Module-1:

LECTURE-1: Introduction to Data

Introduction:

In computerized information system data are the basic resource of the organization. So, proper organization and management for data is required for organization to run smoothly. Database management system deals the knowledge of how data stored and managed on a computerized information system. In any organization, it requires accurate and reliable data for better decision making, ensuring privacy of data and controlling data efficiently.

The examples include deposit and/or withdrawal from a bank, hotel, airline or railway reservation, purchase items from supermarkets in all cases, a database is accessed.

What is data?

Data are the known facts or figures that have implicit meaning. It can also be defined as it is the representation of facts, concepts or instructions in a formal manner, which is suitable for understanding and processing. Data can be represented in alphabets (A-Z, a-z), digits (0-9) and using special characters (+,-.#,$, etc)

e.g: 25, "ajit" etc.

Information:

Information is the processed data on which decisions and actions are based. Information can be defined as the organized and classified data to provide meaningful values.

Eg: "The age of Ravi is 25"

File:

File is a collection of related data stored in secondary memory.

File Oriented Approach:

The traditional file oriented approach to information processing each application has a separate master file and its own set of personal file. In file oriented approach the program dependent on the files and files dependent upon the programs.

Disadvantages of file oriented approach:

1) **Data redundancy and inconsistency:**
   The same information may be written in several files. This redundancy leads to higher storage and access cost. It may lead data inconsistency that is the various copies of the same data may present at multiple places for example a changed customer address may be reflected in single file but not else where in the system.

2) **Difficulty in accessing data :**
   The conventional file processing system do not allow data to be retrieved in a convenient and efficient manner according to user choice.

3) **Data isolation :**
   Because data are scattered in various files and files may be in different formats with new application programs to retrieve the appropriate data is difficult.

4) **Integrity Problems:**

Developers enforce data validation in the system by adding appropriate code in the various application program. How ever when new constraints are added, it is difficult to change the programs to enforce them.

5) **Atomicity:**
It is difficult to ensure atomicity in a file processing system when transaction failure occurs due to power failure, networking problems etc. (atomicity: either all operations of the transaction are reflected properly in the database or non are)

6) **Concurrent access:**
In the file processing system it is not possible to access the same file for transaction at same the time.

7) **Security problems:**
There is no security provided in file processing system to secure the data from unauthorized user access.

LECTURE-2: DBMS

Database:

A database is organized collection of related data of an organization stored in formatted way which is shared by multiple users.

The main feature of data in a database are:

1. It must be well organized
2. It is related
3. It is accessible in a logical order without any difficulty
4. It is stored only once

For example consider the roll no, name, address of a student stored in a student file. It is collection of related data with an implicit meaning. Data in the database may be persistent, integrated and shared.

Persistent:

If data is removed from database due to some explicit request from user to remove.

Integrated:

A database can be a collection of data from different files and when any redundancy among those files are removed from database is said to be integrated data.

Sharing Data:

The data stored in the database can be shared by multiple users simultaneously without affecting the correctness of data.

Why Database:

In order to overcome the limitation of a file system, a new approach was required. Hence a database approach emerged. A database is a persistent collection of logically related data. The initial attempts were to provide a centralized collection of data. A database has a self describing nature. It contains not only the data sharing and integration of data of an organization in a single database.

A small database can be handled manually but for a large database and having multiple users it is difficult to maintain it. In that case a computerized database is useful.

The advantages of database system over traditional, paper based methods of record keeping are:

- **Compactness**: No need for large amount of paper files
- **Speed**: The machine can retrieve and modify the data more faster way then human being
- **Less drudgery**: Much of the maintenance of files by hand is eliminated
- **Accuracy:** Accurate, up-to-date information is fetched as per requirement of the user at any time.

Database Management System (DBMS):

A database management system consists of collection of related data and refers to a set of programs for defining, creation, maintenance and manipulation of a database.

Function of DBMS:

1. **Defining database schema**: it must give facility for defining the database structure also specifies access rights to authorized users.
2. **Manipulation of the database:** The dbms must have functions like insertion of record into database, updation of data, deletion of data, retrieval of data
3. **Sharing of database:** The DBMS must share data items for multiple users by maintaining consistency of data.
4. **Protection of database:** It must protect the database against unauthorized users.
5. **Database recovery:** If for any reason the system fails DBMS must facilitate data base recovery.

Advantages of DBMS:

Reduction of redundancies:
Centralized control of data by the DBA avoids unnecessary duplication of data and effectively reduces the total amount of data storage required avoiding duplication in the elimination of the inconsistencies that tend to be present in redundant data files.

Sharing of Data:
A database allows the sharing of data under its control by any number of application programs or users.

Data Integrity:
Data integrity means that the data contained in the database is both accurate and consistent. Therefore data values being entered for storage could be checked to ensure that they fall with in a specified range and are of the correct format.

Data Security:
The DBA who has the ultimate responsibility for the data in the dbms can ensure that proper access procedures are followed including proper authentication to access to the DataBase System and additional check before permitting access to sensitive data.

Conflict Resolution:
DBA resolve the conflict on requirements of various user and applications. The DBA chooses the best file structure and access method to get optional performance for the application.

Data Independence:
Data independence is usually considered from two points of views; physically data independence and logical data independence.

Physical Data Independence allows changes in the physical storage devices or organization of the files to be made without requiring changes in the conceptual view or any of the external views and hence in the application programs using the data base.

Logical Data Independence indicates that the conceptual schema can be changed without affecting the existing external schema or any application program.

Disadvantage of DBMS:

1. DBMS software and hardware (networking installation) cost is high
2. The processing overhead by the dbms for implementation of security, integrity and sharing of the data.
3. Centralized database control
4. Setup of the database system requires more knowledge, money, skills, and time.
5. The complexity of the database may result in poor performance.

LECTURE-3: 3 level Architecture of DBMS

Database Basics:

Data Item:
The data item is also called as field in data processing and is the smallest unit of data that has meaning to its users.
Eg: “e101”, ”sumit”

Entities and attributes:
An entity is a thing or object in the real world that is distinguishable from all other objects
Eg: Bank, employee, student

Attributes are properties are properties of an entity.
Eg: Empcode, ename, rolno, name

Logical data and physical data :
Logical data are the data for the table created by user in primary memory.
Physical data refers to the data stored in the secondary memory.

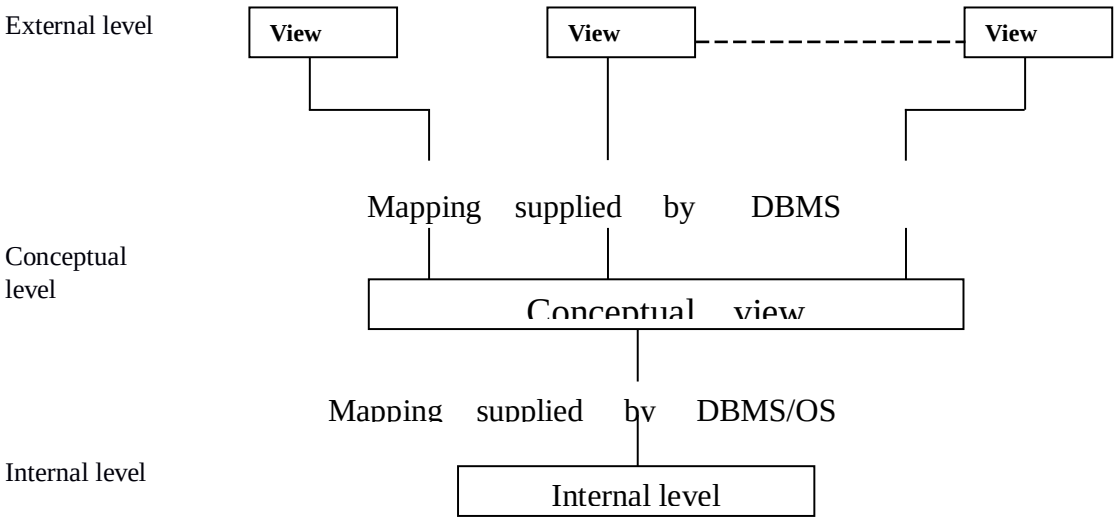
Schema and sub-schema :
A schema is a logical data base description and is drawn as a chart of the types of data that are used. It gives the names of the entities and attributes and specify the relationships between them.

A database schema includes such information as :

- Characteristics of data items such as entities and attributes .
- Logical structures and relationships among these data items .
- Format for storage representation.
- Integrity parameters such as physical authorization and back up policies.

A *subschema* is derived schema derived from existing schema as per the user requirement. There may be more then one subschema create for a single conceptual schema.

Three Level Architecture of DBMS :



A database management system that provides three level of data is said to follow three-level architecture .

- External level
- Conceptual level
- Internal level

External Level :
The external level is at the highest level of database abstraction . At this level, there will be many views define for different users requirement. A view will describe only a subset of the database. Any number of user views may exist for a given global schema(coneptual schema).

For example, each student has different view of the time table. the view of a student of BTech (CSE) is different from the view of the student of Btech (ECE). Thus this level of abstraction is concerned with different categories of users.
Each external view is described by means of a schema called sub schema.

Conceptual Level :
At this level of database abstraction all the database entities and the relationships among them are included. One conceptual view represents the entire database. This conceptual view is defined by the conceptual schema.

The conceptual schema hides the details of physical storage structures and concentrate on describing entities, data types, relationships, user operations and constraints.

It describes all the records and relationships included in the conceptual view. There is only one conceptual schema per database. It includes feature that specify the checks to relation data consistency and integrity.

Internal level :
It is the lowest level of abstraction closest to the physical storage method used. It indicates how the data will be stored and describes the data structures and access methods to be used by the database. The internal view is expressed by internal schema.

The following aspects are considered at this level:

1. Storage allocation e.g: B-tree, hashing
2. Access paths eg. specification of primary and secondary keys, indexes etc
3. Miscellaneous eg. Data compression and encryption techniques, optimization of the internal structures.

Database Users :

Naive Users :
Users who need not be aware of the presence of the database system or any other system supporting their usage are considered naïve users . A user of an automatic teller machine falls on this category.

Online Users :
These are users who may communicate with the database directly via an online terminal or indirectly via a user interface and application program. These users are aware of the database system and also know the data manipulation language system.

Application Programmers :
Professional programmers who are responsible for developing application programs or user interfaces utilized by the naïve and online user falls into this category.

Database Administration :
A person who has central control over the system is called database administrator .
The function of DBA are :

1. Creation and modification of conceptual Schema definition
2. Implementation of storage structure and access method.
3. Schema and physical organization modifications .
4. Granting of authorization for data access.
5. Integrity constraints specification.
6. Execute immediate recovery procedure in case of failures
7. Ensure physical security to database

Database language :

1) **Data definition language (DDL) :**
   DDL is used to define database objects .The conceptual schema is specified by a set of definitions expressed by this language. It also gives some details about how to implement this schema in the physical devices used to store the data. This definition includes all the entity sets and their associated attributes and their relationships. The result of DDL statements will be a set of tables that are stored in special file called data dictionary.

2) **Data Manipulation Language (DML) :**
   A DML is a language that enables users to access or manipulate data stored in the database. Data manipulation involves retrieval of data from the database, insertion of new data into the database and deletion of data or modification of existing data.

   There are basically two types of DML:
   - **Procedural**: Which requires a user to specify what data is needed and how to get it.
   - **Non-Procedural:** which requires a user to specify what data is needed with out specifying how to get it.

3) **Data Control Language (DCL):**
   This language enables user to grant authorization and canceling authorization of database objects.

LECTURE-4: Elements of DBMS

Elements of DBMS:

DML Pre-Compiler:
It converts DML statements embedded in an application program to normal procedure calls in the host language. The pre-complier must interact with the query processor in order to generate the appropriate code.

DDL Compiler:
The DDL compiler converts the data definition statements into a set of tables. These tables contains information concerning the database and are in a form that can be used by other components of the dbms.

File Manager:
File manager manages the allocation of space on disk storage and the data structure used to represent information stored on disk.

Database Manager:
A database manager is a program module which provides the interface between the low level data stored in the database and the application programs and queries submitted to the system.

The responsibilities of database manager are:

1. **Interaction with File Manager**: The data is stored on the disk using the file system which is provided by operating system. The database manager translate the different DML statements into low-level file system commands so the database manager is responsible for the actual storing, retrieving and updating of data in the database.

2. **Integrity Enforcement:** The data values stored in the database must satisfy certain constraints (eg: the age of a person can't be less then zero). These constraints are specified by DBA. Data manager checks the constraints and if it satisfies then it stores the data in the database.

3. **Security Enforcement:** Data manager checks the security measures for database from unauthorized users.

4. **Backup and Recovery:** Database manager detects the failures occur due to different causes (like disk failure, power failure, deadlock, software error) and restores the database to original state of the database.

5. **Concurrency Control:** When several users access the same database file simultaneously, there may be possibilities of data inconsistency. It is responsible of database manager to control the problems occur for concurrent transactions.

Query Processor:
The query processor used to interpret to online user's query and convert it into an efficient series of operations in a form capable of being sent to the data manager for execution. The query processor

uses the data dictionary to find the details of data file and using this information it create query plan/access plan to execute the query.

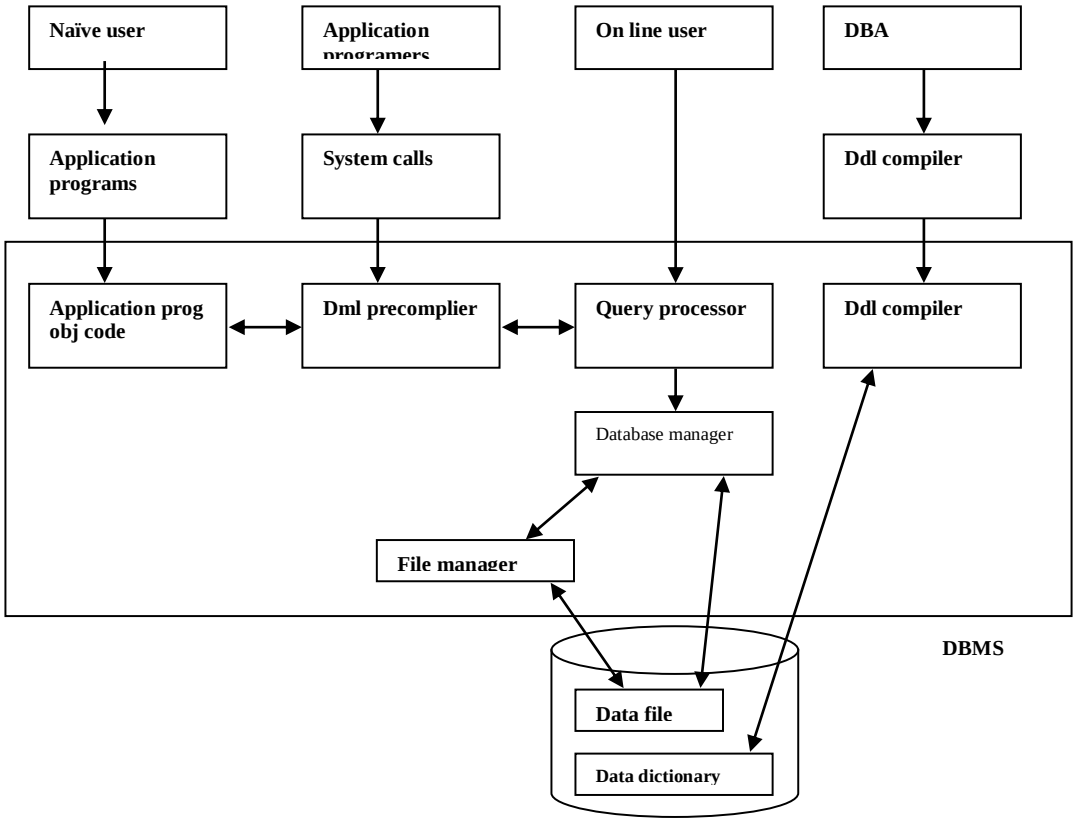
Data Dictionary:

Data dictionary is the table which contains the information about database objects. It contains information like

1. external, conceptual and internal database description
2. description of entities, attributes as well as meaning of data elements
3. synonyms, authorization and security codes
4. database authorization

The data stored in the data dictionary is called *meta data*.

DBMS STRUCTURE:



differences between a File-Processing System and a DBMS.

Some major differences between a database management system and a file-processing system

- Both systems contain a collection of data and a set of programs which access that data. A database management system coordinates both the physical and the logical access to the data, whereas a file-processing system coordinates only the physical access.

- A database management system reduces the amount of data duplication by ensuring that a physical piece of data is available to all programs authorized to have access to it, where as data written by one program in a file-processing system may not be readable by another program.

- A database management system is designed to allow flexible access to data (i.e., queries), whereas a file-processing system is designed to allow predetermined access to data (i.e., compiled programs).

- A database management system is designed to coordinate multiple users accessing the same data at the same time. A file-processing system is usually designed to allow one or more programs to access different data files at the same time. In a file-processing system, a file can be accessed by two programs concurrently only if both programs have read-only access to the file.

Que: Explain the difference between physical and logical data independence.

Ans:

- Physical data independence is the ability to modify the physical scheme without making it necessary to rewrite application programs. Such modifications include changing from unblocked to blocked record storage, or from sequential to random access files.

- Logical data independence is the ability to modify the conceptual scheme without making it necessary to rewrite application programs. Such a modification might be adding a field to a record; an application program's view hides this change from the program.

Que: List five responsibilities of a database management system. For each responsibility, explain the problems that would arise if the responsibility were not discharged.

Ans: A general purpose database manager (DBM) has five responsibilities:

a. interaction with the file manager.
b. integrity enforcement.
c. security enforcement.
d. backup and recovery.
e. concurrency control.

If these responsibilities were not met by a given DBM (and the text points out that sometimes a responsibility is omitted by design, such as concurrency control on a single-user DBM for a micro computer) the following problems can occur, respectively:

a. No DBM can do without this, if there is no file manager interaction then nothing stored in the files can be retrieved.

b. Consistency constraints may not be satisfied, when account balances could go below the minimum allowed, employees could earn too much overtime (e.g.,hours > 80) or, airline pilots may fly more hours than allowed by law.

c. Unauthorized users may access the database, or users authorized to access part of the

database may be able to access parts of the database for which they lack authority. For example, a high school student could get access to national defense secret codes, or employees could find out what their supervisors earn.

d. Data could be lost permanently, rather than at least being available in a consistent state that existed prior to a failure.

e. Consistency constraints may be violated when intgrity constraints failed in a transaction. For example, incorrect bank balances might be reflected due to simultaneous withdrawals and deposits, and so on.

Five main functions of a database administrator are:

- To create the scheme definition
- To define the storage structure and access methods
- To modify the scheme and/or physical organization when necessary
- To grant authorization for data access
- To specify integrity constraints

Six major steps in setting up a database for a particular enterprise are:

- Define the high level requirements of the enterprise (this step generates a document known as the system requirements specification.)
- Define a model containing all appropriate types of data and data relationships.
- Define the integrity constraints on the data.
- Define the physical level.
- For each known problem to be solved on a regular basis (e.g., tasks to be carried out by clerks or Web users) define a user interface to carry out the task, and write the necessary application programs to implement the user interface.
- Create/initialize the database.

LECTURE-5: ER-MODEL

Data Model:
The data model describes the structure of a database. It is a collection of conceptual tools for describing data, data relationships and consistency constraints and various types of data models such as

1. Object based logical model
2. Record based logical model
3. Physical model

Types of data model:

1. Object based logical model
    a. ER-model
    b. Functional model
    c. Object oriented model
    d. Semantic model
2. Record based logical model
    a. Hierarchical database model
    b. Network model
    c. Relational model
3. Physical model

Entity Relationship Model (ER Model)
The entity-relationship data model perceives the real world as consisting of basic objects, called entities and relationships among these objects. It was developed to facilitate database design by allowing specification of an enterprise schema which represents the overall logical structure of a data base.

Main Features of ER-MODEL:

- Entity relationship model is a high level conceptual model
- It allows us to describe the data involved in a real world enterprise in terms of objects and their relationships.
- It is widely used to develop an initial design of a database
- It provides a set of useful concepts that make it convenient for a developer to move from a basic set of information to a detailed and description of information that can be easily implemented in a database system
- It describes data as a collection of entities, relationships and attributes.

Basic Concepts:
The E-R data model employs three basic notions : entity sets, relationship sets and attributes.

Entity Sets:
An entity is a “thing” or “object” in the real world that is distinguishable from all other objects. For example, each person in an enterprise is an entity. An entity has a set properties and the values for some set of properties may uniquely identify an entity. BOOK is entity and its properties (called as attributes) bookcode, booktitle, price etc.

An entity set is a set of entities of the same type that share the same properties, or attributes. The set

of all persons who are customers at a given bank.

Attributes:
An entity is represented by a set of attributes. Attributes are descriptive properties possessed by each member of an entity set.

Customer is an entity and its attributes are **customerid, custmername, custaddress** etc.

An attribute as used in the E-R model, can be characterized by the following attribute types.

a) **Simple and Composite Attribute:**
Simple attributes are the attributes which can't be divided into sub parts, e.g. customerid, empno
Composite attributes are the attributes which can be divided into subparts, e.g. name consisting of first name, middle name, last name and address consisting of city, pincode, state.

b) **Single-Valued and Multi-Valued Attribute:**
The attribute having unique value is single –valued attribute, e.g. empno, customerid, regdno etc.
The attribute having more than one value is multi-valued attribute, eg: phone-no, dependent name, vehicle.

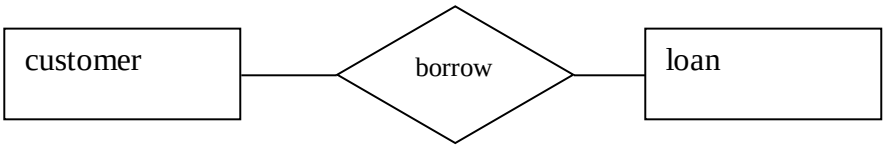
c) **Derived Attribute:**
The values for this type of attribute can be derived from the values of existing attributes, e.g. age which can be derived from currentdate – birthdate and experience_in_year can be calculated as currentdate-joindate.

d) **NULL Valued Attribute:**
The attribute value which is not known to user is called NULL valued attribute.

Relationship Sets:
A relationship is an association among several entities. A relationship set is a set of relationships of the same type. Formally, it is a mathematical relation on n>=2 entity sets. If E_1, E_2…E_n are entity sets, then a relation ship set R is a subset of

$$\{(e_1,e_2,\ldots e_n) \mid e_1 \in E_1, e_2 \in E_2.., e_n \in E_n\}$$

where $(e_1,e_2,\ldots e_n)$ is a relation ship.

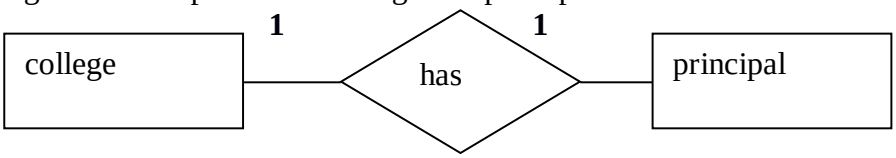


Consider the two entity sets customer and loan. We define the relationship set borrow to denote the association between customers and the bank loans that the customers have.

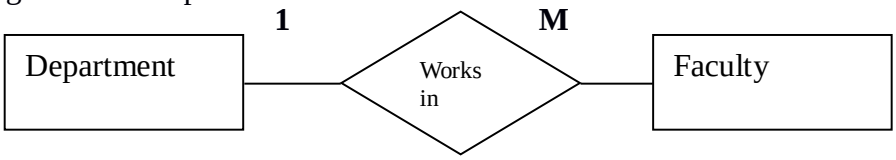
Mapping Cardinalities:
Mapping cardinalities or cardinality ratios, express the number of entities to which another entity can be associated via a relationship set. Mapping cardinalities are most useful in describing binary relationship sets, although they can contribute to the description of relationship sets that involve more than two entity sets. For a binary relationship set R between entity sets A and B, the mapping

cardinalities must be one of the following:

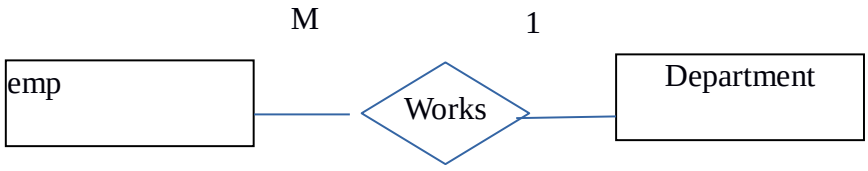
1. One to One:
An entity in A is associated with at most one entity in B, and an entity in B is associated with at most one entity in A.
Eg: relationship between college and principal



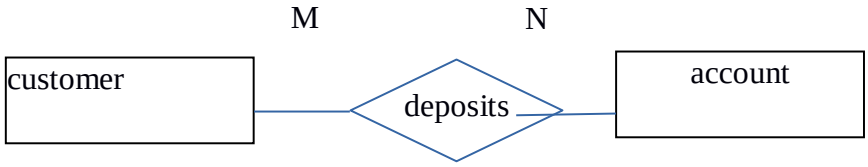
2. One to Many:
An entity in A is associated with any number of entities in B. An entity in B is associated with at the most one entity in A.
Eg: Relationship between department and faculty



3. Many to One:
An entity in A is associated with at most one entity in B. An entity in B is associated with any number in A.



4. Many to Many:
Entities in A and B are associated with any number of entities from each other.



More about Entities and Relationship:

Recursive Relationships:
When the same entity type participates more than once in a relationship type in different roles, the relationship types are called recursive relationships.

Participation Constraints:
The participation constraints specify whether the existence of any entity depends on its being related to another entity via the relationship. There are two types of participation constraints

a) Total : When all the entities from an entity set participate in a relationship type, is called total participation. For example, the participation of the entity set student on the relationship set must 'opts' is said to be total because every student enrolled must opt for a course.

b) Partial: When it is not necessary for all the entities from an entity set to particapte in a relationship type, it is called partial participation. For example, the participation of the entity set student in 'represents' is partial, since not every student in a class is a class representative.

Weak Entity:
Entity types that do not contain any key attribute, and hence can not be identified independently are called weak entity types. A weak entity can be identified by uniquely only by considering some of its attributes in conjunction with the primary key attribute of another entity, which is called the identifying owner entity.

Generally a partial key is attached to a weak entity type that is used for unique identification of weak entities related to a particular owner type. The following restrictions must hold:

- The owner entity set and the weak entity set must participate in one to may relationship set. This relationship set is called the identifying relationship set of the weak entity set.
- The weak entity set must have total participation in the identifying relationship.

Example:
Consider the entity type Dependent related to Employee entity, which is used to keep track of the dependents of each employee. The attributes of Dependents are: name, birthdate, sex and relationship. Each employee entity set is said to its own the dependent entities that are related to it. However, not that the 'Dependent' entity does not exist of its own, it is dependent on the Employee entity.

Keys:

Super Key:
A super key is a set of one or more attributes that taken collectively, allow us to identify uniquely an entity in the entity set. For example , customer-id, (cname, customer-id), (cname, telno)

Candidate Key:
In a relation R, a candidate key for R is a subset of the set of attributes of R, which have the following properties:

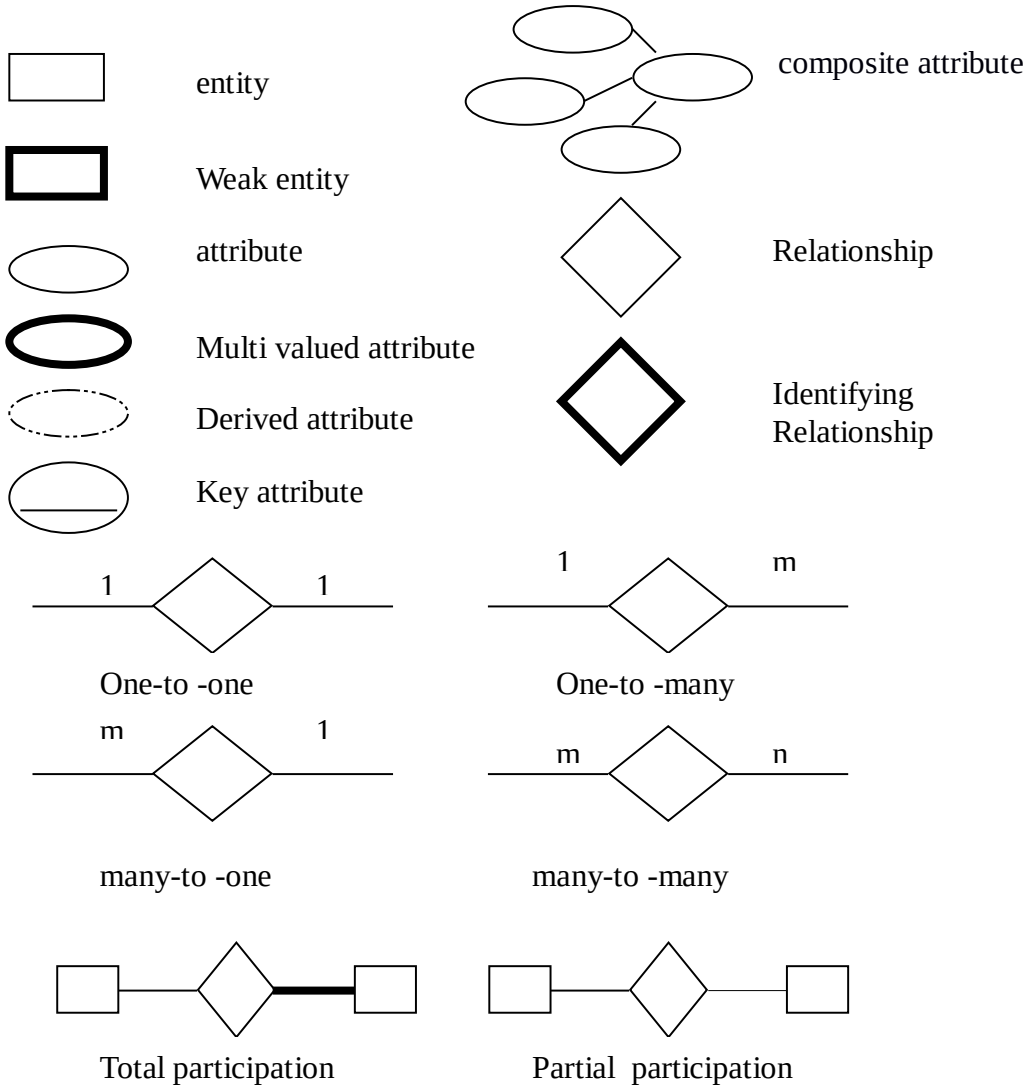
1. *Uniqueness: N*o two distinct tuples in R have the same values for the candidate key
2. *Irreducible:* No proper subset of the candidate key has the uniqueness property that is the candidate key. Eg: (cname,telno)

Primary Key:
The primary key is the candidate key that is chosen by the database designer as the principal means of identifying entities within an entity set. The remaining candidate keys if any, are called *Alternate Key*.

LECTURE-6: ER-DIAGRAM:

The overall logical structure of a database using ER-model graphically with the help of an ER-diagram.

Symbols use ER- diagram:



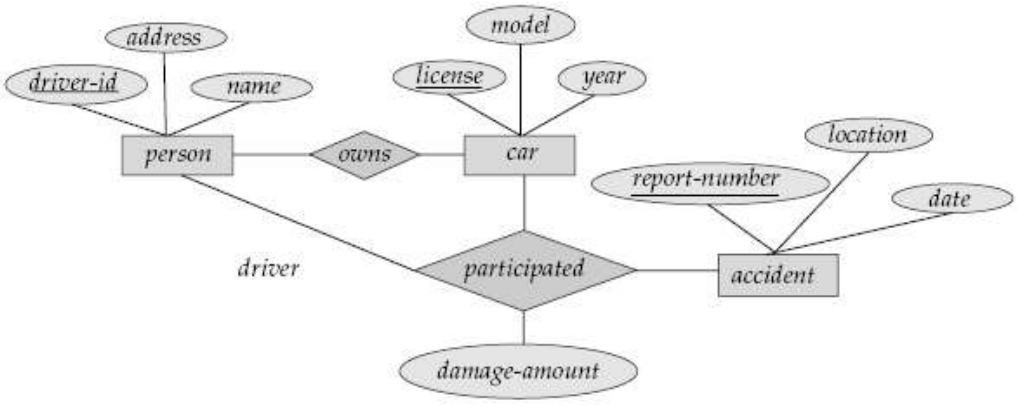


Figure 2.1 E-R diagram for a Car-insurance company.

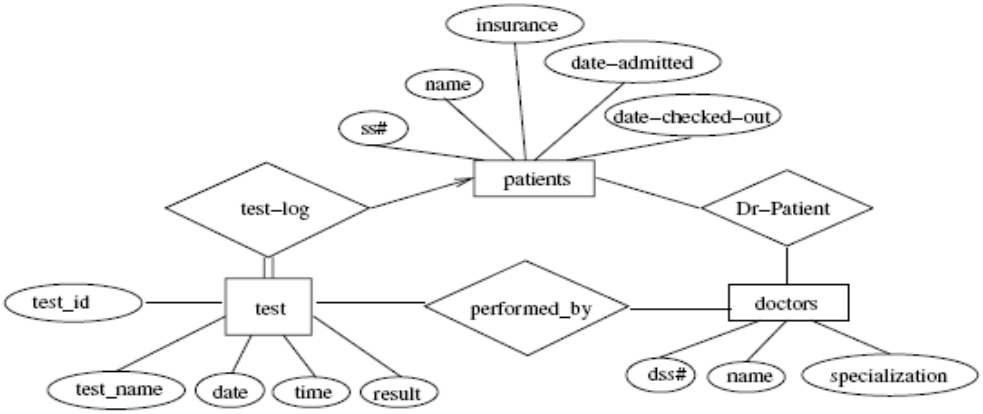


Figure 2.2 E-R diagram for a hospital.

A Univeristy registrar's office maintains data about the following entities:

(a) Course, includeing number,title,credits,syllabus and prereqisites

(b) course offering,including course number,year,semester,section number,instructor timings, and class room

(c) Students including student-id,name and program

(d) Instructors, including identification 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 appropriate modeled.

Construct an E-R diagram for the registrar's office. Document all assumptions that you may make about the mapping constratints

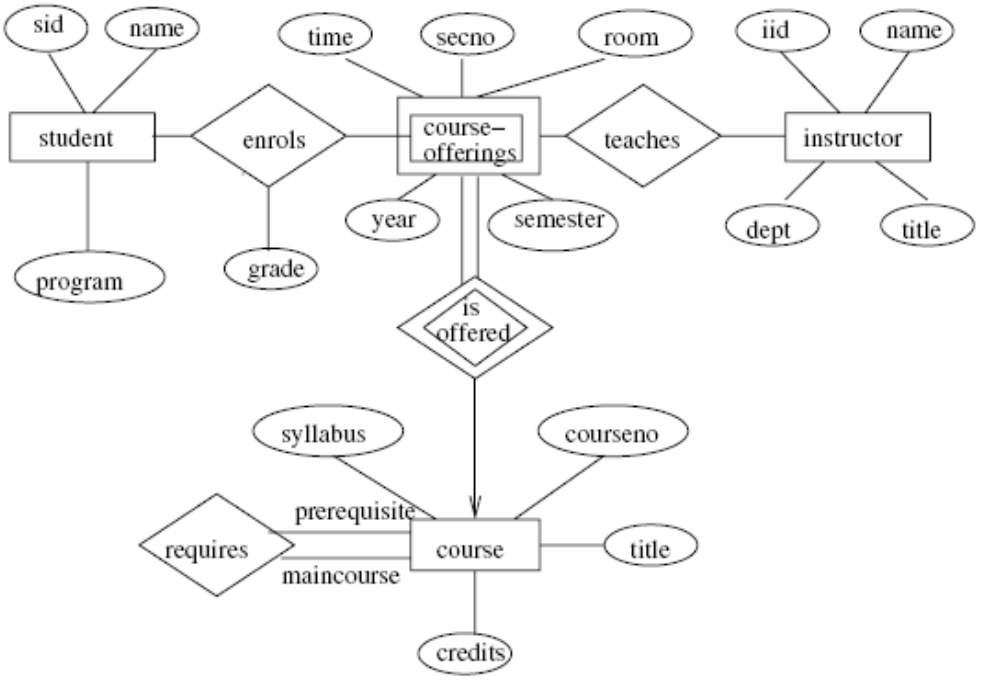


Figure 2.3 E-R diagram for a university.

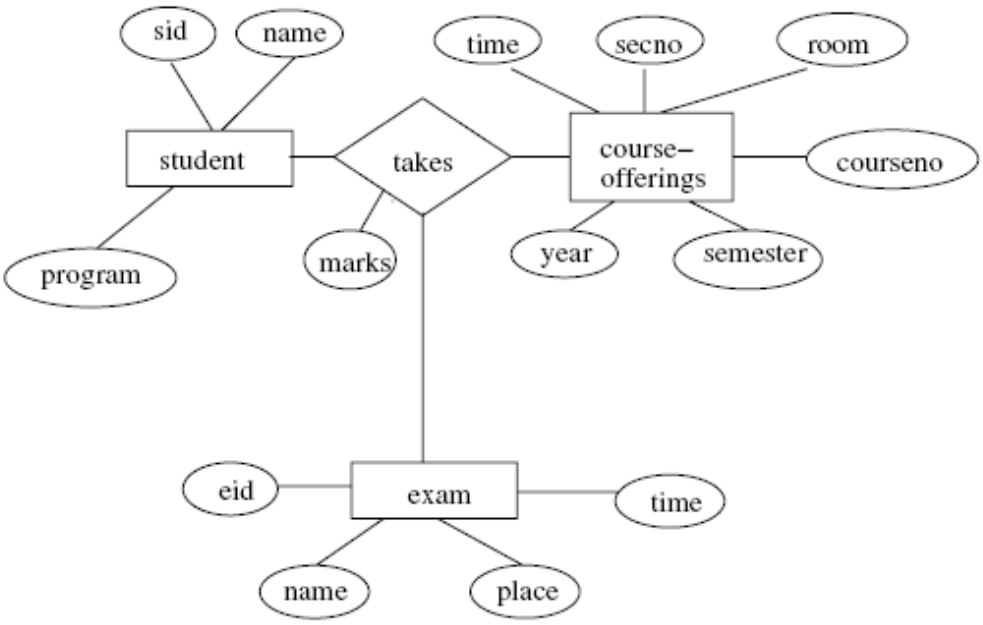


Figure 2.4 E-R diagram for marks database.

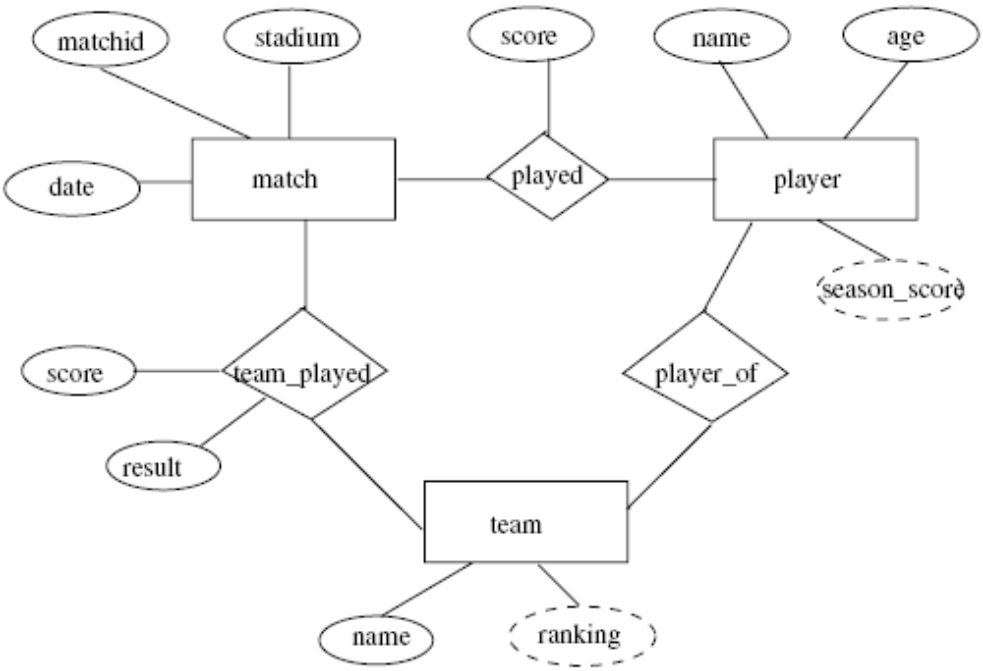


Figure 2.7 E-R diagram for all teams statistics.

Cosidet a university database for the scheduling of class rooms 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 would be defined, along with relationship sets to replae 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

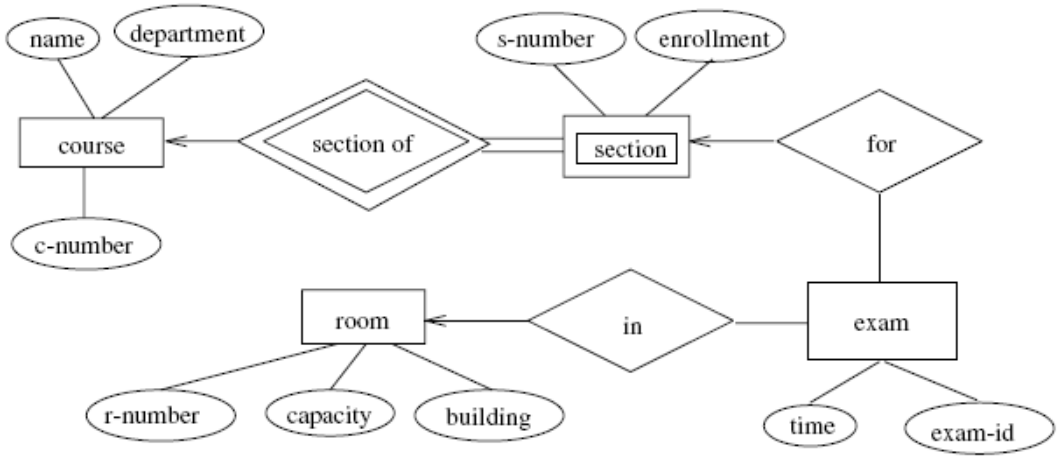


Figure 2.12 E-R diagram for exam scheduling.

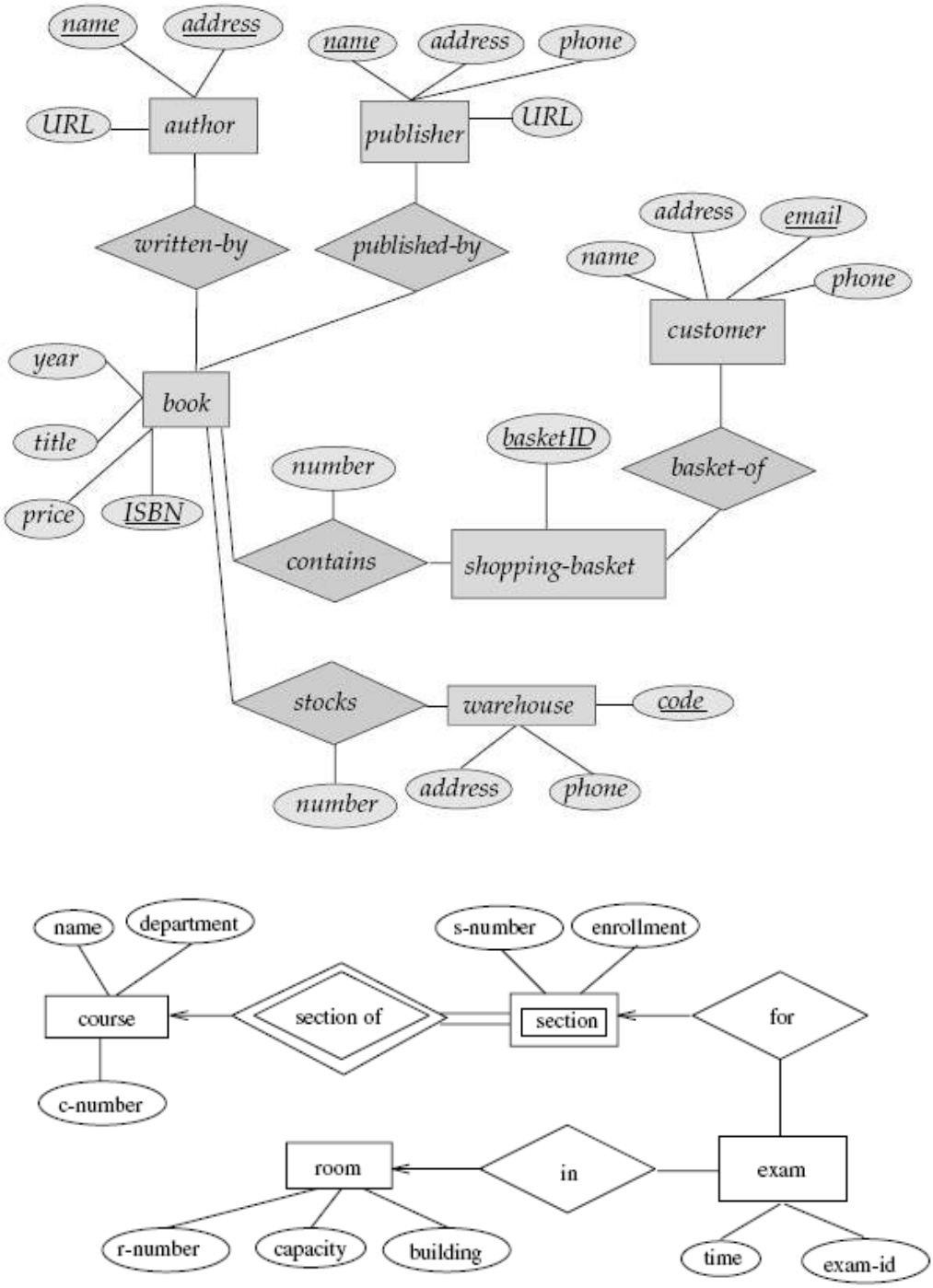


Figure 2.12 E-R diagram for exam scheduling.

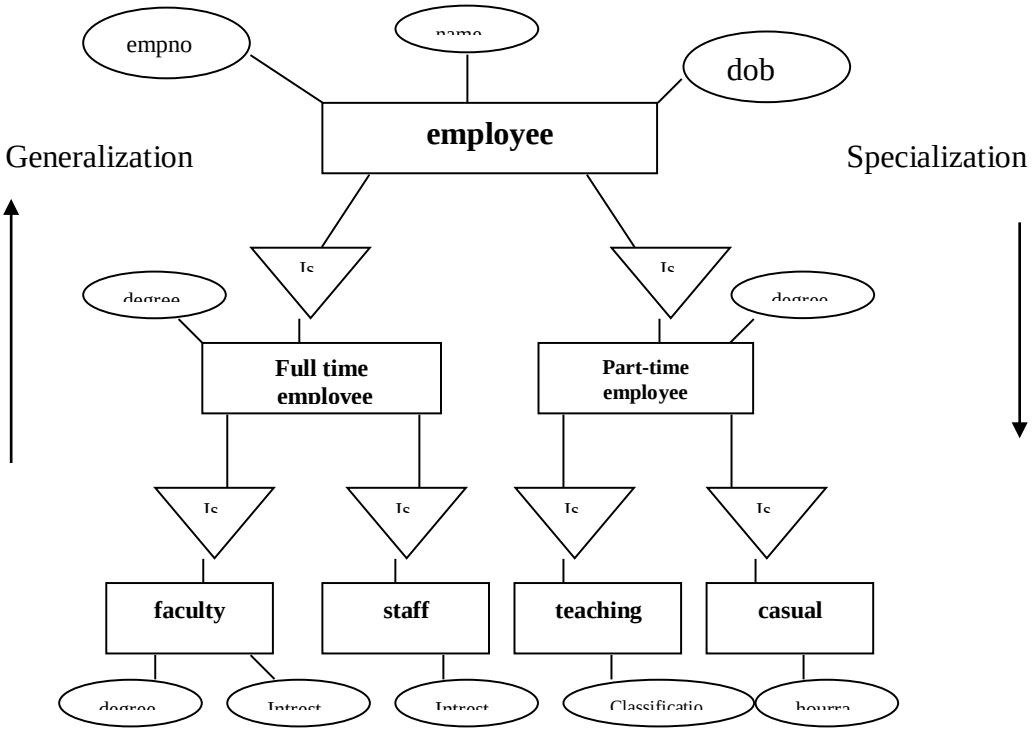
LECTURE-7: Advanced ER-Diagram:

Abstraction is the simplification mechanism used to hide superfluous details of a set of objects. It allows one to concentrate on the properties that are of interest to the application. There are two main abstraction mechanism used to model information:

Generalization and specialization:

Generalization is the abstracting process of viewing set of objects as a single general class by concentrating on the general characteristics of the constituent sets while suppressing or ignoring their differences. It is the union of a number of lower-level entity types for the purpose of producing a higher-level entity type. For instance, student is a generalization of graduate or undergraduate, full-time or part-time students. Similarly, employee is generalization of the classes of objects cook, waiter, and cashier. Generalization is an IS_A relationship; therefore, manager IS_AN employee, cook IS_AN employee, waiter IS_AN employee, and so forth.

Specialization is the abstracting process of introducing new characteristics to an existing class of objects to create one or more new classes of objects. This involves taking a higher-level, and using additional characteristics, generating lower-level entities. The lower-level entities also inherits the, characteristics of the higher-level entity. In applying the characteristics size to car we can create a full-size, mid-size, compact or subcompact car. Specialization may be seen as the reverse process of generalization addition specific properties are introduced at a lower level in a hierarchy of objects.



EMPLOYEE(**empno**,name,dob)
FULL_TIME_EMPLOYEE(**empno**,salary)
PART_TIME_EMPLOYEE(**empno**,type)

Faculty(**empno**,degree,intrest)
Staff(**empno**,hour-rate)
Teaching (**empno**,stipend)

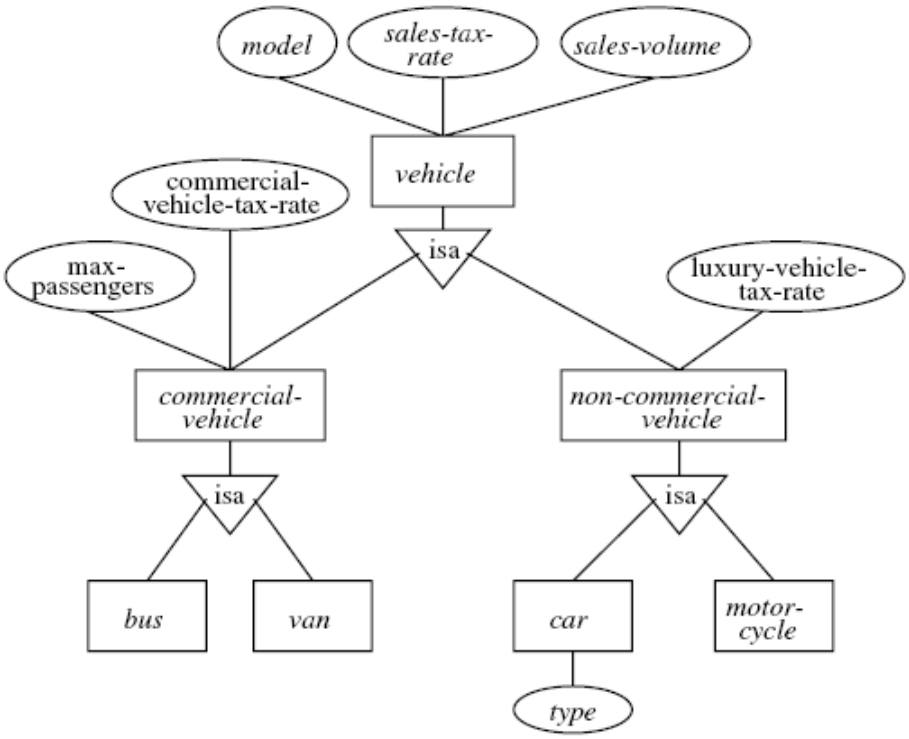
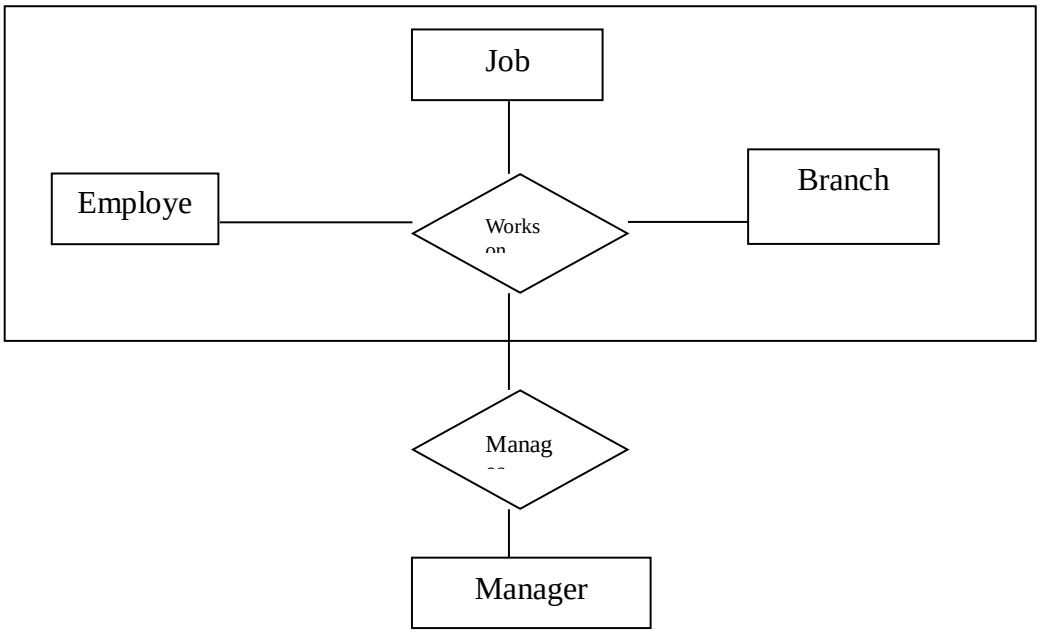


Figure 2.19 E-R diagram of motor-vehicle sales company.

Aggregation:

Aggregation is the process of compiling information on an object, there by abstracting a higher level object. The entity person is derived by aggregating the characteristics of name, address, ssn. Another form of the aggregation is abstracting a relationship objects and viewing the relationship as an object.



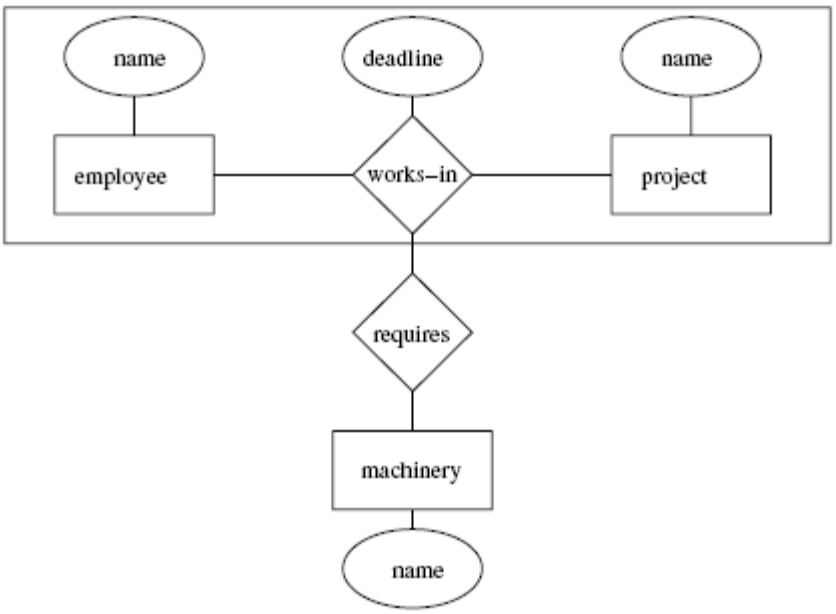


Figure 2.8 E-R diagram Example 1 of aggregation.

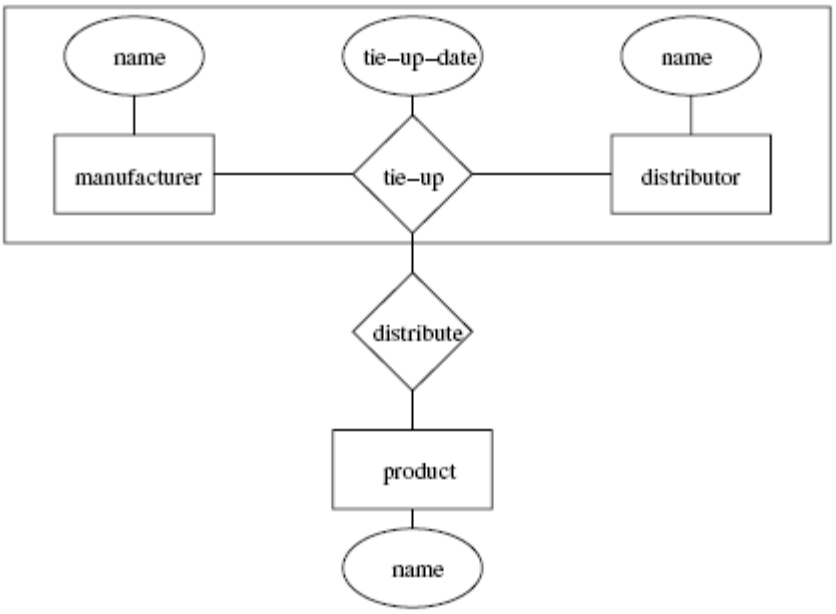
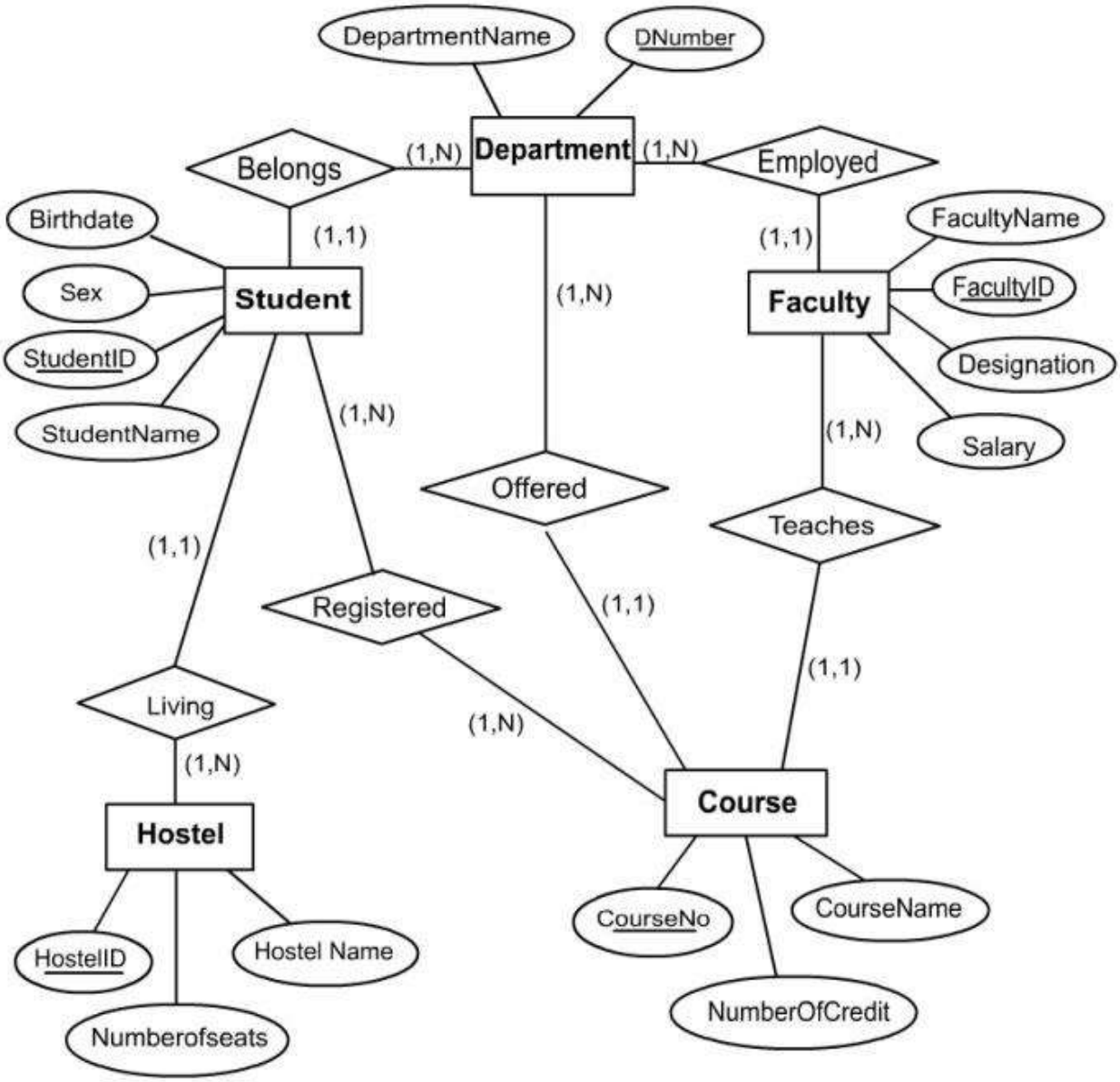


Figure 2.9 E-R diagram Example 2 of aggregation.

ER- Diagram For College Database



LECTURE-8: Conversion of ER-Diagram to Relational Database

Conversion of Entity Sets:

1. For each strong entity type E in the ER diagram, we create a relation R containing all the single attributes of E. The primary key of the relation R will be one of the key attribute of R.

   STUDENT(rollno (primary key),name, address)
   FACULTY(id(primary key),name ,address, salary)
   COURSE(course-id,(primary key),course_name,duration)
   DEPARTMENT(dno(primary key),dname)

2. For each weak entity type W in the ER diagram, we create another relation R that contains all simple attributes of W. If E is an owner entity of W then key attribute of E is also include In R. This key attribute of R is set as a foreign key attribute of R. Now the combination of primary key attribute of owner entity type and partial key of the weak entity type will form the key of the weak entity type

   GUARDIAN((rollno,name) (primary key),address,relationship)

Conversion of Relationship Sets:

Binary Relationships:

- **One-to-One Relationship:**
  For each 1:1 relationship type R in the ER-diagram involving two entities E1 and E2 we choose one of entities(say E1) preferably with total participation and add primary key attribute of another E as a foreign key attribute in the table of entity(E1). We will also include all the simple attributes of relationship type R in E1 if any, For example, the department relationship has been extended tp include head-id and attribute of the relationship.
  DEPARTMENT(D_NO,D_NAME,HEAD_ID,DATE_FROM)

- **One-to-Many Relationship:**
  For each 1:N relationship type R involving two entities E1 and E2, we identify the entity type (say E1) at the N-side of the relationship type R and include primary key of the entity on the other side of the relation (say E2) as a foreign key attribute in the table of E1. We include all simple attribute (or simple components of a composite attribute of R (if any) in the table E1)

  For example:
  The works in relationship between the DEPARTMENT and FACULTY. For this relationship choose the entity at N side, i.e, FACULTY and add primary key attribute of another entity DEPARTMENT i.e., DNO as a foreign key attribute in FACULTY.

  FACULTY(CONTAINS WORKS_IN RELATIOSHIP)
  (ID, NAME, ADDRESS, BASIC_SAL, DNO)

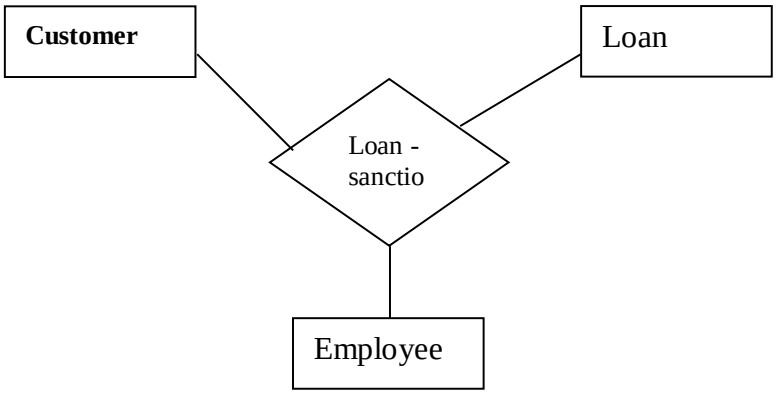
- **Many-to-Many Relationship:**
  For each M:N relationship type R, we create a new table (say S) to represent R, we also include the primary key attributes of both the participating entity types as a foreign key attribute in S. Any simple attributes of the M:N relationship type (or simple components as a composite attribute) is also included as attributes of S.

For example:
The M:N relationship taught-by between entities COURSE and FACULTY should be represented as a new table. The structure of the table will include primary key of COURSE and primary key of FACULTY entities.

TAUGHT-BY (ID (primary key of FACULTY table), course-id (primary key of COURSE table)

- **N-ary Relationship:**

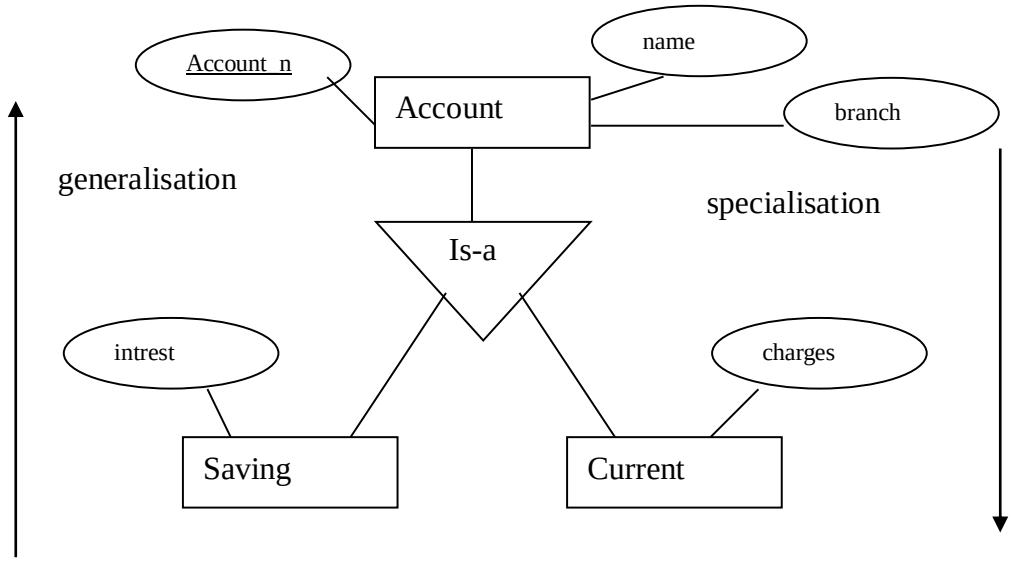
For each N-ary relationship type R where n>2, we create a new table S to represent R, We include as foreign key attributes in S the primary keys of the relations that represent the participating entity types. We also include any simple attributes of the N-ary relationship type (or simple components of complete attribute) as attributes of S. The primary key of S is usually a combination of all the foreign keys that reference the relations representing the participating entity types.



LOAN-SANCTION (cusomer-id, loanno, empno, sancdate, loan_amount)

- **Multi-Valued Attributes:**

For each multivalued attribute 'A', we create a new relation R that includes an attribute corresponding to plus the primary key attributes k of the relation that represents the entity type or relationship that has as an attribute. The primary key of R is then combination of A and k.
For example, if a STUDENT entity has rollno, name and phone number where phone number is a multivalued attribute then we will create table PHONE (rollno, phoneno) where primary key is the combination. In the STUDENT table we need not have phone number, instead if can be simply (rollno, name) only.
PHONE(rollno, phoneno)



- **Converting Generalisation /Specification Hierarchy to Tables:**

  A simple rule for conversion may be to decompose all the specialized entities into table in case they are disjoint, for example, for the figure we can create the three tables as:

  Account (account_no, name, branch, balance)

  Saving_Account (account-no, intrest)

  Current_Account (account-no, charges)

LECTURE-9: Record Based Logical Model

Hierarchical Model:

- A hierarchical database consists of a collection of *records* which are connected to one another through *links.*
- A record is a collection of fields, each of which contains only one data value.
- A link is an association between precisely two records.
- The hierarchical model differs from the network model in that the records are organized as collections of trees rather than as arbitrary graphs.

Tree-Structure Diagrams:

- The schema for a hierarchical database consists of
  - *boxes,* which correspond to record types
  - *lines,* which correspond to links
- Record types are organized in the form of a *rooted tree.*
  - No cycles in the underlying graph.
  - Relationships formed in the graph must be such that only one-to-many or one-to-one relationships exist between a parent and a child.

Database schema is represented as a collection of tree-structure diagrams.

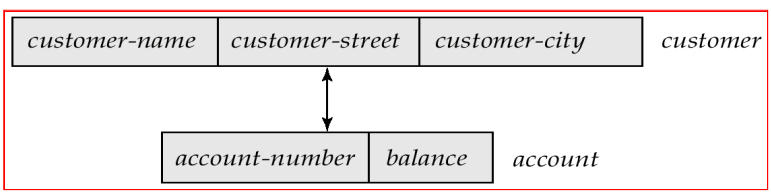
- *single* instance of a database tree
- The root of this tree is a dummy node
- The children of that node are actual instances of the appropriate record type

When transforming E-R diagrams to corresponding tree-structure diagrams, we must ensure that the resulting diagrams are in the form of rooted trees.

Single Relationships:

- Example of E-R diagram with two entity sets, *customer* and *account*, related through a binary, one-to-many relationship *depositor.*
- Corresponding tree-structure diagram has
  - the record type *customer* with three fields: *customer-name*, *customer-street*, and *customer-city.*
  - the record type *account* with two fields: *account-number* and *balance*
  - the link *depositor,* with an arrow pointing to *customer*

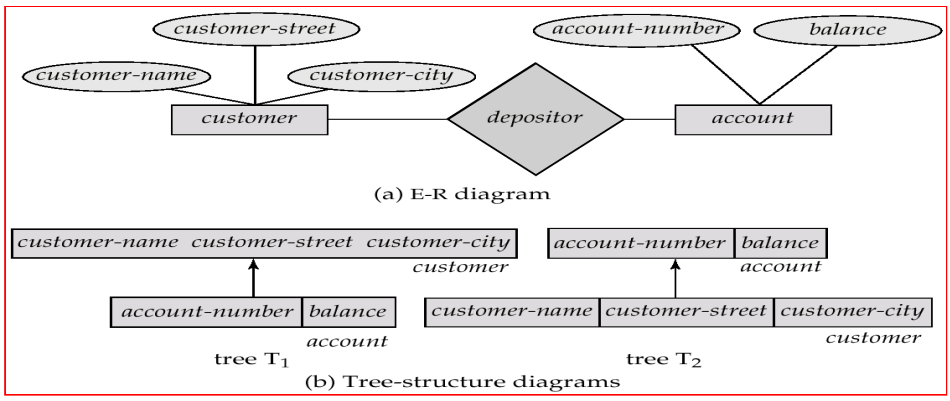
- If the relationship *depositor* is one to one, then the link *depositor* has two arrows.



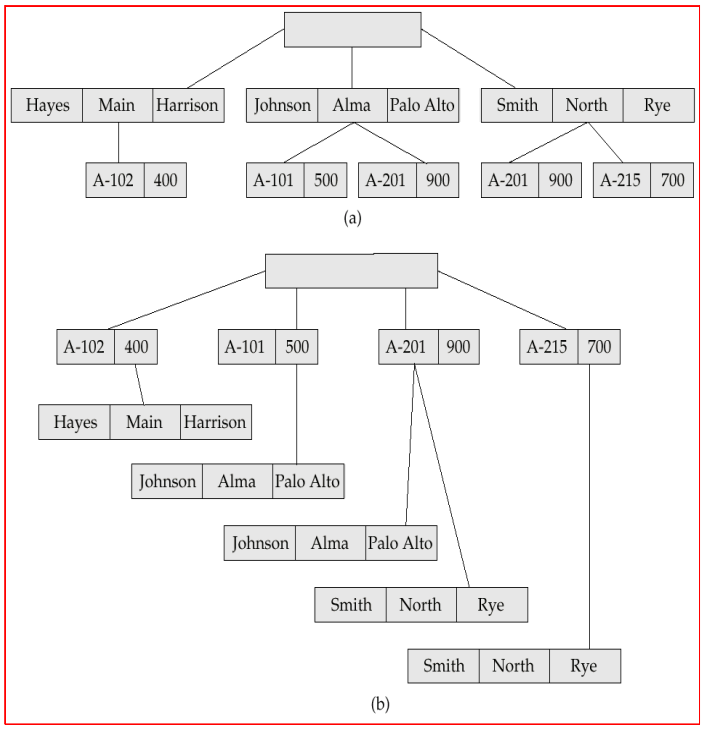
- Only one-to-many and one-to-one relationships can be directly represented in the hierarchical mode.

Transforming Many-To-Many Relationships:

- Must consider the type of queries expected and the degree to which the database schema fits the given E-R diagram.
- In all versions of this transformation, the underlying database tree (or trees) will have replicated records.



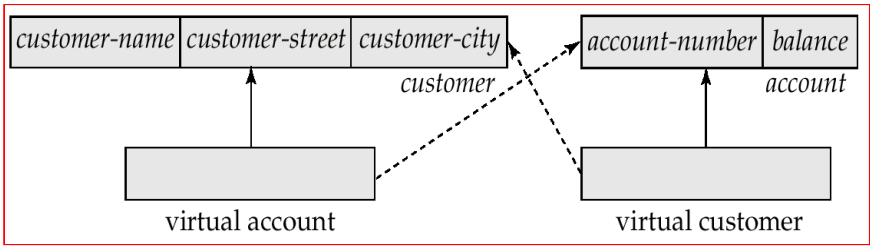
- Create two tree-structure diagrams, *T*1, with the root *customer,* and *T*2, with the root *account.*
- In *T*1, create *depositor,* a many-to-one link from *account* to *customer.*
- In *T*2, create *account-customer,* a many-to-one link from *customer* to *account.*



Virtual Records:

- For many-to-many relationships, record replication is necessary to preserve the tree-structure organization of the database.
- Data inconsistency may result when updating takes place
- Waste of space is unavoidable

- *Virtual record* — contains no data value, only a logical pointer to a particular physical record.
- When a record is to be replicated in several database trees, a single copy of that record is kept in one of the trees and all other records are replaced with a virtual record.
- Let *R* be a record type that is replicated in *T*1, *T*2, . . ., *Tn*. Create a new virtual record type *virtual-R* and replace *R* in each of the *n* – 1 trees with a record of type *virtual-R*.
- Eliminate data replication in the following diagram ; create *virtual-customer* and *virtual-account.*
- Replace *account* with *virtual-account* in the first tree, and replace *customer* with *virtual-customer* in the second tree.
- Add a dashed line from *virtual-customer* to *customer*, and from *virtual-account* to *account*, to specify the association between a virtual record and its corresponding physical record.



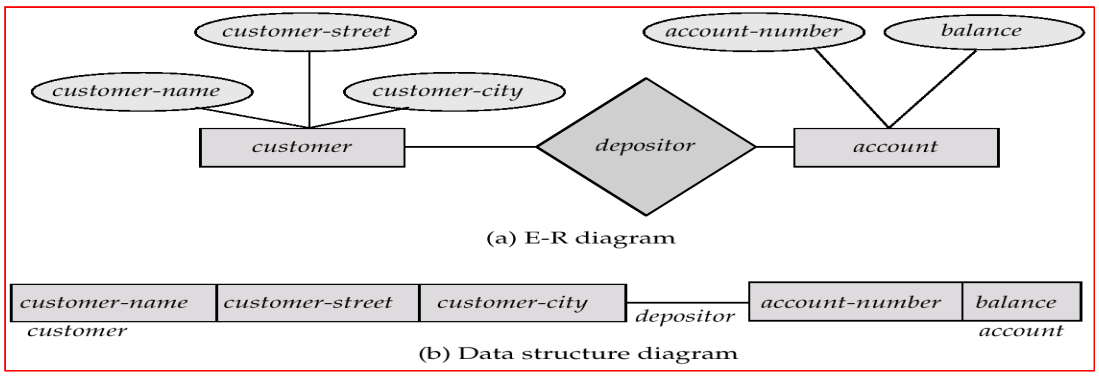
Network Model:

- Data are represented by collections of *records*.
  - similar to an entity in the E-R model
  - Records and their fields are represented as *record type*
- type *customer* = record
  customer-name: string;
  customer-street: string;
  customer-city: string;
  type *account* = record type
  account-number: integer;
  balance: integer;
- end
  end
- Relationships among data are represented by *links*
  - similar to a restricted (binary) form of an E-R relationship
  - restrictions on links depend on whether the relationship is many-to-many, many-to-one, or one-to-one.

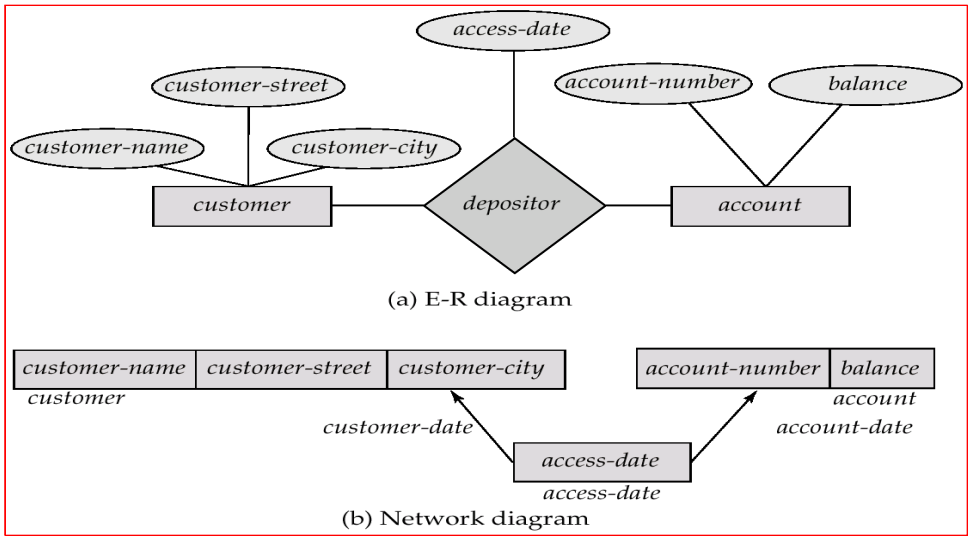
Data-Structure Diagrams:

- Schema representing the design of a network database.
- A data-structure diagram consists of two basic components:
  - Boxes, which correspond to record types.
  - Lines, which correspond to links.
- Specifies the overall logical structure of the database.

For every E-R diagram, there is a corresponding data-structure diagram.



Since a link cannot contain any data value, represent an E-R relationship with attributes with a new record type and links**.**



To represent an E-R relationship of degree 3 or higher, connect the participating record types through a new record type that is linked directly to each of the original record types.

1. Replace entity sets *account*, *customer,* and *branch* with record types *account*, *customer,* and *branch*, respectively.
2. Create a new record type *Rlink* (referred to as a *dummy* record type).
3. Create the following many-to-one links:
    - *CustRlink* from *Rlink* record type to *customer* record type
    - *AcctRlnk* from *Rlink* record type to *account* record type
    - *BrncRlnk* from *Rlink* record type to *branch* record type