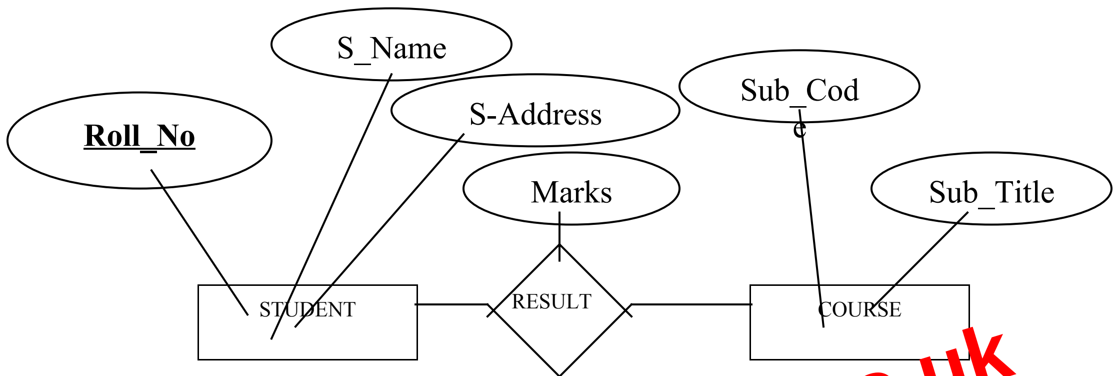
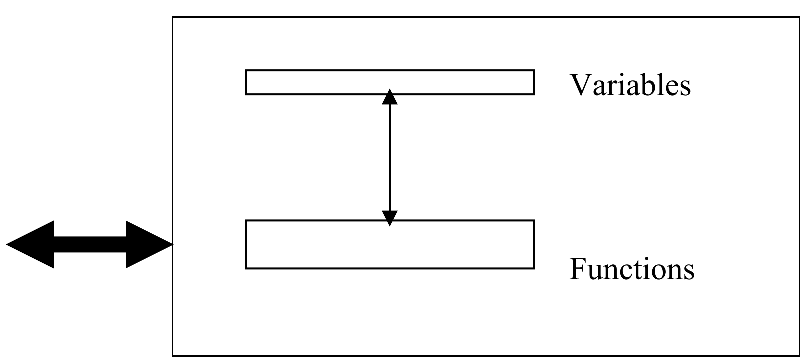
- E-R Model also specifies certain constraints, like Mapping Cardinalities i.e. whether the relationship is one-to-one, one-to-many, many-to-one or many-to-many.

- The E-R Diagram below depicts two Entity Sets "STUDENT", "COURSE" and a relationship set "RESULT" indicating the marks obtained by students in different Courses.



(ii) **Object-Oriented Model:** Like the E-R Model, this model also models a database as a Collection of Objects. An Object Body encapsulates Data (Variables) as well as Methods (Functions) to manipulate the Data (Variables). The Objects that contain same Type of Data Variables and same Type of Functions are grouped together as a Class. Thus, a Class may be viewed as a Type Definition of the Objects. The only way an Object "A" can access the Data Items of another Object "B" is by invoking the Methods of "B". "A" can accomplish this by making calls to the methods of "B", through B's Interface. The methods defined within an object are made visible to the external world, through its Interface.



(giving details of all students), TEACHER (giving details of all teachers), COURSE (giving details of all courses); and three tables to represent relationships i.e. COURSE-TEACHER (indicating relationships – OFFERED BY and OFFERS), COURSE-STUDENT (indicating relationships ATTENDS and ATTENDED BY) and TEACHER-STUDENT (indicating relationships TAUGHT BY and TEACHES).

STUDENT

Roll_No	S_Name	Branch	Semester	Section	S_Address

COURSE

Sub_Code	Sub_Title	Semester	Branch	Contact_Hrs

TEACHER

Fac_Code	Fac_Name	Desig	Dept	Fac_Address

COURSE-TEACHER

Sub_Code	Fac_Code

COURSE-STUDENT

Sub_Code	Roll_No

TEACHER-STUDENT

Fac_Code	Roll_No

The Relational Model has become extremely popular because:-

(a) It is extremely simple and easy to implement.
(b) It has a strong mathematical foundation.
(c) It has been highly standardized.

SCHEMAS AND INSTANCES

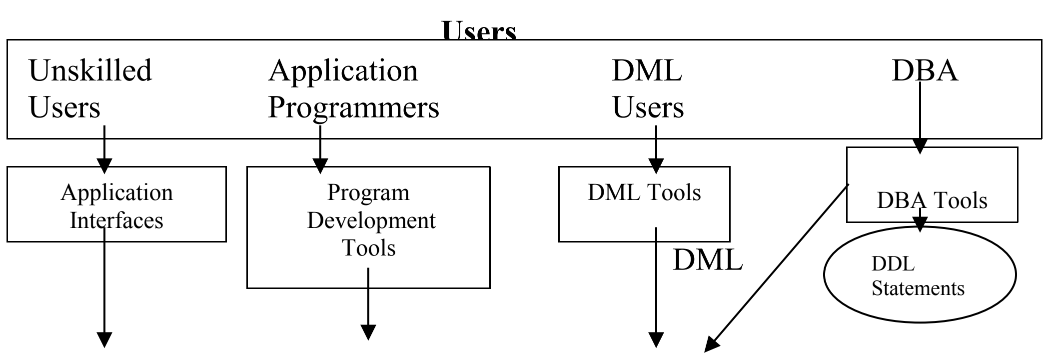
There are two types of DMLs:-

(i) **<u>Procedural DMLs.</u>** A query in procedural DML requires the user to specify not only "what data is required to be extracted from the database" but also to specify "how to extract those data".

(ii) **<u>Non-Procedural DMLs.</u>** A Query in Non-Procedural DML requires the user to specify **only** "what data is needed", **without** specifying how to get those data.

Non-procedural DMLs are easier to learn and to use than the procedural DMLs. However, since non-Procedural DMLs do not specify "how to get the data", the queries in Non-Procedural DMLs may not generate as efficient code as the equivalent queries in Procedural DMLs. This limitation of Non-Procedural DMLs is overcome by performing query optimization at the System Level.

<u>OVERALL STRUCTURE OF DBMS</u>



(b) **Restricting unauthorized access** The user access rights are stored in the data dictionary. Whenever, any query is received from any user, it is checked for valid access rights. If access rights exist, the query is processed else it is rejected as ‘Invalid Query’. This prevents unauthorized access of data.

(c) **Providing Multiple User-Interfaces** A DBMS provides various types of user interfaces for various categories of users:-

- Query Languages (like SQL) for skilled users
- Programming Languages (like PL/SQL) for application programmers
- Menus, Forms for Naive Users
- DDL for Database Administrator

(d) **Enforcing of Data Integrity Constraints** The Data Integrity Constraints are stored in the data dictionary itself. Whenever, some data is inserted/updated/deleted, the data constraints are automatically applied to the related data items and invalid operations are rejected.

(e) **Supporting Concurrent Access** A DBMS supports concurrent access by multiple users. Despite concurrent access by multiple users, database consistency is maintained.

(f) **Providing backup & recovery** A DBMS supports data backup & recovery in case of failures.

(g) **Reduced Application Development Time** Development time of a new application using DBMS is of the order of 15 – 25% as compared to the time needed in development of equivalent applications in a traditional file processing system.

(h) **Easy Adaptability** A database system can be easily adapted to changed requirement, with minimal time and cost implications.

(i) **Potential for enforcing Standards** It permits the Database Administrator (DBA) to define & enforce standards among the database users. The standards can be defined for naming conventions, formats of data items, display formats or report structures etc.

Lower Level Entity Sets or Sub Classes

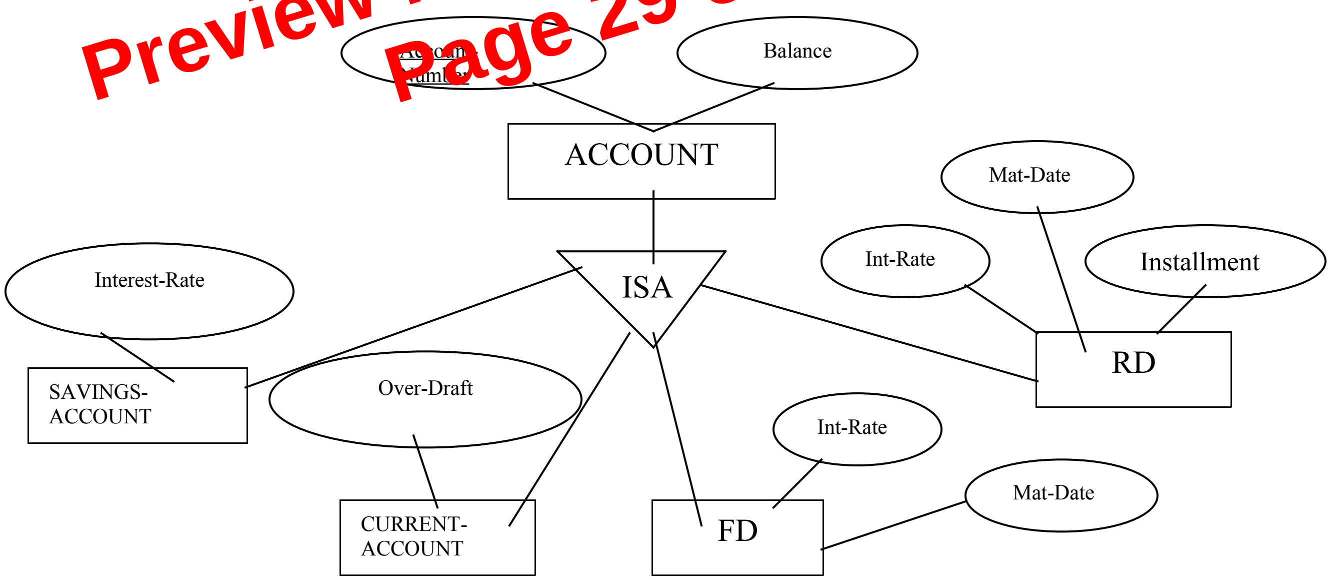
In the above example, an Entity Set E has been specialized into Sub-groups designated as E_1, E_2 ….. E_n. E is called "Super Class" or "Higher Level Entity Set" and the entity sets E_1, E_2 ….. E_n are called "Sub Classes" or "Lower Level Entity Sets" of E. The common attributes of all sub entity sets are represented with the super entity sets. And the distinct attributes of each sub entity set are represented with the sub entity set.

The relationship of Higher Level Entity Set with its Lower Level Entity Sets is called ISA relationship. It is read as "is a".

Inheritance of Attributes in Specialization

Each Sub Class will inherit the Attributes of its Super Class; plus it will have its own distinct Attributes. Like in the above case, each lower entity set will inherit attributes A_1 and A_2 of the Super Class E.

Example:- Consider an entity set *ACCOUNT* with attributes *Account-Number* and *Balance*. The Entity Set ACCOUNT may be specialized into different types of accounts like *SAVINGS-ACCOUNT*, *CURRENT-ACCOUNT*, *FIXED-DEPOSIT (FD) and RECURRING-DEPOSIT (RD)*. The *SAVINGS-ACCOUNT* may have an attribute *Interest-Rate* and *CURRENT-ACCOUNT* may have attribute *Over-Draft*. Similarly, FD and RD have distinct attributes of their own.



Specialization Constraints

Disjoint Vs Overlapping Specialization

Disjoint. It implies that an entity does not belong to more than one lower-level entity set i.e. an *account* is either *savings-account* or *current-account* but not both.

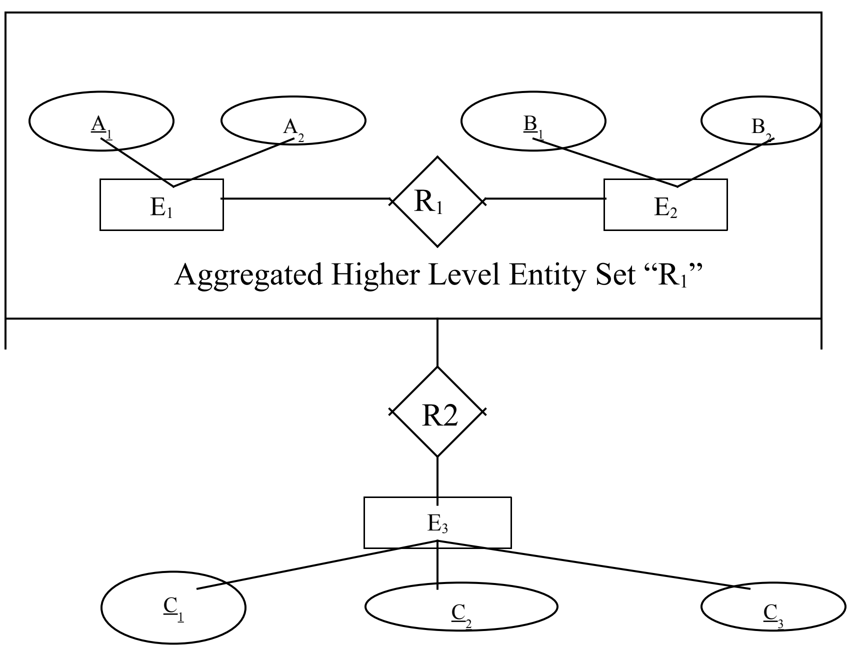
Overlapping. In overlapping generalizations, an entity may belong to more than one lower-level entity sets within a single generalization.

Total Vs Partial Specialization

Total Each higher level entity must belong to a lower-level entity set.
Partial. Some higher-level entities may not belong to any lower-level entity set.

Generalization. Specialization is a top-down approach; whereas Generalization is exactly inverse of that. Generalization refers to the process of fusing several distinct entity sets into a single Higher Level Entity Set, on the basis of commonality of their attributes. Then the fused sets form sub classes or lower level entity sets. The common attributes of the Lower Level Entity Sets will be assigned to the Higher Level Entity Set. Thus, generalization is a process, which proceeds in a bottom-up manner, in which multiple entities are synthesized into a single higher-level entity set, on the basis of their common features. The higher-level entity set is termed as super-class and lower level entity set is termed as sub-class. As regards E-R Diagram, both Specialization and Generalization are represented exactly in the same manner.

Aggregation. One limitation of E-R Model is that it fails to express relationships among relationship sets or relationship between a relationship set on one side and an entity set on the other side. Aggregation provides a solution in this case. Aggregation is an abstraction through which relationships are treated as higher-level entities, which can then participate in relationships with other Entity Sets or with other relationship sets. For example the relationship between R1 and E_3 as indicated below.



The rows in the DEPOSITOR table have one-to-one mapping onto the rows in ACCOUNT Table i.e. with the "Many-Side Entity Set" Table. That is, the first row of DEPOSITOR maps onto the fourth row of ACCOUNT, the second row of DEPOSITOR maps onto the first row of ACCOUNT, the third row of DEPOSITOR maps onto the second row of ACCOUNT, the fourth row of DEPOSITOR maps onto the third row of ACCOUNT, the fifth row of DEPOSITOR maps onto the fifth row of ACCOUNT and the last row of DEPOSITOR maps onto the last row of ACCOUNT table. Thus, the descriptive attribute Date-Of-Operation can be shifted to ACCOUNT (The "Many-Side" Entity Set) and the DEPOSITOR Table can be with the ACCOUNT Table (i.e. with the table of the "Many-Side" Entity Set), without losing any information. The resultant ACCOUNT table will also include the Primary Key C-Id of CUSOMER table and descriptive attribute DOO of the DEPOSITOR table. The resulting set of tables will then be:-

CUSTOMER

C-Id	C-Name	C-address
C-001	Ajay	320, Sector-26, Noida
C-220	Vijay	110,Sector-8, RKP
C-310	Ram	120,Sector-25, Noida
C-505	Shyam	303,Sector-22,RKP

ACCOUNT

Account-Number	Balance	Branch-Name	Customer_Id	Date_of_Operation
A-101	10000	Sec-18	C-220	23-Dec-2006
A-203	30000	Sec-26	C-310	03-Feb-2007
A-305	50000	CP	C-505	27-Dec-2007
A-310	25000	RKP	C-101	10-Jan-2007
A-550	35000	CP	C-101	22-Dec-2006
A-670	60000	Sec-18	C-310	01-Jan-2007

(3) **Many-to-One Relationship** Suppose there is many-to-one relationship between CUSTOMER and ACCCOUNT, which implies that each account can be "Joint" but each customer can hold only one account. In this case, the table DEPOSITOR can be combined with "Many-Side" Entity-Set table CUSTOMER.

Example