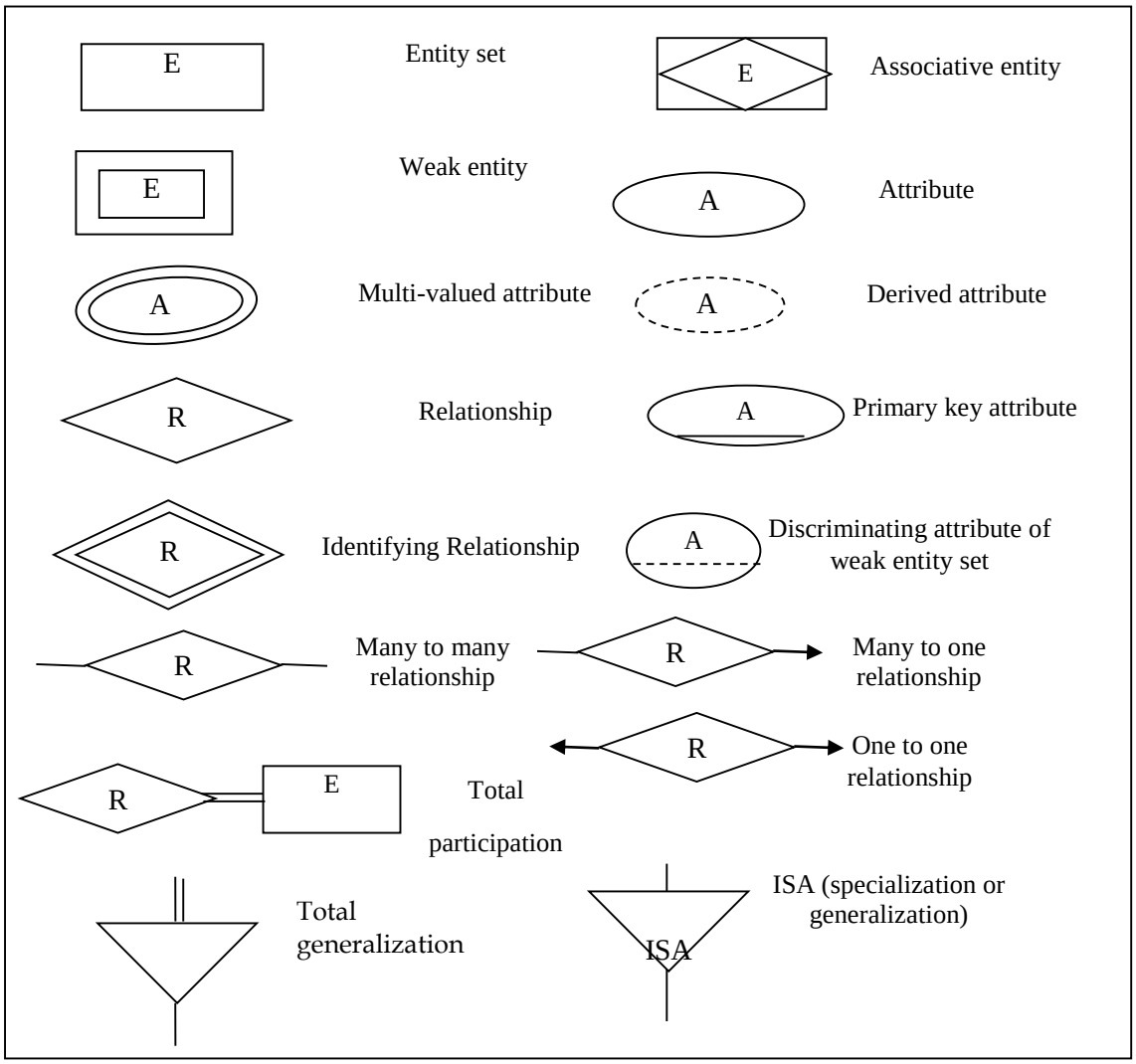
Unit 2

The Entity Relationship, Extended Entity Relationship Model And Object Model

E-R Model

It is developed to facilitate database design by allowing the specification of an enterprise schema, which represents overall logical structure of database. E-R model is useful in mapping the meanings and interactions of real word objects. Basic objects called entities. Relationship among objects called relationship. E-R model keep the record of entities, their attributes and relationship among those entities.

Symbols Used in ER Diagram



E-R data model consists of three basic notions.

a. Entity sets
b. Relationship sets
c. Attributes

Entity Sets

An entry set is a set of entities of the same type that share the same properties, or attributes for example the set of all persons who are customer at a given bank can be defined as the entity set customer, loan etc.

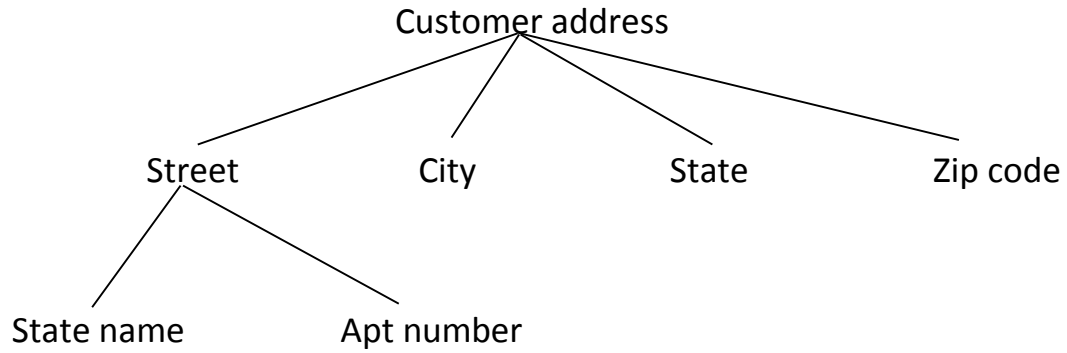
Attributes

An entity is represented by set of characteristics called attributes. Each attributes has set of values called domain. E.g. possible attributes of loan entity set has loan number and loan amount similarly the domain of attribute might be a set of a positive integers.

Types of attributes

1.Simple Vs. Composite:

Simple attributes are these they are not divided into sub parts. Composite attributes can be divided into subpart. A composite attribute is made of one or more simple or composite attributes. E.g. name is made of first name, middle name and last name and where name is composite attribute and first name, middle name and last name are simple attribute. It may come in hierarchy.



2. Single valued Vs. multivalued

A single valued attribute is described by one value e.g. age of a person. A multivalued attribute is described by many values e.g. color attribute of a multicolored bird.

3. Stored Vs. Derived

An attribute which value can be derived from the value of other attribute is called a derived attribute e.g. age can be derived from date of birth and current date. If this is not a case then called stored values e.g. roll no.

4. Null attributes

Attributes that do not have any applicable values are said to have null values. They are also called unknown and missing values. For example, if a social security value of a particular customer is null, we assume, the value is missing.

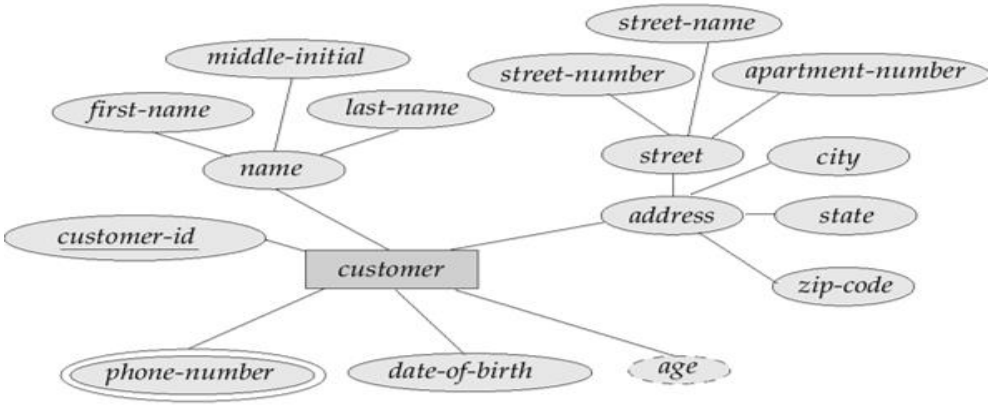
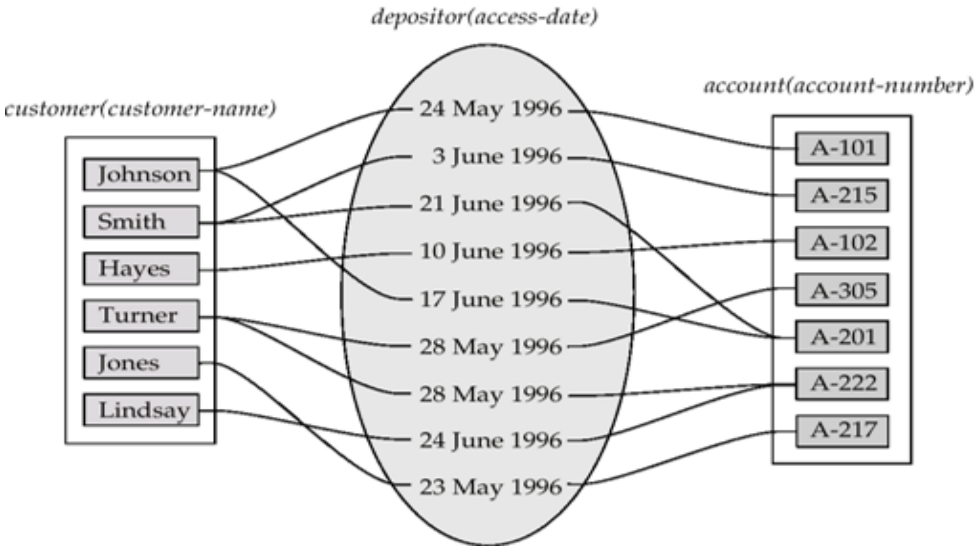


Fig: E-R diagram with *composite*, *multivalued*, and *derived* attributes

Descriptive Attributes

A relationship set may also have attributes called descriptive attributes. For example, the depositor relationship set between entity sets customer and account may have the attribute access-date. See in fig below. A relationship instance in a given relationship set must be uniquely identifiable from other relationship instances, without using descriptive attributes.



Relationship sets

A relationship is an association between several entities. A relationship set is a set of relationships of the same type. Mathematically For non-distinct entity set n ≥ 2. If E1, E2… En are entity sets then relationship set R is the subset of {(e1, e2, e3… en) / e1 ∈ E1, e2∈E2……en∈En}.

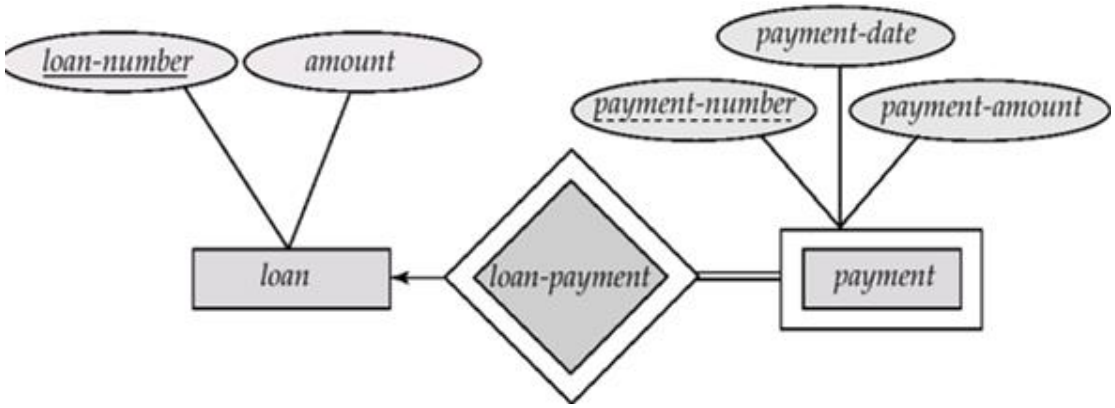
Strong and Weak Entity Set

An entity set may not have sufficient attributes to form a primary key. Such an entity set is termed as a **weak entity set**. An entity set that has a primary key is termed as a **strong entity set**.

For a weak entity set to be meaningful, it must be associated with another entity set, called the identifying or owner entity set, using one of the key attribute of owner entity set. The weak entity set is said to be existence dependent on the identifying entity set. The relationship associating the weak entity set with the identifying entity set is called the identifying relationship. The identifying relationship is many-to-one form the weak entity set to the identifying entity set, and the participation of the weak entity set in the relationship set is total.

Although a weak entity set does not have a primary key, we use discriminator (or partial key) as a set of attributes that allows the distinction to be made among all the entities in the weak entity set.

In the figure below, payment-number is partial key and (loan-number, payment-number) is primary key for payment entity set.



Constraints on ER Model

Relationship sets in ER model usually have certain constraints that limit the possible combinations of entities that may involve in the corresponding relationship set. Database content must confirm these constraints. The most important constraints are: mapping cardinalities and participation constraints.

Mapping Cardinality Constraints

ER model constraint that describes maximum number of possible relationship occurrences for an entity set participating in a given relationship type is called mapping cardinality. It is also termed as cardinality ratio. On the basis of cardinality ratio, relationships can be categorized into: One-to-One, One-to- Many, Many-to-One, and Many-to-Many. We express cardinality constraints by drawing either a directed line (→), signifying "one," or an undirected line (—), signifying "many," between the relationship set and the entity set.

1. One-to-One Relationship

If every entity in A is associated with at most one entity in B and vice-versa then the relationship is called one-to-one relationship. The following figure shows one to one mapping cardinality between entity sets A and B. For example every bank has only one CEO and a person can be CEO of only one bank therefore it shows one-to-one relationship between Bank and CEO.

2. One-to-Many Relationship

If an entity in A can be associated with any number (zero or more) of entities in B but every entity in B can be associated with at most one entity in A, and then it is called one-to-many relationship. For example, a mother can have any number of children but children can have only one mother therefore it shows one-to-many relationship between mother and child.

3. Many-to-One Relationship

If very entity in A can be associated only one of entities in B but an entity in B can be associated with any number of entities in A, then it is called many-to-one relationship. For example, a Book is always published by only one publisher but a publisher can publish any number of books therefore it shows many-to-one relationship between books and publication.

4. Many-to-Many Relationship

If an entity in A can be associated with any number of entities in B and vice versa then it is called many-to-many relationship. For example, a student can enroll into more than one subject and a subject can be enrolled by many students therefore it shows many-to-many relationship between students and courses.

Participation Constraints

Constraint on ER model that determines whether all or only some entity occurrences participate in a relationship is called participation constraint. It specifies whether the existence of an entity depends on its being related to another entity via the relationship type. There are two types of participation constraints:

- Total Participation Constraints and
- Partial Participation Constraints.

The participation of an entity set A in a relationship set R is said to be **total** if every entity in A participates in relationship at least once.

On the other hand, the participation of an entity set A in a relationship set R is said to be **partial** if only some of the members of an entity set A participate in relationship.

Total participation and partial participation is denoted by single line and double line in ER diagrams respectively.

For example, consider Customer and Loan entity sets in a banking system, and a relationship set borrower between them indicates that only some of the customers have Loan but every Loan should be associated with some customer. Therefore, there is total participation of entity set Loan in the relationship set borrower but participation of entity set customer is partial in relationship set borrower. Here, Loan entity set cannot exist without Customer entity set but existence of Customer entity set is independent of Loan entity set.

Keys

Set of one or more attributes whose values are distinct for each individual entity in the entity set is called key, and its values can be used to identify each entity uniquely. There are different types of keys which are:

- Super key
- Candidate key
- Primary key
- Composite key
- Foreign key

Super Key

A supper key is a set of one or more attributes allow us to identify uniquely in entity set. E.g. social security number attribute of a entity set customer is distinguish from one customer entity to another. Similarly, customer name and social-security is a supper key for an entity set customer. The customer name of entity customer is not super key because several people might have the same name.

Candidate key

A candidate key of an entity set is a minimal super key. That is a super key which does not have any proper subset is called candidate key. For example, student-id is candidate key of the entity set student but set of attributes {roll-number, name, program, semester, section} is not candidate key of the entity set student because it has proper subset {roll-number, program, semester section} which is also key. All candidate keys are super keys but vice versa is not true. Any candidate key other than the one chosen as a primary key is known as alternate key.

Primary key

A primary key is a candidate key that is chosen by the database designer as the principle means of uniquely identifying entities within an entity set. There may exist several candidate keys, one of the candidate keys is selected to be the primary key. For example, entity set student have two

candidate keys: student-id, and {roll-number, program, semester section}, if database designer chooses student-id for the purpose of uniquely identifying entities within entity set then it becomes primary key. Primary key must satisfy following two characteristics:

- It cannot be null
- It cannot be duplicate

Composite Key

If a primary key contains more than one attribute, then it is called composite key. For example, if database designer chooses student-id as primary key then it not composite key but if database designer chooses {roll-number, program, semester section} as primary key then it is also called composite key.

Foreign key

A foreign key (FK) is an attribute or combination of attributes that is used to establish and enforce relationship between two relations (table). A set of attributes that references primary key of another table is called foreign key. For example, if a student enrolls in program then program-id (primary key of relation program) can be used as foreign key in student relation,

Student

S-ID	Name	Address	Program-ID
S-12	Pawan	Joshi	C002
S-14	Yamman	Karki	C021
S-51	Abin	Saud	C321
S-11	Binak	Singh	C112

Foreign Keys

Relationships

Primary Keys

Program

Program-ID	Program-Name
C002	BBA
C021	B. Sc CSIT
C112	BIM
C321	B. ed.

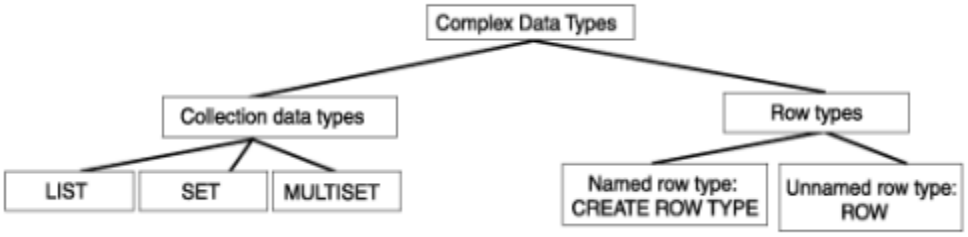
Fig: Primary key and foreign key

Complex Data Types

Any data that does not fall into the traditional field structure (alpha, numeric, dates) of a relational DBMS are called complex data. Examples of complex data types are bills of materials, word processing documents, maps, time-series, images and video. In a relational DBMS, complex data types are stored in a large object (LOB), but either the client application or some middleware is

required to process the data. In an object DBMS or an object-relational DBMS, complex data types are stored as objects that are integrated into and activated by the DBMS.

Complex data types are those that include record type and collections which are used to handle data in record format or in an array format. A complex data type is usually a composite of other existing data types. For example, you might create a complex data type whose components include built-in types, opaque types, distinct types, or other complex types. An important advantage that complex data types have over user-defined types is that users can access and manipulate the individual components of a complex data type.



Extended E-R model (EER model)

The ER modeling concepts are not sufficient for representing new database applications, which have more complex requirements than do the more traditional applications. To model modern complex systems such as, such as databases for engineering design and manufacturing (CAD/CAM), telecommunications, complex software systems, and Geographic Information Systems (GIS), among many other applications, it is not enough. Thus, modeling of such systems can be made easy with the help of extended ER model. The EER model includes all of the concepts introduced by the ER model. Additionally it includes the concepts of a subclass and super class, along with the concepts of specialization and generalization. There are basically four concepts of EER-Model:

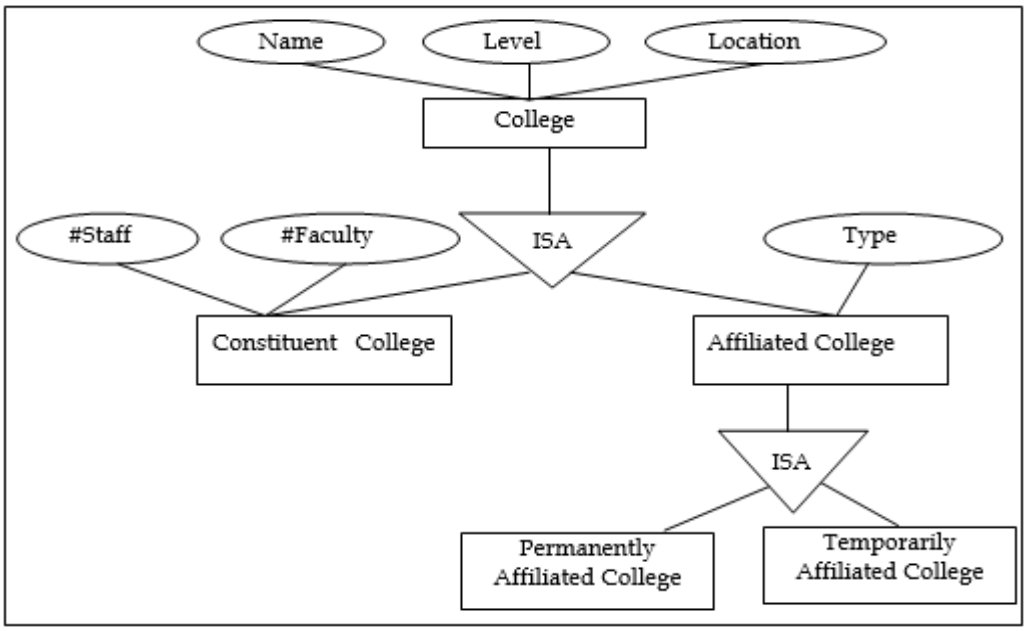
- Subclass/super class relationship
- Specialization and Generalization
- categories (UNION types)
- Aggregation

Subclass/super class relationship

An entity set may have a number of sub-groupings of its entities that are meaningful. Here, the entity set is called superclass while the subgroupings are known as subclasses of the superclass. We call the relationship between a superclass and any one of its subclass a **superclass/subclass** or simply **class/subclass** relationship. An entity that is a member of subclass inherits all attributes

from its superclass. That is, there is type inheritance in subclass. The entity also inherits all the relationships in which the superclass participates.

Consider the example of entity set college. We can divide the entity set college into two subgroups: constituent colleges and affiliated colleges. Further affiliated colleges can be divided into two subgroups: permanently affiliated colleges and temporarily affiliated colleges. Here the entity set college is superclass of subgroups entity sets constituent colleges and affiliated colleges and the subgroups are called subclasses of the superclass entity set college. In ER diagram we can represent superclass/subclass relationship by using ISA triangle as below:



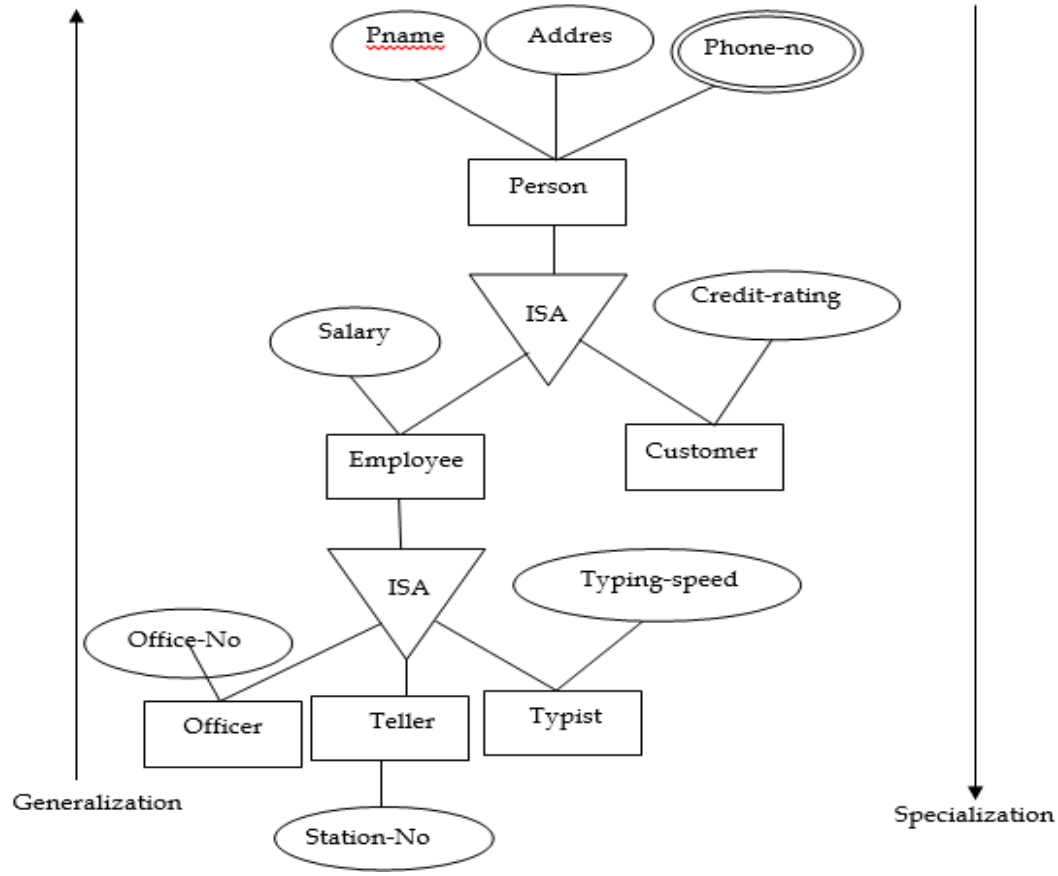
Specialization and Generalization

The process of defining a set of subclasses from a superclass is known as specialization. The set of subclasses is based upon some distinguishing characteristics of the entities in the superclass. It is a top-down design process. For Example, the entity set person can be specialized into entity sets employee, and customer. Further, on the basis of job type the entity set employee can be divided into entity sets manager, teller, typist etc. Thus, specialization may create hierarchy. Attributes of a subclass are called specific attributes. For example, the subclass entity set may have attribute 'typing-speed'.

There may have several specializations of the same superclass. For example, on the basis of pay type the entity set employee can be specialized entity sets "Full-time employees" and "Part-time" employees.

Generalization is the reverse of the specialization. Several classes with common features are generalized into a super class; original classes become its subclasses. It is a bottom-up design process. Here, we combine a number of entity sets that share the same features into a higher-level entity set. Designer applies generalization is to emphasize the similarities among the entity sets

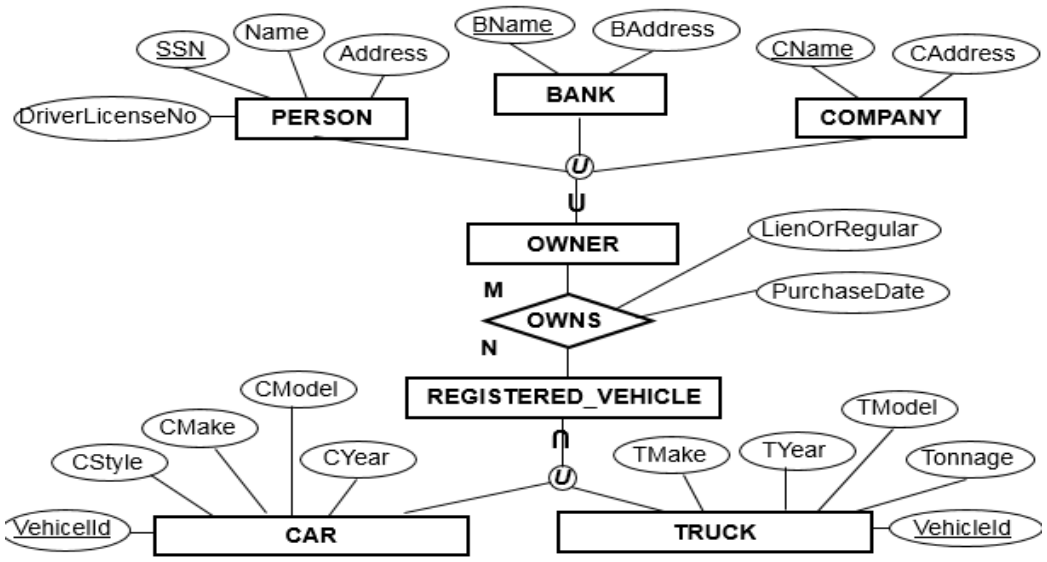
and hide their differences. Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way. The terms specialization and generalization are used interchangeably.



Categories (Union Types)

It is possible that single superclass/subclass relationship has more than one super classes representing different entity types. In this case, the subclass will represent a collection of objects that is the union of distinct entity types; we call such a subclass a union type or a **category**.

A category has two or more super classes that may represent distinct entity types, whereas non-category superclass/subclass relationships always have a single superclass.



Aggregation

Sometimes we have to model a relationship between a collection of entities and relationships. Basic ER model cannot express relationships among relationship sets and relationships between relationship sets and entity sets. Aggregation is an abstraction through which relationship sets are treated as high-level entity sets and can participate in relationship sets. It allows relationships between relationships.

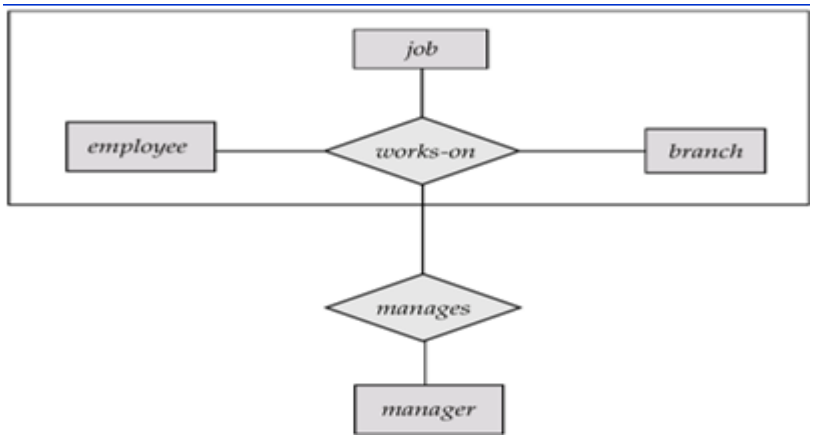


Fig: Aggregation

Constraints on Generalization/Specialization

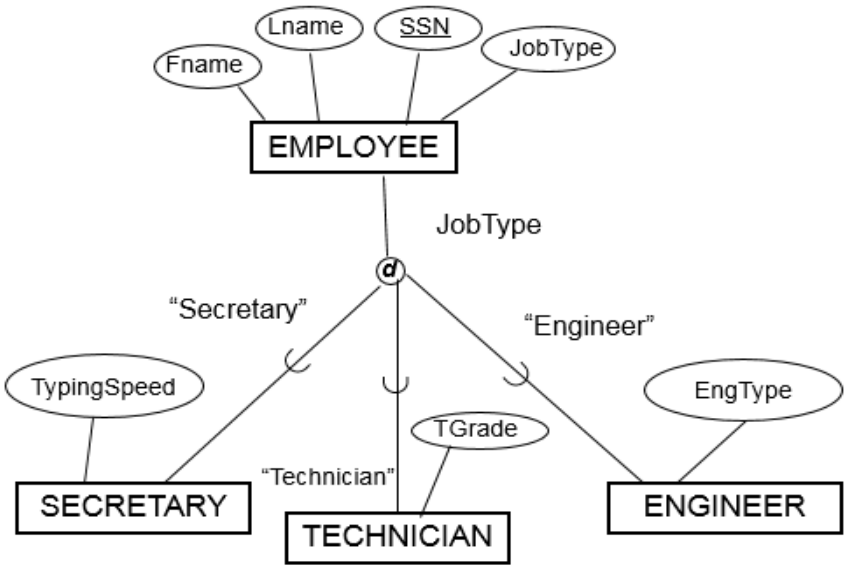
To model real world more accurately by using ER diagram we need to keep certain constraints on it. Constraints on which entities can be members of a given lower-level entity set are discussed below.

- Condition defined constraint
- Disjoint vs. Overlap Constraints

- A total specialization vs. a partial specialization

Condition defined constraint

If we can determine exactly those entities that will become members of each subclass by a condition, the subclasses are called **predicate-defined (or condition-defined**) subclasses. Here, condition is a constraint that determines subclass members. For example, if the EMPLOYEE entity type has an attribute **JobType**, we can specify the condition of membership in the SECRETRY subclass by a condition (JobType="secretary"), which we call defining predicate of the subclass.
We display predicate-defined subclass by writing the predicate condition next to the line that connects the subclass to the specialization circle.
If all subclasses have membership condition on the same attribute of the superclass, then it is called an **attribute defined-subclass**. And, the attribute is called the **defining attribute.**



Disjoint vs. Overlap Constraints

If an entity can be a member of at most one of the subclasses of the specialization, then the subclasses are called disjoint. In EER diagram, d in the circle stands for disjoint. For example, specialized subclasses 'Electronic Book' and 'Paperback Book' are disjoint subclasses of the super class entity set Book.
If the same entity may be a member of more than one subclass of the specialization, then the subclasses are said to be **overlapped**. For example, specialized subclasses 'Action Movie' and 'Comedy Movie' are overlapped subclasses of the super class entity set Movie. This is because there also movies called "Action Comedy" that belong both of the above subclasses.

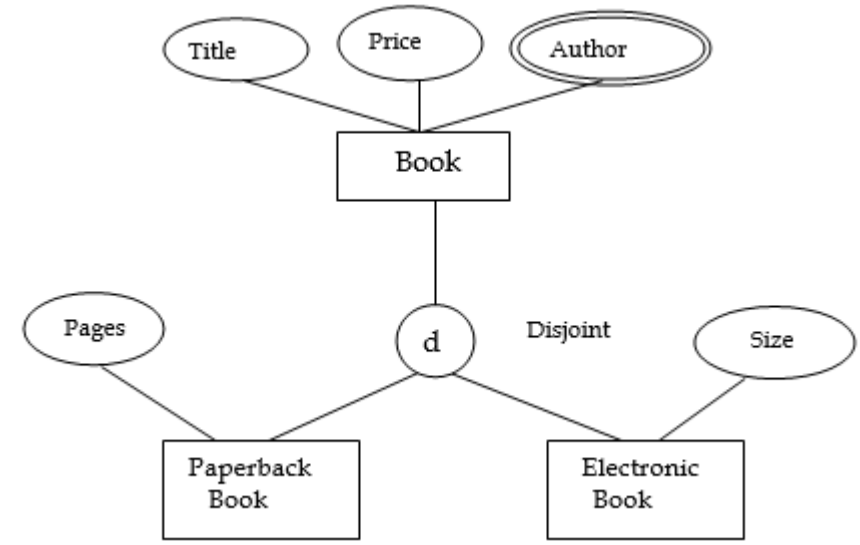


Fig: Disjoint constraint

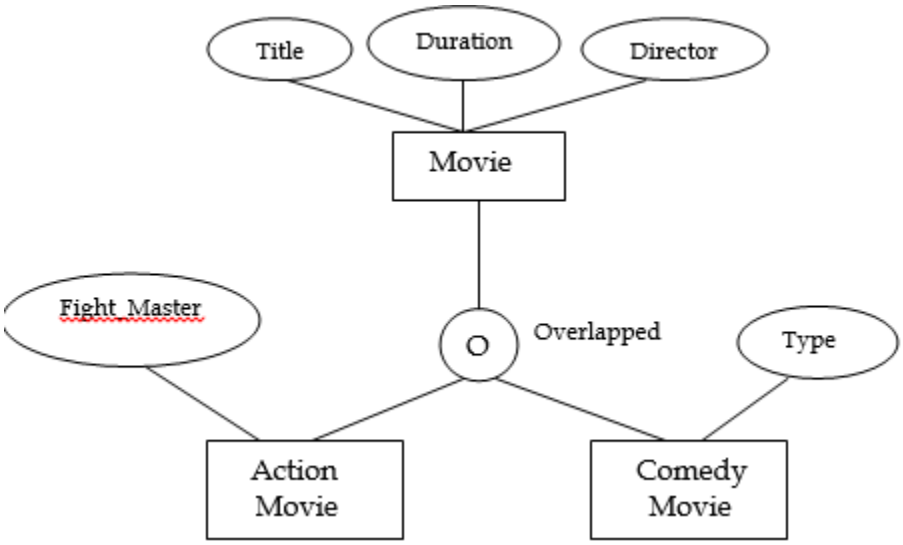
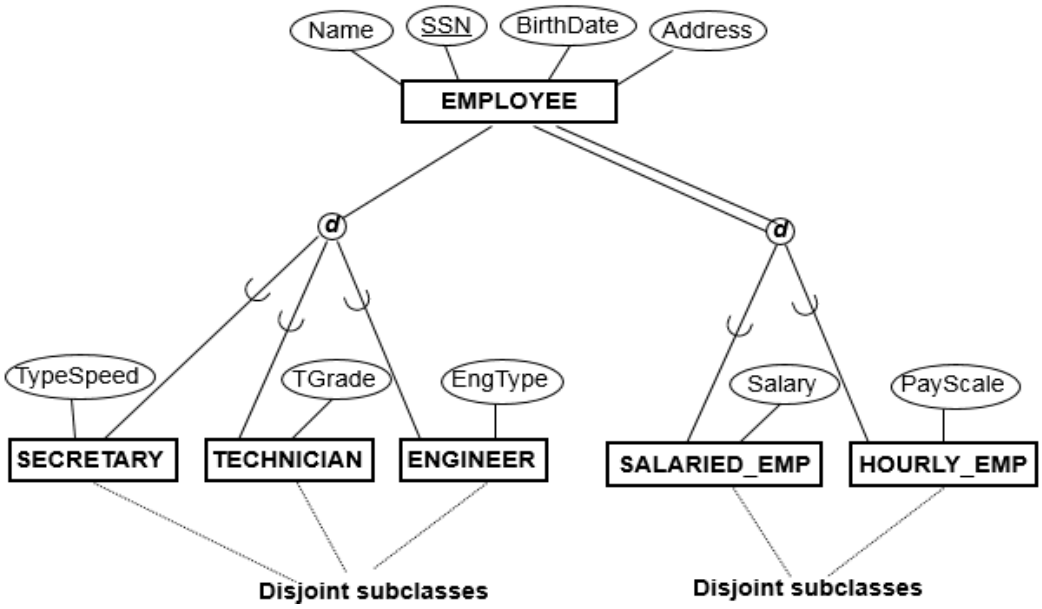


Fig: Overlapped constraint

A total specialization vs. a partial specialization

A total specialization constraint specifies that every entity in the superclass must be a member of some subclass in the specialization. It is represented by a double line connecting the superclass to the circle.

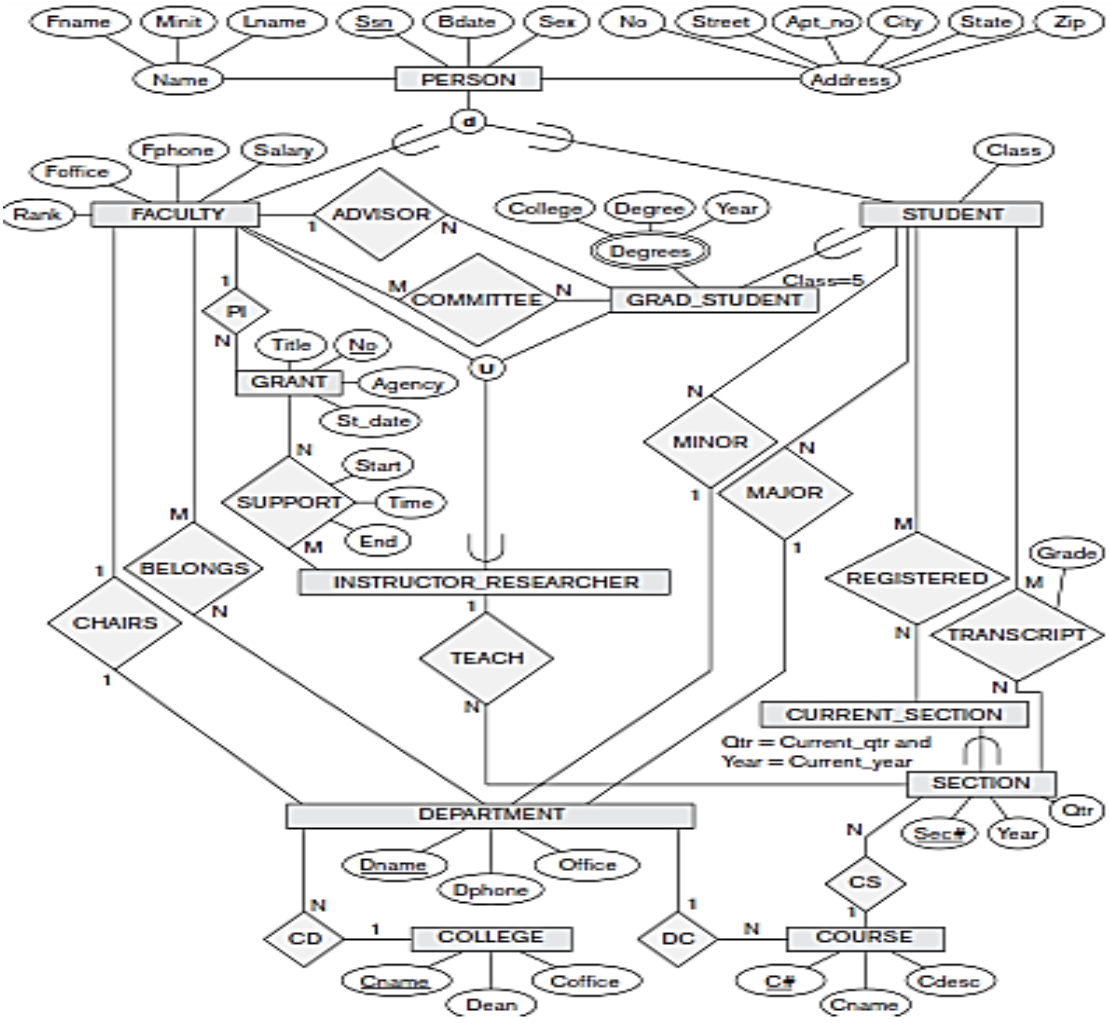
A partial specialization allows an entity not to belong to any of the subclasses, using a single line in EER. e.g., if some EMPLOYEE entities, for example, **sales representatives**, do not belong to any of the subclasses {SECRETARY, ENGINEER, TECHNICIAN}, then the specialization is partial. It is represented by a single line connecting the superclass to the circle. Partial generalization is the default.



Some insertion/deletion rules for specialization/generalization

- Deleting an entity from a superclass
  - it is automatically deleted from all the subclasses to which it belongs
- Inserting an entity in a superclass
  - it is mandatorily inserted in all predicate-defined (or attribute-defined) subclasses for which it satisfies the defining predicate
- Inserting an entity in a superclass of a total specialization
  - it is mandatorily inserted in at least one of subclasses

Example: The UNIVERSITY Database Example



ER-to-Relational Mapping Algorithm

- Step 1: Mapping of Regular Entity Types
- Step 2: Mapping of Weak Entity Types
- Step 3: Mapping of Multivalued attributes.
- Step 4: Mapping of Composite attributes.
- Step 5: Mapping of Binary 1:1 Relation Types
- Step 6: Mapping of Binary 1: N Relationship Types.
- Step 7: Mapping of Binary M: N Relationship Types.
- Step 8: Mapping of N-ary Relationship Types.

Mapping EER Model constructs to-Relational Algorithm

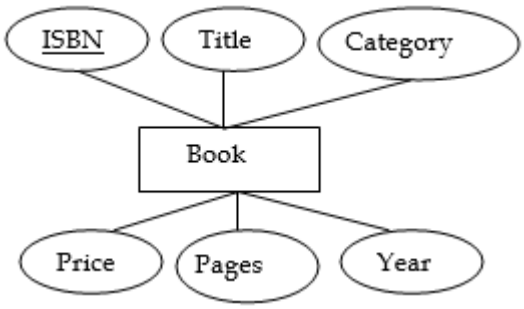
Step 1: Mapping Specialization or Generalization to relational model

Step 2: Mapping of Union Types (Categories) to relational model

Step 3: Mapping aggregation to relational model

Mapping regular entity set to relational model

An entity set is mapped to a relation in a straightforward way: Each simple attribute of the entity set becomes an attribute (column) of the table and primary key of the entity set becomes primary key of the relation. Domain of each attribute of the relation must be known.

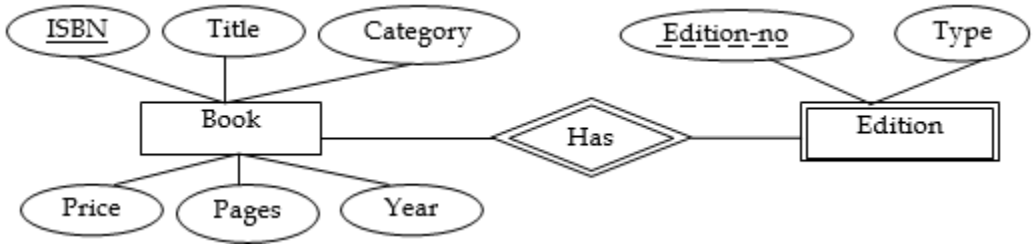


ISBN	Title	Category	Price	Pages	Year

Mapping of Weak Entity Types

A weak entity set does not have its own primary key and always participates in one-to-many relationship with owner entity set and has total participation. For a weak entity set create a relation that contains all simple attributes (or simple components of composite attributes). In addition, relation for weak entity set contains primary key of the owner entity set as foreign key and its primary key is formed by combining partial key (discriminator) and primary key of the owner entity set.

Example: Consider the weak entity set 'Edition' with two attributes edition_no and type, where edition_no is partial key.

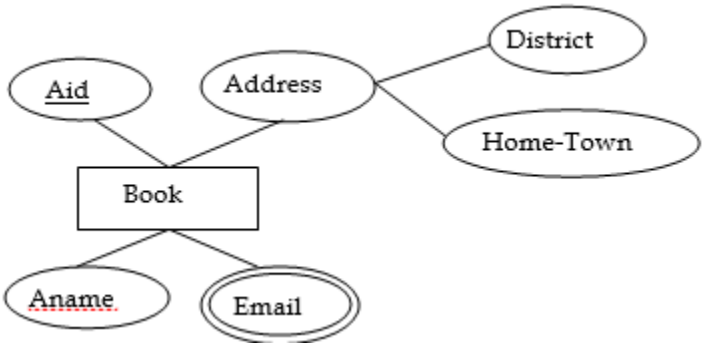


ISBN	Edition-no	Type

Mapping of Multivalued and composite attributes

If an entity has composite attributes, no separate attribute (column) is created for composite attribute itself rather attributes (columns) are created for component attributes of the composite

attribute. For a multivalued attribute, separate relation is created with primary key of the entity set and multivalued attribute itself.

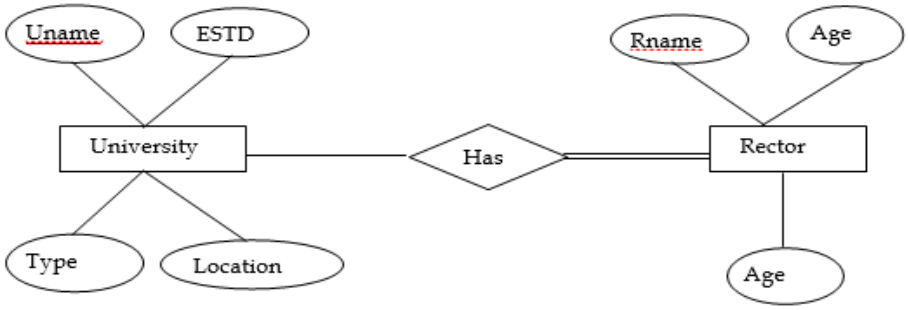


Aid	Aname	District	Home_Town

Aid	Email

Mapping of Binary 1:1 Relation Types

For a binary one to one relationship set separate relation is not created rather primary key of an entity set is included in relation for another entity set as a foreign key. It is better to include foreign key into the entity set with total participation in the relationship.

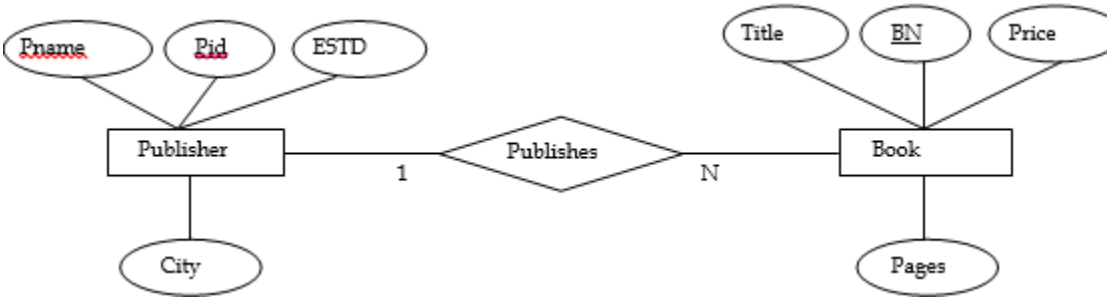


Uname	ESTD	Type	Location

Rname	Uname	Age	Sex

Mapping of Binary 1: N Relation Types

For binary one-to-many relationship identify the relation that represent the participating entity type at the N-side of the relationship type and then include primary key of one side entity set into many side entity set as foreign key. Separate relation is created for the relationship set only when the relationship set has its own attributes

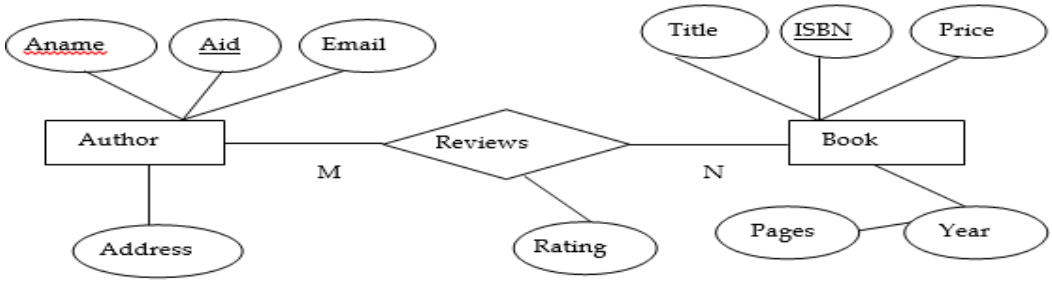


Pid	Pname	ESTD	City

ISBN	Title	Price	Pages	Pid

Mapping of Binary M: N Relation Types

For a binary Many-to-Many relationship type, separate relation is created for the relationship type. Primary key for each participating entity set is included as foreign key in the relation and their combination will form the primary key of the relation. Besides this, simple attributes of the many-to-many relationship type (or simple components of composite attributes) is included as attributes of the relation.



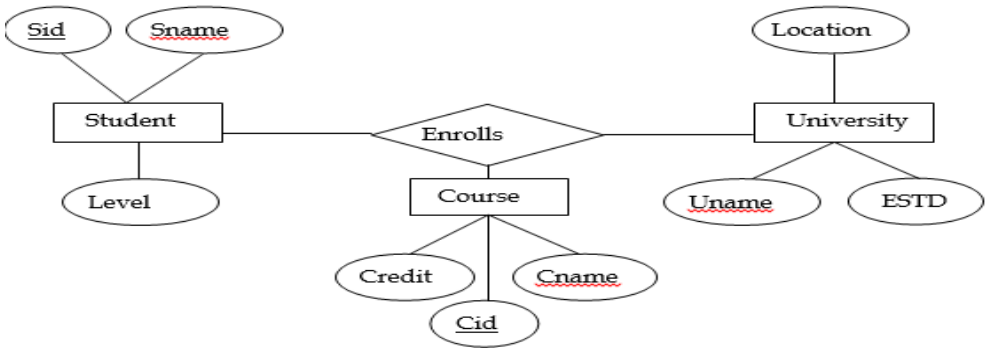
ISBN	Title	Pages	Price	Year

Aid	Aname	Email	Address

ISBN	Aid	Rating

Mapping of N-ary Relationship Types

For each n-ary relationship set for n>2, a new relation is created. Primary keys of all participating entity sets are included in the relation as foreign key attributes. Besides this all simple attributes of the n-ary relationship set (or simple components of composite attributes) are included as attributes of the relation.



Sid	Sname	Level

Uname	ESTD	Location

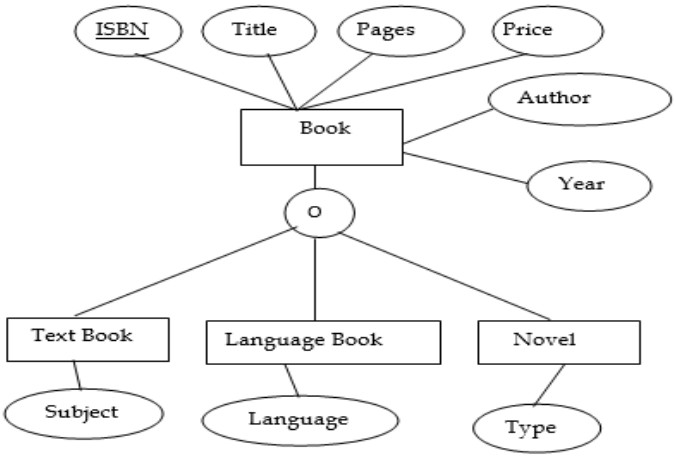
Cid	Cname	Credit

Sid	Uname	Cid

Representing Specialization/Generalization

There are two approaches for translating ER diagrams with specialization or generalization into relations.

- A relation is created for higher level entity set using above discussed methods and then separate relation is created for each of the lower level entity sets. Relation for a subclass entity set includes all of its attributes and key attributes of superclass entity set.
- Another approach is to create relations only for lower level entity sets. Here, relation for a subclass entity set includes all attributes of superclass entity set and all of its own attributes. This approach is possible only when subclasses are disjoint and complete



ISBN	Title	Author	Pages	Price	Year

ISBN	Subject

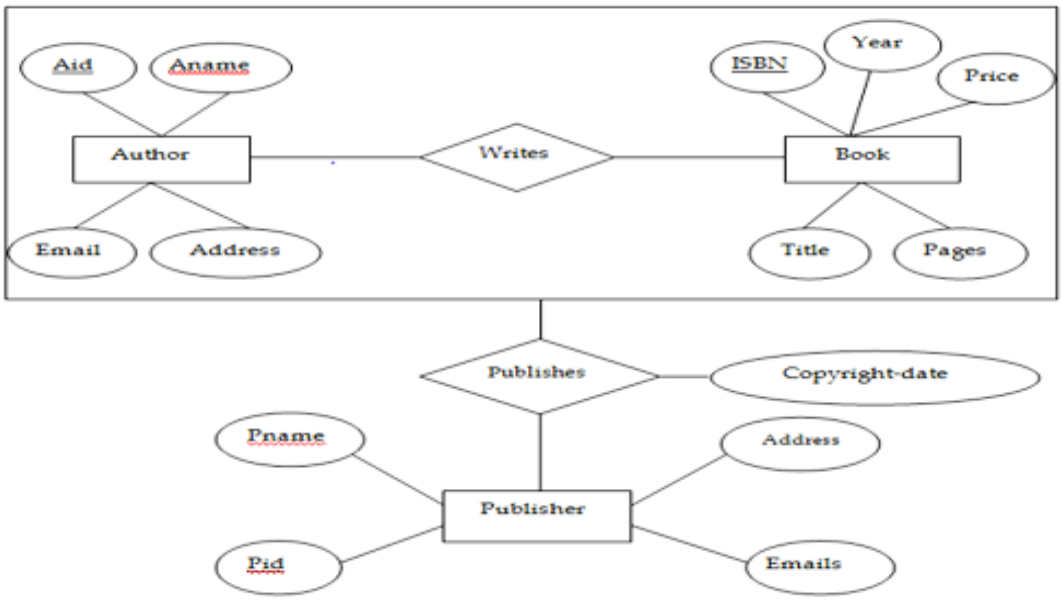
ISBN	Language

ISBN	Type

Representing Aggregation

To represent aggregation, create a table containing

- Primary key of aggregated relationship
- Primary key of the associated entity set
- Any descriptive attributes of the relationship set



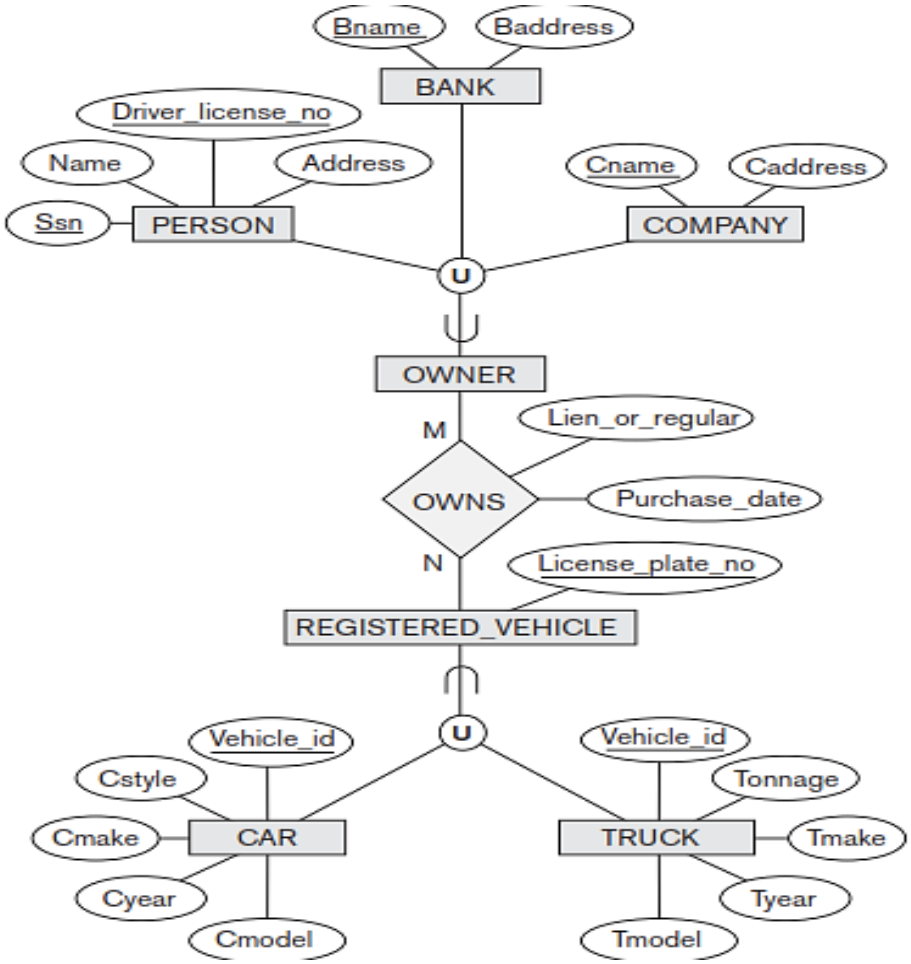
Aid	ISBN	Pid	Copyright_date

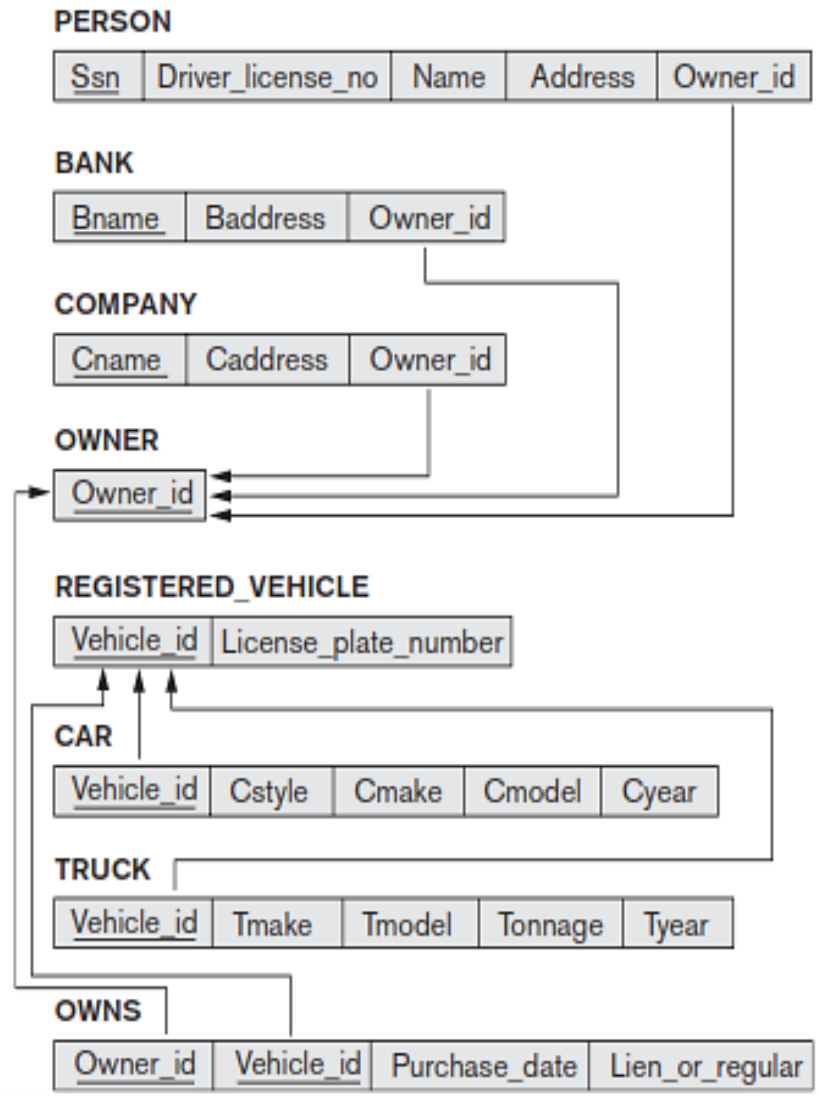
Mapping of Union Types (Categories) to relational model

For mapping a category whose defining super-classes have different keys, it is customary to specify a new key attribute, called a **surrogate key**, when creating a relation to correspond to the category. The keys of the defining classes are different, so we cannot use any one of them exclusively to identify all entities in the category. In our example in Figure below, we create a relation OWNER to correspond to the OWNER category, and include any attributes of the category

in this relation. The primary key of the OWNER relation is the **surrogate key**, which we called Owner_id. We also include the **surrogate key** attribute Owner_id as foreign key in each relation corresponding to a superclass of the category, to specify the correspondence in values between the **surrogate key** and the key of each superclass. Notice that if a particular PERSON (or BANK or COMPANY) entity is not a member of OWNER, it would have a NULL value for its Owner_id attribute in its corresponding tuple.

Two categories (union types): OWNER and REGISTERED_VEHICLE are shown in fig. below.





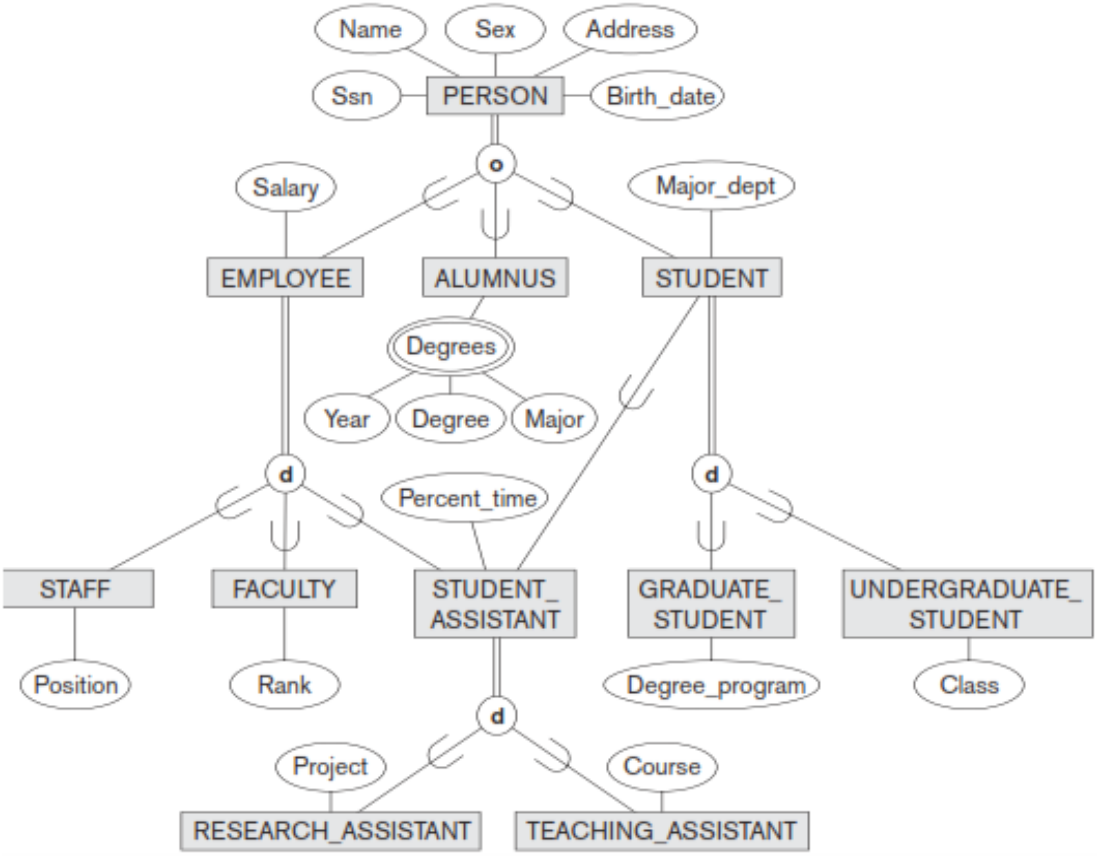
Object Modeling Using UML Class Diagrams

Object modeling methodologies, such as UML (Universal Modeling Language) and OMT (Object Modeling Technique) are becoming increasingly popular. Although these methodologies were developed mainly for software design, a major part of software design involves designing the databases that will be accessed by the software modules. Hence, an important part of these methodologies—namely, the class diagrams are similar to EER diagrams in many ways. Unfortunately, the terminology often differs. We briefly review some of the notation, terminology, and concepts used in UML class diagrams, and compare them with EER terminology and notation. Figure below shows how the COMPANY ER database schema can be displayed using UML notation.

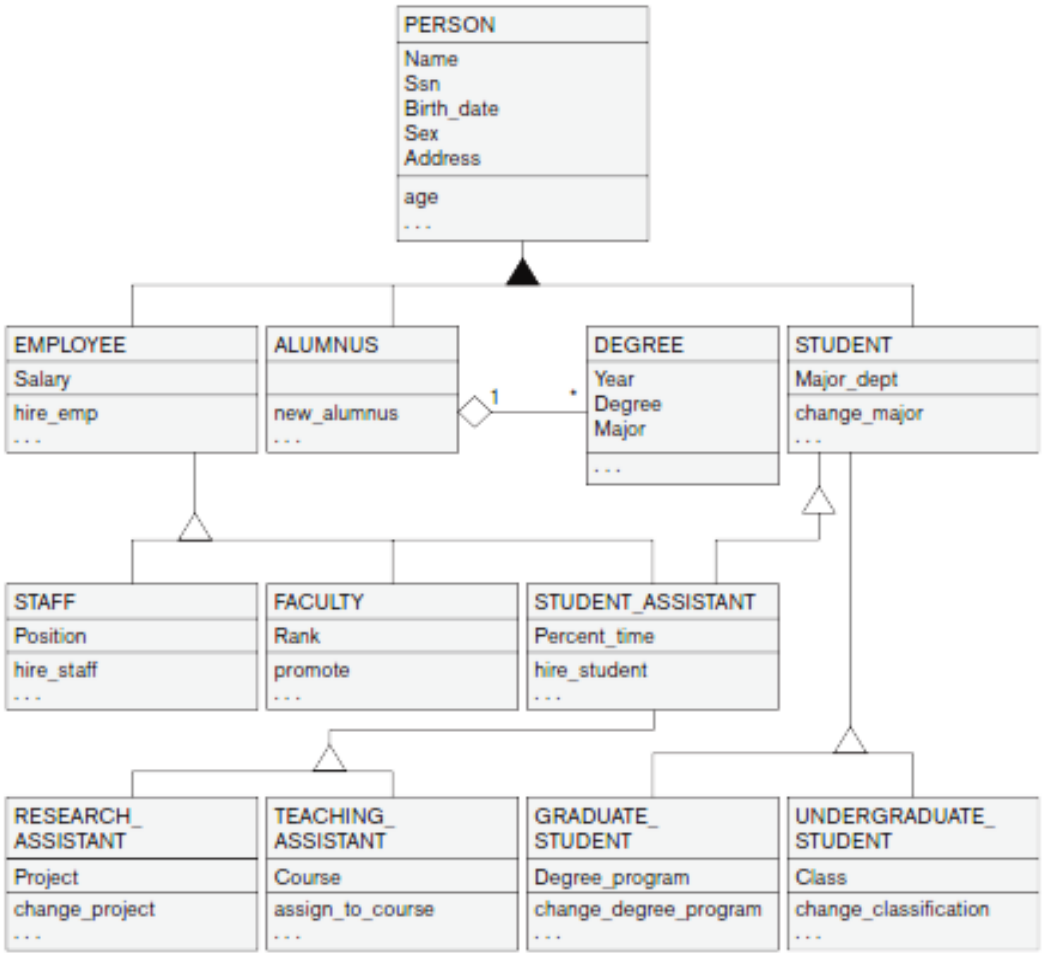
In UML class diagrams, a class is displayed as a box that includes three sections: the top section gives the class name; the middle section includes the attributes for individual objects of the class; and the last section includes operations that can be applied to these objects. Operations are not specified in EER diagrams. Consider the EMPLOYEE class. Its attributes are Name, Ssn, Bdate, Sex, Address, and Salary. The designer can optionally specify the domain of an attribute if desired.

A multivalued attribute will generally be modeled as a separate class, as illustrated by the LOCATION class Relationship types are called associations in UML terminology, and relationship instances are called links. A binary association (binary relationship type) is represented as a line connecting the participating classes (entity types), and may (optional) have a name. A relationship attribute, called a link attribute, is placed in a box that is connected to the association's line by a dashed line.

In UML, there are two types of relationships: **association** and **aggregation**. Figure below illustrates the UML notation for generalization/specialization by giving a possible UML class diagram corresponding to the EER diagram



A UML class diagram corresponding to the EER diagram shown in Figure above is represented by following diagram.



Assignment

1. Construct ER diagram and then translate into a set of **Nepal premier database (NPL)**

Requirements:

- The NPL has many teams,
- Each team has a name, a city, a coach, a captain, and a set of players,
- Each player belongs to only one team,
- Each player has a name, role, a skill level, and a set of injury records
- A team captain is also a player
- A game is played between two teams (referred to as host_team and guest_team) and has a date (such as May 11th, 1999).

2. Construct EER diagram and then map into a set of **University Database**.

Requirements:

- Professors have an Citizenship number, a name, an age, a rank, and a research specialty.

- Projects have a project number, a sponsor name, a starting date, an ending date, and a budget.
- Graduate students have Citizenship number, a name, an age, and a degree program (e.g., M.S. Or Ph.D.).
- Each project is managed by one professor (known as the project's principal investigator).
- Each project is worked on by one or more professors (known as the project's co investigators).
- Professors can manage and/or work on multiple projects.
- Each project is worked on by one or more graduate students (known as the project's research assistants).
- When graduate students work on a project, a professor must supervise their work on the project. Graduate students can work on multiple projects, in which case they will have a (potentially different) supervisor for each one.
- Departments have a department number, a department name, and a main office.
- Departments have a professor (known as the chairman) who runs the department.
- Professors' work in one or more departments, and for each department that they work in, a time percentage is associated with their job.
- Graduate students have one major department in which they are working on their degree.

1. Construct an ER diagram for a car-insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents.
2. Construct an ER diagram for a hospital with a set of patients and a set of doctors. Associate with each patient a log of the various tests and examinations conducted.
3. Construct an ER diagram of the library system in your college.
4. Construct an ER diagram to maintain data about students, instructors, semester, and courses in a college.
5. Construct an ERD to record the marks that students get in different exams of different course offerings.