CHAPTER-2

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 model 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

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 data base 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 baseid 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(calles 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, for example, can be defined as the entity set customer.

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
eg: customerid,empno
composite attributes are the attributes which can be divided into subparts.
eg: name consisting of first name, middle name, last name
address consisting of city,pincode,state

b) **single-valued and multi-valued attribute:**
The attribute having unique value is single –valued attribute
eg: 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
eg: age which can be derived from (currentdate-birthdate)
experience_in_year can be calculated as (currentdate-joindate)

d) **NULL valued attribute:**
The attribute value which is unknown to user is called NULL valued attribute.

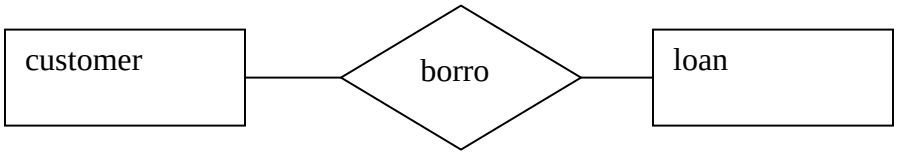
Relationship sets:

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 E1,E2…En are entity sets, then a relation ship set R is a subset of

{(e1,e2,…en)|e1Є E1,e2 Є E2..,en Є En}

where (e1,e2,…en) 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.

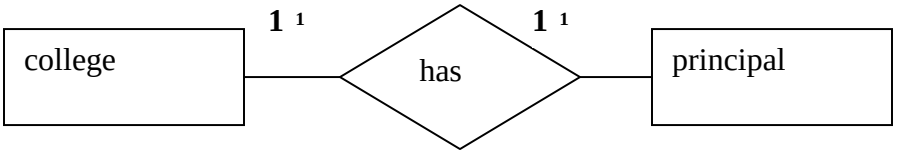
Mapping cardinalities are most useful in describing binary relationship sets, although they can contribute to the description of relationship sets that involve more than two entity sets.

For a binary relationship set R between entity sets A and B, the mapping cardinalities must be one of the following:

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.

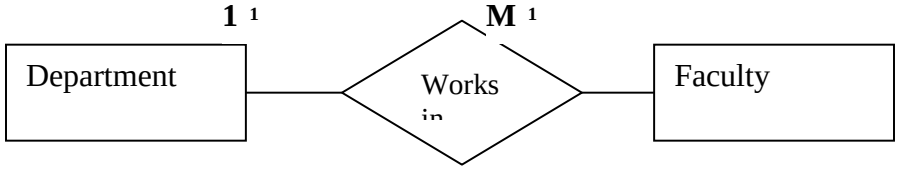
Eg: relationship between college and principal



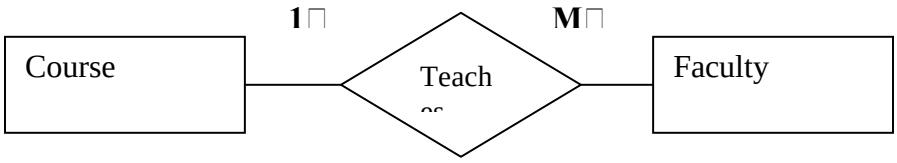
One to many:

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.

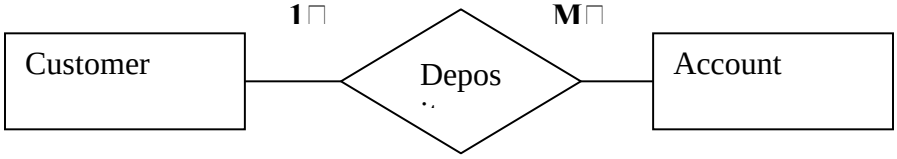
Eg: Relationship between department and faculty



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.



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
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.
Partial:
When it is not necessary for all the entities from an entity set to particapte ion a relationship type, it is called 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 ,birthrate, sex and relationship. Each employee entity set is said to its own the dependent entities that are related to it. How ever, not that the 'dependent' entity does not exist of its own., it is dependent on the employee entity. In other words we can say that in case an employee leaves the organization all dependents related to without the entity 'employee'. Thus it is a weak 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:

- *Uniqueness:* no two distinct tuples in R have the same values for the candidate key
- *Irreducible:* No proper subset of the candidate key has the uniqueness property that is the candidate key.

Eg: (cname,telno)

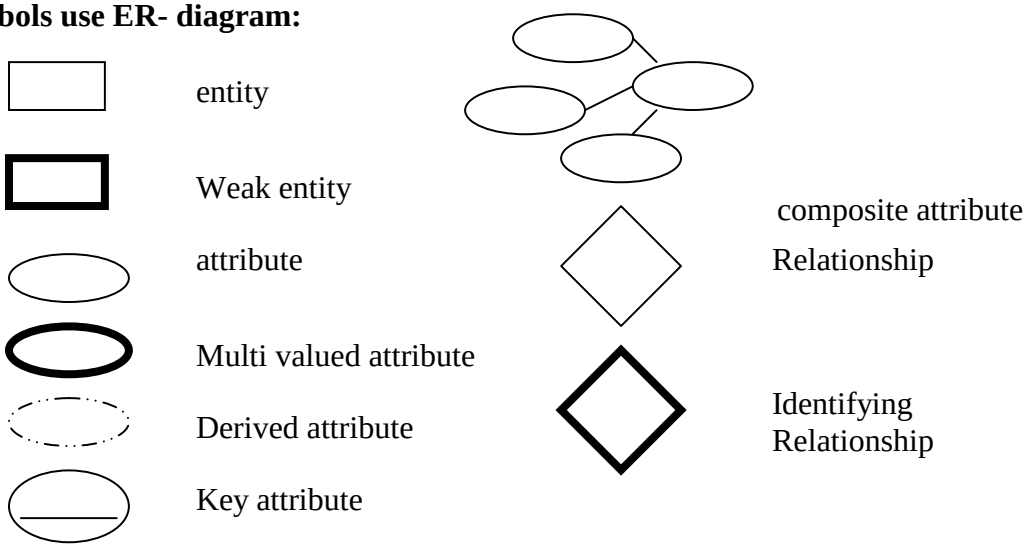
Primary key:

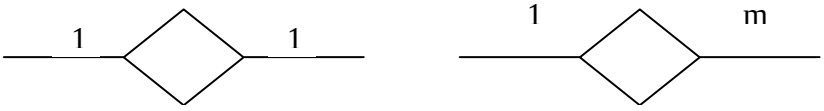
The primary key is the candidate key that is chosen by the database designer as the principal means of identifying entities with in an entity set. The remaining candidate keys if any, are called *alternate key*.

ER-DIAGRAM:

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

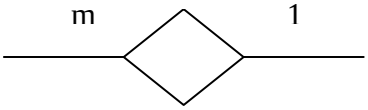
Symbols use ER- diagram:



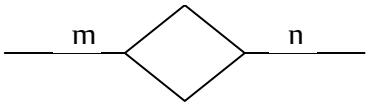


One-to -one

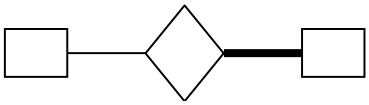
One-to -many



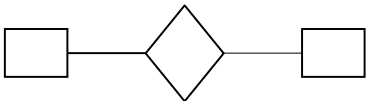
many-to -one



many-to -many



Total participation



Partial participation

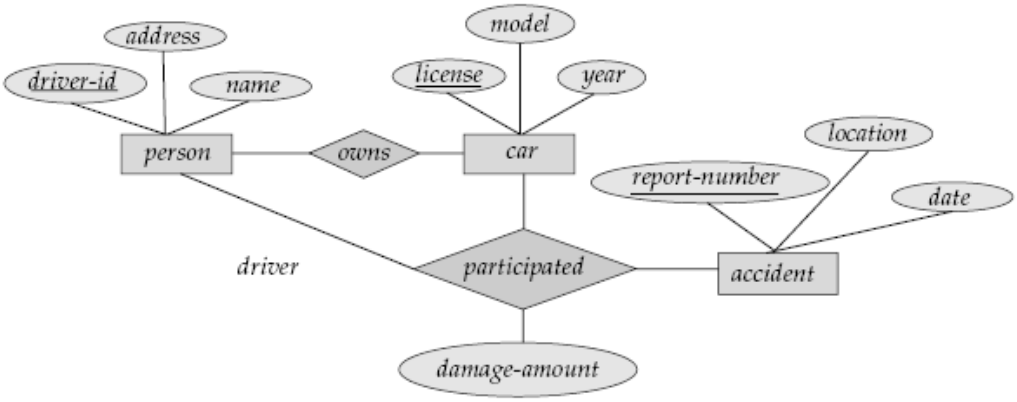


Figure 2.1 E-R diagram for a Car-insurance company.

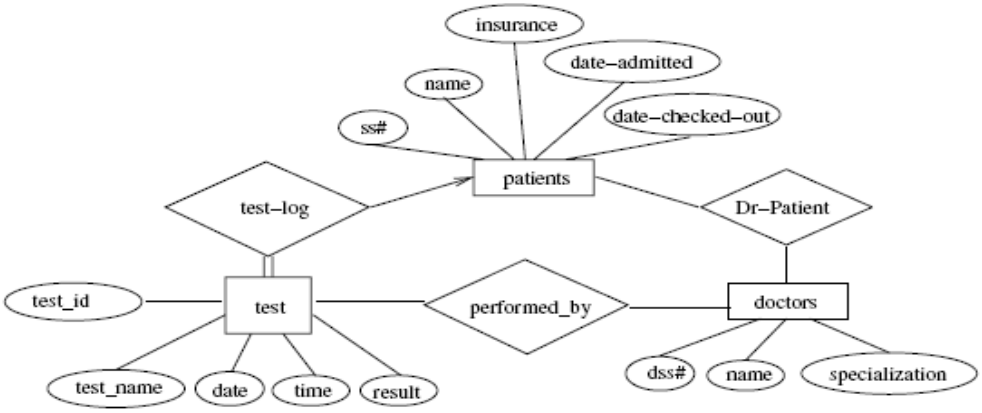


Figure 2.2 E-R diagram for a hospital.

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 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 appropriately modeled.

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

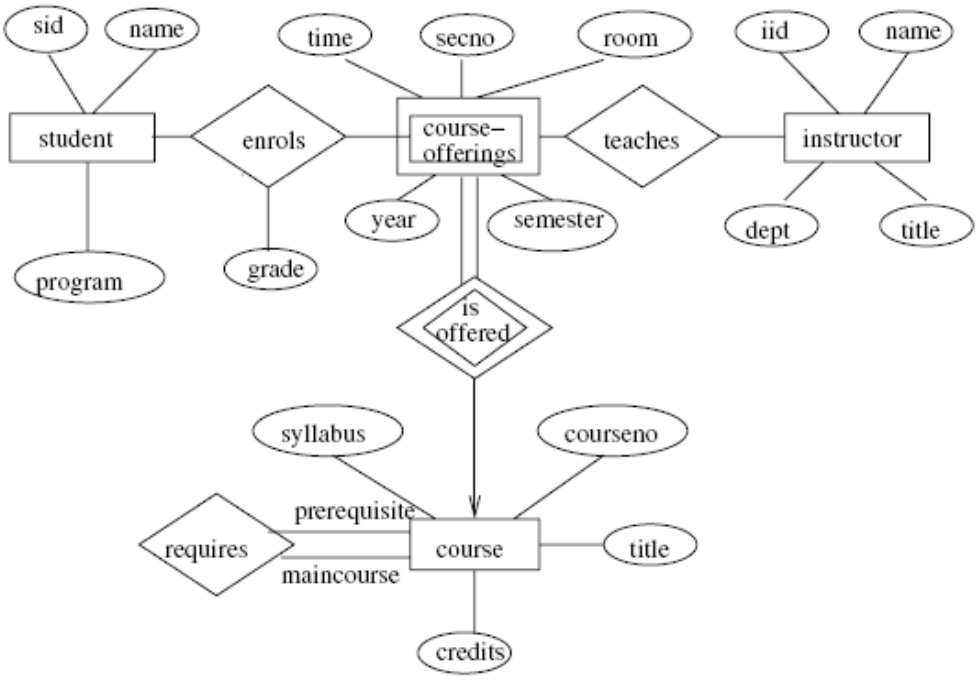


Figure 2.3 E-R diagram for a university.

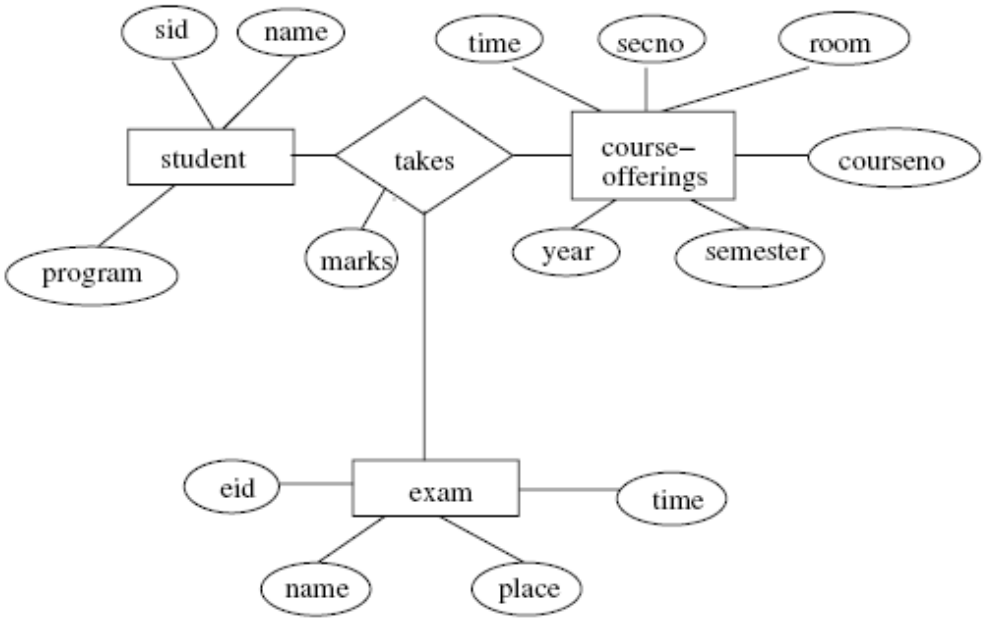


Figure 2.4 E-R diagram for marks database.

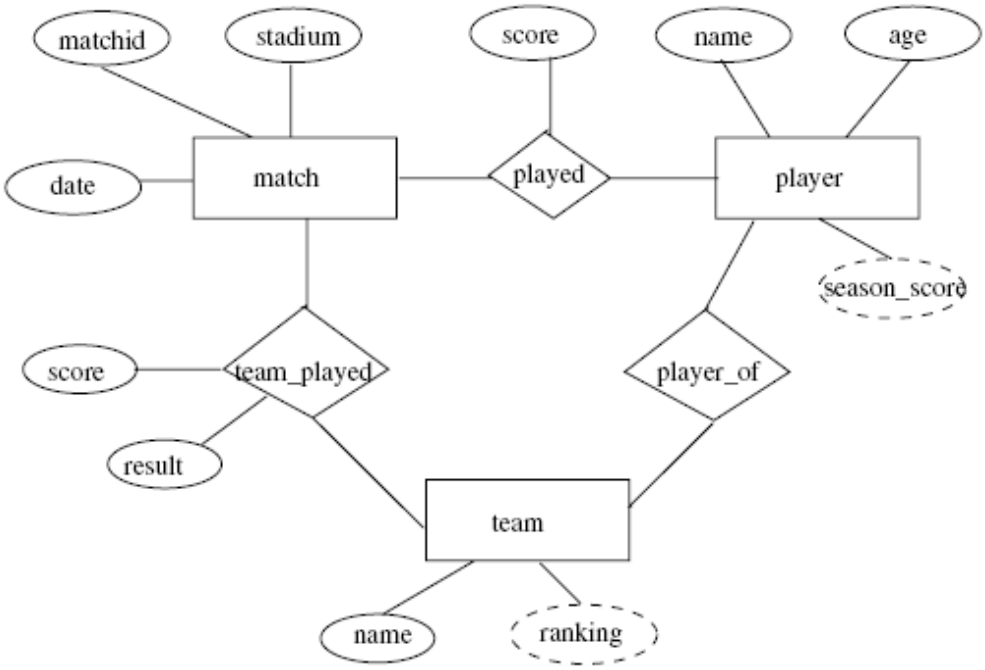


Figure 2.7 E-R diagram for all teams statistics.

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*

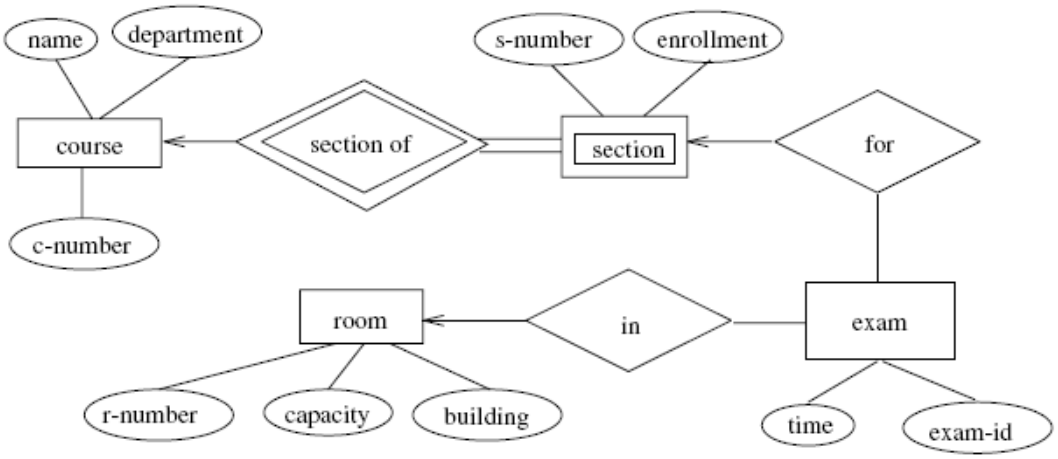
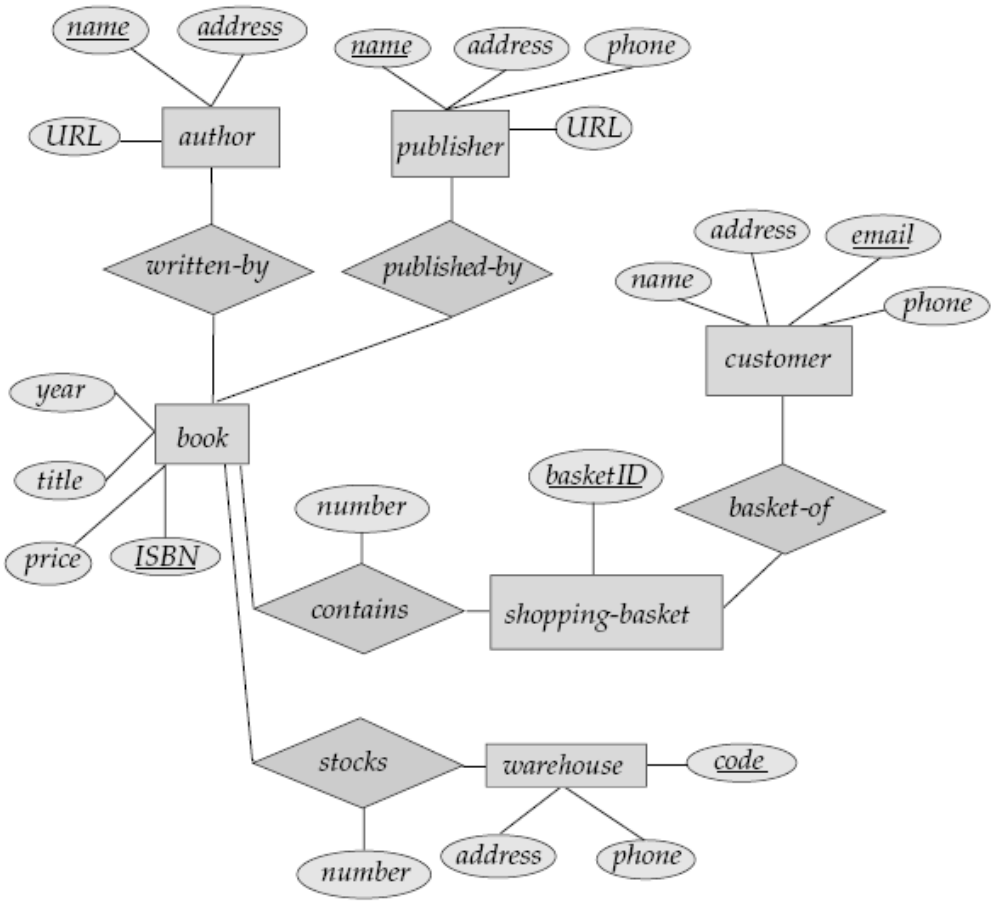


Figure 2.12 E-R diagram for exam scheduling.



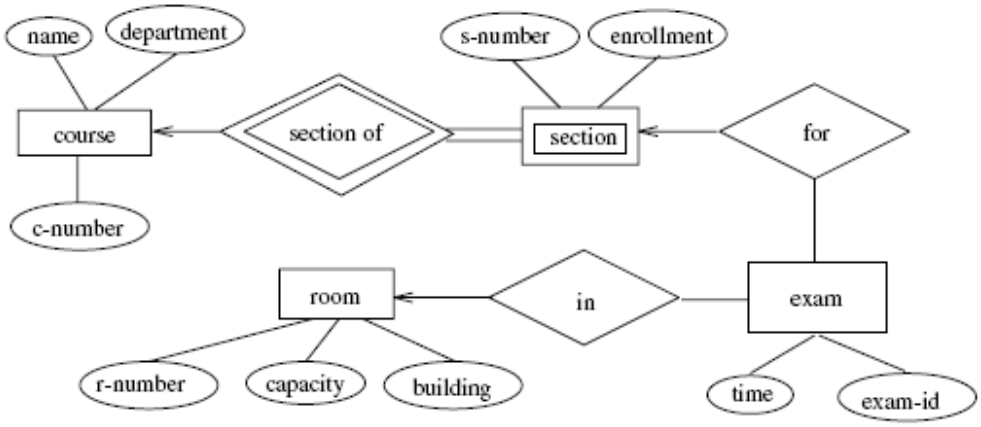


Figure 2.12 E-R diagram for exam scheduling.

Advanced ER-diagram:

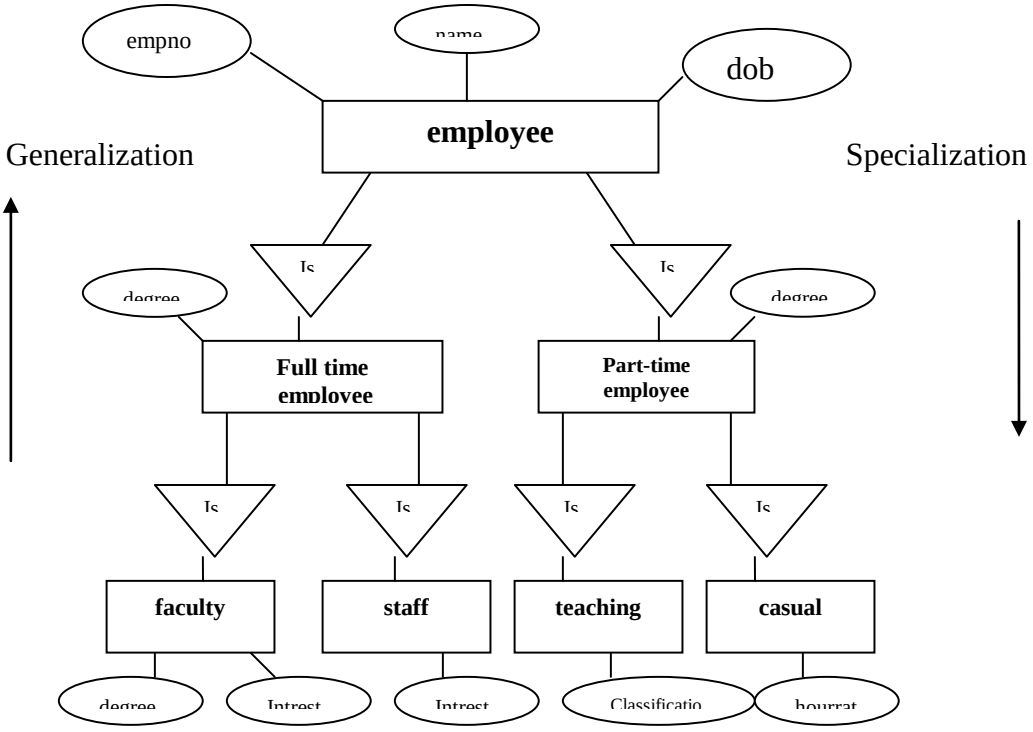
Abstraction is the simplification mechanism used to hide superfluous details of a set of objects. It allows one to concentrate on the properties that are of interest to the application.

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)

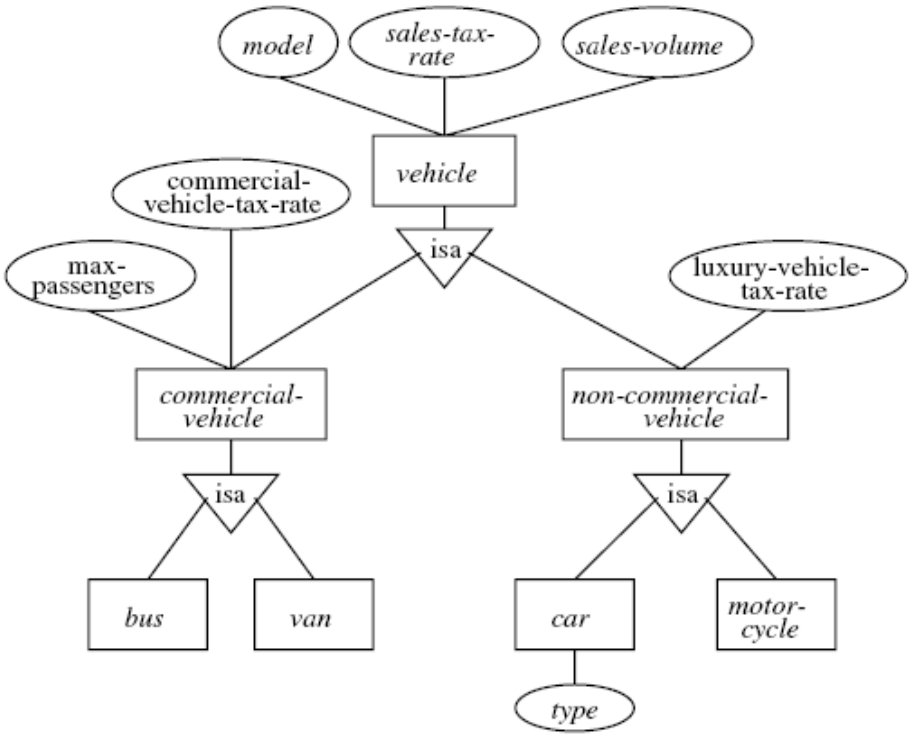
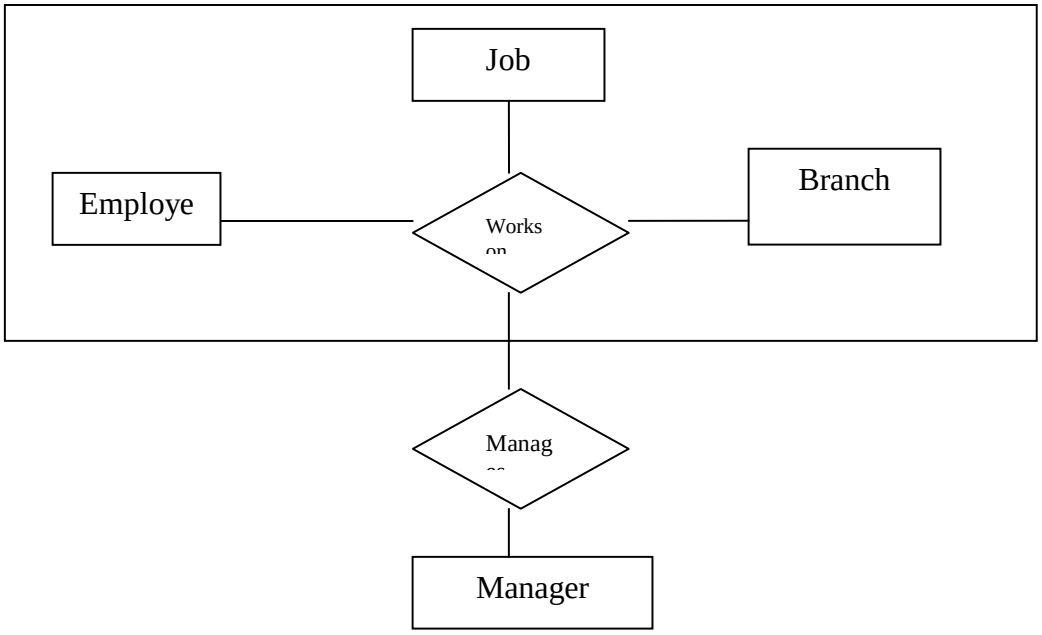


Figure 2.19 E-R diagram of motor-vehicle sales company.

Aggregation:

Aggregation is the process of compiling information on an object, there by abstracting a higher level object. In this manner, 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.


Job
Employe
Works
Branch
Manag
Manager

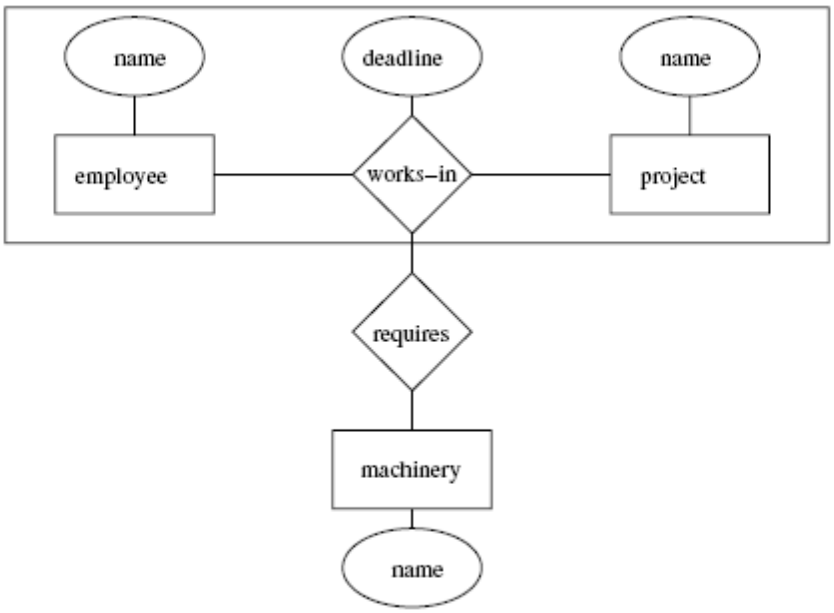


Figure 2.8 E-R diagram Example 1 of aggregation.

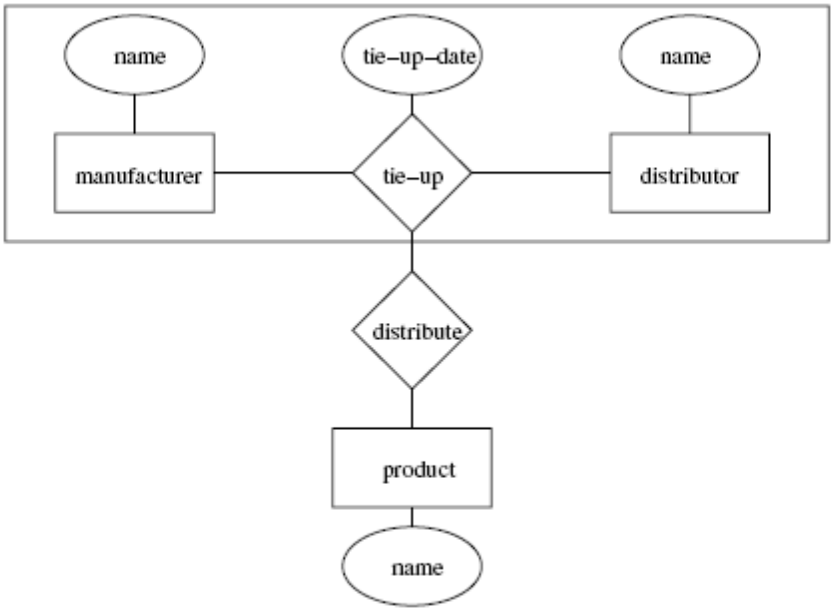
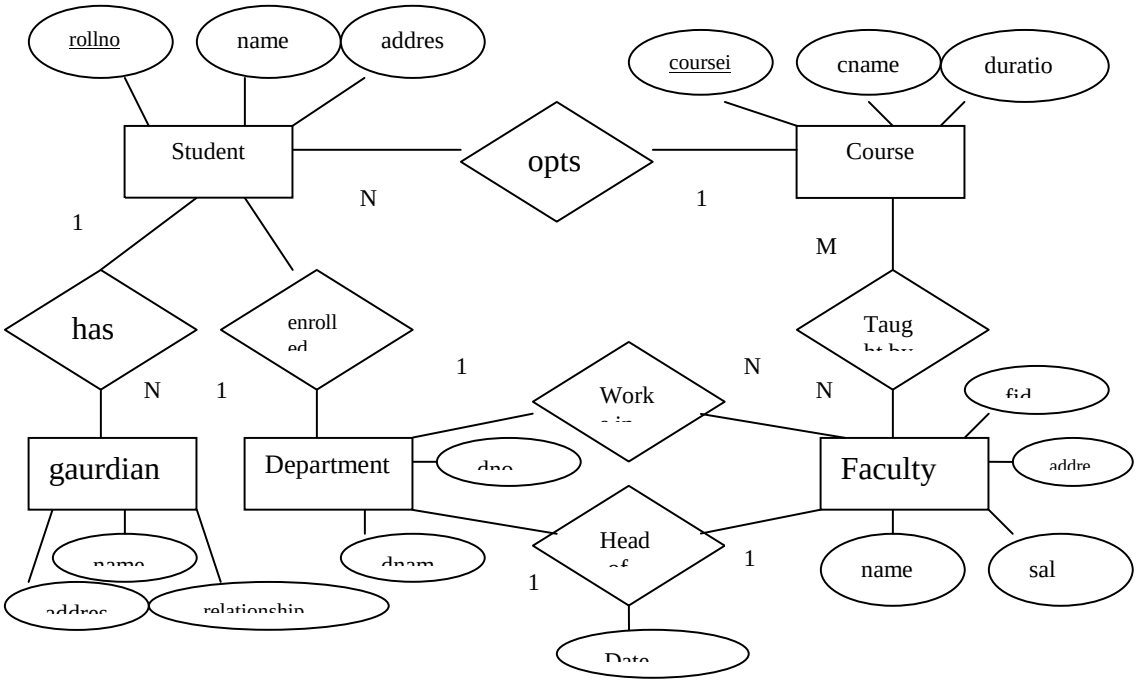


Figure 2.9 E-R diagram Example 2 of aggregation.

Explain the distinctions among the terms primary key, candidate key, and superkey.

Answer: A *superkey* is a set of one or more attributes that, taken collectively, allows us to identify uniquely an entity in the entity set. A superkey may contain extraneous attributes. If *K* is a superkey, then so is any superset of *K*. A superkey for which no proper subset is also a superkey is called a *candidate key*. It is possible that several distinct sets of attributes could serve as candidate keys. The *primary key* is one of the candidate keys that is chosen by the database designer as the principal means of identifying entities within an entity set.

ER- Diagram For College Database



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 he 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, ie, DNO as a foreign key attribute in FACULTY.

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

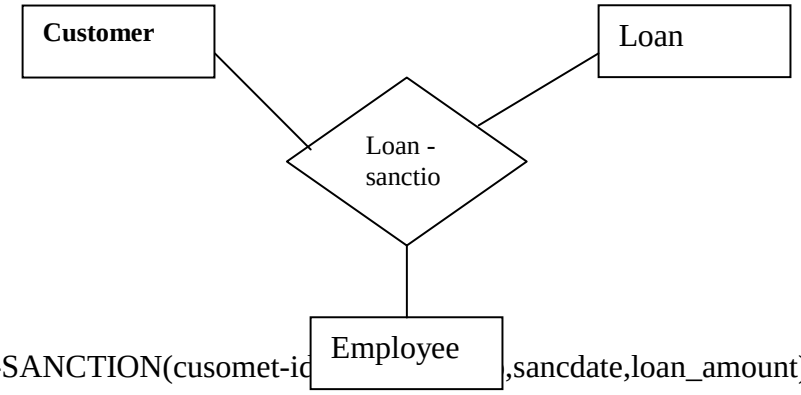
- **Many-to-many relationship:**

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 shod 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:**

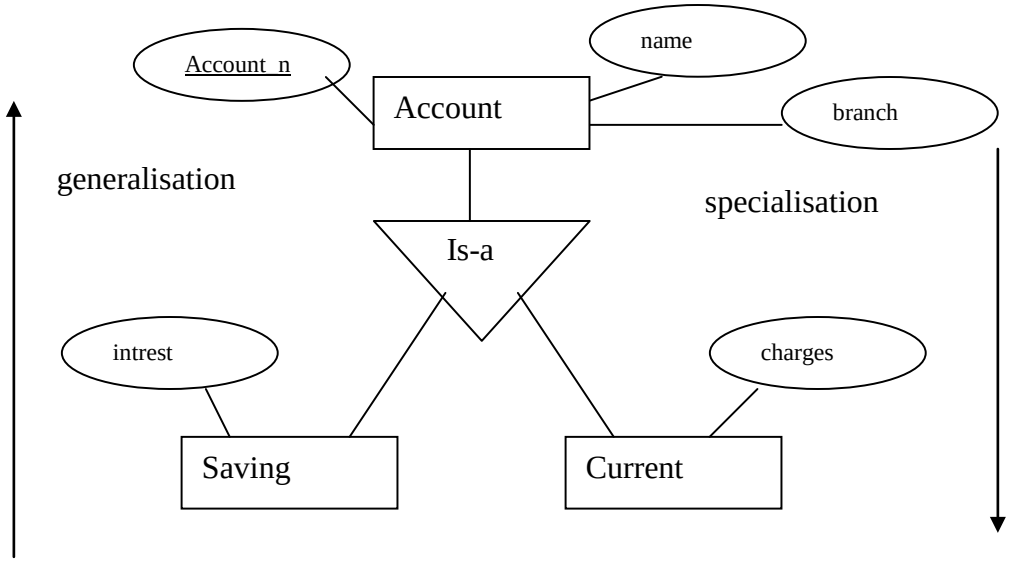
For each n-anry relationship type R where n>2, we create a new table S to represent R, We include as foreign key attributes in s the primary keys of the relations that represent the participating entity types. We also include any simple attributes of the n-ary relationship type(or simple components of 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(cusomet-id ,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 numer is a multivalued attribute the 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 two table as:

Account(account_no,name,branch,balance)
Saving account(account-no,intrest)
Current_account(account-no,charges)

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.

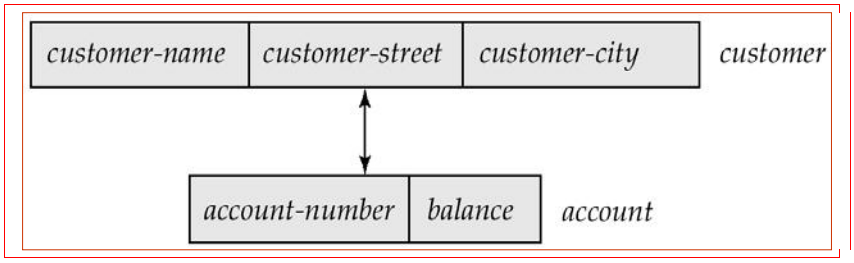
- *single* instance of a database tree
- The root of this tree is a dummy node
- The children of that node are actual instances of the appropriate record type

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 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*

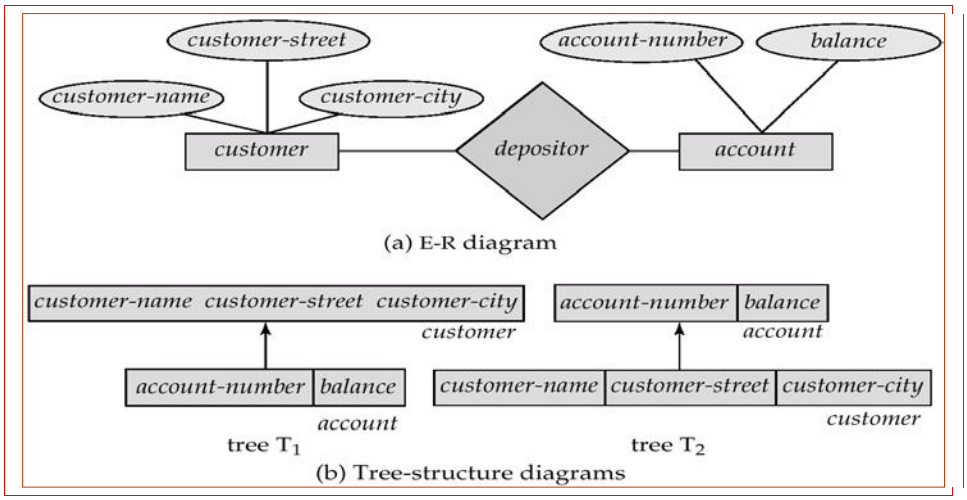
- If the relationship *depositor* is one to one, then the link *depositor* has two arrows.



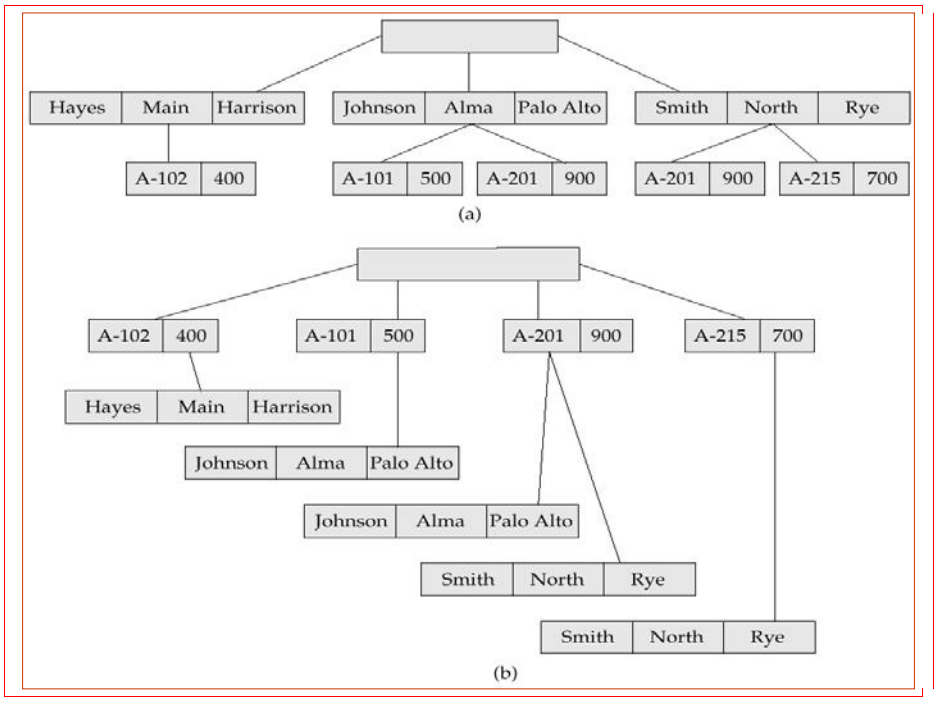
- Only one-to-many and one-to-one relationships can be directly represented in the hierarchical mode.

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.

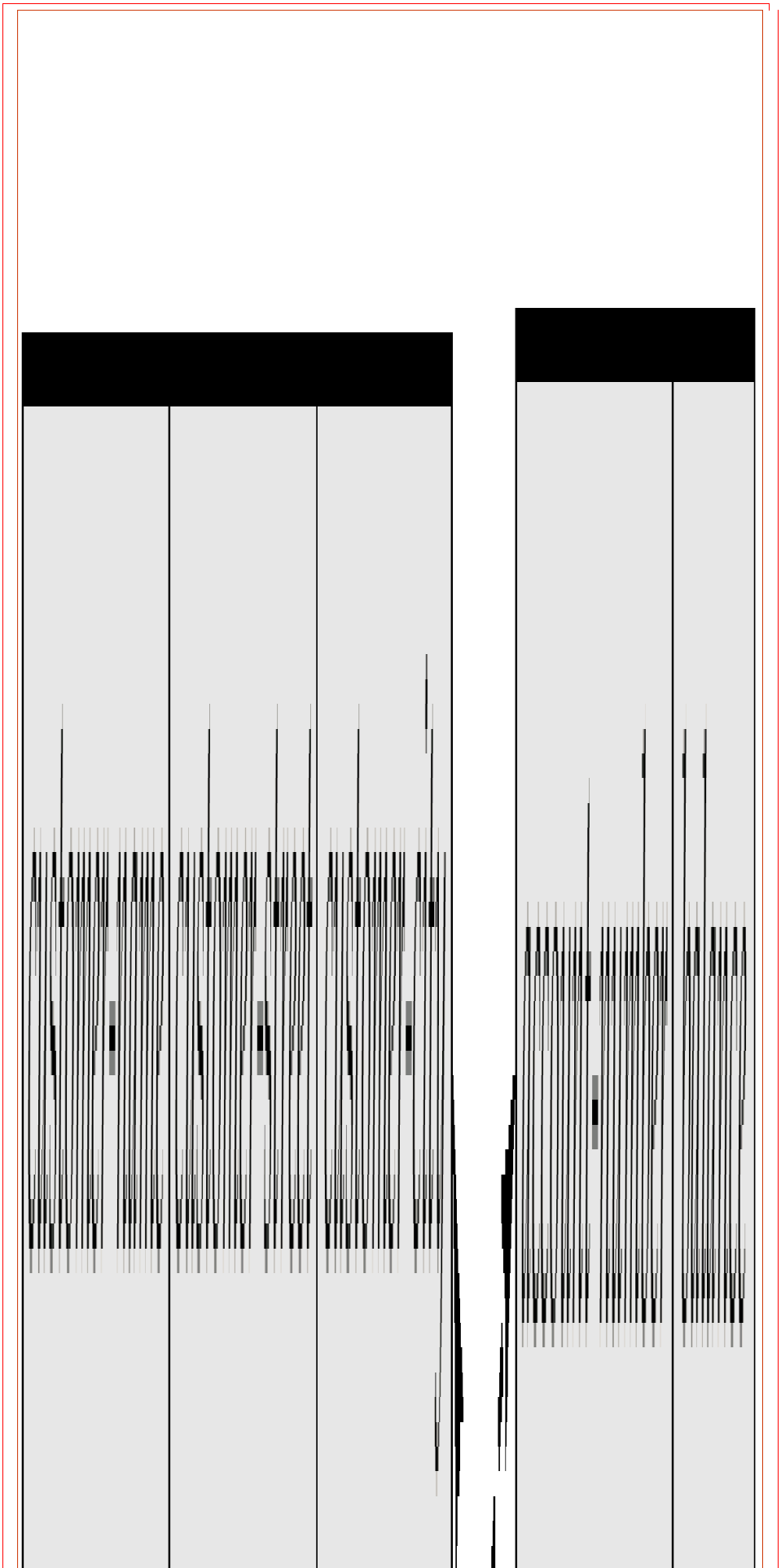


- Create two tree-structure diagrams, *T*1, with the root *customer*, and *T*2, with the root *account.*
- In *T*1, create *depositor*, a many-to-one link from *account* to *customer.*
- In *T*2, create *account-customer*, a many-to-one link from *customer* to *account.*



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 diagram shown on page B.11; 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.



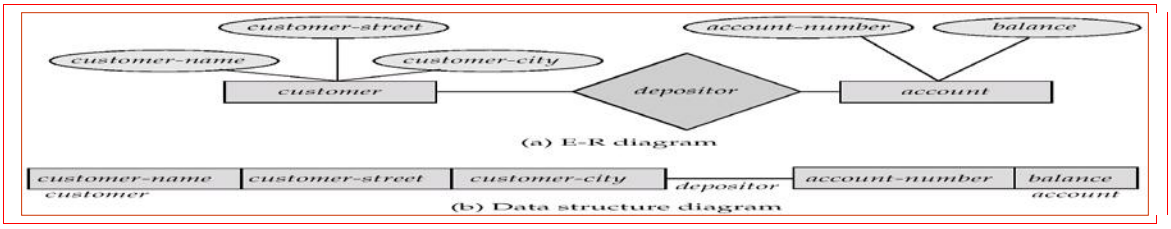
Network Model:

- 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-many, many-to-one, or one-to-one.

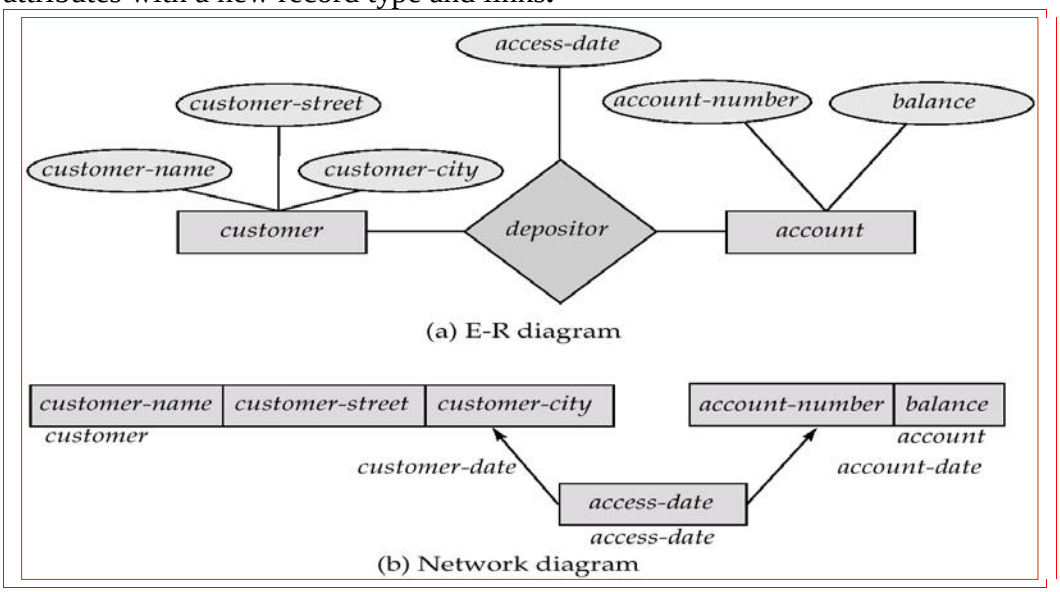
Data-Structure Diagrams:

- 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.

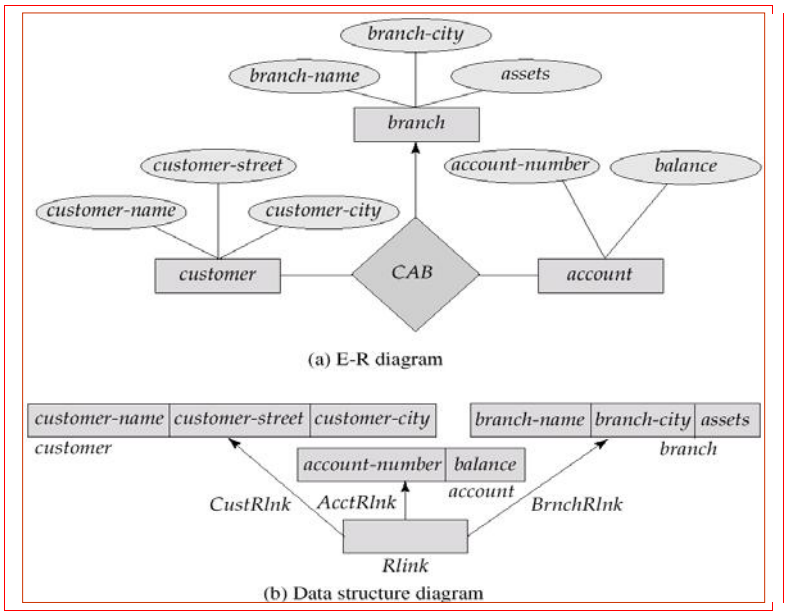


Since a link cannot contain any data value, represent an E-R relationship with attributes with a new record type and links.



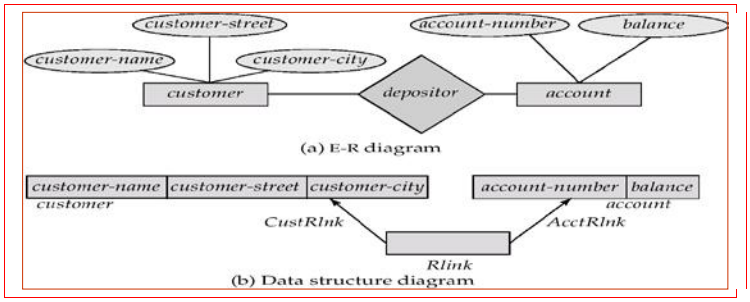
To represent an E-R relationship of degree 3 or higher, connect the participating record types through a new record type that is linked directly to each of the original record types.

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



The DBTG CODASYL Model:

- All links are treated as many-to-one relationships.
- To model many-to-many relationships, a record type is defined to represent the relationship and two links are used.



DBTG Sets:

- The structure consisting of two record types that are linked together is referred to in the DBTG model as a *DBTG set*
- In each DBTG set, one record type is designated as the *owner*, and the other is designated as the *member*, of the set.
- Each DBTG set can have any number of *set occurrences* (actual instances of linked records).
- Since many-to-many links are disallowed, each set occurrence has precisely one owner, and has zero or more member records.
- No member record of a set can participate in more than one occurrence of the set at any point.
- A member record can participate simultaneously in several set occurrences of *different* DBTG sets.