Study Unit 2
Database Design with Entity-Relationship Modelling

Learning Outcomes

By the end of this unit, you should be able to:

1. Explain the importance and purpose of a data model
2. Interpret any given Entity Relationship diagram
3. Diffentiate the various design patterns in daa modelling
4. Choose design patterns for a given set of data requirements
5. Construct conceptual (ER diagram) from a statement requirement to solve common design problems

Overview

In this study unit, we look into designing a database when given a set of data requirements. The data requirements are first depicted pictorially via a data model using the Entity Relationship Model notation. In particular, the Crow Foot notation adopted by the course text will be discussed.

The data requirements are depicted in a data model using the various types of entities: strong, weak and identifying entities, and various types of relationships: strong, weak and identifying relationships. For each relationship, the maximum and minimum cardinalities are determined from the data requirements so that we know how many instances of another entity one instance of an entity may relate to. When the data model is verified to be correct, that is, the data requirements are correct, the entities and relationships will be transformed into relations with referential integrity constraint, to be covered in Study Unit 3.

We look also at the various design patterns in data modelling. Knowing these patterns will help us use time-tested solutions for specific data requirements in our data model.

This study unit covers chapter 5 of the course text. It is estimated that the student will spend about 4 hours to read the course text chapter 5 in conjunction with the study notes, to work out the activities, and self-assessment questions in the study notes included at suitable junctures to test understanding of the contents covered. It is advisable to use the study notes to guide the reading of the chapter in the textbook, attempt the questions, and then check the text and/or other sources for the accuracy and completeness of your answers.

Chapter 1 Data Model

1.1 Motivation

In a new application development, the data requirements are first gathered and then analysed and represented in non-ambiguous notations such as Entity-Relationship diagram or UML class diagram as a data model.

Data models are useful for verifying data requirements as well. Making changes on the pictorial representation is also easier than on a database. When the data model is correct and complete, it is transformed into a logical data model, and then implemented as a physical model, that is, a database.

Read

Kroenke, D and D. Auer. (2016). *Database Processing: Fundamentals, Design and Implementation Edition 14*. Pearson, 220.

Activity 1

Reproduced from Question 5.3 of the course text.

Explain how a data model is like a building blueprint. What is the advantage of making changes during the data modelling stage?

1.1.1 Data Model or Conceptual Model

A data model or a conceptual model is a pictorial representation of the data requirements of an application. The representation is in a generalised, non-ambiguous and non-DBMS specific notation. The representation is also called an entity relationship (ER) diagram or ER model.

1.1.2 Logical Model or Logical Schema

A logical model uses the specific data structure of a technology. For example, in relational databases, the logical model is represented in the form of relations, attributes, keys, referential integrity constraints and other constraints.

Relations in the logical model are normalized to BCNF and 4NF in preparation of its implementation as a physical model, on a specific Database Management System (DBMS).

A logical model is platform-independent. It is covered in Study Unit 3.

1.1.3 Physical Model or Physical Schema

The physical model is an implementation of the logical model. A specific DBMS is chosen to implement a physical model or physical schema for the logical model.

Physical design or physical schema is also covered in Study Unit 3.

1.2 Notations in Entity Relationship Model

The data model is most often represented as an entity relationship model. There are many versions of entity relationship model, such as the original Chen's model, the

extended ER model, the UML class diagram and Crow Foot notation. The Crow Foot notation is adopted by the course text.

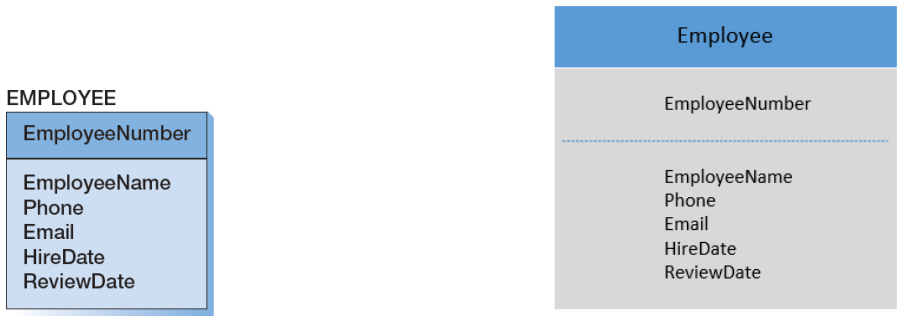
Read

Kroenke, D and D. Auer. (2016). *Database Processing: Fundamentals, Design and Implementation Edition 14*. Pearson, 235-236.

1.2.1 Entity

An entity is something for which data must be recorded, e.g., employee, department, and project.

The notation for an entity is a rectangle, as shown in Figure 2.1.



(a) (b)

Figure 2.1 Entity Employee

(Source for Figure 2.1 (a): Kroenke, D and D. Auer. (2016). Database Processing: Fundamentals, Design and Implementation Edition 14. Pearson, Figure 5-2 (b) Figure 2.1 (b) Entity Employee equivalent using Visio)

Activity 2

Modified from Question 5.5 of the course text.

Define entity. Give three example of an entity for an application for Real Estate Agency.

1.2.2 Attributes and Identifiers

The attributes of an entity are the characteristics about the entity relevant to the application, for which data must be recorded.

For example, the relevant characteristics of an employee, as shown in Figure 2.1, are the employee number, the employee name, phone, email, date the employee is hired and the date that employee has a review.

An identifier identifies a specific entity instance or an occurrence of the entity. Two employee instances or occurrences are shown in Figure 2.2. The identifier employee number identifies the specific employee. For example, employee number E123 identifies Tom Peters who joined on 12 February 2000 and whose review is on 15 April 2019.

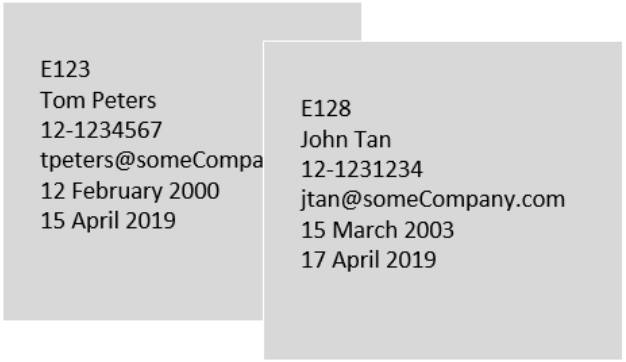


Figure 2.2 Two entity instances of Employee

The concept of an identifier is similar to the concept of a primary key. Note that an employee name may not be unique and so, employee name cannot be an identifier of the entity Employee.

The identifier of an entity can be made up of one or more attributes. Identifiers are placed within one compartment in the rectangle for that entity, and the remaining attributes are placed another compartment in the same rectangle.

For example, the identifier for employees is their employee numbers is placed in one compartment, as shown in Figure 2.1. The remaining attributes are placed in another compartment, after the compartment for the identifier.

1.2.3 Relationships

Entities can be related to one another. Two entities are related when their instances are related.

For example, an employee (an Employee instance) works in a particular department (a Department instance) of the company. The data model must represent relationships that are relevant to the application.

Relationships are represented with a line, to show the connected entities have a relationship. The line must be named appropriately so that the relationship can be understood and verified. In addition, the line must show the minimum and maximum cardinalities, described in Section 1.2.4.

Figure 2.3 shows two relations between the entities Employee and Department. The first relationship is `worksIn`. The relationship `worksIn` is about Employee instances who work in Department instances. An Employee instance is related to a Department instance if the Employee instance works in the Department instance. The `worksIn` relationship can be read as an employee works in a department, or alternatively, the department's work is performed by the employee.

The second relationship is `manages`. The relationship `manages` is about Employee instances who manage Department instances. An Employee instance is related to a Department instance if the Employee instance manages the Department instance. Similarly, the `manages` relationship can be read as an employee manages a department, or alternatively, the department is managed by the employee.

It is important to note that the identifier of one must not be duplicated as attribute(s) in the related entity. As a tool for analyzing and verifying data requirements, a data model must not include foreign keys in the entities. The relationships are depicted by lines between the entities. As mentioned, foreign keys are introduced only in logical models.

Not having foreign keys simplifies making changes to an incorrect relationship in a data model. In such a case, the line for the relationship can simply be removed without having the need to check for and remove duplicated attributes acting as foreign keys. Apart from the case of id-dependent entities discussed in Section 1.2.5, a data model that shows duplicated attributes is incorrect.

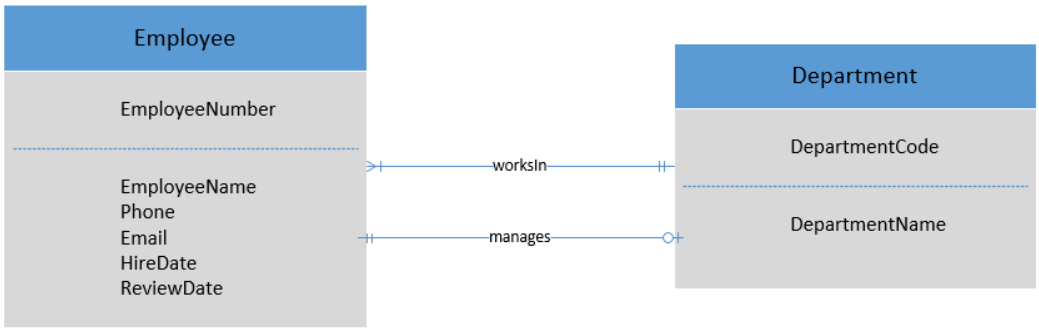


Figure 2.3 Two relationships between the entities Employee and Department

1.2.4 Minimum and Maximum Cardinalities

The minimum cardinality tells us how many entity instances must participate in a relationship whereas the maximum cardinality tells us how many entity instances can participate in a relationship.

Two sets of minimum and maximum cardinalities for each relationship between two entities must be shown on the relationship, with one set on each end of a relationship.

The Crow Foot notation for minimum and maximum cardinalities is shown in Figure 2.4.

Symbol	Meaning	Numeric Meaning
	Mandatory—One	Exactly one
	Mandatory—Many	One or more
	Optional—One	Zero or one
	Optional—Many	Zero or more

Figure 2.4 Crow Foot notation for minimum and maximum cardinalities

(Source: Kroenke, D and D. Auer. (2016). Database Processing: Fundamentals, Design and Implementation Edition 14. Pearson, Figure 5-8)

Consider the relationship `worksIn` in Figure 2.3. On the entity Department side of the relation, the minimum and maximum cardinalities are mandatory-one. This means that each Employee instance must work in a minimum of one (mandatory) department, and can work in a maximum of one department.

On the entity Employee side of the relation, the minimum and maximum cardinalities are mandatory-many. This means that each Department instance must have a minimum of one (mandatory) employee working in the department, and can have a maximum of many employees working in the department.

Now consider the relationship `manages` in Figure 2.3. On the entity Department side of the relation, the minimum and maximum cardinalities are optional-one. This means that each Employee instance must manage a minimum of zero (optional) department, and can manage a maximum of one department.

On the entity Employee side of the relation, the minimum and maximum cardinalities are mandatory-one. This means that each Department instance must have a minimum of one (mandatory) employee managing the department, and can have a maximum of one employee managing the department.

Read

Kroenke, D and D. Auer. (2016). *Database Processing: Fundamentals, Design and Implementation Edition 14*. Pearson, 229-235.

Activity 3

Reproduced from Question 5.19 of the course text.

In the Real Estate Agency application, show the data model in Crow Foot notation for the below data requirements:

a) each AGENT must use an agency car when on agency business. Further, to keep costs down the agency keeps exactly enough cars for the agents. Therefore, each AGENT must have a CAR, and each CAR must be assigned to an AGENT. This is an M-M relationship.

b) each CLIENT must be assigned to an AGENT, but there may be AGENTs who currently have no CLIENTs. This is an M-O (same as O-M, but seen reversed) relationship.

c) CLIENTs may be seeing AGENTs without currently being interested in specific PROPERTY, and a PROPERTY may be listed without any CLIENT currently being interested in it. This is an O-O relationship.

1.2.5 Strong and Weak Entities and ID-dependent Entities

The existence of an entity may depend on the existence of another entity. If the existence of an entity does not depend on the existence of another entity, the strong entity is strong. Otherwise, the entity is weak. Whether an entity is strong or weak is dependent on the problem domain.

Consider a car rental application. The entities in the data model include car, customer and rental. For the car rental application, the existence of a car is independent of the existence of all other entities, whether customer, rental or car. Thus, the entity car is strong.

In a different application, the existence of a car can be dependent on the existence of some other entities. For example, in a staff parking application, only staff can get a parking license for his car. The entities in the data model for this application include ataff and car. The entity car is weak as its existence depends on the existence of the entity staff. When a staff resigns, both details of the staff and his car should be removed.

A weak entity may also be an id-dependent entity. The identifier of an id-dependent entity includes the identifier of the strong entity its existence depends on. For example, the employee number of staff may be used as a part of the identifier for car. However, it is not always necessary to make a weak entity id-dependent. Using the same example, the car registration number alone can identify a car.

The relationship that an id-dependent entity has with the strong entity its existence depends on is also called an identifying relationship. Identifying relationships on the data model are shown as solid lines while non-identifying relationships are shown as dashed lines as shown in Figure 2.5.

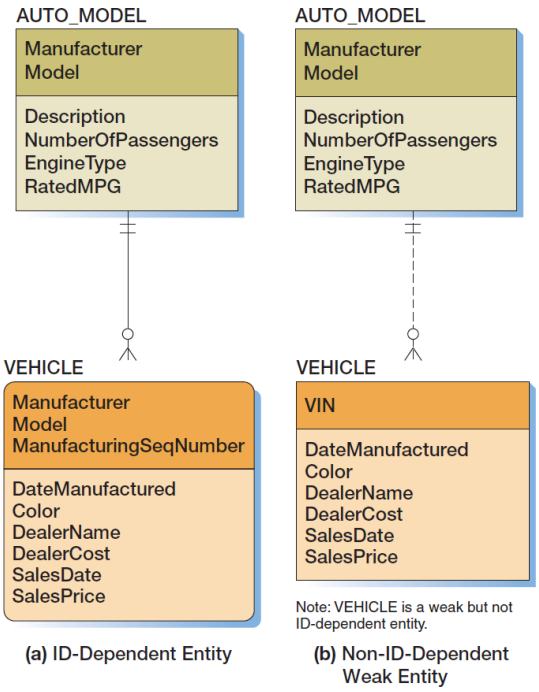


Figure 2.5 Identifying and non-identifying relationships

(Source: Kroenke, D and D. Auer. (2016). Database Processing: Fundamentals, Design and Implementation Edition 14. Pearson, Figure 5-11)

Read

Kroenke, D and D. Auer. (2016). *Database Processing: Fundamentals, Design and Implementation Edition 14*. Pearson, 238-240.

Activity 4

Reproduced from Question 5.29 of the course text.

What distinguishes a weak entity from a strong entity that has a required relationship to another entity?

1.2.6 Subtype and Supertype

Two entities can have a subtype-supertype relationship if one entity is a specialization of the other entity. For example, the entity undergraduate is a subtype of the supertype student as shown in Figure 2.6. Undergraduate is a specialization of student.

The cross within the circle in Figure 2.6 denotes that the subtype-supertype relationship is exclusive. This means that the supertype can relate to only one subtype. Thus, a student can either be an undergraduate or a graduate student, but not both.

The double bar under the circle denotes that subtype-supertype relationship is total or mandatory. This means that each instance of the supertype must relate to one subtype instance, and so, every student must be an undergraduate or a graduate student. There is no student who is not an undergraduate and who is also not a graduate student.

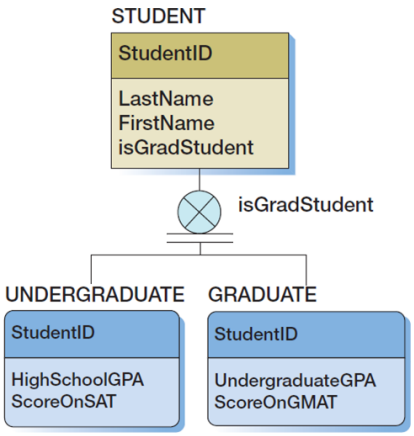


Figure 2.6 Total and exclusive subtype-supertype relationships

(Source: Modified from Kroenke, D and D. Auer. (2016). Database Processing: Fundamentals, Design and Implementation Edition 14. Pearson, Figure 5-13)

When there is no cross within the circle as seen in Figure 2.7(b), the subtype-supertype relationship is inclusive. This means that the supertype can relate to one

or more subtype. For example, a student can be from the hiking club and also be from the sailing club at the same time.

The single bar under the circle denotes that subtype-supertype relationship is partial or optional. This means that an instance of the supertype need not relate to any subtype instance, and so, a student need not be from the hiking club nor need he be from the sailing club.

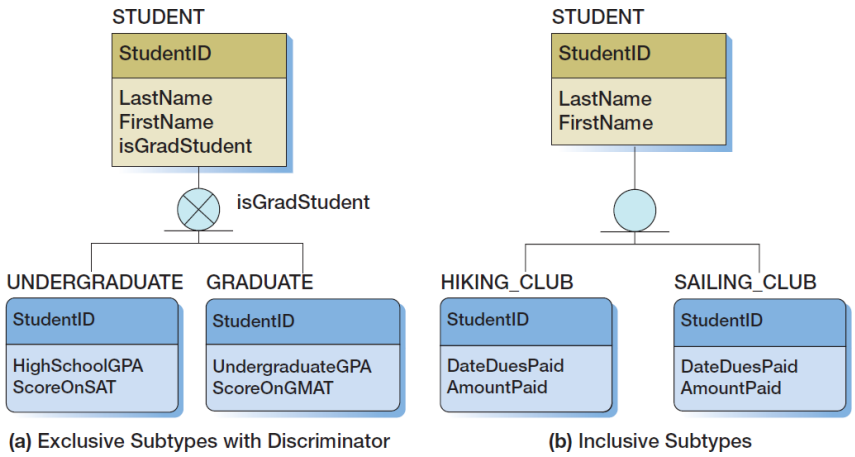


Figure 2.7 (a) Partial and exclusive subtype-supertype relationships
(b) Partial and inclusive subtype-supertype relationships

(Source: Kroenke, D and D. Auer. (2016). Database Processing: Fundamentals, Design and Implementation Edition 14. Pearson, Figure 5-13)

Read

Kroenke, D and D. Auer. (2016). *Database Processing: Fundamentals, Design and Implementation Edition 14*. Pearson, 240-242.

Activity 5

Reproduced from Question 5.33 of the course text.

Design an E-R model for an event planner, using subtype entities to indicate the fee for planning different types of events such as business conferences, weddings, expositions and exhibitions.

Chapter 2 Design Patterns in Data Modelling

Data requirements for an application come from users, and may be presented in data entry forms as data to be captured, as well as reports as data transformed from data captured in the system.

Some data requirements recur in many applications, and so, patterns for the entities and their relationship for these data requirements have been proposed for them. The patterns are also called design patterns, in particular, design patterns in data modelling as the patterns are specifically those for data modelling. Design patterns incorporate best practices, which are time-tested.

Knowing what design patterns are available for recurring data requirements speeds up the data modelling process, besides improving the quality of the data model. We adopt the set of design patterns in the course text.

2.1 Strong Entity Pattern

When two related entities are both strong, the relationship between them is termed strong. Strong relationships can have different minimum and maximum cardinalities. The authors of the course text have named the patterns according to the maximum cardinality.

2.1.1 1:1 Strong Entity Relationship

This pattern is used when the maximum cardinality on both strong entities is 1:1. This means there is at most one instance of one entity for each instance of the other entity when looking at the maximum cardinality in either direction of the relationship.

An example of two strong entities is club member and locker. If a club member can rent at most one locker, and a locker can be rented by at most one club member, then the two entities have a 1:1 strong entity relationship, as shown in Figure 2.8.

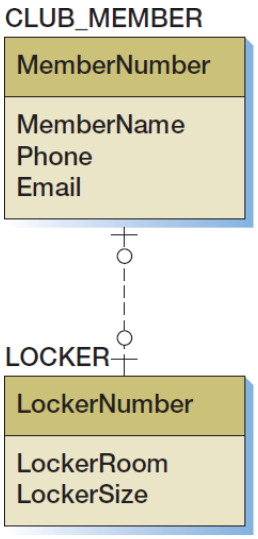


Figure 2.8 1:1 strong entity relationship

(Source: Kroenke, D and D. Auer. (2016). Database Processing: Fundamentals, Design and Implementation Edition 14. Pearson, Figure 5-16)

2.1.2 1:N Strong Entity Relationship

This pattern is used when the maximum cardinality on one strong entity is 1, and the maximum cardinality on the other strong entity is N, where N means many. This means when looking at the maximum cardinality in one direction of the relationship, there is at most one instance of one entity for each instance of the other entity, and when looking at the maximum cardinality in the other direction of the relationship, there can be many instances of one entity for each instance of the other entity.

An example of two strong entities is club member and club uniform. If a club member can have many club uniforms, and a club uniform can be given to at most one club member, then the two entities have a 1:N strong entity relationship, as shown in Figure 2.9.

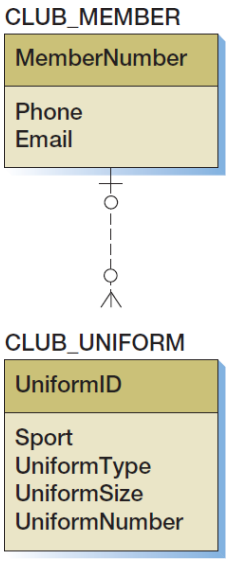


Figure 2.9 1:N strong entity relationship

(Source: Kroenke, D and D. Auer. (2016). Database Processing: Fundamentals, Design and Implementation Edition 14. Pearson, Figure 5-18)

2.1.3 N:M Strong Entity Relationship

This pattern is used when the maximum cardinality on both strong entities is many, as both N and M mean many, in this case.

This means there can be many instances of one entity for each instance of the other entity when looking at the maximum cardinality in either direction of the relationship.

An example of two strong entities is companies and parts. If a company supplies many parts, and a part can be supplied by companies, then the two entities have a N:M strong entity relationship, as shown in Figure 2.10.

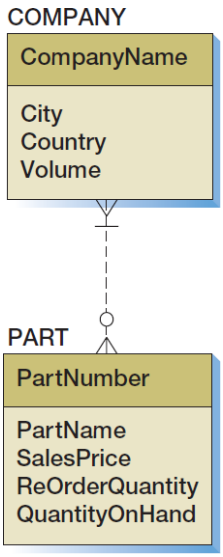


Figure 2.10 1:N strong entity relationship

(Source: Kroenke, D and D. Auer. (2016). Database Processing: Fundamentals, Design and Implementation Edition 14. Pearson, Figure 5-20)

Read

Kroenke, D and D. Auer. (2016). *Database Processing: Fundamentals, Design and Implementation Edition 14*. Pearson, 243-247.

Activity 6

Reproduced from Question 5.44 of the course text.

Design an E-R model for a fitness club that enrolls members in a variety of classes run by different trainers.

2.2 ID-Dependent Relationships

ID-dependent relationship involves one or more strong entities and one weak entity. For the patterns in this section, the weak entity always has an identifying relationship to each of the strong entities its existence depends on.

Note that the minimum and maximum cardinality on the side of the strong entity is always mandatory one, that is, each instance of the weak entity is related to exactly one instance of each of the strong entities. The maximum cardinality on the side of the weak entity is optional many, that is, an instance of each strong entity can relate to no instance or to many instances of the weak entity.

2.2.1 Association Pattern and the Associative Entity

When attributes result from a relationship between two or more strong entities, the association pattern is used.

For example, as shown in Figure 2.11, a student may enroll in many modules, and a module can have many students enrolled. The year that a student is enrolled in a module and the grade that he obtains for the module are a result of the relationship between a student and a module.

An entity called an associative entity is introduced for these attributes. In this example, the associative entity is StudentGrade.

If only one grade per module is recorded for each student, then the identifier for the associative entity, StudentGrade is (studentNumber, moduleCode), the composite of the identifiers of the student and module. The attributes of StudentGrade are grade and the year the module is taken for the given year.

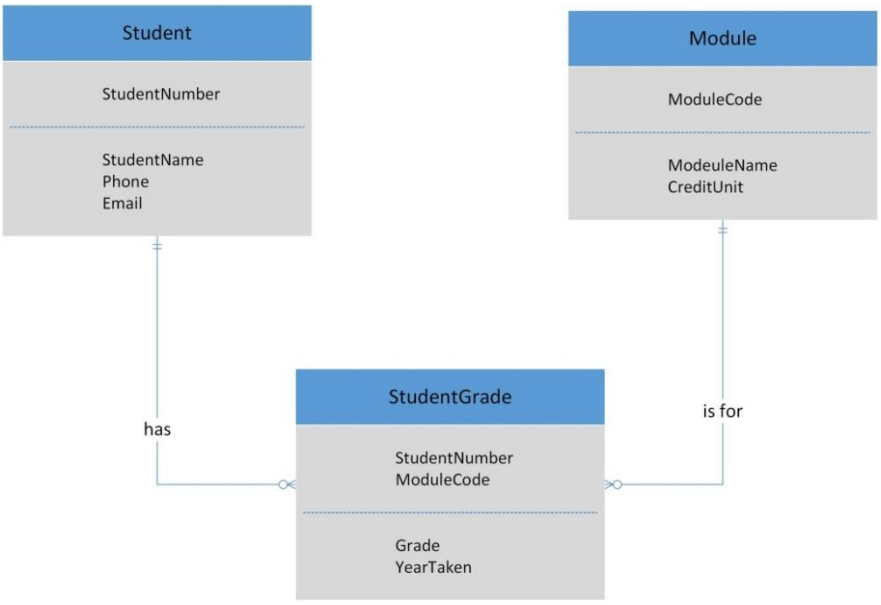


Figure 2.11 Associative entity with composite key student number and module code

However, if we wish to record the grade for all the attempts a student has made for any of his modules, then the identifier for the associative entity, StudentGrade is (studentNumber, moduleCode, yearTaken) as shown in Figure 2.12., to allow the grades of different attempts in different years to be recorded. The attributes of StudentGrade is grade.

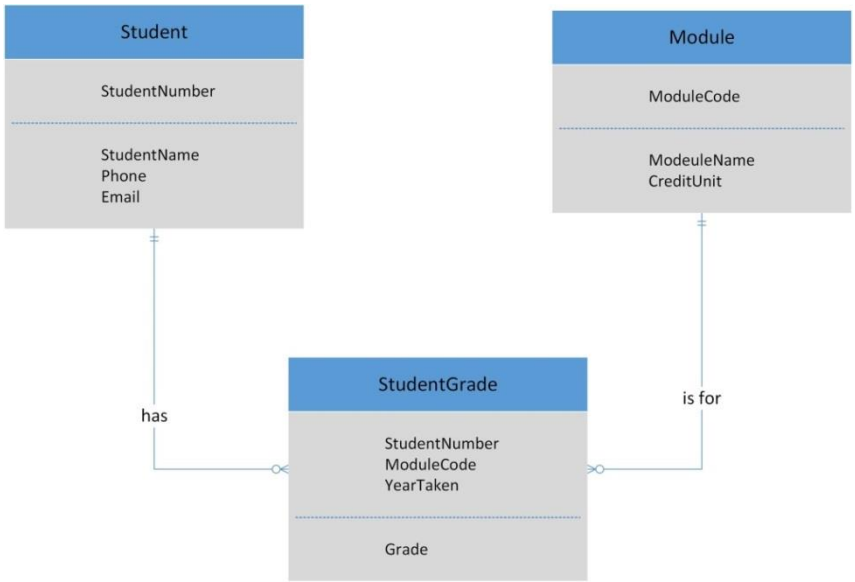


Figure 2.12 Associative entity with composite key student number, module code and year taken

2.2.2 Multivalued Attribute Pattern

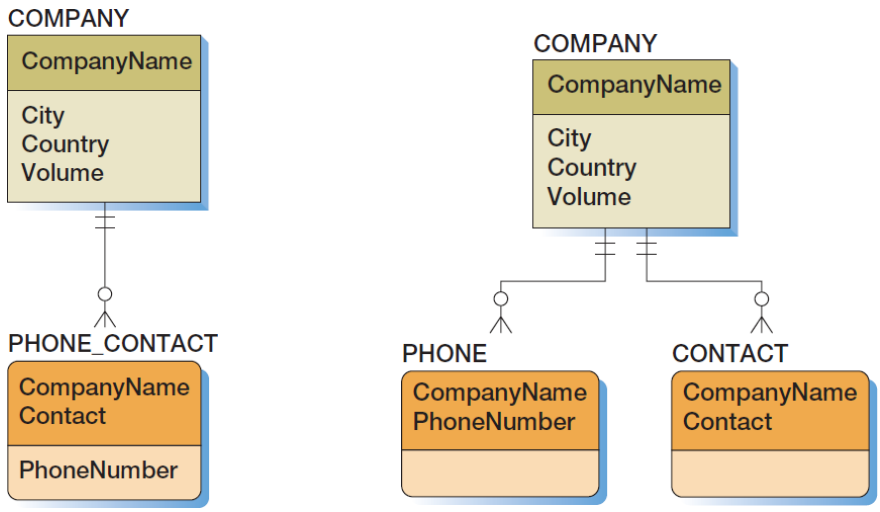
Multivalued attribute is a form of multivalued dependency discussed in Study Unit 1. Each multivalued attribute results in data duplication, and so, it is a source of modification anomaly. When an entity has a multivalued attribute, the multivalued attribute pattern is used.

The multivalued attribute pattern is in an attempt to normalise such entities by introducing an entity for each multivalued attribute. In addition, the identifier of new entity includes the identifier of the entity that has the multivalued attribute.

Consider a company that has many contact persons, each can be reached through one or more phone numbers. The entity company has a repeating group - contact (person name) and phone number.

Consider the case where a company has many contact persons. each reached through a specific phone number. The multivalued attribute in this case is a contact person with the associated phone number, as shown in Figure 2.13(a).

However, if contact persons share a common pool of phone numbers, then there are two multivalued attributes for the company – one for the contact persons and another for the pool of phone numbers, as shown in Figure 2.13(b).



(a) (b)

Figure 2.13 Multivalued attribute pattern

(Source: Kroenke, D and D. Auer. (2016). Database Processing: Fundamentals, Design and Implementation Edition 14. Pearson, Figure 5-29 for Figure 2.13(a), Figure 5-27 for Figure 2.13(b))

2.2.3 Archetype/Instance Pattern

Sometimes, the values of some attributes of an entity are repeated in many of its instances.

For example, when a yacht sold by a company can be described by its length and number of rooms it has, as well as the price it was sold for and the date sold. However, the length and number of rooms a yacht has actually depends on its design or model.

There is a transitive dependency; the design or model of the yacht determines its length and number of rooms it has but not the price the yacht was sold for and the date sold. Both the hull number of the yacht and the design jointly determine the price the yacht was sold for and the date sold. The Transitive dependency is a source of modification anomalies, as discussed in Study Unit 1.

The transitive dependency is handled via the archetype-instance pattern. The original entity is split into an entity for yacht design and an entity for the yacht. The yacht design is the archetype entity and the yacht is the instance entity.

The two entities derived from removing the transitive dependency result in one weak entity and one strong entity, as a yacht cannot exist without having a yacht design. The identifier of the weak entity includes the identifier of the strong entity.

Consider another example where a school puts students into different sections (or tutorial groups) for a same class (module). If there is only one entity which combines both the attributes of a section and the class it is for, the entity will have the following attributes: day of the week, time the section meets, a professor who conducts the section as well as details of the class conducted such as a description of the class and the number of hours required to complete the class.

If there are many sections for a class, the details of the class is repeated for each of the sections.

The archetype-instance pattern suggests two entities – class and section, rather than simply one entity, to remove the transitive dependency. The two entities derived from removing the transitive dependency in this application result in one weak

entity and one strong entity, as a section cannot exist without the class the section is for.

The identifiers of the strong entities can be included in the identifier of the weak entity as shown in Figure 2.14. However, the weak entities need not be id-dependent if we are able to find an attribute in the weak entity to be its identifier, as shown in Figure 2.15.

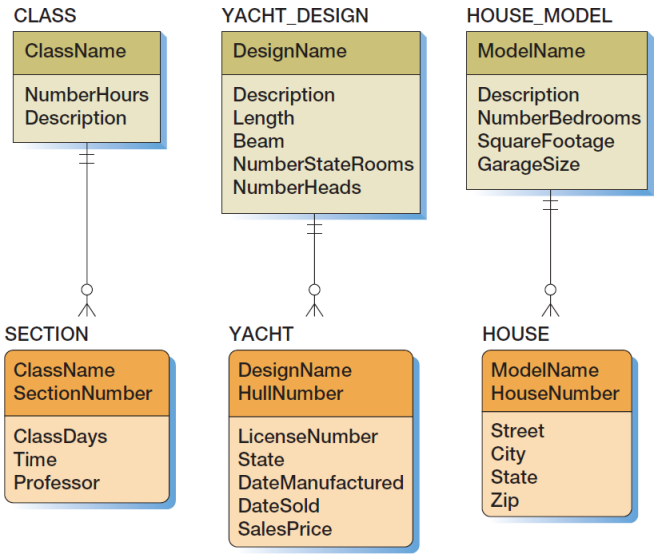


Figure 2.14 Archetype/Instance pattern with identifying relationship

(Source: Kroenke, D and D. Auer. (2016). Database Processing: Fundamentals, Design and Implementation Edition 14. Pearson, Figure 5-30)

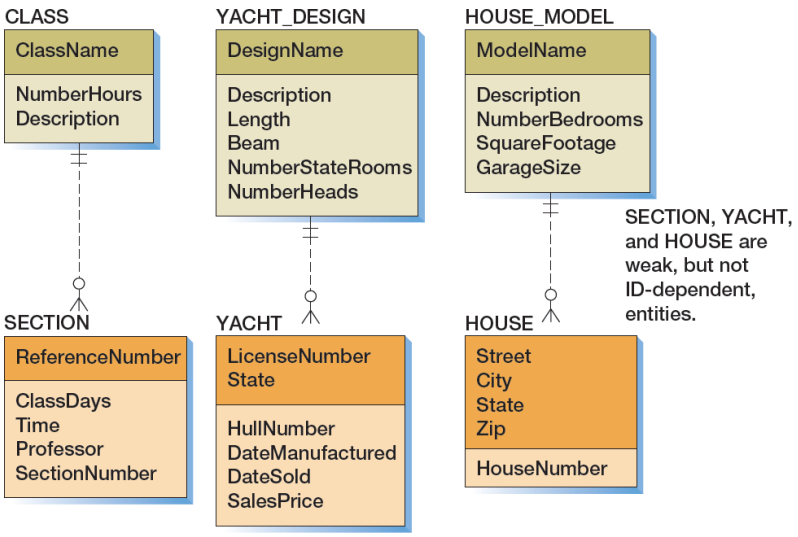


Figure 2.15 Archetype/Instance pattern with non-identifying relationship

(Source: Kroenke, D and D. Auer. (2016). Database Processing: Fundamentals, Design and Implementation Edition 14. Pearson, Figure 5-31)

Read

Kroenke, D and D. Auer. (2016). *Database Processing: Fundamentals, Design and Implementation Edition 14*. Pearson, 247-253.

Activity 7

Modified from Question 5.50 of the course text.

a) Identify the archetype and the instance in this data requirement: Data about 2 entities: MOVIE and PART must be recorded; with PART is an ID-dependent entity. Part of the identifier of the entity PART includes the identifier of the entity MOVIE (MovieName).
b) Design an E-R diagram that has an ID-dependent entity.
c) Change the E-R diagram to a weak but non ID-dependent entity.

2.3 Mixed Identifying and Non-identifying Relationship

The patterns in this section involve a weak entity that has an identifying relationship with one strong entity but a non-identifying relationship with another strong entity. Furthermore, the existence of the weak entity is dependent on the existence of one strong entity but not on the existence of the other strong entity.

2.3.1 Line-Item Pattern

The line-item pattern has a part-whole relationship where the whole has many parts, and the existence of the parts depend on the existence of the whole. A line-item is a part of a whole, therefore, it is a weak entity. In addition, the line-item also has a non-identifying relationship with another strong entity.

Some examples of the line-item are as follows:

- In an order, there are many order lines, each order line describes the ordered item such as the unit price and quantity.

  Each ordered item is also related to an item that a customer can order.

- In a quotation, there are many quotation lines, each quotation line describes the quotation item such as the unit price and minimum quantity to qualify for the unit price.

  Each quotation item is also related to a product that a company sells.

- In a vehicle service, there are many service lines, each service line describes the service item such as the cost for the service item.

  Each service item is also related to a service that is provided by the car servicing agency.

Refer to Figure 2.16 for an example of the line-item pattern.

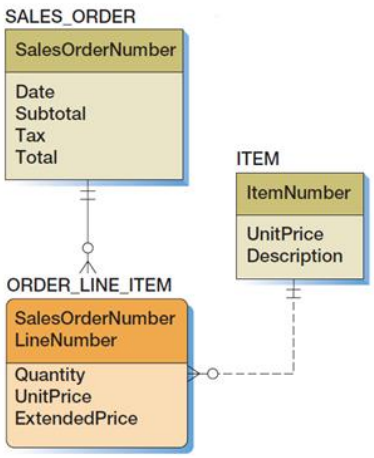


Figure 2.16 Line-Item pattern

(Source: Modified from Kroenke, D and D. Auer. (2016). Database Processing: Fundamentals, Design and Implementation Edition 14. Pearson, Figure 5-33)

Sales order is the whole and order line item is the part. An order line item describes the item sold. It has an identifying relationship with sales order. But it has a non-identifying relationship with the items that a company sells.

Note that a sales order can have many order line items, so sale order number alone is not enough to identify the order line item. Another identifying attribute is needed to identify the order line order, and typically, a running number is used, such as line number for the order line item within a sales order.

2.3.2 Other Mixed Pattern

The other mixed pattern is a generalized case of the line-item pattern. In the general case, an entity has a multivalued composite group, and each element in the composite group is a part of the whole, the entity.

For example, consider Figure 2.17.

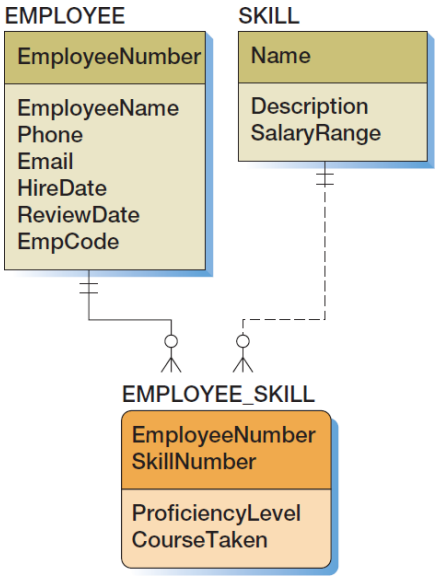


Figure 2.17 Other mixed pattern

(Source: Modified from Kroenke, D and D. Auer. (2016). Database Processing: Fundamentals, Design and Implementation Edition 14. Pearson, Figure 5-35)

A company may wish to track the different skills that it requires. Each skill has a description and the salary range an employee having the skill can command. A employee (the whole) has many skills (the parts).

Each employee skill is attained at a proficiency level through a course the employee undertook. The employee skill has an identifying relationship with the employee and a non-identifying relationship with the skills tracked by the company.

Read

Kroenke, D and D. Auer. (2016). *Database Processing: Fundamentals, Design and Implementation Edition 14*. Pearson, 253-256.

Activity 8

Modified from Question 5.52 of the course text.

Give an example of the line-item pattern as it could be used to describe the contents of a shipment. Assume that the shipment includes the names and quantities of various items as well as each item's insured value. Place the insurance value per item in an ITEM entity.

2.4 For-Use-By Pattern

The For-Use-By pattern is used whenever some attributes of an entity are required only under certain circumstances, that is, these attributes are 'for use by' someone or something only.

When those circumstances do not happen, the attributes do not have values or rather, they have null values.

Null values can have many interpretations such as

- someone forgot the value,
- someone forgot to fill up that value, or

- the value is inappropriate or the value.

To avoid null values because the values are inappropriate for a circumstance, the for-use-by pattern should be used, The for-use pattern differentiate between attributes that are always needed and those attributes that are specialized (for special circumstances only).

Attributes that are always needed are put in one entity called the supertype, and those attributes that are used under special circumstances, are put in one or more entities called the subtypes. Each use results in one subtype.

For example refer to Figure 2.18 for the example of fishing license.

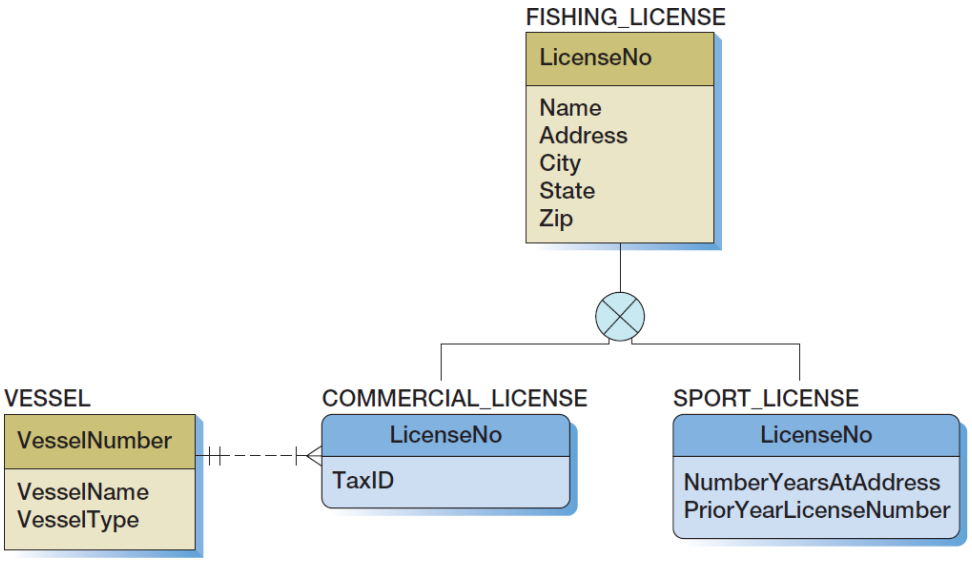


Figure 2.18 For-Use-By pattern

(Source: Kroenke, D and D. Auer. (2016). Database Processing: Fundamentals, Design and Implementation Edition 14. Pearson, Figure 5-37)

A fishing license for commercial use is taxable whereas a fishing license for sport use is not taxable. Furthermore, there may be other attributes for a fishing license for sport use that are not applicable to fishing license for commercial use such as the number of years a fishing license is given to an address. As there are two uses or special circumstances, two subtypes are introduced..

2.5 Recursive Relationship

A recursive relationship is a relationship that an entity has with itself. This means that an instance of the entity is related to another instance of the same entity.

2.5.1 1:1 Recursive Relationship

This pattern is used when the maximum cardinality on both ends of the recursive relationship is 1:1. This means there is at most one instance of the entity is related to each instance of the same entity.

An example of an application that has a 1:1 recursive relationship is one for marriage registration in a monogamous society. The 1:1 recursive relationship between adults is 'is married to', as shown in Figure 2.19. An adult can be married to at most one adult. Note that some adults can be unmarried, therefore the minimum cardinality on both ends is optional.

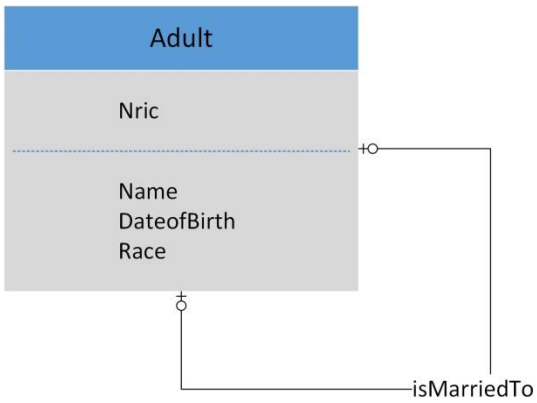


Figure 2.19 1:1 recursive relationship

2.5.2 1:N Recursive Relationship

This pattern is used when the maximum cardinality on one end of the recursive relationship is 1, and the maximum cardinality on the other end of the recursive relationship is N or many. This means there is at most one instance of the entity is

related to each instance of the same entity on one end, and there can be many instances of the entity is related to each instance of the same entity on other end.

An example of an application that has a 1:N recursive relationship is one for leave application where a manager who is an employee, needs to approve the leave application of employees he manages. The 1:N recursive relationship between employees is 'manages', as shown in Figure 2.20. An adult can be married to at most one adult.

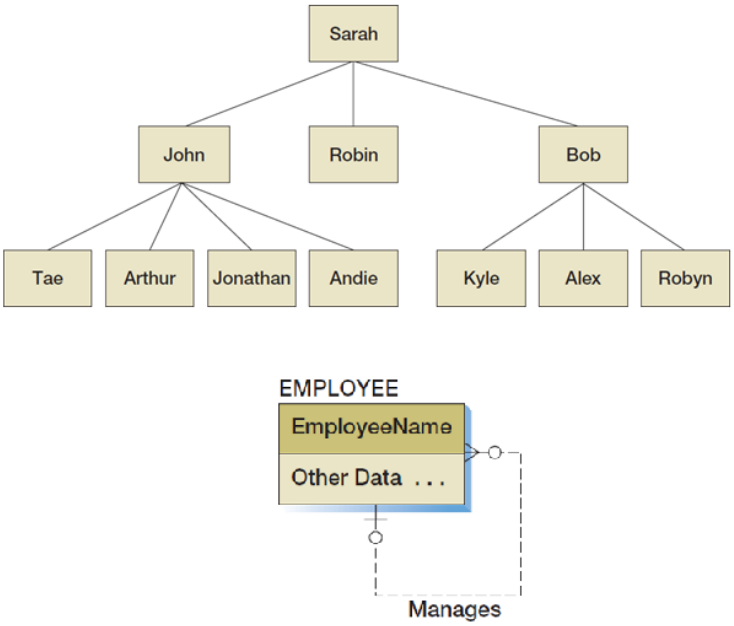


Figure 2.20 1:N recursive relationship

(Source: Kroenke, D and D. Auer. (2016). Database Processing: Fundamentals, Design and Implementation Edition 14. Pearson, Figure 5-40 and Figure 5-41)

2.5.3 N:M Recursive Relationship

This pattern is used when the maximum cardinality on both ends of the recursive relationship is many. This means there can be many instances of the entity that related to each instance of the same entity.

An example of a manufacturing application in Figure 2.21 has a N:M recursive relationship.

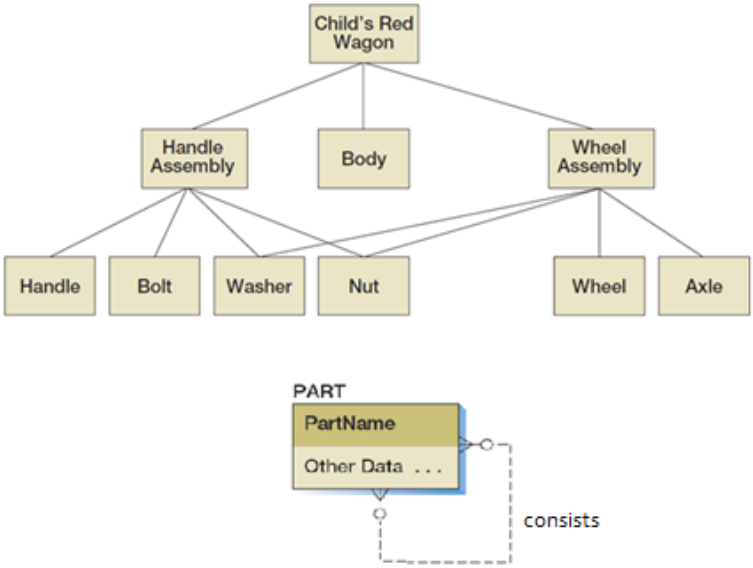


Figure 2.21 N:M recursive relationship

(Source: Kroenke, D and D. Auer. (2016). Database Processing: Fundamentals, Design and Implementation Edition 14. Pearson, Figure 5-42 and Figure 5-43)

The N:M recursive relationship between parts is 'consists', A part can consist many parts. A part can also be consisted in many parts.

Read

Kroenke, D and D. Auer. (2016). *Database Processing: Fundamentals, Design and Implementation Edition 14*. Pearson, 256-260.

Activity 9

Modified from Question 5.54 of the course text.

Represent the recursive relationships given the data requirements"

a) 1:1 Recursive:

   To be a member of the New City Club, you must have a sponsor, who can sponsor at most one other member. Therefore each member is associated with exactly one sponsor, who is also a member with a sponsor

b) 1:N Recursive:

   Customers of the All Stuff Warehouse may refer one or more other customers to the business. Therefore, there are many referred customers with exactly one referring customer, who may also have been referred by another customer. However, customers may not have been referred, and customers do not have to make referrals.

c) N:M Recursive:

   In the medical community in New Town, doctors treat each other. Given the number of medical specialties, it is no wonder that any one doctor is treated by many other doctors, while at the same time that doctor has treated many other doctors him- or herself. However, a doctor does not have to be treated by another doctor in the community, nor does the doctor have to be treating another community doctor.

Quiz

1. Which statement about entity is false?

a. An entity is something that users want to track.

b. An entity instance is the occurrence of a particular entity.

*c. A group of entity instance of a given type is called an entity class.

d. An identifier of an entity instance is one or more attributes that name or identify entity instances.

2. You are given an E-R diagram with two entities, ORDER and CUSTOMER, and are asked to draw the relationship between them.

If a given customer can place zero or more orders and a given order must be placed by exactly one customer, what is the maximum cardinality for relationship between the two entities ORDER:CUSTOMER?

a. 1:N

*b. N:1

c. 0:1

d. 1:1

3. Which statement about minimum cardinality is false?

*a. A relationship's minimum cardinality indicates the minimum number of entity instances that can participate in the relationship.

b. A relationship's minimum cardinality indicates whether or not an entity must participate in the relationship.

c. In an E-R model, the two types of minimum cardinality are mandatory or optional.

d. In an E-R model, minimum cardinality is represented by either a 0 or 1.

4. Give the term for an entity whose existence depends on the presence of another entity, but whose identifier does not include the identifier of the other entity.

a. Strong entity

*b. Weak entity

c. ID-dependent entity

d. Weak and ID-dependent

5. Which statement is false about subtype entities?

a. Subtypes may be exclusive or inclusive.

b. The supertype and subtypes will have the same identifier.

c. Subtypes are used to avoid a situation in which some attributes are required to be null.

*d. Always add a discriminator to the supertype class to determine which subtype should be used.

6. Which statement is false about design patterns?

a. They are the common patterns observed in forms and reports.

b. They are time-tested patterns allowing us to map forms and reports to the data model.

c. They help reveal what missing information should be gathered from the users.

*d. The structure of the data model does not determine the patterns in users' forms and reports.

Formative Assessment

1. Give the pattern best suited for this scenario:

A Class (with attributes: subject name, level, duration) may be taught in different Offering (with attributes: day, time, tutor)

a. Archetype/instance pattern

Correct! Class is the archetype and the Offering is the instance. Refer to textbook page 252.

b. 1:N strong entity relationship pattern

Incorrect. Offering is a weak entity, depending on the presence of Class. Refer to textbook page 245.

c. Association pattern

Incorrect. Association pattern involves three entities. Refer to textbook page 247-248.

d. Multivalued attribute pattern

Incorrect. Offering is not an attribute of Class. Refer to textbook page 249.

2. Give the pattern best suited for this scenario:

A Course may have several prerequisites. Each prerequisite is itself a course and may be a prerequisite for several courses.

a. Archetype/instance

Incorrect. Prerequisite is not an instance of Course. Refer to textbook page 252.

b. N:M recursive relationship

Correct! A Prerequisite is also a Course. So, the relationship is recursive. Course has many prerequisites and prerequisite has many courses. Refer to textbook pages 246-247.

c. Association pattern

Incorrect. Association pattern involves three entities. Refer to textbook pages 247-248.

D. Multivalued attribute pattern

Incorrect. Prerequisite is not an attribute of Course. Refer to textbook page 249

3. How is an association pattern represented in an E-R model?

a. As an ID-dependent entity with a 1:1 relationship to one other entity.

Incorrect. Association pattern involves two parent entities. Refer to textbook pages 247-248.

b. As a weak but not ID-dependent entity with a 1:1 relationship to one other entity

Incorrect. Association pattern involves three entities. Refer to textbook pages 247-248.

c. As a strong entity with a 1:1 relationship to one other entity

Incorrect. Association pattern involves three entities. Refer to textbook pages 247-248.

d. As an ID-dependent entity with a 1:N relationship to each one of its two parent entities

Correct! Association pattern involves three entities. Refer to textbook pages 247-248.

4. What type of relationship does one get when an entity has a relationship with itself?

a. Supertype/subtype relationship

Incorrect. An entity cannot be the supertype and the subtype at the same time. Refer to textbook pages 256-257.

b. Archetype/instance relationship

Incorrect. An entity cannot be the archetype and instance at the same time. Refer to textbook pages 252-253.

c. Recursive relationship

Correct! The relationship is defined as recursive when an entity relates to itself. Refer to textbook page 257.

d. Association pattern

Incorrect. Association pattern involves three entities. Refer to textbook pages 247-248.

5. Which scenario displays the Line-Item pattern?

a. A salesperson makes a sale to a customer.

Incorrect. This is an association pattern where salesperson and customer are the strong entities and sale is a weak entity. Refer to textbook pages 247-248.

b. A quotation for a list of cleaning services.

Correct! A quotation typically has data about the quotation itself, such as the validity period, data about the requester of the quotation, the supplier of the quotation and then data about each cleaning service the quotation is made for. Refer to textbook page 254.

c. A list of grades for the various courses taken by students.

Incorrect. This is an association pattern where student and course are the strong entities and grade is a weak entity. Refer to textbook pages 247-248.

d. A list of parts making a part.

Incorrect. Entity that relates to itself is defined as recursive. Refer to textbook page 257.

Summary

In this study unit, we map data requirements into a data model. As a tool for analyzing and verifying data requirements, a data model must not include foreign keys in the entities. The relationships are depicted by lines between the entities.

An entity is something for which data must be recorded. Strong entities do not depend on other entities to exist unlike weak entities.

Two entities are related when their instances are related. A relationship can be identifying or non-identifying. The maximum and minimum cardinalities tell us many instances of another entity that one instance of an entity may relate to.

There are design patterns that can be used in data modelling. In particular, there are design patterns for relationship between strong entities, between strong and weak entities and within the same entity.

Book

Author(s)	Year	Book Title	Edition	Publisher
Kroenke, D., & Auer, D. J.	2016	*Database Processing – Fundamentals, Design, and Implementation*	14	Pearson

Solutions or Suggested Answers

Activity 1

Before a building is actually constructed, it is carefully planned and designed. That work is documented in the building blueprint. Similarly, before a database is actually created in a DBMS, it needs to be carefully planned and designed. The work of planning and designing a database is documented in a data model.

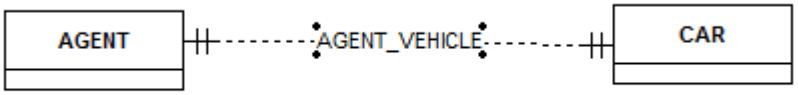
The advantage of making changes during the data modelling stage is that it is easier, simpler, faster, and cheaper to make changes at that stage of database development.

Activity 2

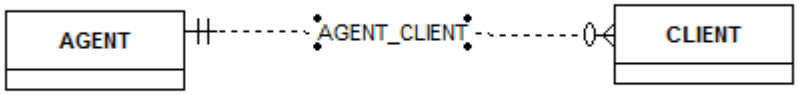
An entity is something that the users want to track, and is readily identifiable in their environment. Real Estate Agency example entities are agent, property and client.

Activity 3

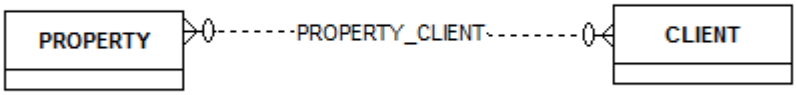
a)



b)



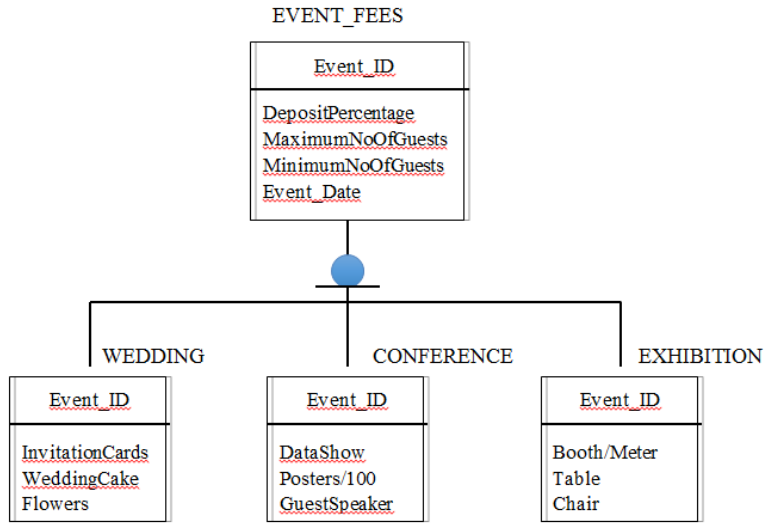
c)



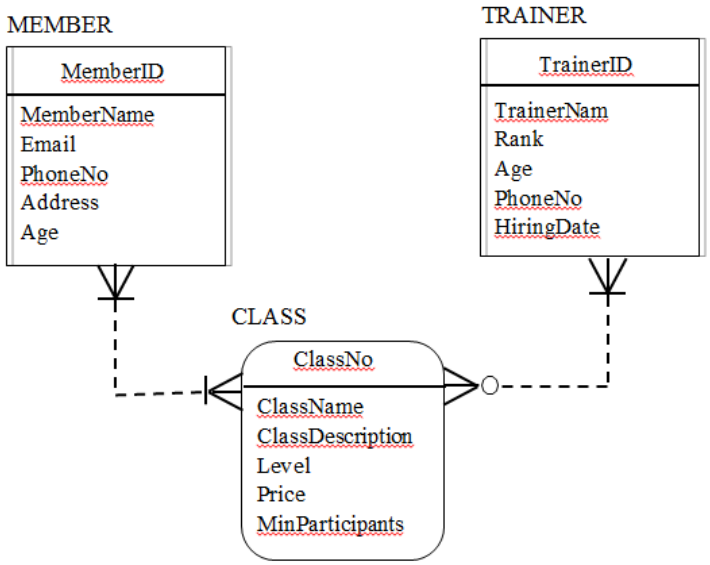
Activity 4

A strong entity that has a required relationship with another entity and, it can and will exist without the presence of the other, strong entity. A weak entity cannot and does not exist without the presence of the other, strong entity.

Activity 5

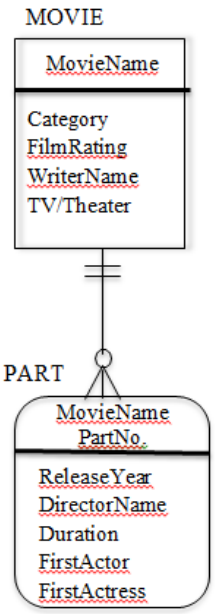


Activity 6

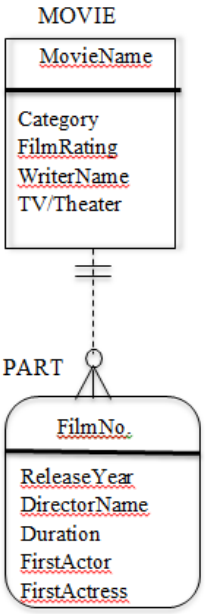


Activity 7

Movie is the archetype and part is the instance.

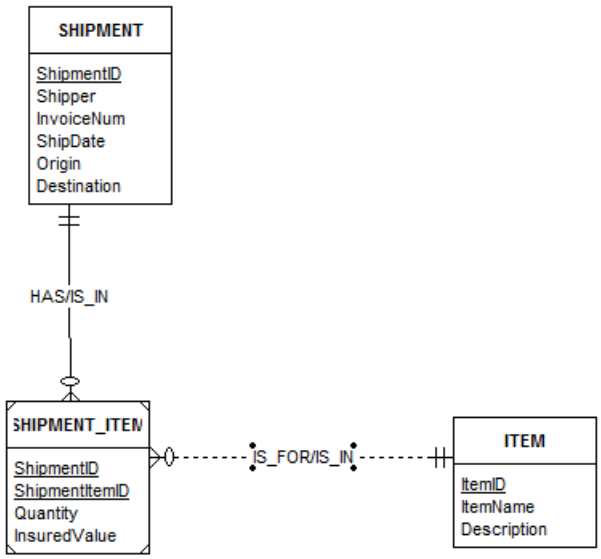


PART: ID-Dependent Entity



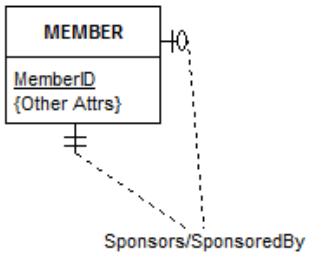
PART: Non-ID-Dependent Entity, Weak Entity

Activity 8

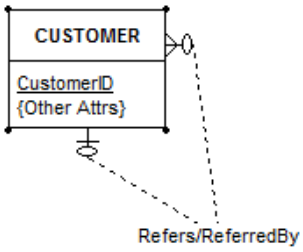


Activity 9

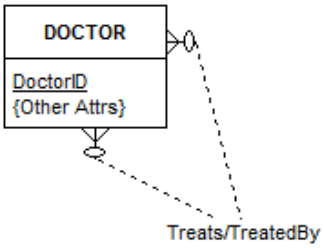
a)



b)


CUSTOMER
CustomerID
{Other Attrs}
Refers/ReferredBy

c)


DOCTOR
DoctorID
{Other Attrs}
Treats/TreatedBy