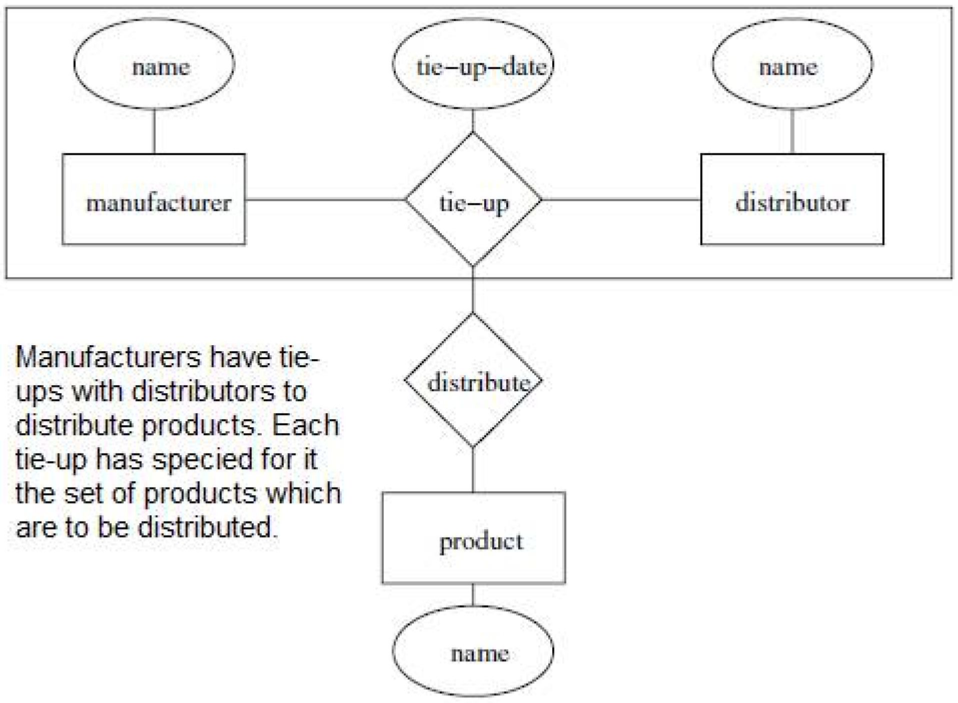
Aggregation is the process of compiling information on an object, thereby abstracting a higher level object. So,an entity Person is derived by aggregating the characteristics name, house_no., city and social security number(SSN).



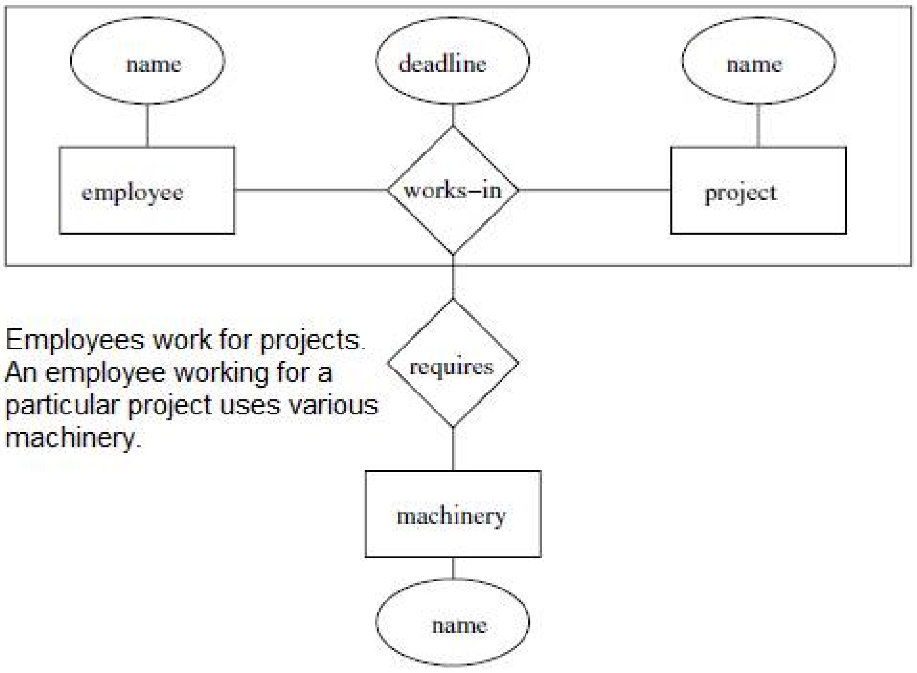
Another form of aggregation says that it is abstracting a relationship between objects and viewing the relationship as an object. So, in Figure 2, the Enrollment relationship between entities student and Course entity can be viewed as entity REGISTRATION.



In simple terms, Aggregation is a process when relation between two entity is treated as a single entity. **Another Example of Aggregation :**



Manufacturers have tie-ups with distributors to distribute products. Each tie-up has specied for it the set of products which are to be distributed.

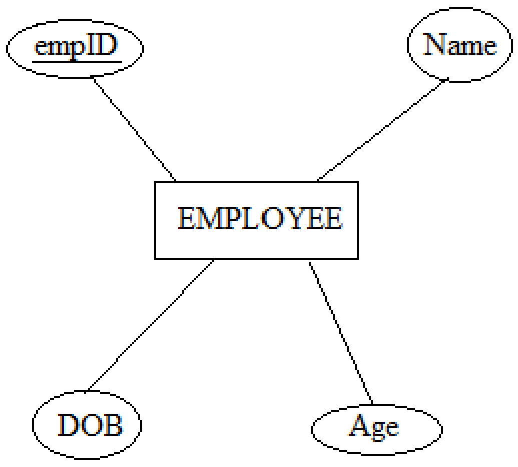


Employees work for projects. An employee working for a particular project uses various machinery.

Supertype and Subtype with Example

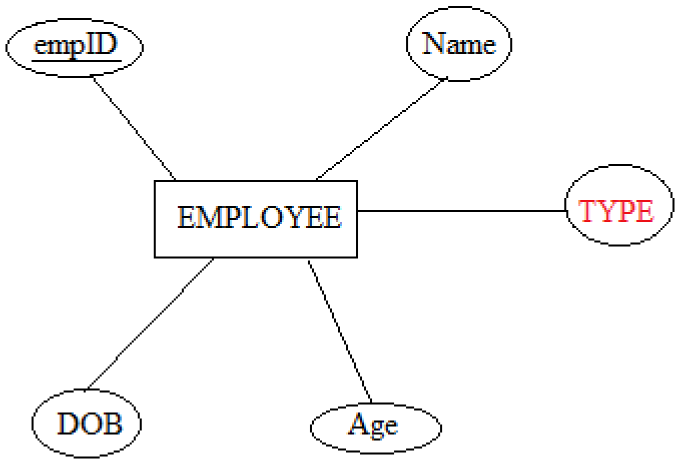
In the previous post (entity and its types) we define the concept of super and sub types. Let us thoroughly understand this concept.

- Let us assume an entity named EMPLOYEE. The entity employee has some attributes-empID, Name, Birth Date, Age etc.



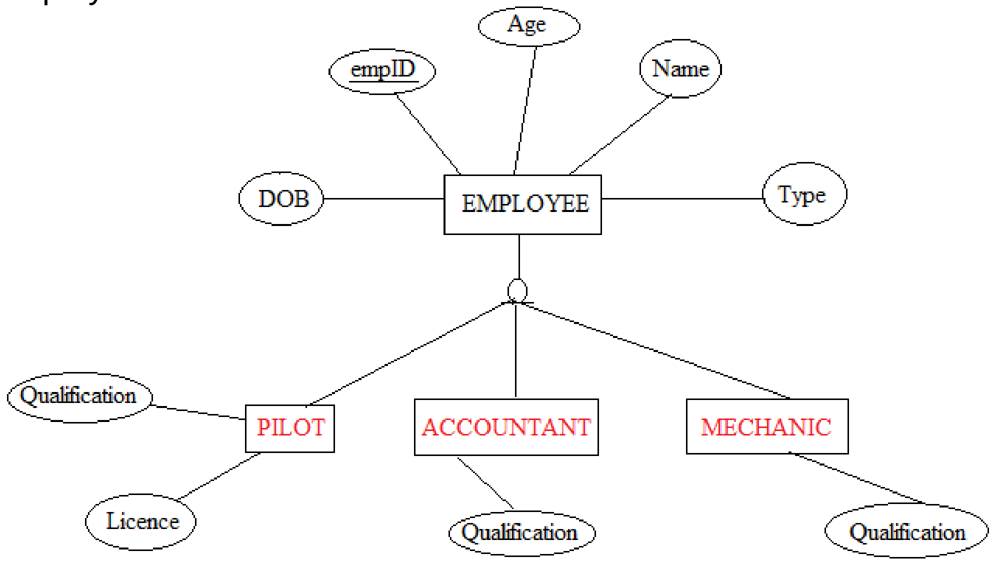
But sometimes we will have special employees that have some different attributes to other employees.

- Imagine the employees at airline company, there are different type of employees including Accountants, Mechanics, Pilots, Flight attendants etc. A quick solution would be to include another attribute named "type" to the entity EMPLOYEE.



- But the type attribute would work for some cases and in some cases, it doesn't work as it create problems. For example, an accountant has a CPA(Certified Public Accountant) that no other employee type has. A pilot has a flying qualification and a pilot licence that no other employee has. A mechanic has a qualification of aircraft engineering that no other employee has.
- So, this type of situation can be handled using the concept of sub-type and super type because we can use sub-types to give unique attributes to those types of

employees.



- Each sub-type(child) will automatically inherit all the attributes from the super type(parent) and each sub-type have some unique attributes that a parent and the other sub-type doesn't have.
- So EMPLOYEE-super-type(parernt) and PILOT, ACCOUNTANT, MECHANICS, FLIGHT_ATTENDANTS-sub-types(child)

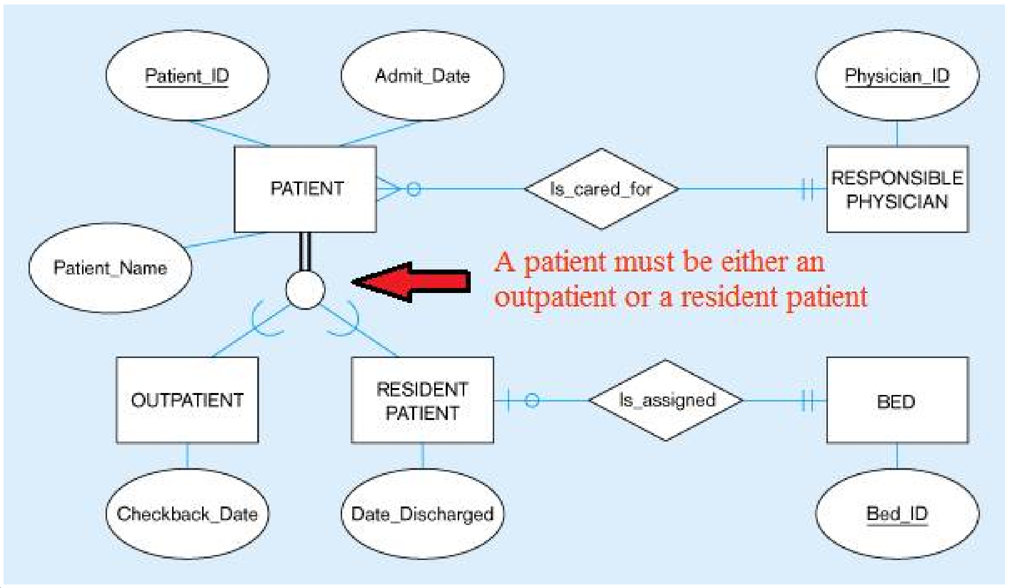
Business Rules for Supertype and subtype Relationships (or) Constraints in Supertype and Subtype Discriminators :

- Total Specialization
- Partial Specialization
- Disjoint Rule
- Overlap Rule

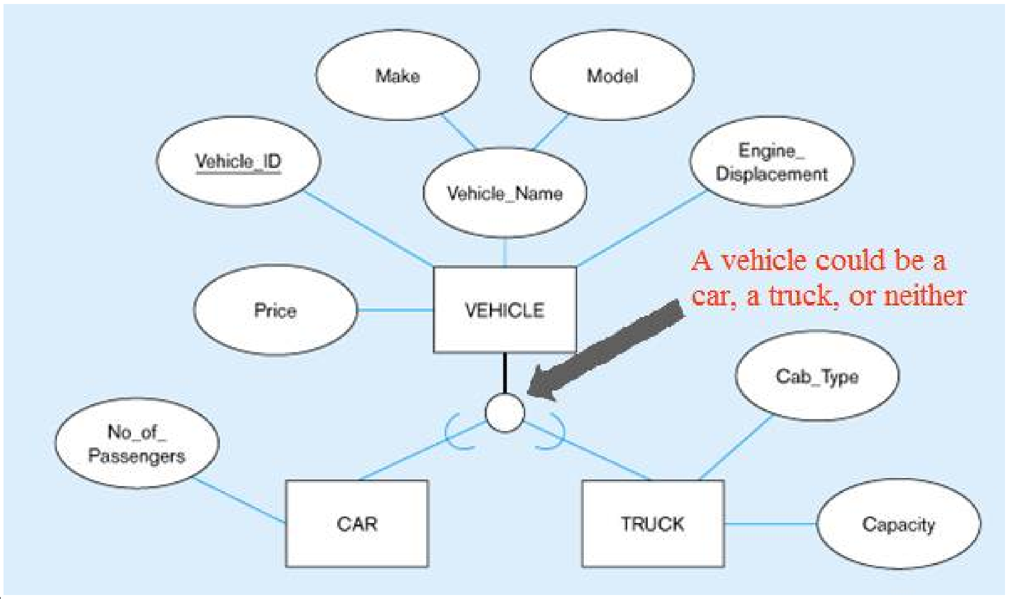
The first two rules i.e. Total Specialization Rule and Partial Specialization Rule **are Completeness Constraints. The Last two rules** i.e. Disjoint Rule and Overlap Rule **are Disjoint Constraints.**

Completeness Constraints :

1) <u>Total specialization</u> specifies that each entity instance of the supertype must be a member of some subtype in the specialization/generalization. Total Specialization is shown in EER Diagrams by a double line.
<u>Example</u> : In following Fig. A PATIENT must either be an OUTPATIENT or a RESIDENT PATIENT.



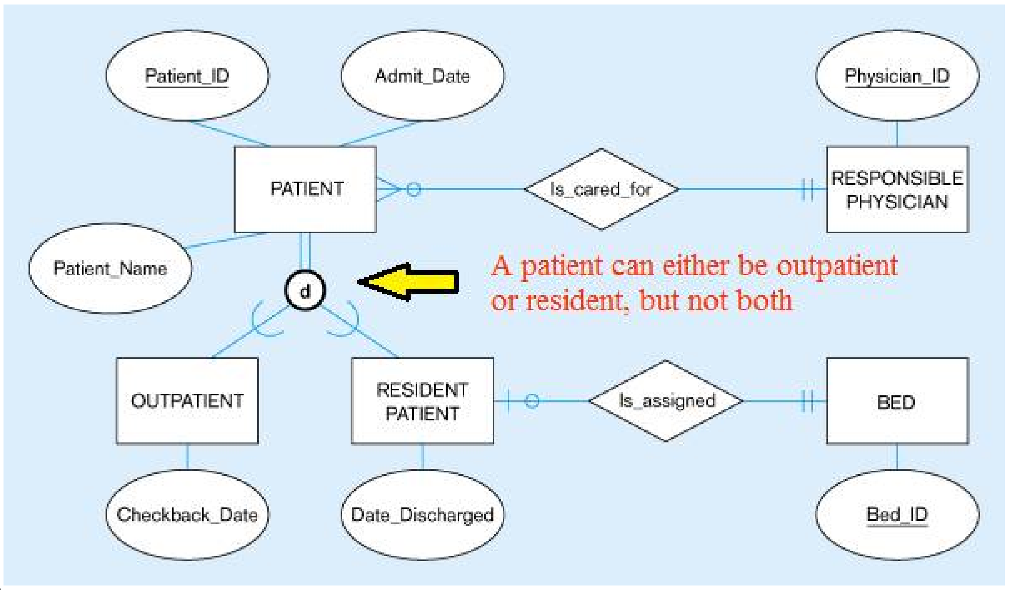
2) <u>Partial specialization</u> specifies that an entity instance of the supertype does not have to belong to any subtype, and may or may not be an instance of one of the subtypes. Partial Specialization is shown in EER Diagrams by a single line.
<u>Example</u> : In the following Fig. If a VEHICLE is a car it will appear as an instance of CAR, and if a truck as an instance of TRUCK. However, if the vehicle is a motorcycle it cannot appear as an instance of any subtype.



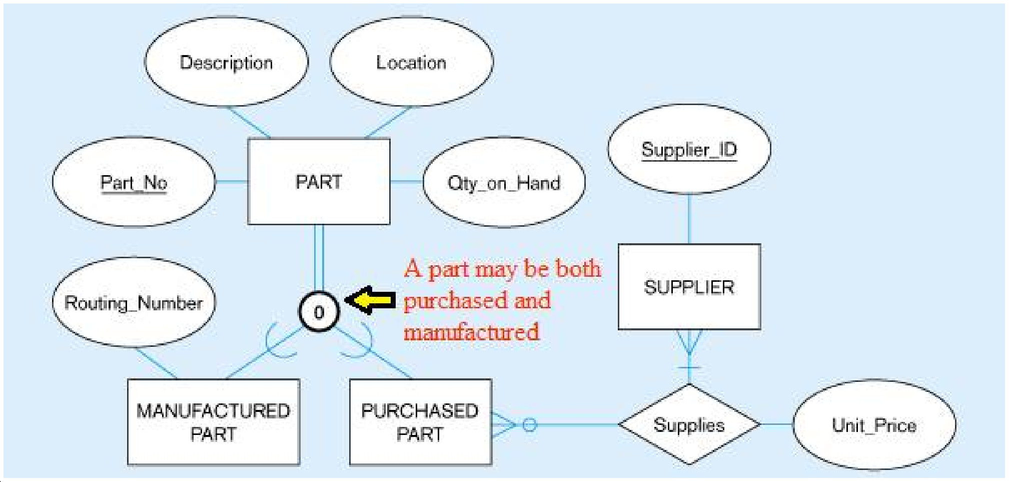
Disjoint Constraints :

3) <u>Disjoint rule</u> specifies that if an entity instance of the supertype is a member of one subtype, it cannot simultaneously be a member of any other subtype. The overlap is specified by placing an 'd' in the circle.

<u>Example</u> : Following Fig. shows that at any one time a PATIENT must be either an outpatient or a resident patient but cannot be both



4) <u>Overlap rule</u> specifies that an entity instance can simultaneously be a member of two (or more) subtypes. The overlap is specified by placing an 'o' in the circle.
<u>Example</u> : A part may be both purchased and manufactured

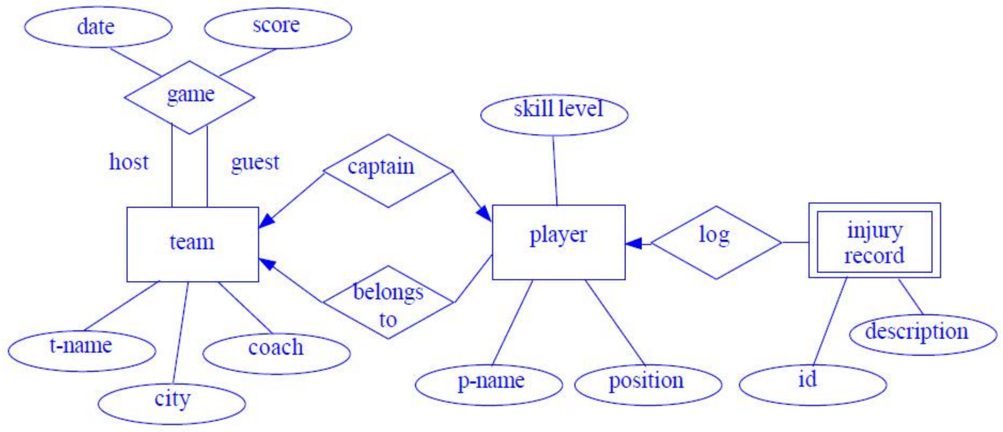


er diagram example

Some er diagram example : Question 1 : Suppose you are given the following requirements for a simple database for the National Hockey League (NHL):

- the NHL has many teams,
- each team has a name, a city, a coach, a captain, and a set of players,
- each player belongs to only one team,
- each player has a name, a position (such as left wing or goalie), a skill level, and a set of injury records,
- a team captain is also a player,
- a game is played between two teams (referred to as host_team and guest_team) and has a date (such as May 11th, 1999) and a score (such as 4 to 2).

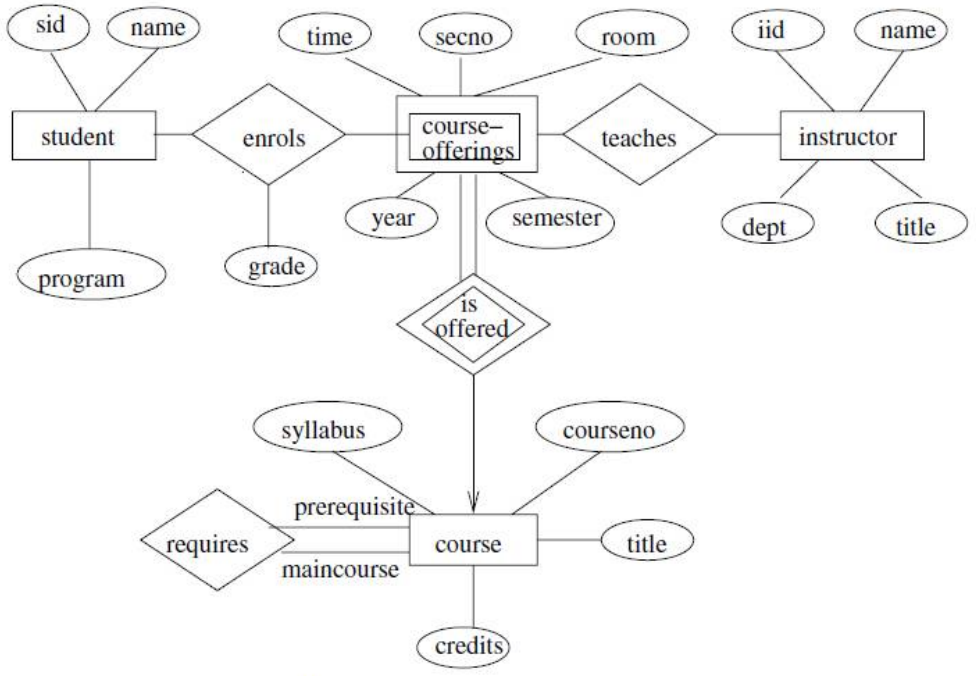
Construct a clean and concise ER diagram for the NHL database. **Answer :**



Question 2: A university registrar's office maintains data about the following entities:

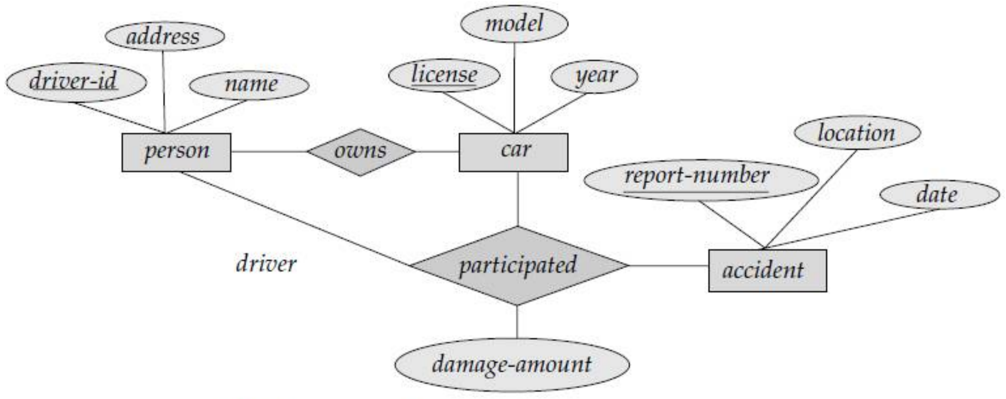
1. courses, including number, title, credits, syllabus, and prerequisites;
2. course offerings, including course number, year, semester, section number, instructor(s), timings, and classroom;
3. students, including student-id, name, and program;
4. instructors, including identi-cation number, name, department, and title.

Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled. Construct an E-R diagram for the registrar's of-ce.Document all assumptions that you make about the mapping constraints. **Answer :**



E-R diagram for a university.

Question 3: (a) Construct an E-R diagram for a car-insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents. **Answer :**



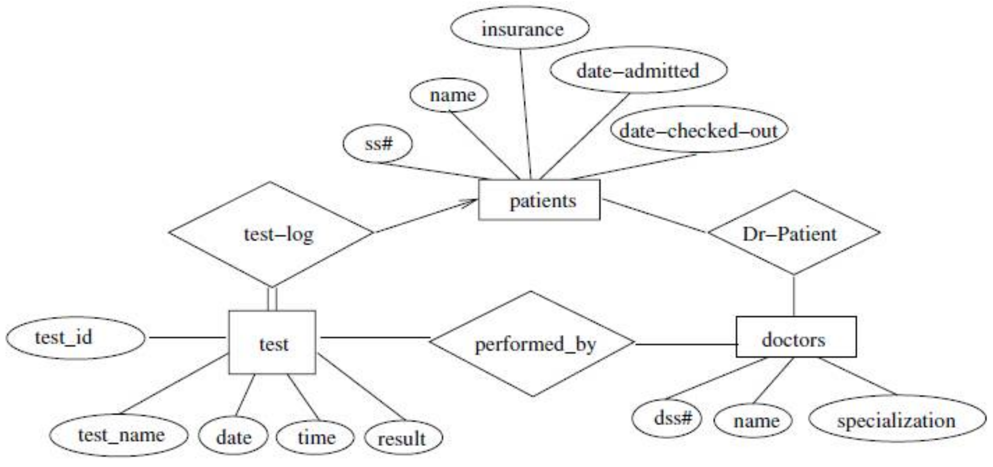
E-R diagram for a Car-insurance company.

(b) Construct appropriate tables for the above ER Diagram ?
Car insurance tables:
person (<u>driver-id</u>, name, address)

car (<u>license</u>, year,model)

accident (<u>report-number</u>, date, location)

participated(<u>driver-id</u>, <u>license</u>, <u>report-number</u>, damage-amount)
Question 4: (a) Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted. **Answer :**



E-R diagram for a hospital.

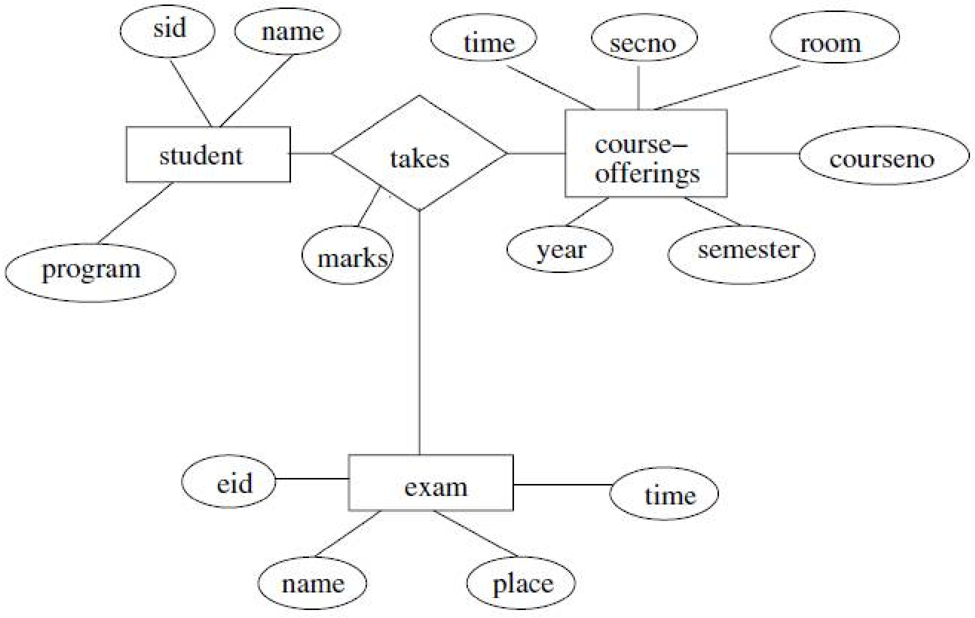
(b) Construct appropriate tables for the above ER Diagram :
Patient(SS#, name, insurance)

Physician (name, specialization)

Test-log(SS#, test-name, date, time)

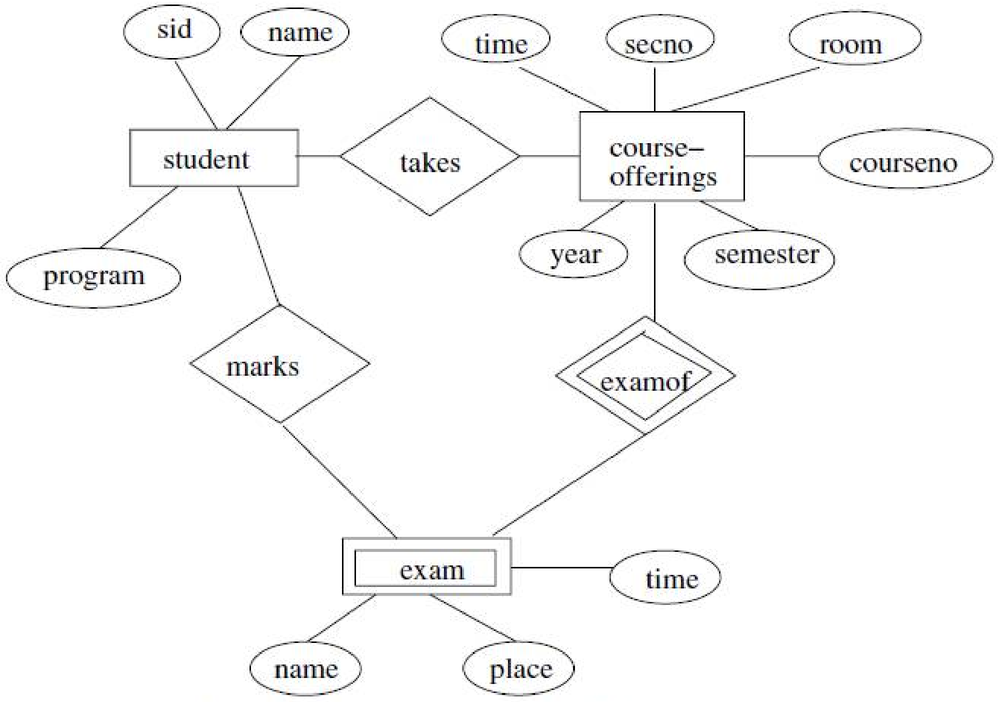
Doctor-patient (physician-name, SS#)

Patient-history(SS#, test-name, date)
Question 5: Consider a database used to record the marks that students get in different exams of different course offerings. **a)** Construct an E-R diagram that models exams as entities, and uses a ternary relationship, for the above database.



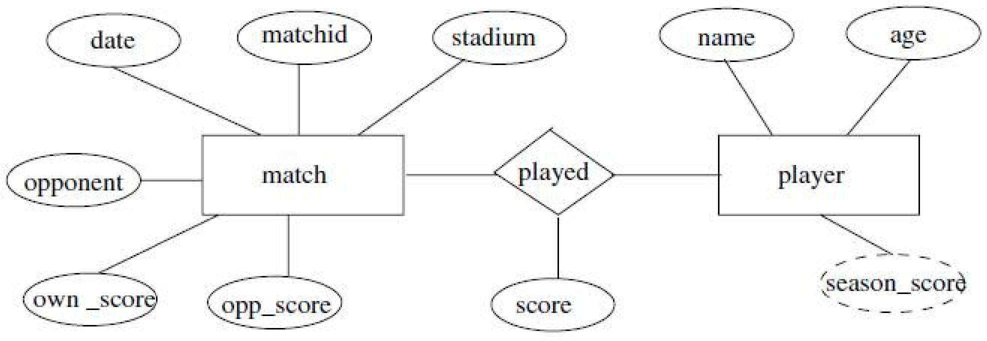
Answer a : E-R diagram for marks database.

b) Construct an alternative E-R diagram that uses only a binary relationship between students and course-offerings. Make sure that only one relationship exists between a particular student and course-offering pair, yet you can represent the marks that a student gets in different exams of a course offering.



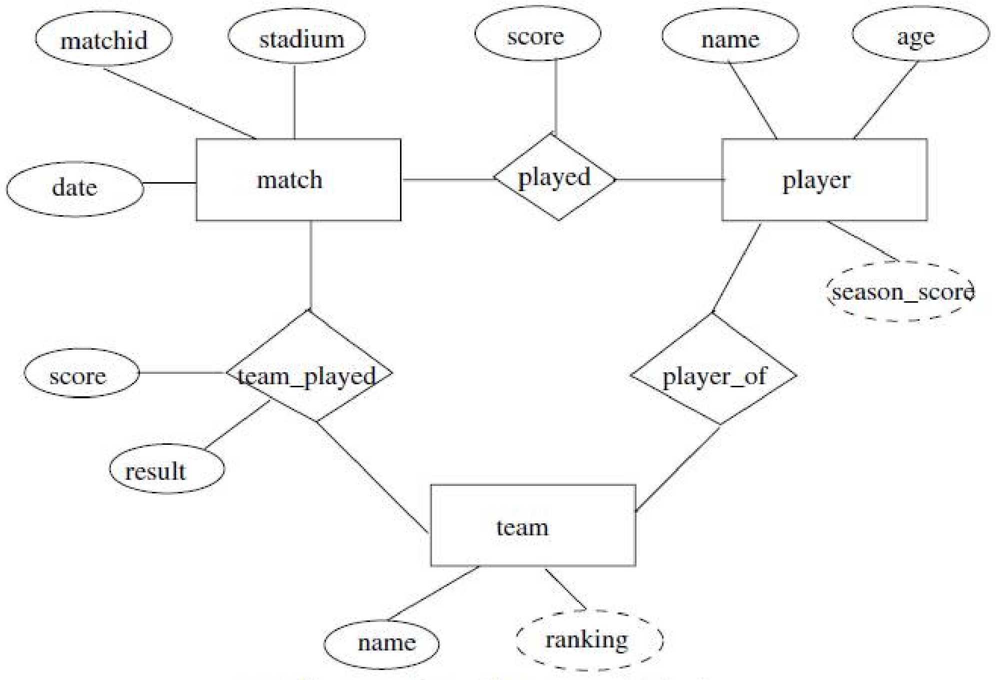
Answer b: Another E-R diagram for marks database.

Question 6: Design an E-R diagram for keeping track of the exploits of your favorite sports team. You should store the matches played, the scores in each match, the players in each match and individual player statistics for each match. Summary statistics should be modeled as derived attributes. **Answer :**



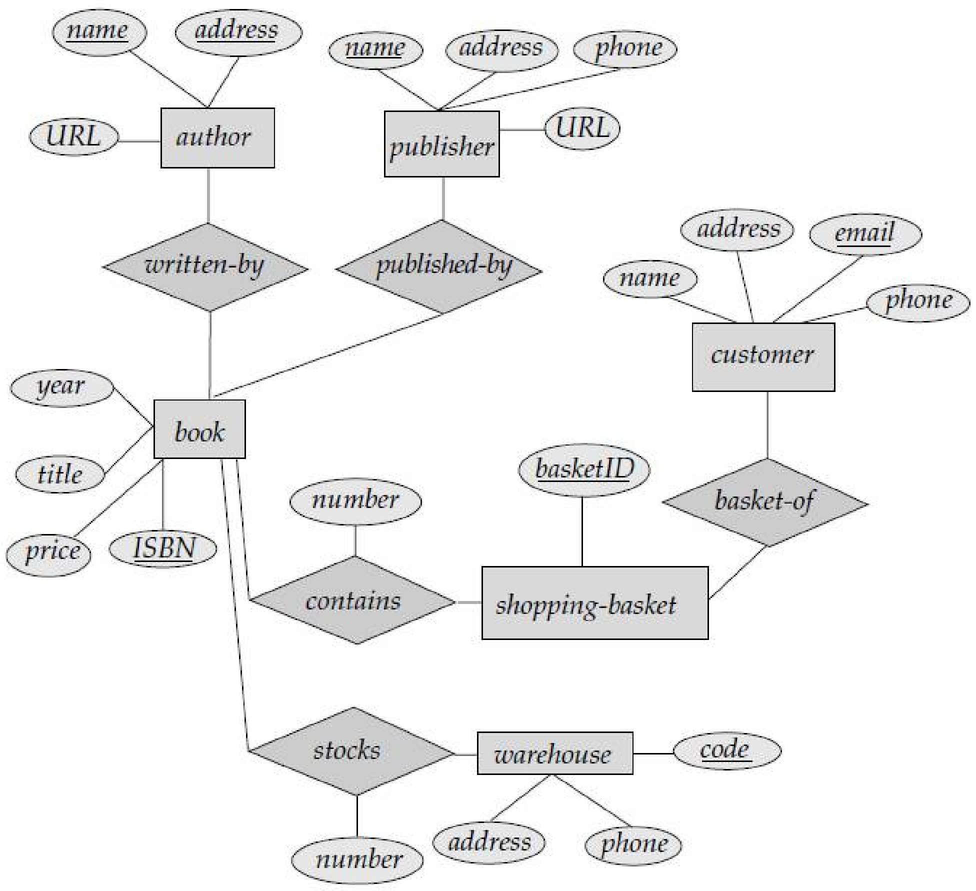
E-R diagram for favourite team statistics.

Question 7 : Extend the E-R diagram of the previous question to track the same information for all teams in a league. **Answer :**



E-R diagram for all teams statistics.

Question 8 : Draw the E-R diagram which models an online bookstore.



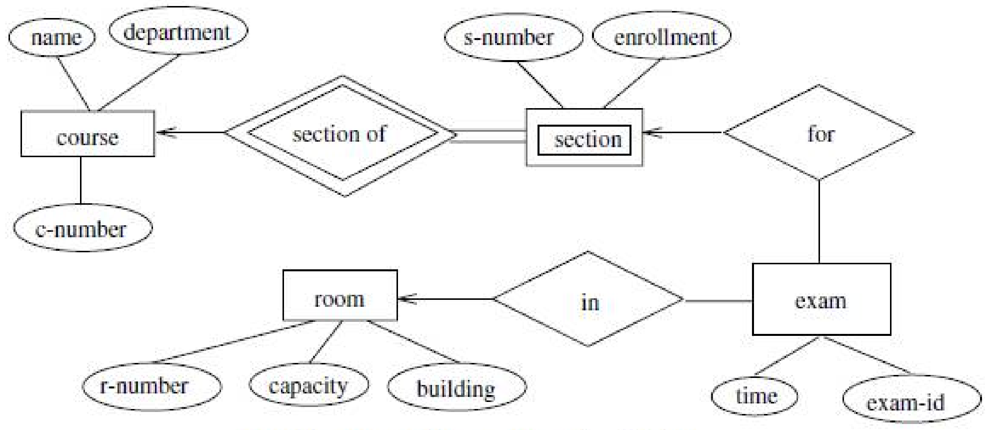
Answer :

ER Diagram for Online BookStore

Question 9 : Consider a university database for the scheduling of classrooms for -final exams. This database could be modeled as the single entity set exam, with attributes course-name, section-number, room-number, and time. Alternatively, one or more additional entity sets could be defined, along with relationship sets to replace some of the attributes of the exam entity set, as

- course with attributes name, department, and c-number
- section with attributes s-number and enrollment, and dependent as a weak entity set on course
- room with attributes r-number, capacity, and building

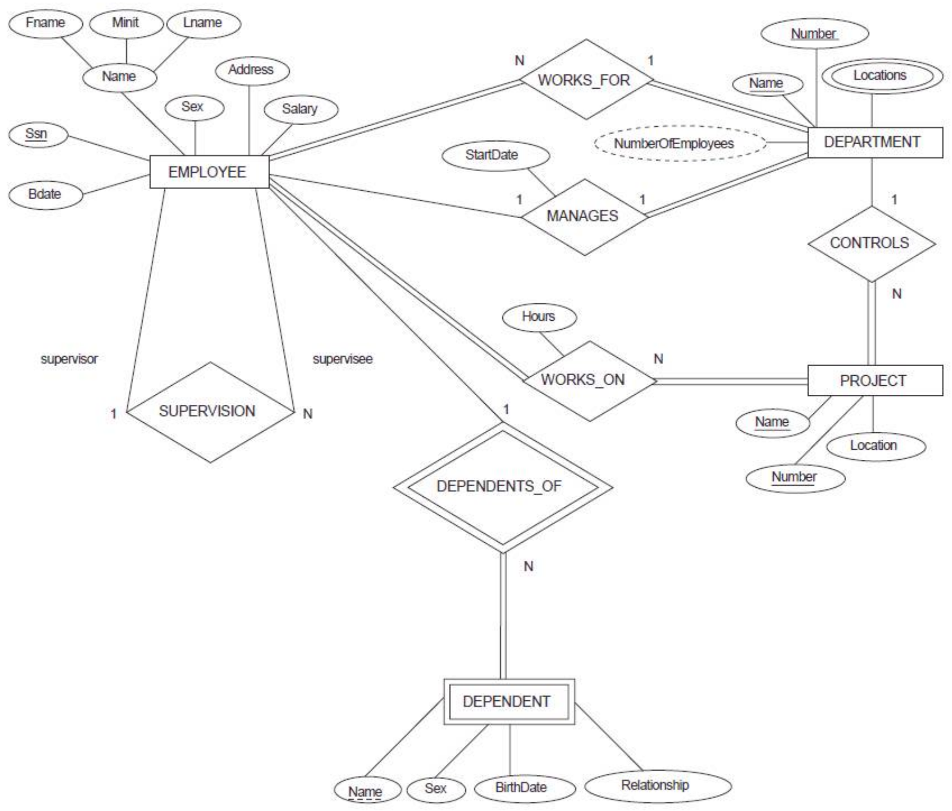
Show an E-R diagram illustrating the use of all three additional entity sets listed. **Answer :**



E-R diagram for exam scheduling.

Question 10 : Construct an ER Diagram for Company having following details :

- Company organized into DEPARTMENT. Each department has unique name and a particular employee who manages the department. Start date for the manager is recorded. Department may have several locations.
- A department controls a number of PROJECT. Projects have a unique name, number and a single location.
- Company's EMPLOYEE name, ssno, address, salary, sex and birth date are recorded. An employee is assigned to one department, but may work for several projects (not necessarily controlled by her dept). Number of hours/week an employee works on each project is recorded; The immediate supervisor for the employee.
- Employee's DEPENDENT are tracked for health insurance purposes (dependent name, birthdate, relationship to employee).



Answer :

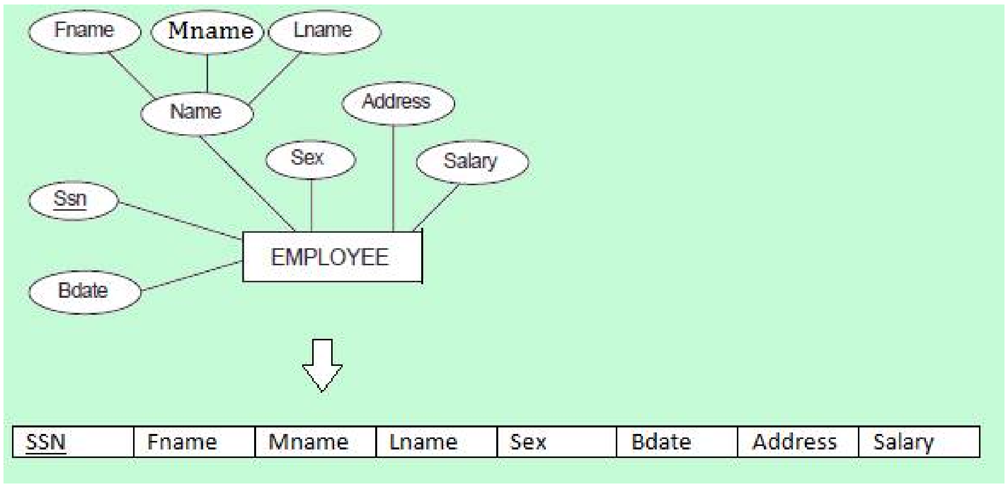
Conversion of er model to relational model - Minimisation of ER Diagram - (7-STEPS)

To convert the er model to relational model there are 7 Steps to be followed, which are

1. **Conversion of Strong Entities -**
2. **Conversion of Weak Entities**
3. **Conversion of one to one Relationships**
4. **Conversion of One to Many Relationships**
5. **Conversion of Many to Many Relationships**
6. **Conversion of n-ary Relationships**
7. **Conversion of Multivalued Attribute**

1. Conversion of Strong Entities -

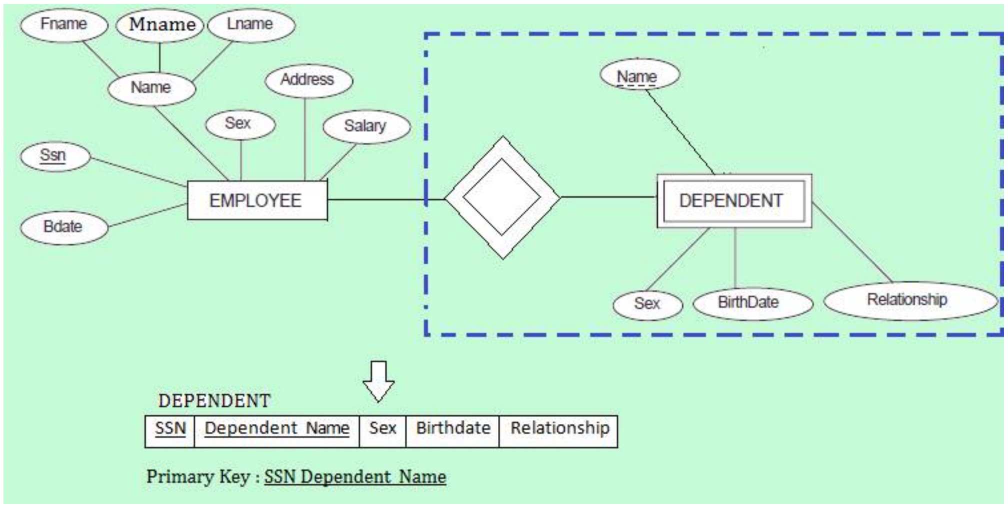
- For each strong entity in ERD, create a separate table with the same name.
- Create all simple Attributes
- Break the Composite attributes into simple attributes and create them.
- Choose a Primary Key for the table.



<u>SSN</u>	Fname	Mname	Lname	Sex	Bdate	Address	Salary

2. Conversion of Weak Entities -

- For each weak entity, create a separate table with the same name.
- Include Primary Key of the strong entity as a foreign key in the table.
- Select the Primary Key attributes of strong entity and the partial Key attribute of the weak entity, and declare them as primary key



DEPENDENT

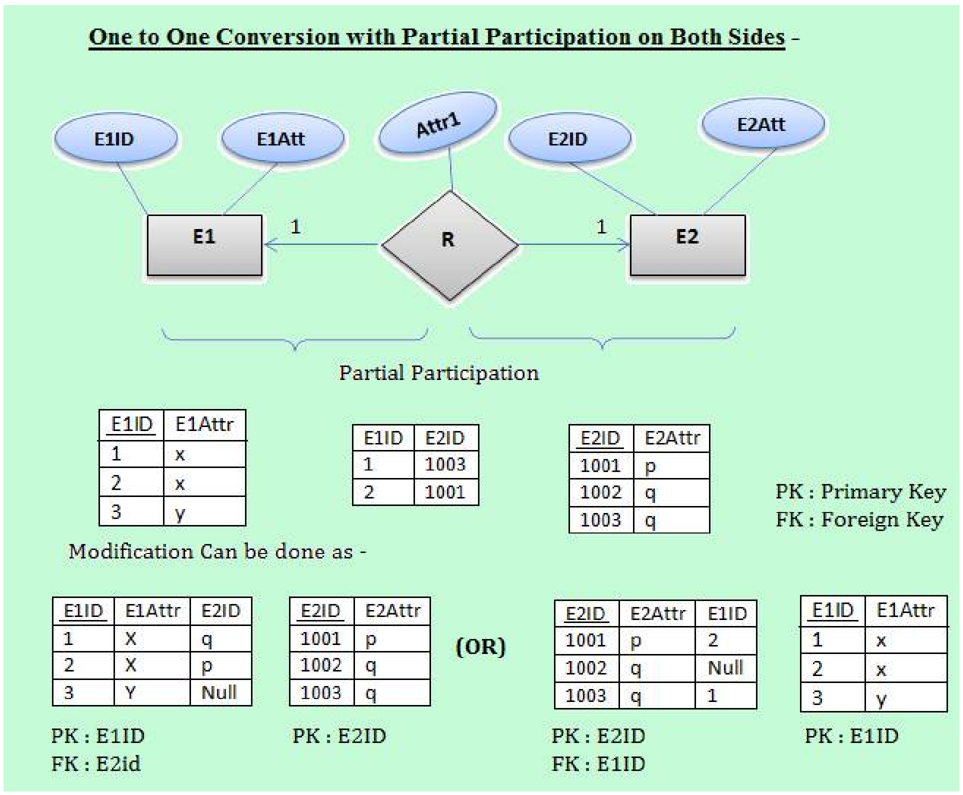
<u>SSN</u>	<u>Dependent Name</u>	Sex	Birthdate	Relationship

Primary Key : <u>SSN Dependent Name</u>

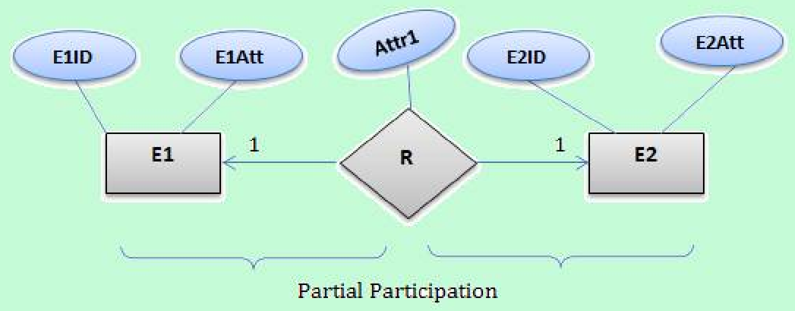
3. Conversion of One to One Relationships -

There are two possible approaches on the basis of Participation Constraints -

- Partial Participation on Both Sides - For each One to One Cardinality between E1 and E2 with partial participation on both sides, modify either E1 or E2 to include the primary key of other table as a foreign key. So, 1:1 cardinality with partial participation on both sides can be minimised into two relations only.



One to One Conversion with Partial Participation on Both Sides -



E1ID	E1Attr
1	x
2	x
3	y

E1ID	E2ID
1	1003
2	1001

E2ID	E2Attr
1001	p
1002	q
1003	q

PK : Primary Key
FK : Foreign Key

Modification Can be done as -

E1ID	E1Attr	E2ID
1	X	q
2	X	p
3	Y	Null

PK : E1ID
FK : E2id

E2ID	E2Attr
1001	p
1002	q
1003	q

PK : E2ID

(OR)

E2ID	E2Attr	E1ID
1001	p	2
1002	q	Null
1003	q	1

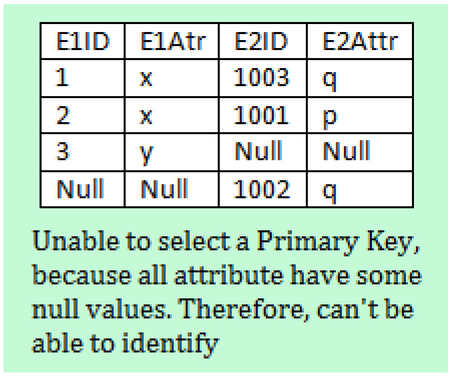
PK : E2ID
FK : E1ID

E1ID	E1Attr
1	x
2	x
3	y

PK : E1ID

- If we try to minimize the above ERD in a single table, i.e. E1RE2, then it contains too many NULL values, and therefore, we are not be able to select a primary key.

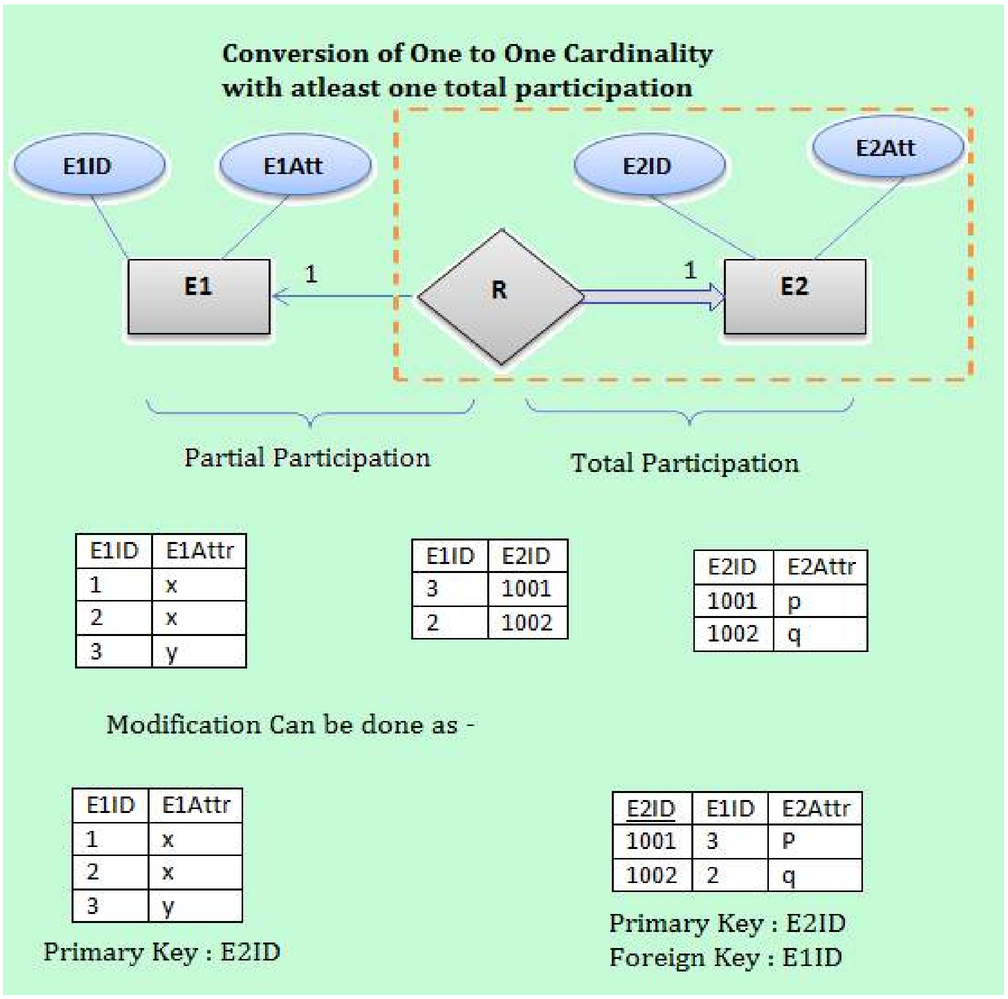
For example,



E1ID	E1Atr	E2ID	E2Attr
1	x	1003	q
2	x	1001	p
3	y	Null	Null
Null	Null	1002	q

Unable to select a Primary Key, because all attribute have some null values. Therefore, can't be able to identify

- Cardinality with atleast one Total Participation - For each One to One Cardinality between E1 and E2 with atleast one total participation, modification is done only on total participation side. So, One to One Cardinality with atleast one Total Participation can be minimized into a single relation.


Conversion of One to One Cardinality
with atleast one total participation
E1ID
E1Att
E2ID
E2Att
E1
R
E2
1
1
Partial Participation
Total Participation
Modification Can be done as -

E1ID	E1Attr
1	x
2	x
3	y

E1ID	E2ID
3	1001
2	1002

E2ID	E2Attr
1001	p
1002	q

Modification Can be done as -

E1ID	E1Attr
1	x
2	x
3	y

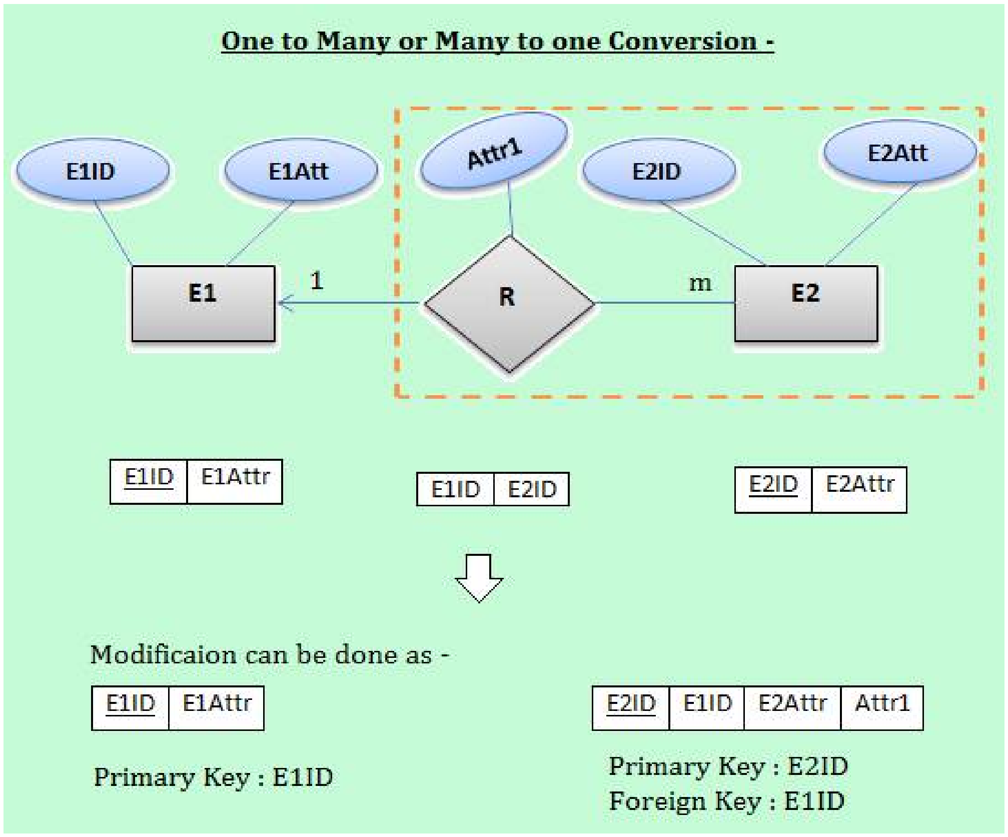
Primary Key : E2ID

E2ID	E1ID	E2Attr
1001	3	P
1002	2	q

Primary Key : E2ID
Foreign Key : E1ID

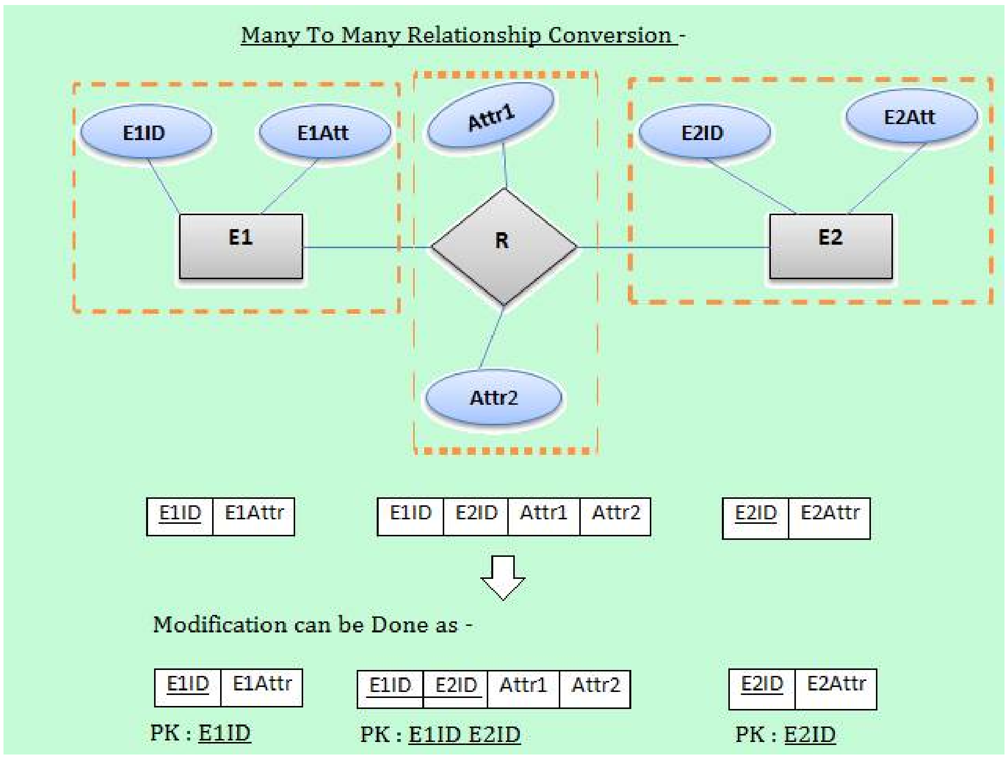
4. Conversion of One to Many or Many to One Relationship -

- For each one to many relationship between E1 and E2, modify many side relation to include from one side as a Foreign Key.

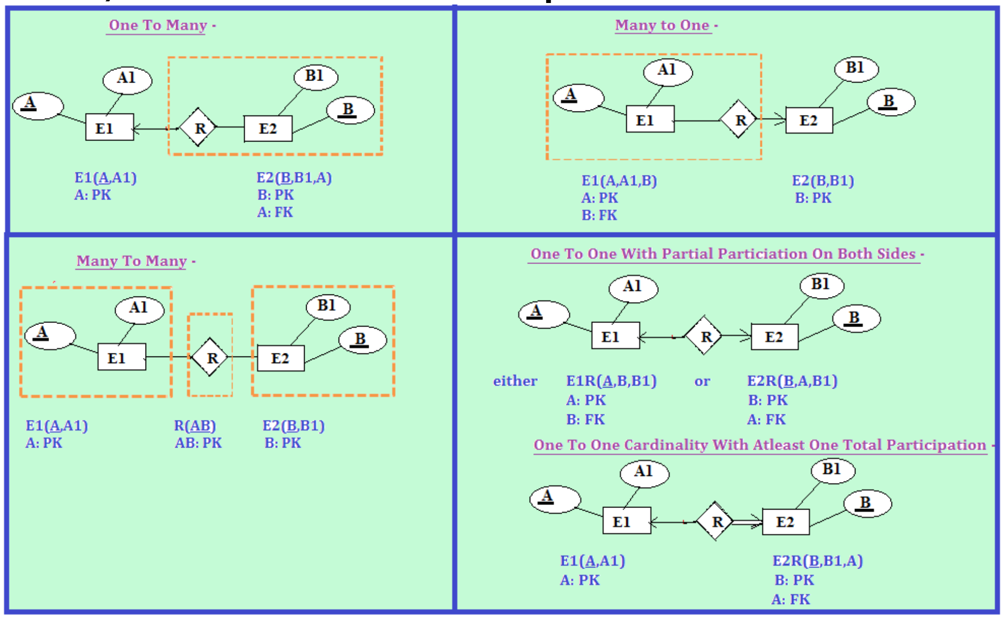


5. Conversion of Many to Many Relationship -

- For each one to many relationship between E1 and E2, create a separate table and include primary key of both the tables as a Foreign Key.
- If relationship is having one or more attributes, then these must also be included in the table.

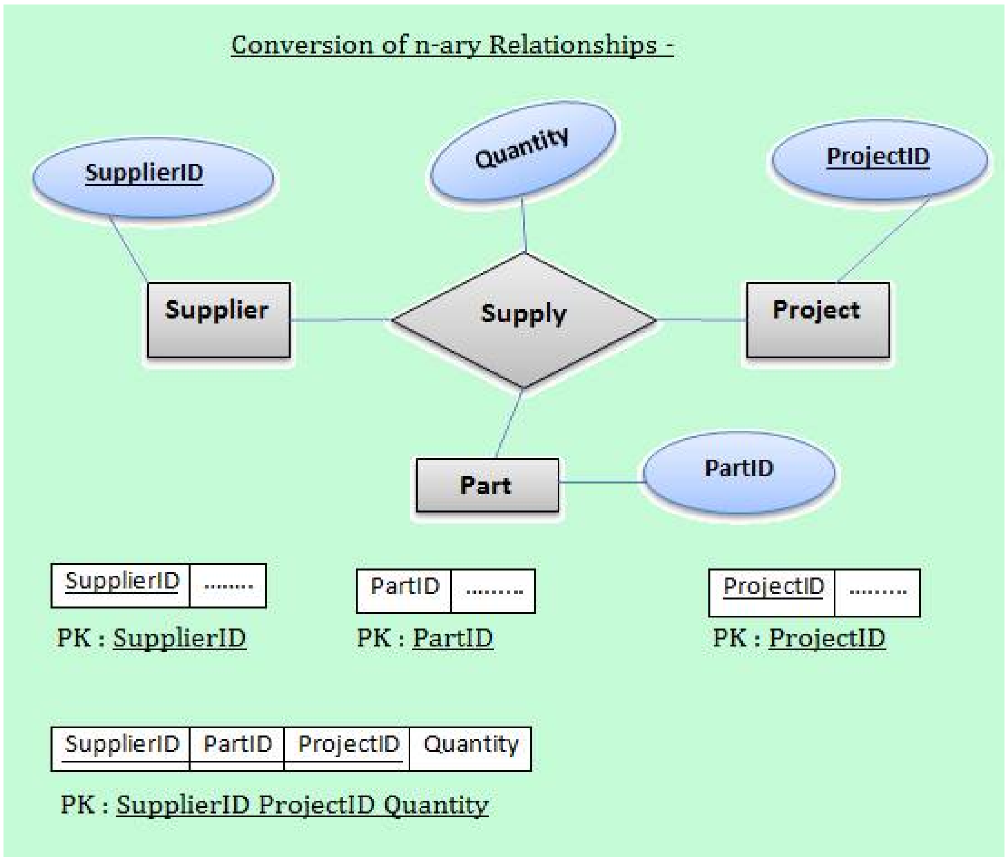

Many To Many Relationship Conversion -
E1ID
E1Att
Attr1
E2ID
E2Att
E1
R
E2
Attr2
E1ID | E1Attr
E1ID | E2ID | Attr1 | Attr2
E2ID | E2Attr
Modification can be Done as -
E1ID | E1Attr
E1ID | E2ID | Attr1 | Attr2
E2ID | E2Attr
PK : E1ID
PK : E1ID E2ID
PK : E2ID

In short, The Conversion of Relationships will be done as -



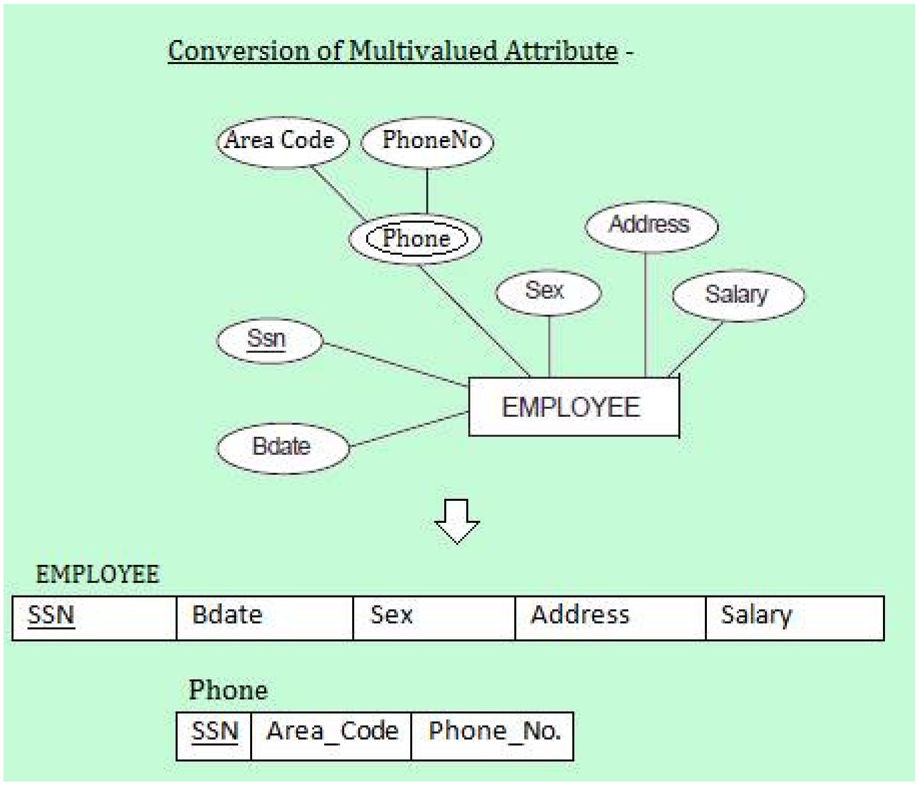
6. Conversion of n-ary Relationship -

- For each n-ary relationship, Create a separate table and include primary keys of all other entities as a foreign key.
- If the relationships has some attributes, then these must also be included in the table.



7. Conversion of multivalued Attributes -

- For each multivalued attributes, create a separate table, then include all of its simple attributes.
- Include the primary key of the original table as a foreign key.



EMPLOYEE

SSN	Bdate	Sex	Address	Salary

Phone

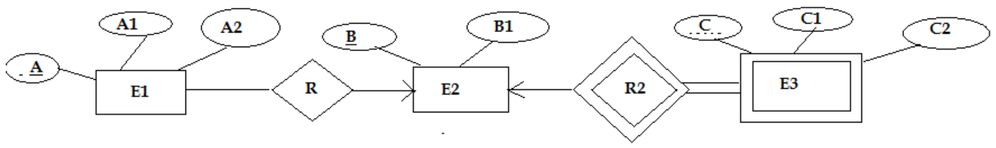
SSN	Area_Code	Phone_No.

All the steps for conversion of er model to relational model defined above, are necessary steps. If you follow all the steps for the conversion, then you will definitely convert er model to relational model. check out in the below link for some questions on er model to relational model.

Conversion of ER Model to Relational Model-Questions

Question 1 :

Consider the following ER Diagram -



a) How many minimum number of tables required ?

Solution :

Step 1 : Conversion of Strong Entities :

E1		
A	A1	A2
Primary Key : A		

E2	
B	B1
Primary Key : B	

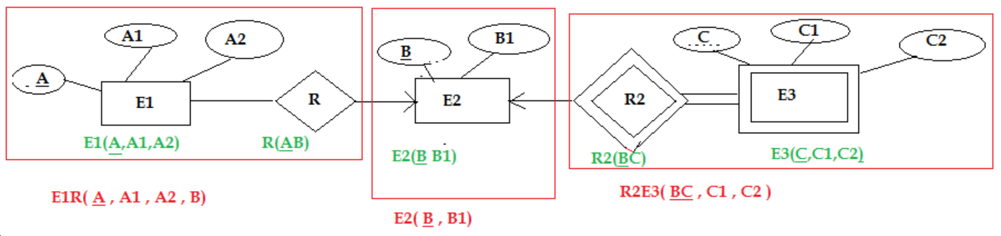
Step 2 : Conversion of Weak Entity :

E3			
B	**C**	C1	C2
Primary Key : BC			

Step 3 : Conversion of Many to One relationship -

E1			
A	A1	A2	B
Primary Key : A Foreign Key : B			

So, the minimum number of tables required for the ER Diagram will be as follows -



= = > 3 tables

b) Which is the common attribute set of each of minimasation table?

Solution :

Attribute B is common in all the tables.

Question 2 :

Movie(<u>id</u>, title, yr) & Actor(<u>id</u>, name, age)

can we join these tables if there is no foreign key given ?

Solution : NO

Question 3 :

E(<u>B</u>,E,F) & R(<u>A</u>,C,D) where,

B: Primary Key of Table E.

A: Primary key for table R and foreign key of table E.

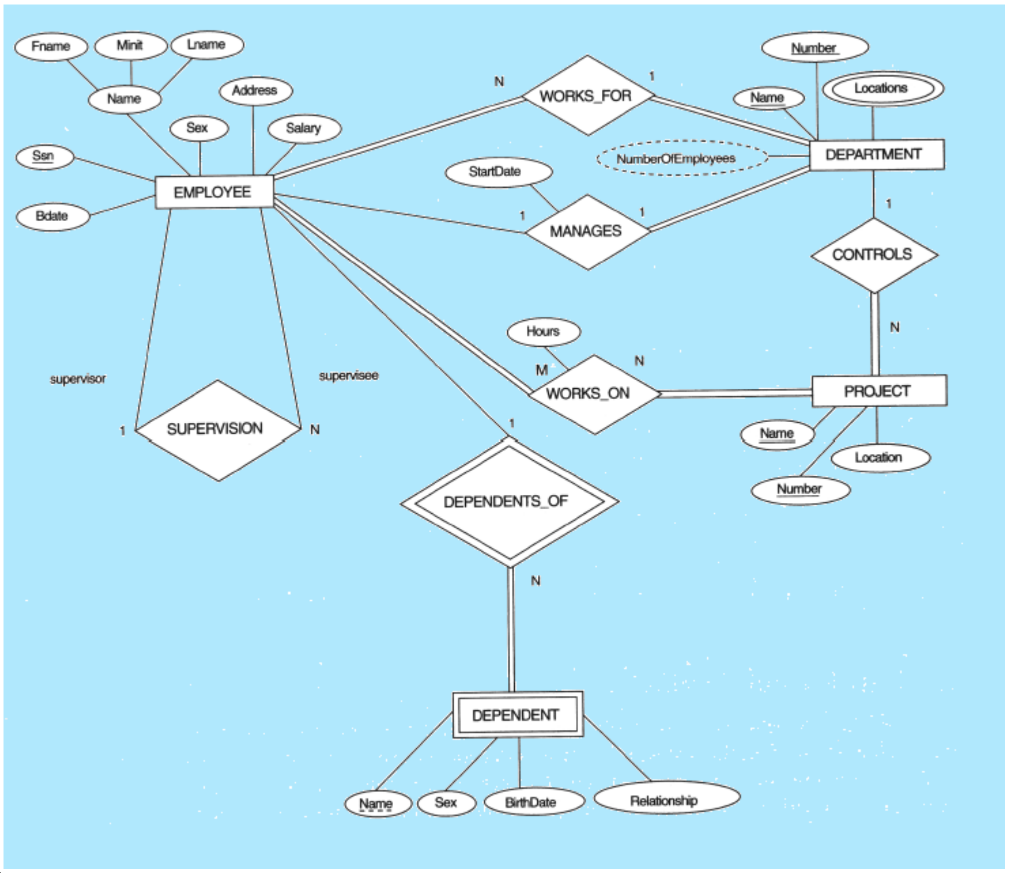
Is it possible to minimize?

Solution :

Yes, ER(B,E,F,C,D) . It is possible to minimize.

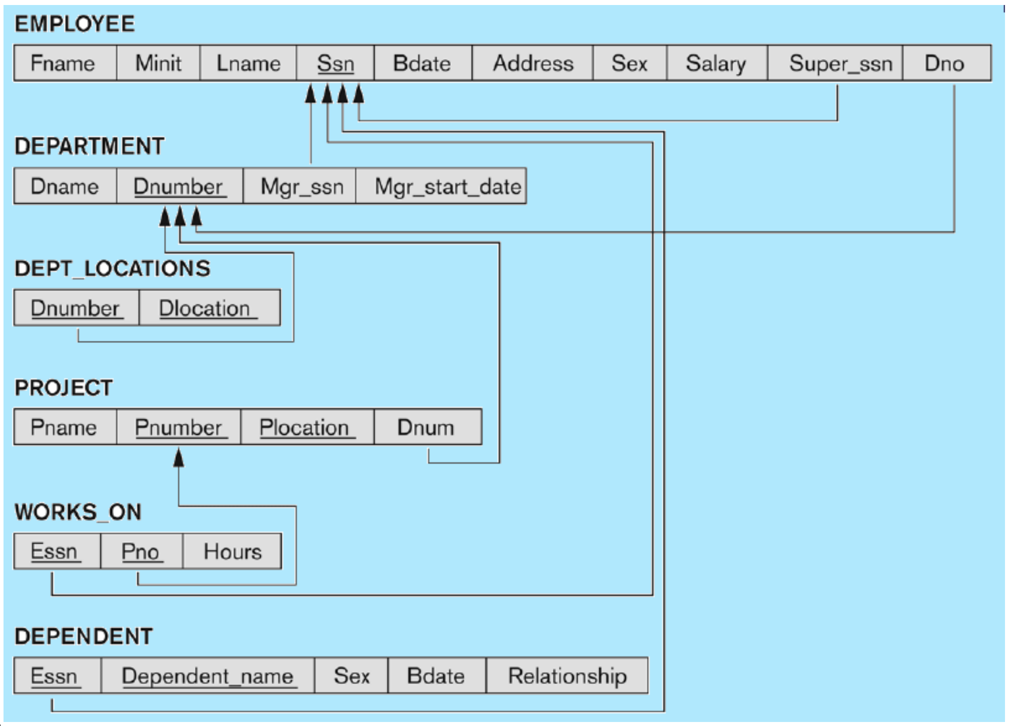
Question 4 :

Consider the following ER Diagram and minimize it into relations.



Solution :

Minimize the ER Diagram into Relations -


EMPLOYEE
Fname | Minit | Lname | Ssn | Bdate | Address | Sex | Salary | Super_ssn | Dno
DEPARTMENT
Dname | Dnumber | Mgr_ssn | Mgr_start_date
DEPT_LOCATIONS
Dnumber | Dlocation
PROJECT
Pname | Pnumber | Plocation | Dnum
WORKS_ON
Essn | Pno | Hours
DEPENDENT
Essn | Dependent_name | Sex | Bdate | Relationship

ER Diagram Examples

Some ER Diagram Examples :

Question 1 :

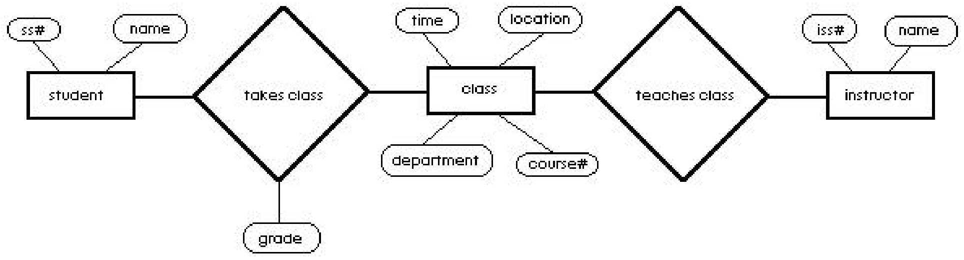
(a) Construct an E-R diagram for the registrar's office. Document all assumptions you make about the mapping constraints.

Assumptions:

- A class meets only at one particular place and time. This diagram does not attempt to model a class meeting at different places or at different times
- There is no guarantee that the database does not have two classes meeting at the same place and time
- Each class has a unique instructor

(b) Construct appropriate tables for the ER Diagram

Solution (a):



-

Solution (b) :

Student(<u>SS#</u>, name)
Class(<u>Course#</u>, <u>time</u>, <u>location</u>, department)
Instructor(<u>Iss#</u>, name)
Teaches-class(<u>Iss#</u>, <u>Course#</u>, <u>time</u>, <u>location</u>)
Takes-class (<u>SS#</u>, <u>Course#</u>, <u>time</u>, <u>location</u>, grade)

Question 2 :

(a) A university registrar's office maintains data about the following entities:

- (a) courses, including number, title, credits, syllabus, and prerequisites;
- (b) course offerings, including course number, year, semester, section number, instructor(s), timings, and classroom;
- (c) students, including student-id, name, and program; and
- (d) instructors, including identif-cation number, name, department, and title.

Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled. Construct an E-R diagram for the registrar's of-ce.Document all assumptions
that you make about the mapping constraints.

(b) Construct appropriate tables for the above ER Diagram :
Answer (a):
In the answer given here, the main entity sets are student, course, course-offering, and instructor. The entity set course-offering is a weak entity set dependent on course. The assumptions made are :
a. a class meets only at one particular place and time. This E-R diagram cannot
model a class meeting at different places at different times.
b. There is no guarantee that the database does not have two classes meeting
at the same place and time.

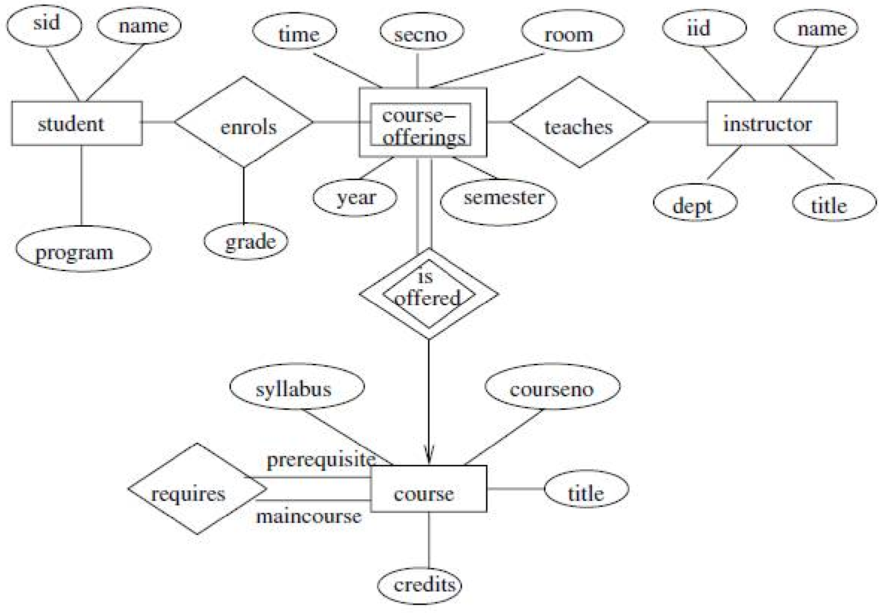


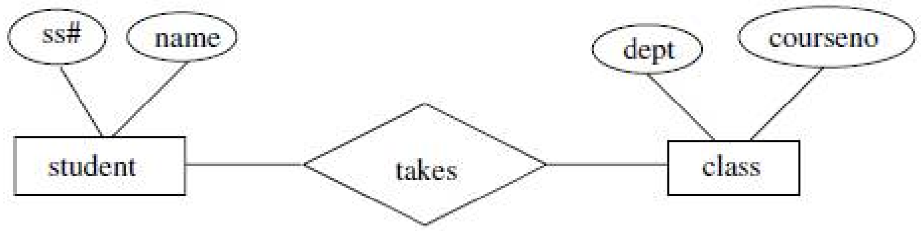
Figure 2.3 E-R diagram for a university.

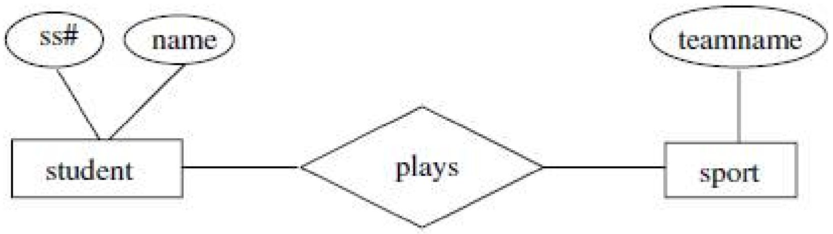
-

Solution (b):

University registrar's tables:
student (<u>student-id</u>, name, program)
course (<u>courseno</u>, title, syllabus, credits)
course-offering (<u>courseno</u>, <u>secno</u>, <u>year</u>, <u>semester</u>, time, room)
instructor (<u>instructor-id</u>, name, dept, title)
enrols (<u>student-id</u>, <u>courseno</u>, <u>secno</u>, <u>semester</u>, <u>year</u>, grade)
teaches (<u>courseno</u>, <u>secno</u>, <u>semester</u>, <u>year</u>, <u>instructor-id</u>)
requires (<u>maincourse</u>, <u>prerequisite</u>)

Question 3 : Consider an E-R diagram in which the same entity set appears several times. Why is allowing this redundancy a bad practice that one should avoid whenever possible?
Answer: By using one entity set many times we are missing relationships in the model. For example, in the following E-R diagram, the students taking classes are the same students who are athletes, but this model will not show that.





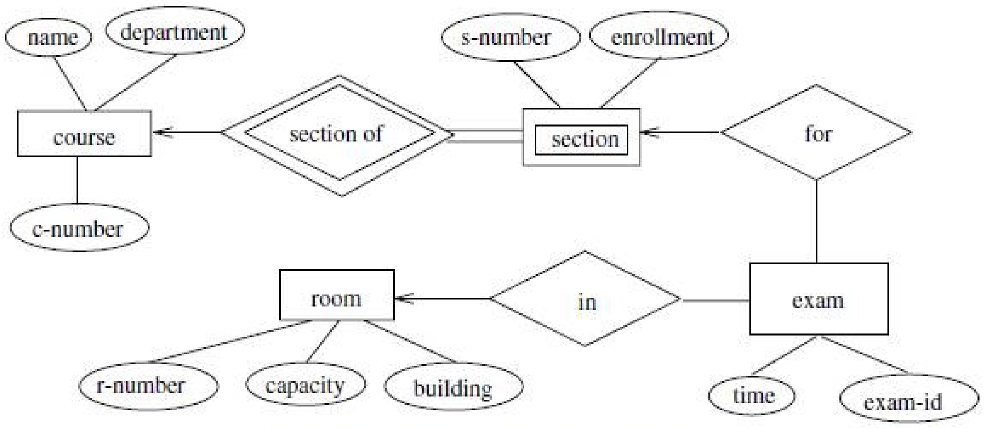
Question 4 :
(a)Consider a university database for the scheduling of classrooms for -final exams. This database could be modeled as the single entity set exam, with attributes course-name, section-number, room-number, and time. Alternatively, one or more additional entity sets could be defined, along with relationship sets to replace some of the attributes of the exam entity set, as

- course with attributes name, department, and c-number
- section with attributes s-number and enrollment, and dependent as a weak entity set on course
- room with attributes r-number, capacity, and building

Show an E-R diagram illustrating the use of all three additional entity sets listed.

(b) Explain what application characteristics would inuence a decision to include or not to include each of the additional entity sets.

Answer (a):



E-R diagram for exam scheduling.

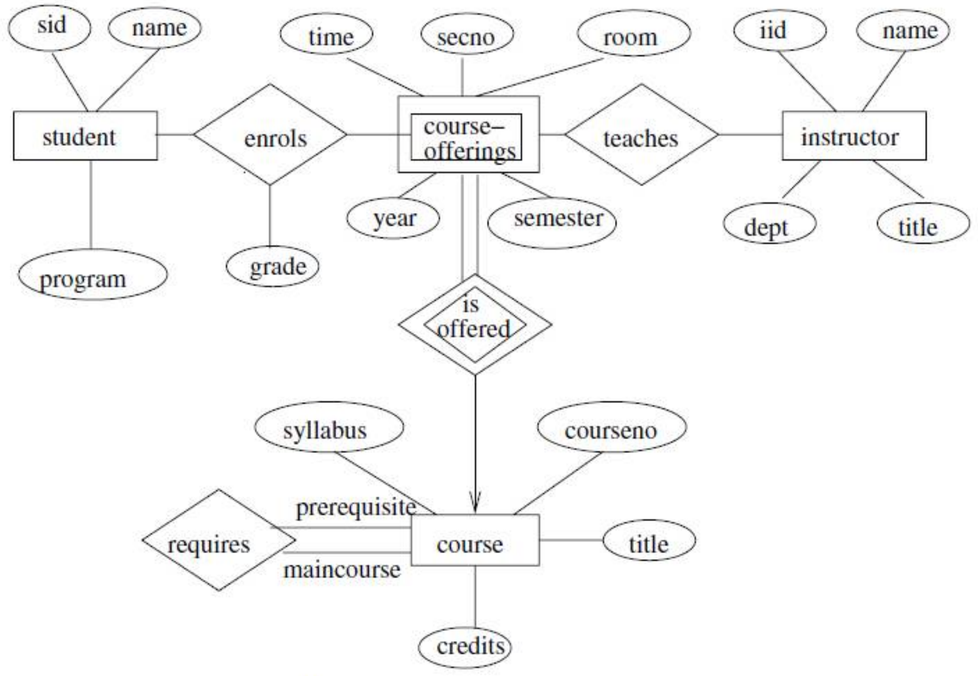
Answer (b) :
The additional entity sets are useful if we wish to store their attributes as part of the database. For the course entity set, we have chosen to include three attributes. If only the primary key (c-number) were included, and if courses have only one section, then it would be appropriate to replace the course (and section) entity sets by an attribute (c-number) of exam. The reason it is undesirable to have multiple attributes of course as attributes of exam is that it would then be dif-cult to maintain data on the courses, particularly if a course has no exam or several exams. Similar remarks apply to the room entity set.

Question 5: A university registrar's office maintains data about the following entities:

1. courses, including number, title, credits, syllabus, and prerequisites;
2. course offerings, including course number, year, semester, section number, instructor(s), timings, and classroom;
3. students, including student-id, name, and program;
4. instructors, including identi-cation number, name, department, and title.

Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled. Construct an E-R diagram for the registrar's of-ce.Document all assumptions that you make about the mapping constraints.

Answer :



E-R diagram for a university.