Homework1

Ruili Yao

2021146833

Chapter1

1.1. Define the following terms: data, database, DBMS, database system, database catalog, program-data independence, user view, DBA, end user, canned transaction, deductive database system, persistent object, meta-data, and transaction-processing application.

**Data:** known facts that can be recorded and that have implicit meaning.

**Database:** Database is a collection of related data or operational data extracted from any firm or organization. In other words, a collection of organized data is called database.

**DBMS (Database Management System):** A database management system (DBMS) is a collection of programs that enables users to create and maintain a database. The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating, and sharing databases among various users and applications.

**Database Systems:** A database system comprises a database of operational data, together with processing functionality required to access and manage that data. The combination of the DBMS and the database is called database systems.

**Database Catalog:** A database catalog contains complete description of the databases, database objects, database structure, details of users and constraints etc. that are stored. This definition is stored in the DBMS catalog, which contains information such as the structure of each file, the type and storage format of each data item, and various constraints on the data.

**Program-data independence:** In traditional file processing, the structure of data files is embedded in the application programs, so any changes to the structure of a file may require changing all programs that access that file. By contrast, DBMS access programs do not require such changes in most cases. The structure of data files is stored in the DBMS catalog separately from the access programs. We call this property program-data independence.

**User view:** A database typically has many users, each of whom may require a different perspective or view of the database. A view may be a subset of the database or it may contain virtual data that is derived from the database files but is not explicitly stored.

**DBA (database administrator):** The DBA is responsible for authorizing access to the database, coordinating and monitoring its use, and acquiring software and hardware resources as needed. The DBA is accountable for problems such as security breaches and poor system response time. In large organizations, the DBA is assisted by a staff that carries out these functions.

**End user:** End users are the people whose jobs require access to the database for querying, updating, and generating reports; the database primarily exists for their use.

**Canned transaction:** Naïve or parametric end users make up a sizable portion of database end users. Their main job function revolves around constantly querying and updating the database, using standard types of queries and updates—called canned transactions.

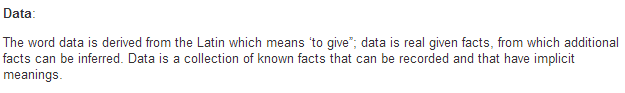
**Deductive database system:** Some database systems provide capabilities for defining deduction rules for inferencing new information from the stored database facts. Such systems are called deductive database systems.

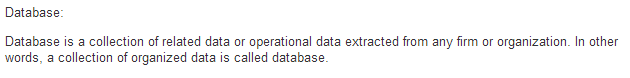
**Persistent object:** Persistent objects are stored in the database and persist after program termination. The typical mechanisms for making an object persistent are naming and reachability.

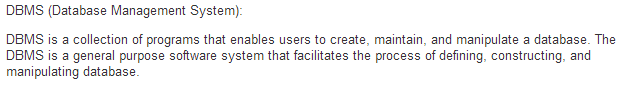
**Meta-data:** The information stored in the catalog is called meta-data, and it describes the structure of the primary database.

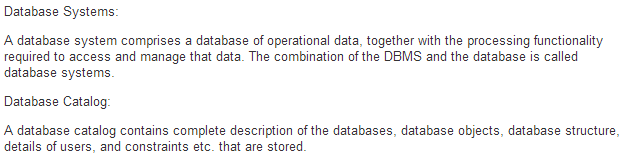
**Transaction-processing application:** The DBMS must include concurrency control software to ensure that several users trying to update the same data do so in a controlled manner so that the result of the updates is correct. For example, when several reservation agents try to assign a seat on an airline flight, the DBMS should ensure that each seat can be accessed by only one agent at a time for assignment to a passenger. These types of applications are generally called online transaction processing (OLTP) applications.

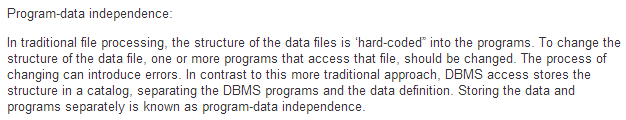
As soon as one user’s update is applied to the database, all other users can immediately see this update. This availability of up-to-date information is essential for many transaction-processing applications.



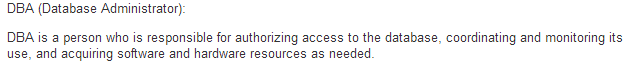


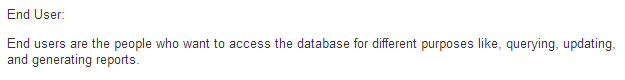




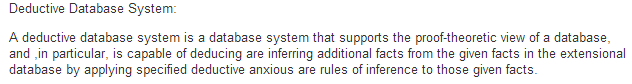


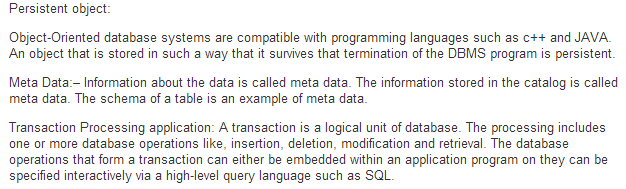












1.3. Discuss the main characteristics of the database approach and how it differs from traditional file systems.

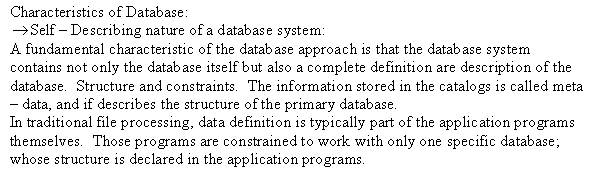
The main characteristics of the database approach versus the file-processing approach are the following:

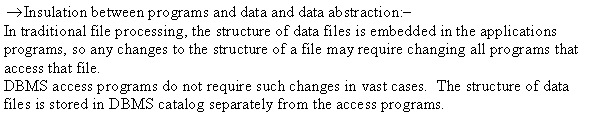
Self-describing nature of a database system

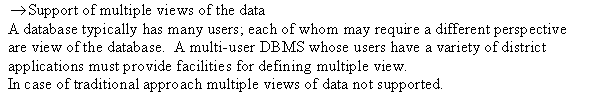
Insulation between programs and data, and data abstraction

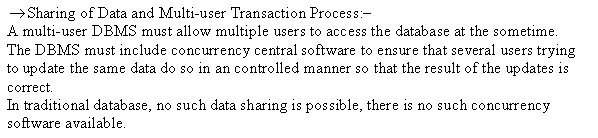
Support of multiple views of the data

Sharing of data and multiuser transaction processing









1.5. What are the different types of database end users? Discuss the main activities of each.

There are several categories of end users:

Casual end users occasionally access the database, but they may need different information each time. They use a sophisticated database query language to specify their requests and are typically middle- or high-level managers or other occasional browsers.

Naïve or parametric end users make up a sizable portion of database end users. Their main job function revolves around constantly querying and updating the database, using standard types of queries and updates—called canned transactions—that have been carefully programmed and tested. The tasks that such users perform are varied:

Bank tellers check account balances and post withdrawals and deposits.

Reservation agents for airlines, hotels, and car rental companies check availability for a given request and make reservations.

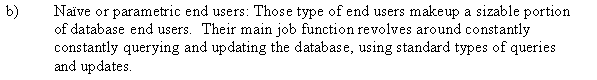
Employees at receiving stations for shipping companies enter package identifications via bar codes and descriptive information through buttons to update a central database of received and in-transit packages.

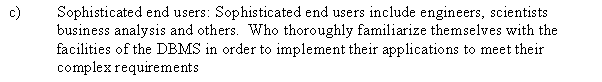
Sophisticated end users include engineers, scientists, business analysts, and others who thoroughly familiarize themselves with the facilities of the DBMS in order to implement their own applications to meet their complex requirements.

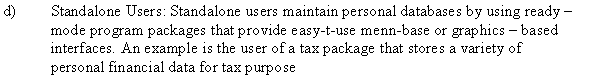
Standalone users maintain personal databases by using ready-made program packages that provide easy-to-use menu-based or graphics-based interfaces. An example is the user of a tax package that stores a variety of personal financial data for tax purposes.











1.6. Discuss the capabilities that should be provided by a DBMS.

Controlling Redundancy

Restricting Unauthorized Access

Providing Persistent Storage for Program Objects

Providing Storage Structures and Search Techniques for Efficient Query Processing

Providing Backup and Recovery

Providing Multiple User Interfaces

Representing Complex Relationships among Data

Enforcing Integrity Constraints

Permitting Inferencing and Actions Using Rules

Additional Implications of Using the Database Approach:

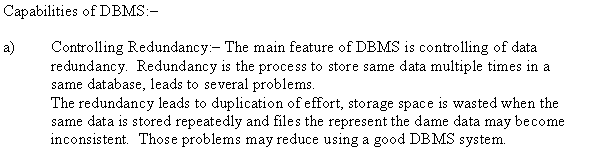
Potential for Enforcing Standards.

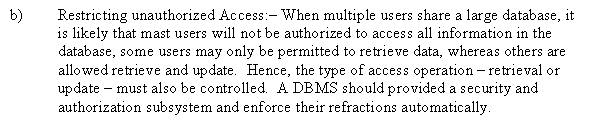
Reduced Application Development Time.

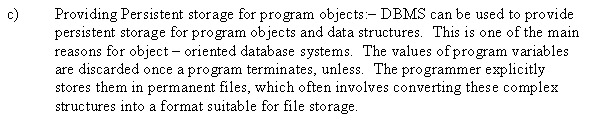
Flexibility.

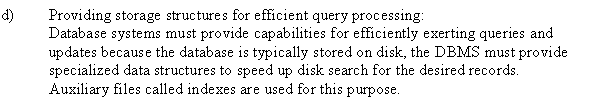
Availability of Up-to-Date Information.

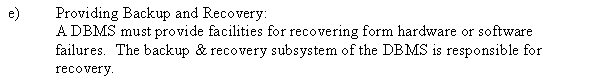
Economies of Scale.

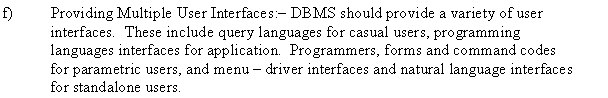


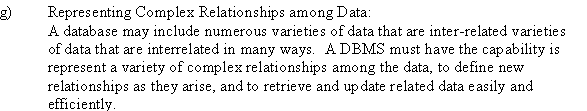


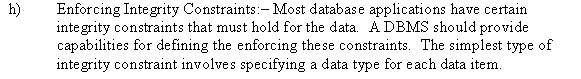






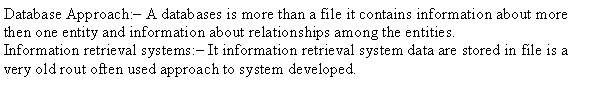






1.7. Discuss the differences between database systems and information retrieval systems.

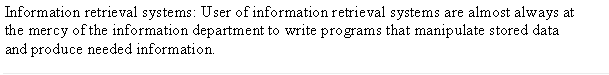
Database technology applies to structured and formatted data that arises in routine applications in government, business, and industry. Database technology is heavily used in manufacturing, retail, banking, insurance, finance, and health care industries, where structured data is collected through forms, such as invoices or patient registration documents. An area related to database technology is Information Retrieval (IR), which deals with books, manuscripts, and various forms of library-based articles. Data is indexed, cataloged, and annotated using keywords. IR is concerned with searching for material based on these keywords, and with the many problems dealing with document processing and free-form text processing. There has been a considerable amount of work done on searching for text based on keywords, finding documents and ranking them based on relevance, automatic text categorization, classification of text documents by topics, and so on. With the advent of the Web and the proliferation of HTML pages running into the billions, there is a need to apply many of the IR techniques to processing data on the Web. Data on Web pages typically contains images, text, and objects that are active and change dynamically. Retrieval of information on the Web is a new problem that requires techniques from databases and IR to be applied in a variety of novel combinations.





























1.8. Identify some informal queries and update operations that you would expect to apply to the database shown in Figure 1.2.

Informal queries:

1. Retrieve the transcript - a list of all courses and grades – of ‘smith’
2. List the name of students who took the section of the ‘Database’ course offered in fall 2005 and their grades in that section.
3. List the prerequisites of the ’Database’ course

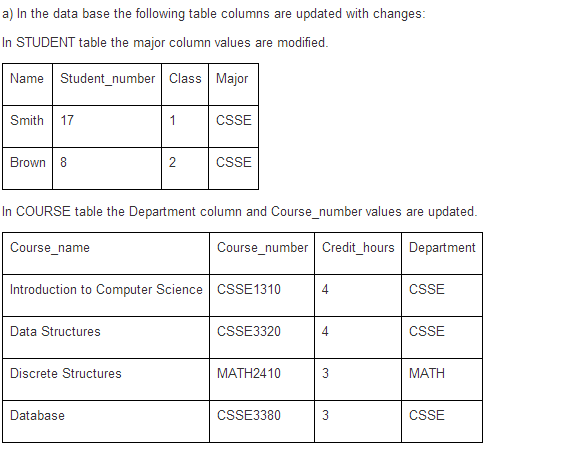
Update operations:

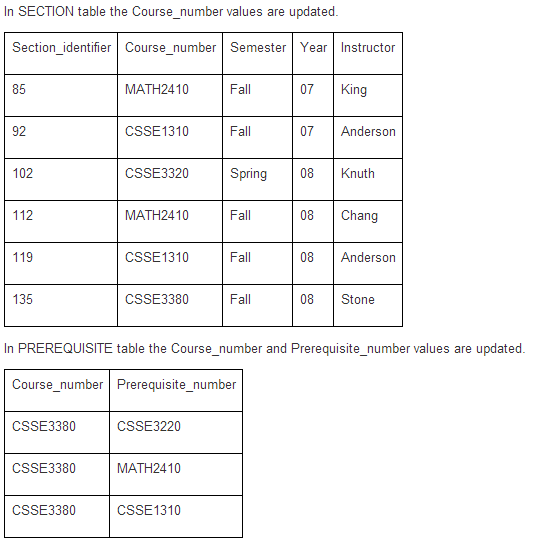
1. Change the class of ‘Smith’ to sophomore
2. Create a new section for the ‘Database’ course for this semester.
3. Enter a grade of ’A’ for ‘Smith’ in the ‘Database’ section of last semester

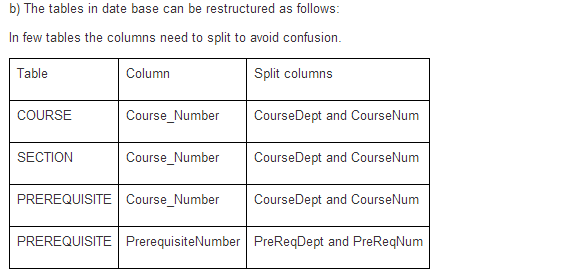
1.14. Consider Figure 1.2.

a. If the name of the ‘CS’ (Computer Science) Department changes to ’CSSE’ (Computer Science and Software Engineering) Department and the corresponding prefix for the course number also changes, identify the columns in the database that would need to be updated.

b. Can you restructure the columns in the COURSE, SECTION, and PREREQUISITE tables so that only one column will need to be updated?







Chapter 2

2.1. Define the following terms: data model, database schema, database state, internal schema, conceptual schema, external schema, data independence, DDL, DML, SDL, VDL, query language, host language, data sublanguage, database utility, catalog, client/server architecture, three-tier architecture, and n-tier architecture.

**Data model:** A data model—a collection of concepts that can be used to describe the structure of a database—provides the necessary means to achieve this abstraction.

**Database schema:** In any data model, it is important to distinguish between the description of the database and the database itself. The description of a database is called the database schema.

**Database state:** The data in the database at a particular moment in time is called a database state or snapshot.

**Internal schema:** The internal level has an internal schema, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

**Conceptual schema:** The conceptual level has a conceptual schema, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.

**External schema:** The external or view level includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group.

**Data independence:** Data independence can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level.

**DDL:** In many DBMSs where no strict separation of levels is maintained, one language, called the data definition language (DDL), is used by the DBA and by database designers to define both schemas. The DDL is used to specify the conceptual schema only. In most DBMSs the DDL is used to define both conceptual and external schemas.

**DML:** The DBMS provides a set of operations or a language called the data manipulation language (DML) for retrieval, insertion, deletion, and modification of the data.

SDL: Another language, the storage definition language (SDL), is used to specify the internal schema.

**VDL:** The view definition language (VDL) is used to specify user views and their mappings to the conceptual schema.

**Query language:** a high-level DML used in a standalone interactive manner is called a query language.

**Host language:** Whenever DML commands, whether high level or low level, are embedded in a general-purpose programming language, that language is called the host language.

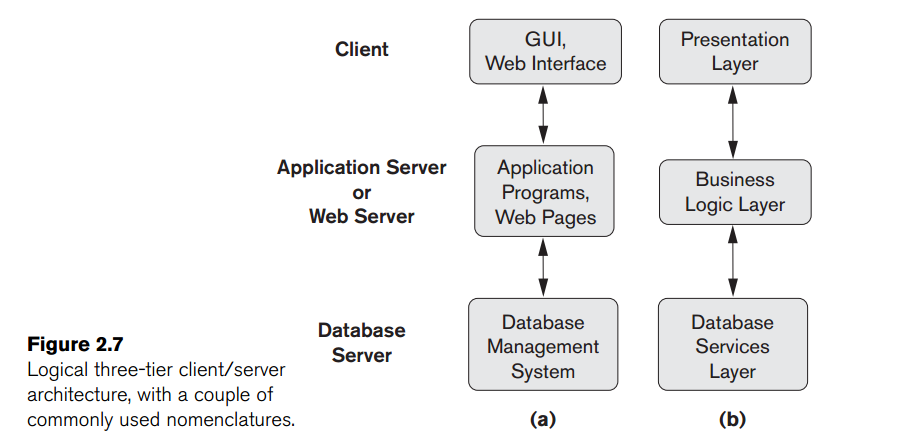
**Data sublanguage:** The DML is called the data sublanguage.

**Database utility:** Most DBMSs have database utilities that help the DBA manage the database system.

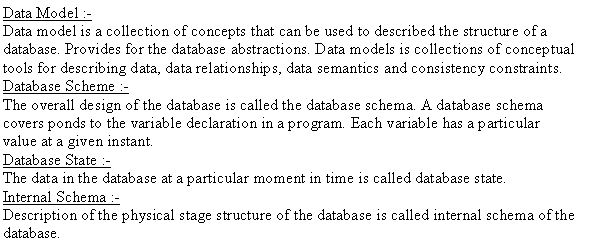
**Catalog:** In the lower part of Figure 2.3, the runtime database process or executes (1) the privileged commands, (2) the executable query plans, and (3) the canned transactions with runtime parameters. It works with the system catalog and may update it with statistics.

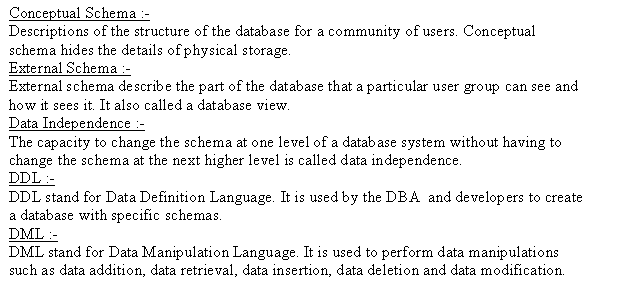
**Client/server architecture:** The client/server architecture was developed to deal with computing environments in which a large number of PCs, workstations, file servers, printers, database servers, Web servers, e-mail servers, and other software and equipment are connected via a network.

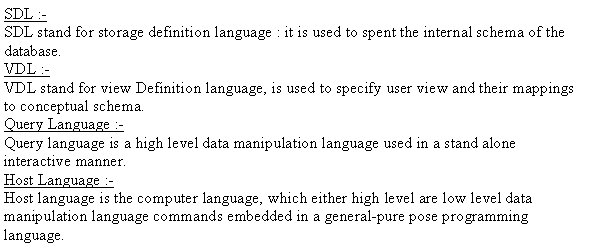
**Three-tier architecture:** Many Web applications use an architecture called the three-tier architecture, which adds an intermediate layer between the client and the database server, as illustrated in Figure 2.7(a).

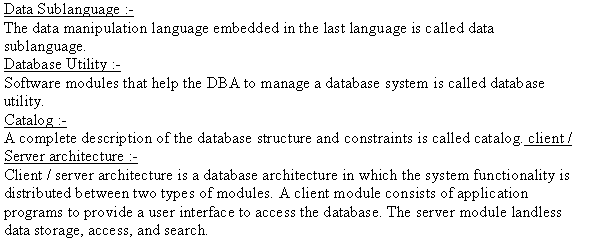


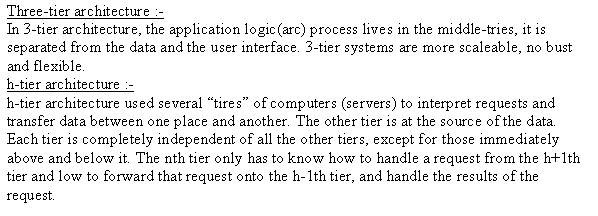
**N-tier architecture:** It is possible to divide the layers between the user and the stored data further into finer components, thereby giving rise to n-tier architectures, where n may be four or five tiers.









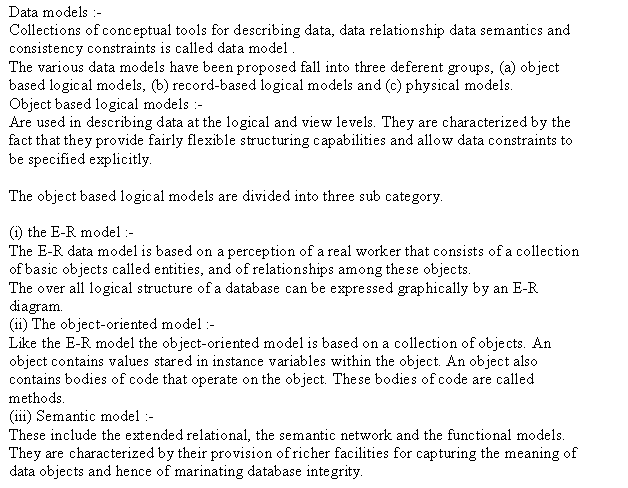


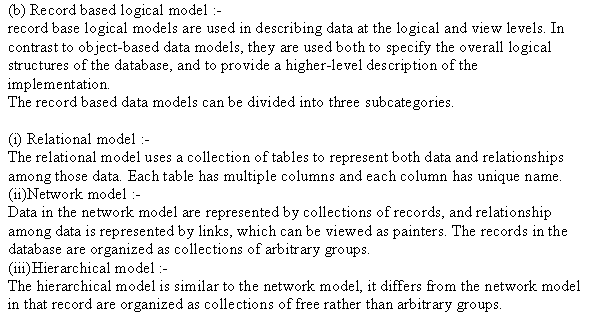
2.2. Discuss the main categories of data models. What are the basic differences between the relational model, the object model, and the XML model?

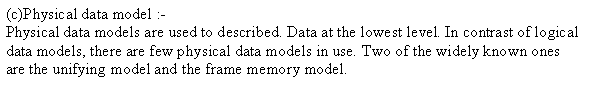
The basic relational data model represents a database as a collection of tables, where each table can be stored as a separate file. Most relational databases use the high-level query language called SQL and support a limited form of user views.

The object data model defines a database in terms of objects, their properties, and their operations. Objects with the same structure and behavior belong to a class, and classes are organized into hierarchies (or acyclic graphs). The operations of each class are specified in terms of predefined procedures called methods. Relational DBMSs have been extending their models to incorporate object database concepts and other capabilities; these systems are referred to as object-relational or extended relational systems.

The XML model has emerged as a standard for exchanging data over the Web, and has been used as a basis for implementing several prototype native XML systems. XML uses hierarchical tree structures. It combines database concepts with concepts from document representation models. Data is represented as elements; with the use of tags, data can be nested to create complex hierarchical structures. This model conceptually resembles the object model but uses different terminology. XML capabilities have been added to many commercial DBMS products.

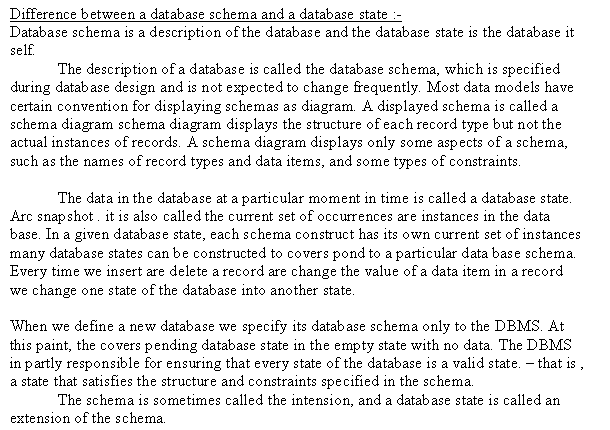






2.3. What is the difference between a database schema and a database state?

When we define a new database, we specify its database schema only to the DBMS. At this point, the corresponding database state is the empty state with no data. We get the initial state of the database when the database is first populated or loaded with the initial data. From then on, every time an update operation is applied to the database, we get another database state. At any point in time, the database has a current state. The DBMS is partly responsible for ensuring that every state of the database is a valid state—that is, a state that satisfies the structure and constraints specified in the schema. Hence, specifying a correct schema to the DBMS is extremely important and the schema must be designed with utmost care. The DBMS stores the descriptions of the schema constructs and constraints—also called the meta-data—in the DBMS catalog so that DBMS software can refer to the schema whenever it needs to. The schema is sometimes called the intension, and a database state is called an extension of the schema.



2.4. Describe the three-schema architecture. Why do we need mappings between schema levels? How do different schema definition languages support this architecture?

1. The internal level has an internal schema, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.

2. The conceptual level has a conceptual schema, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.

Usually, a representational data model is used to describe the conceptual schema when a database system is implemented. This implementation conceptual schema is often based on a conceptual schema design in a high-level data model.

3. The external or view level includes a number of external schemas or user views. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group. As in the previous level, each external schema is typically implemented using a representational data model, possibly based on an external schema design in a high-level data model.

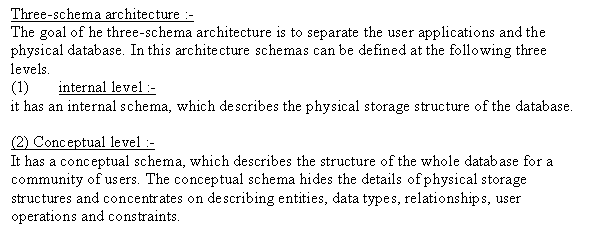
In a DBMS based on the three-schema architecture, each user group refers to its own external schema. Hence, the DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database. If the request is a database retrieval, the data extracted from the stored database must be reformatted to match the user’s external view. The processes of transforming requests and results between levels are called mappings. These mappings may be time-consuming, so some DBMSs—especially those that are meant to support small databases—do not support external views. Even in such systems, however, a certain amount of mapping is necessary to transform requests between the conceptual and internal levels.

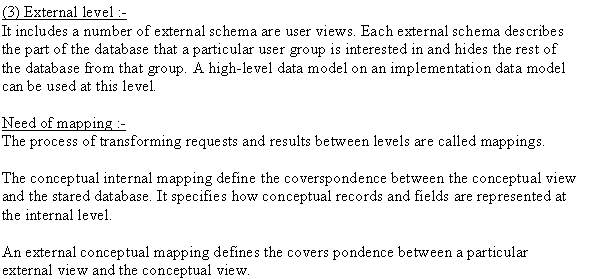
In many DBMSs where no strict separation of levels is maintained, one language, called the data definition language (DDL), is used by the DBA and by database designers to define both schemas.

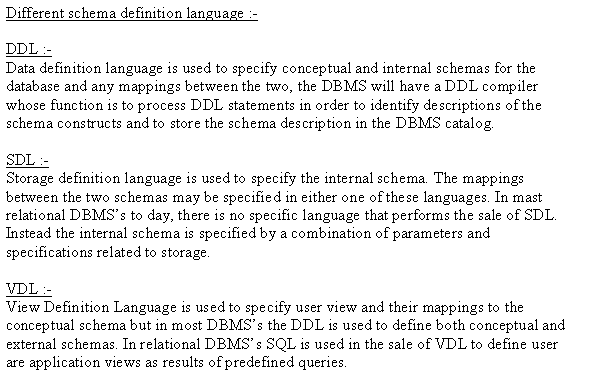
In DBMSs where a clear separation is maintained between the conceptual and internal levels, the DDL is used to specify the conceptual schema only. Another language, the storage definition language (SDL), is used to specify the internal schema. The mappings between the two schemas may be specified in either one of these languages.

For a true three-schema architecture, we would need a third language, the view definition language (VDL), to specify user views and their mappings to the conceptual schema, but in most DBMSs the DDL is used to define both conceptual and external schemas.

The DBMS provides a set of operations or a language called the data manipulation language (DML) for retrieval, insertion, deletion, and modification of the data.





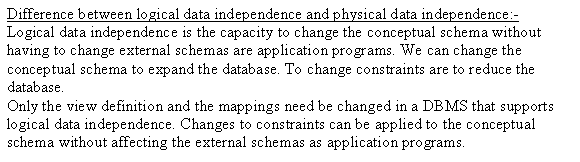


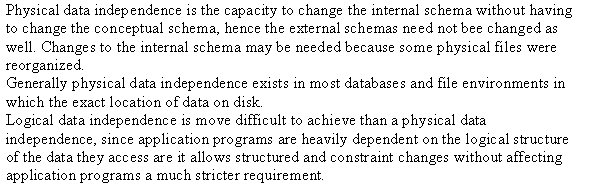
2.5. What is the difference between logical data independence and physical data independence? Which one is harder to achieve? Why?

1. Logical data independence is the capacity to change the conceptual schema without having to change external schemas or application programs. We may change the conceptual schema to expand the database (by adding a record type or data item), to change constraints, or to reduce the database (by removing a record type or data item). Only the view definition and the mappings need to be changed in a DBMS that supports logical data independence. After the conceptual schema undergoes a logical reorganization, application programs that reference the external schema constructs must work as before. Changes to constraints can be applied to the conceptual schema without affecting the external schemas or application programs.
2. Physical data independence is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schemas need not be changed as well. Changes to the internal schema may be needed because some physical files were reorganized—for example, by creating additional access structures—to improve the performance of retrieval or update. If the same data as before remains in the database, we should not have to change the conceptual schema.

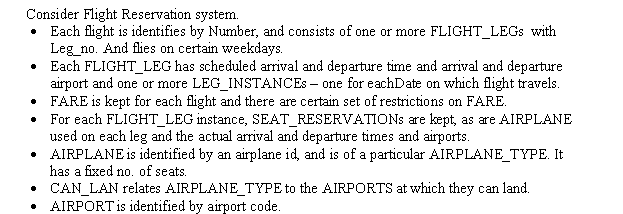
Generally, physical data independence exists in most databases and file environments where physical details such as the exact location of data on disk, and hardware details of storage encoding, placement, compression, splitting, merging of records, and so on are hidden from the user. Applications remain unaware of these details.

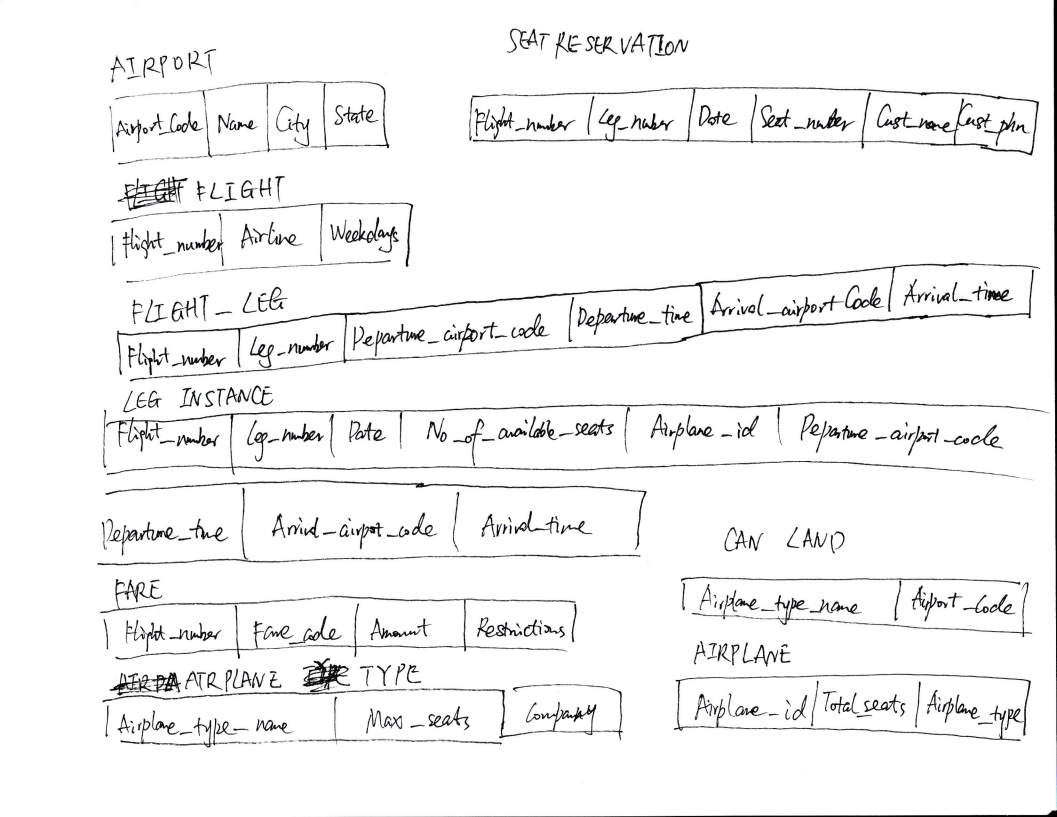
On the other hand, logical data independence is harder to achieve because it allows structural and constraint changes without affecting application programs—a much stricter requirement.

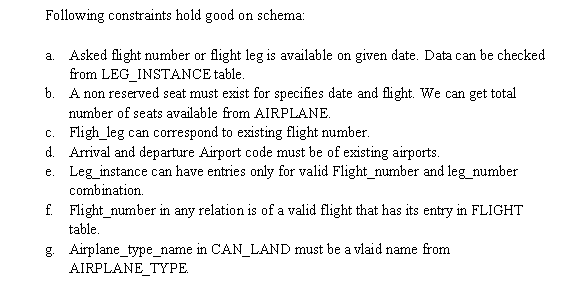




2.13. Choose a database application with which you are familiar. Design a schema and show a sample database for that application, using the notation of Figures 1.2 and 2.1. What types of additional information and constraints would you like to represent in the schema? Think of several users of your database, and design a view for each.





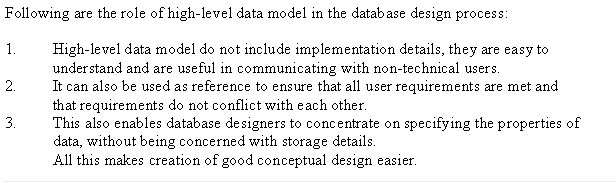


Chapter 7

7.1. Discuss the role of a high-level data model in the database design process.

Once the requirements have been collected and analyzed, the next step is to create a conceptual schema for the database, using a high-level conceptual data model. This step is called conceptual design. The conceptual schema is a concise description of the data requirements of the users and includes detailed descriptions of the entity types, relationships, and constraints; these are expressed using the concepts provided by the high-level data model. Because these concepts do not include implementation details, they are usually easier to understand and can be used to communicate with nontechnical users. The high-level conceptual schema can also be used as a reference to ensure that all users’ data requirements are met and that the requirements do not conflict. This approach enables database designers to concentrate on specifying the properties of the data, without being concerned with storage and implementation details. This makes it is easier to create a good conceptual database design.

During or after the conceptual schema design, the basic data model operations can be used to specify the high-level user queries and operations identified during functional analysis. This also serves to confirm that the conceptual schema meets all the identified functional requirements. Modifications to the conceptual schema can be introduced if some functional requirements cannot be specified using the initial schema.



7.3. Define the following terms: entity, attribute, attribute value, relationship instance, composite attribute, multivalued attribute, derived attribute, complex attribute, key attribute, and value set (domain).

**Entity:** The basic object that the ER model represents is an entity, which is a thing in the real world with an independent existence. An entity may be an object with a physical existence (for example, a particular person, car, house, or employee) or it may be an object with a conceptual existence (for instance, a company, a job, or a university course).

**Attribute:** The particular properties that describe it. For example, an EMPLOYEE entity may be described by the employee’s name, age, address, salary, and job.

**Attribute value:** Attribute value is the real data of a particular entity for each of its attributes. In other word, associated with each real world entities are certain attributes that describe that entity; value of these attributes for any entity is called attribute value. For example, attribute value of first\_name of attribute of student\_name can be Gyanendra.

**Relationship instance:** Each relationship instance r in R is an association of entities, where the association includes exactly one entity from each participating entity type. Each such relationship instance r represent the fact that the entities participating in rare related in some way in the corresponding miniworld situation. For example, in relationship type WORKS\_FOR associates one EMPLOYEE and DEPARTMENT, which associates each employee with the department for which the employee works. Each relationship instance in the relationship set WORKS\_FOR associates one EMPLOYEE and one DEPARTMENT.

**Composite attribute:** Composite attribute is an attribute that can be divided into smaller subparts, which represent more basic attributes with dependent meanings, is called a composite attribute. For example, the Address attribute consists of several domains such as house number, street number, city, country, etc.

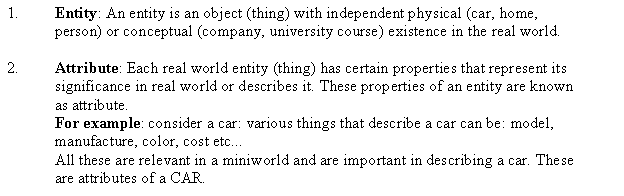
**Multivalued attribute:** A multi-valued attribute can have more than one value at one time. For example, address of a person is a multi-valued attribute since a person can have more than one address such as Present and Permanent address. Upper and lower bounds may be placed on the number of values in a multi-valued attribute. For example, a bank may limit the number of addresses recorded for a single customer to two.

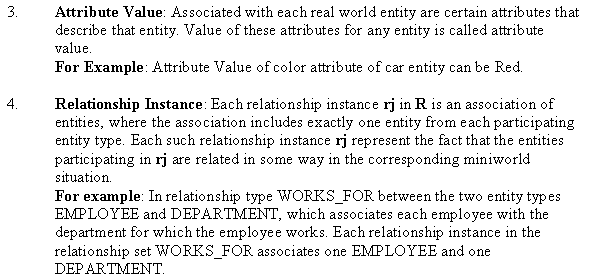
**Derived attribute:** If an attribute’s value can be determined from the values of other attributes, then the attribute is derivable, and is said to be a derived attribute. Example: consider attributes for an employee: birth date, current age; here, age is derivable by subtracting the birth date from the current date.

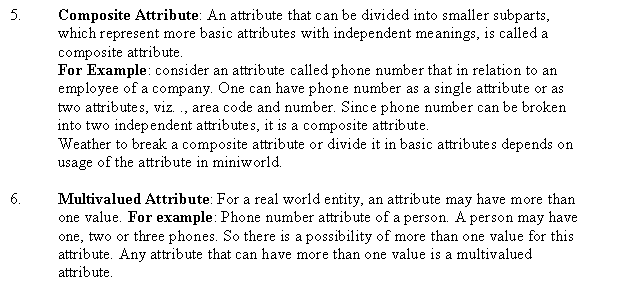
**Complex attribute:** Composite and multivalued attribute can be nested arbitrarily. Arbitrary nesting can be represented by grouping components of a composite attribute between parentheses and separating the components with commas and by displaying multivalued attributes between braces. Such attributes are called composite attributes. For example, if a person has more than one address and each residence has multiple phones and address\_phone attribute can be specifies as: {AddressPhone( (Phone{AreaCode,PhoneNumber)},Address(StreetAddress(Number, Street, ApartmentNumber), City, State, Zip) ) }

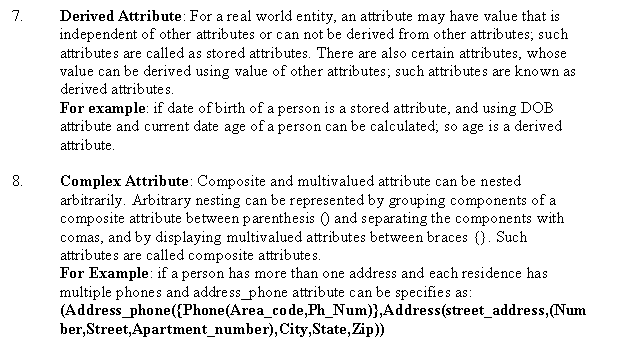
**Key attribute:** Each real world entity is unique in itself. There are certain attributes whose value is different for all similar type of entities. Those attributes are called key attributes. These attributes are used to specify uniqueness constraint in relation. For example, a house has a registration number. This is a key of all entity of house.

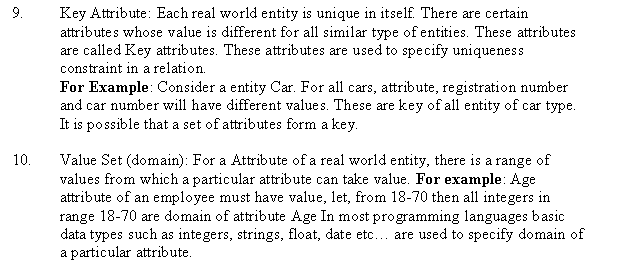
**Value set (domain):** There is a range of values from which a particular attribute can take value for a attribute of a real world entity. For example, salary attribute of an employee must have value, let, from $2000 to $12000, and then all integers in range $2000 to $12000 are domain of attribute salary.











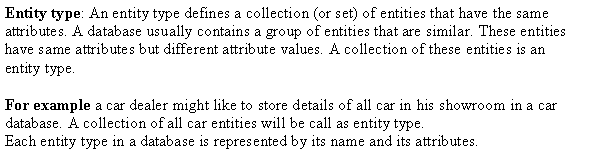
7.4. What is an entity type? What is an entity set? Explain the differences among

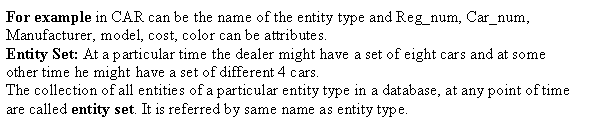
an entity, an entity type, and an entity set.

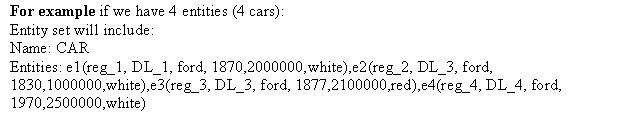
An entity type defines a collection (or set) of entities that have the same attributes.

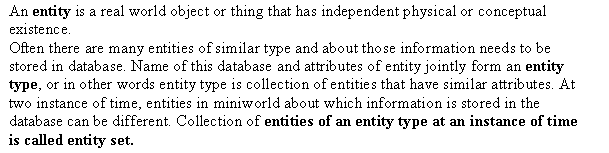
Entity set: The collection of all entities of a particular entity type in the database

In a database, there are a lot of entity type (defines a type, such as employee and department, they are different entities types), and each entity type has a lot of entity sets, then every entity set has a lot of entities.





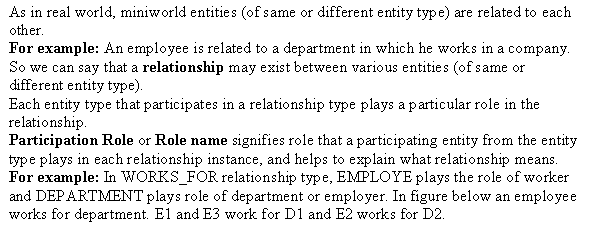


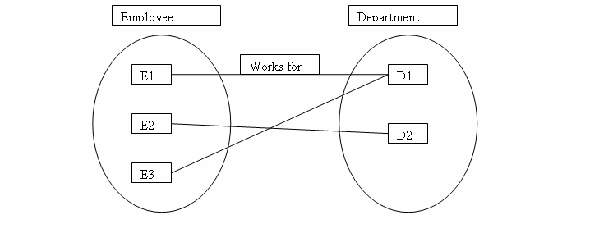


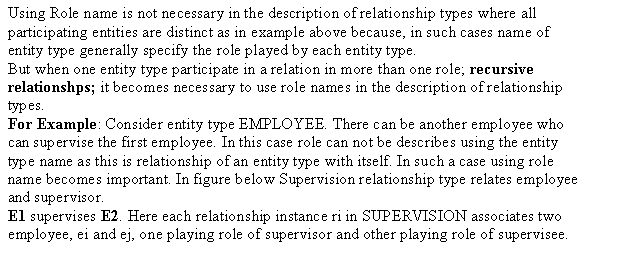
7.7. What is a participation role? When is it necessary to use role names in the

description of relationship types?

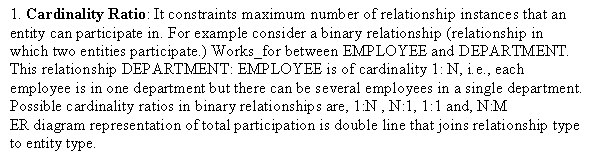
Participation role signifies role that a participating entity from the entity type plays in each relationship instance, and helps to explain that relationship means. Using Role name is not necessary in the description of relationship types where all participating entities are distinct because in such case name of entity types generally specify the role played by each entity type. But when one entity type participate in a relation in more than one role; recursive relationship; it becomes necessary to use role names in the description of relationship type.

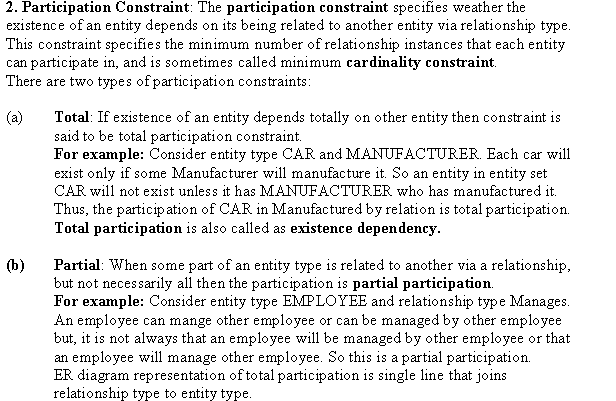






7.8. Describe the two alternatives for specifying structural constraints on relationship types. What are the advantages and disadvantages of each?

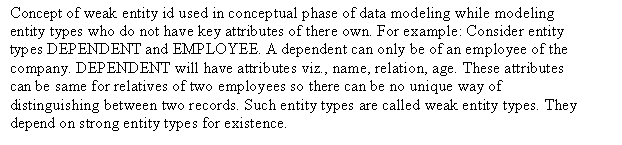


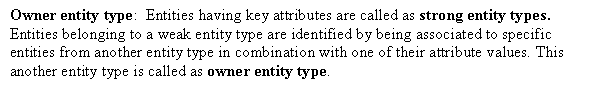


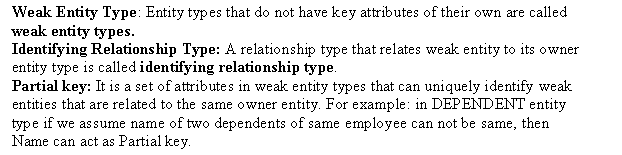
7.12. When is the concept of a weak entity used in data modeling? Define the terms owner entity type, weak entity type, identifying relationship type, and partial key.

Entity types that do not have key attributes of their own are called weak entity types. In contrast, regular entity types that do have key attribute of their own are called strong entity types. Entities belonging to a weak entity type are identified by being related to specific entities from another entity type in combination with one of their attribute values. This another entity type is called the identifying or owner entity type, and we call the relationship type that relates a weak entity type to its owner the identifying relationship of the weak entity type. A weak entity type always has a total participation constraint with respect to its identifying relationship because a weak entity cannot be identified without an owner entity. A weak entity type normally has a partial key, which is the set of attributes that can uniquely identify weak entities that are related to the same owner entity.

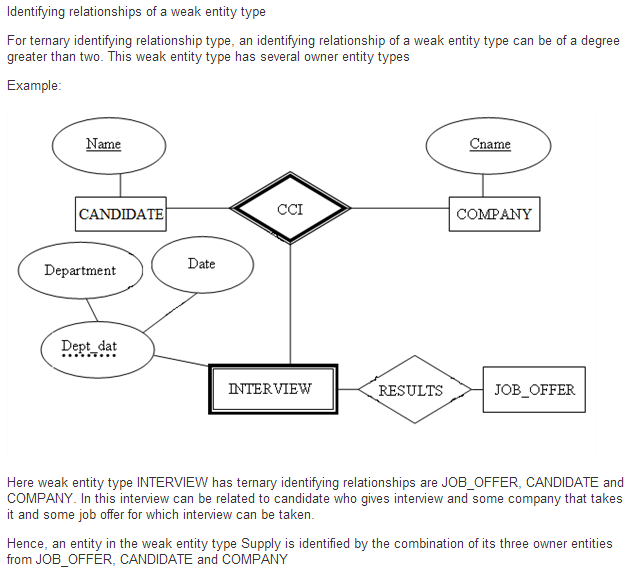
If there are many attributes, the concept of weak entity comes in data modelling. If the weak entity participates independently in relationship types other than its identifying relationship type, then it should not modelled as complex attribute.







7.13. Can an identifying relationship of a weak entity type be of a degree greater than two? Give examples to illustrate your answer.



7.16. Consider the following set of requirements for a UNIVERSITY database that is used to keep track of students’ transcripts. This is similar but not identical to the database shown in Figure 1.2:

a. The university keeps track of each student’s name, student number, Social Security number, current address and phone number, permanent address and phone number, birth date, sex, class (freshman, sophomore,..., graduate), major department, minor department (if any), and degree program (B.A., B.S., ..., Ph.D.). Some user applications need to refer to the city, state, and ZIP Code of the student’s permanent address and to the student’s last name. Both Social Security number and student number have unique values for each student.

b. Each department is described by a name, department code, office number, office phone number, and college. Both name and code have unique values for each department.

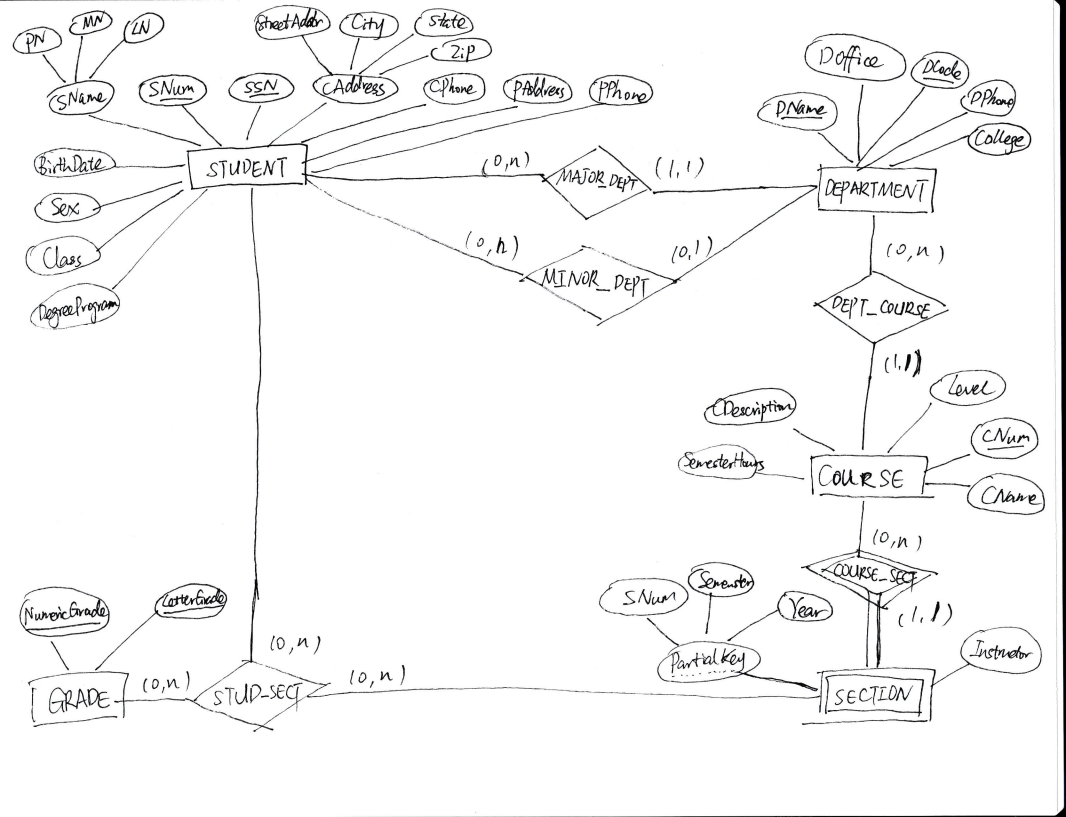
c. Each course has a course name, description, course number, number of semester hours, level, and offering department. The value of the course number is unique for each course.

d. Each section has an instructor, semester, year, course, and section number. The section number distinguishes sections of the same course that are taught during the same semester/year; its values are 1, 2, 3,..., up to the number of sections taught during each semester.

e. A grade report has a student, section, letter grade, and numeric grade (0,

1, 2, 3, or 4).

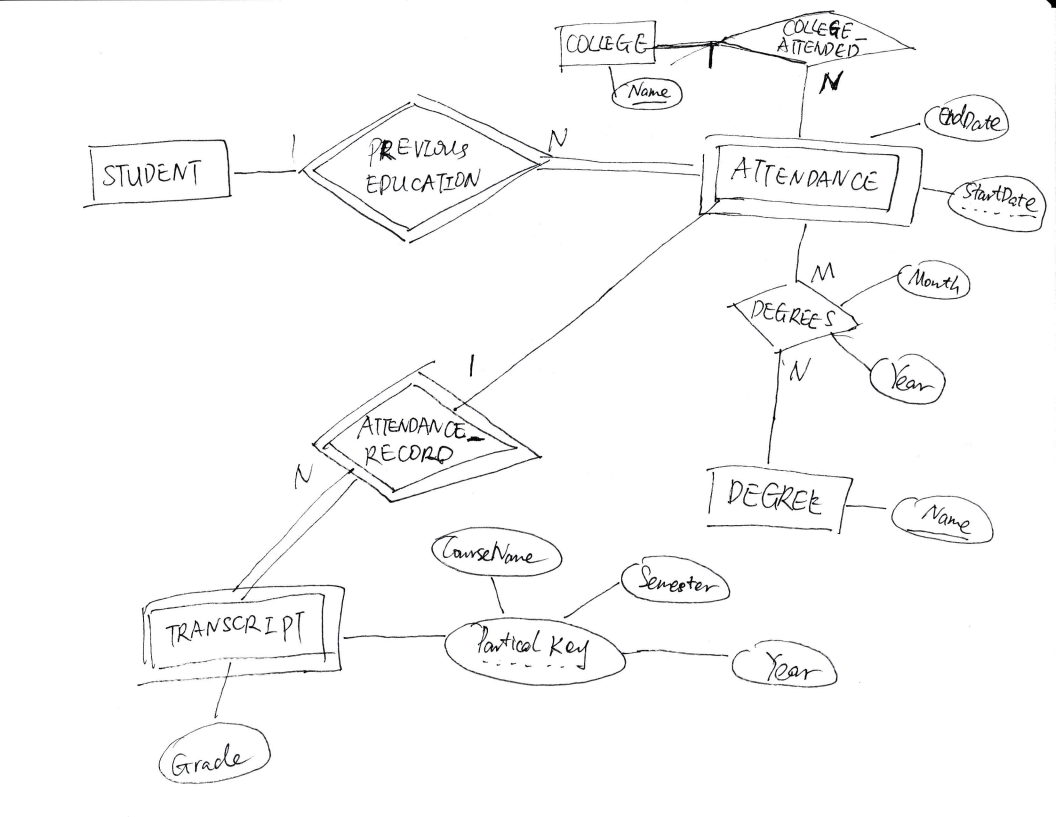
Design an ER schema for this application, and draw an ER diagram for the schema. Specify key attributes of each entity type, and structural constraints on each relationship type. Note any unspecified requirements, and make appropriate assumptions to make the specification complete.



7.18. Show an alternative design for the attribute described in Exercise 7.17 that uses only entity types (including weak entity types, if needed) and relationship types.

This example illustrates a perceived weakness of the ER model, which is: how does the database designer decide what to model as an entity type and what to model as a relationship type. In our solution, we created a weak entity type ATTENDANCE; each (weak) entity in ATTENDANCE represents a period in which a STUDENT attended a particular COLLEGE, and is identified by the STUDENT and the StartDate of the period.

Hence, the StartDate attribute is the partial key of ATTENDANCE. Each ATTENDANCE entity is related to one COLLEGE and zero or more DEGREEs (the degrees awarded during that attendance period). The TRANSCRIPT of the STUDENT during each attendance period is modeled as a weak entity type, which gives the records of the student during the attendance period. Each (weak) entity in TRANSCRIPT gives the record of the student in one course during the attendance period, as shown in the ER diagram below. Other ER schema designs are also possible for this problem.



7.23. Consider the ER diagram shown in Figure 7.21 for part of a BANK database. Each bank can have multiple branches, and each branch can have multiple accounts and loans.

a. List the strong (nonweak) entity types in the ER diagram.

b. Is there a weak entity type? If so, give its name, partial key, and identifying relationship.

c. What constraints do the partial key and the identifying relationship of the weak entity type specify in this diagram?

d. List the names of all relationship types, and specify the (min, max) constraint on each participation of an entity type in a relationship type. Justify your choices.

e. List concisely the user requirements that led to this ER schema design.

f. Suppose that every customer must have at least one account but is restricted to at most two loans at a time, and that a bank branch cannot have more than 1,000 loans. How does this show up on the (min, max) constraints?

(a) Entity types: BANK, ACCOUNT, CUSTOMER, LOAN

(b) Weak entity type: BANK-BRANCH. Partial key: BranchNo. Identifying relationship: BRANCHES.

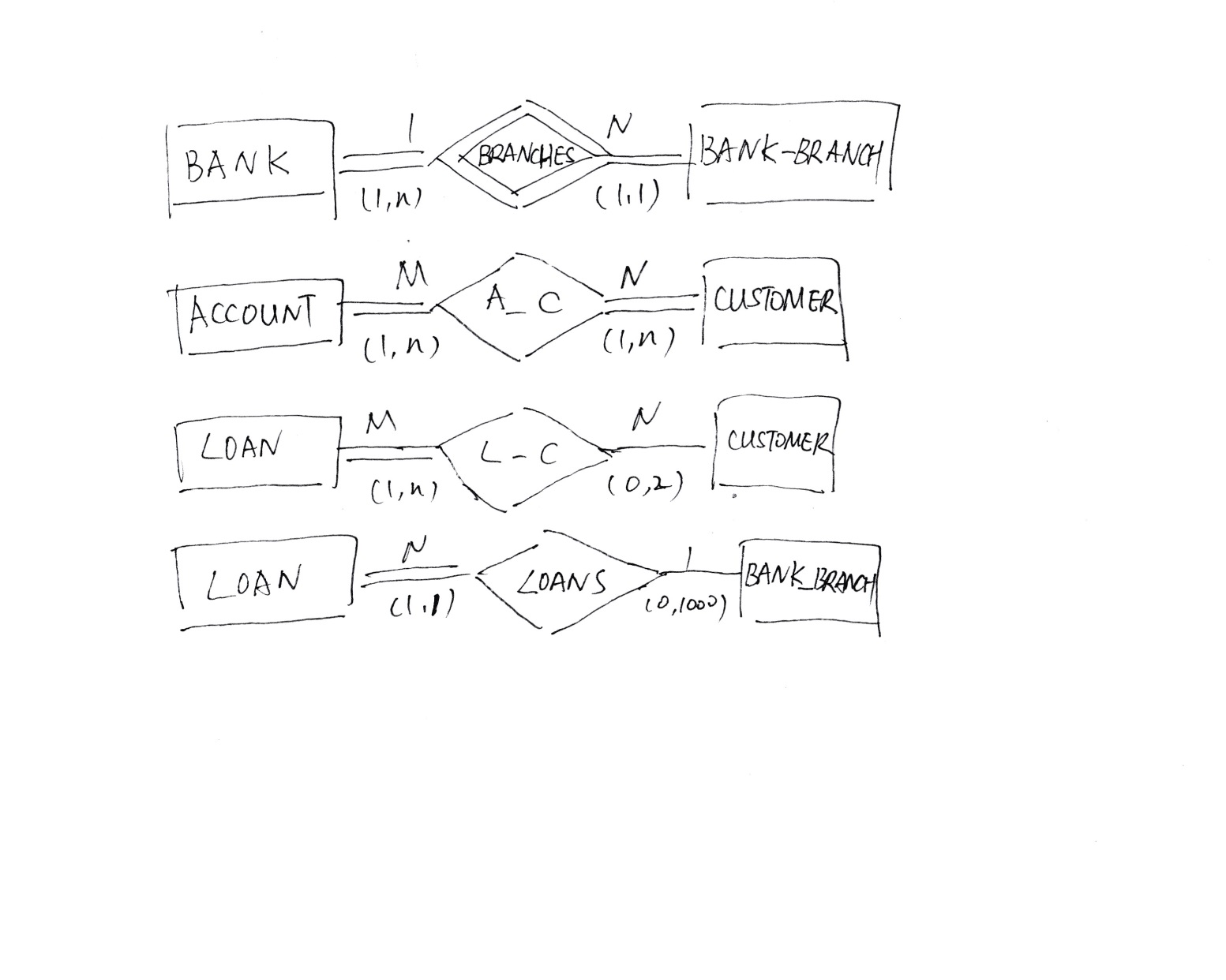
(c) The partial key BranchNo in BANK-BRANCH specifies that the same BranchNo value ay occur under different BANKs. The identifying relationship BRANCHES specifies that BranchNo values are uniquely assigned for those BANK-BRANCH entities that are related to the same BANK entity. Hence, the combination of BANK Code and BranchNo together

constitute a full identifier for a BANK-BRANCH.

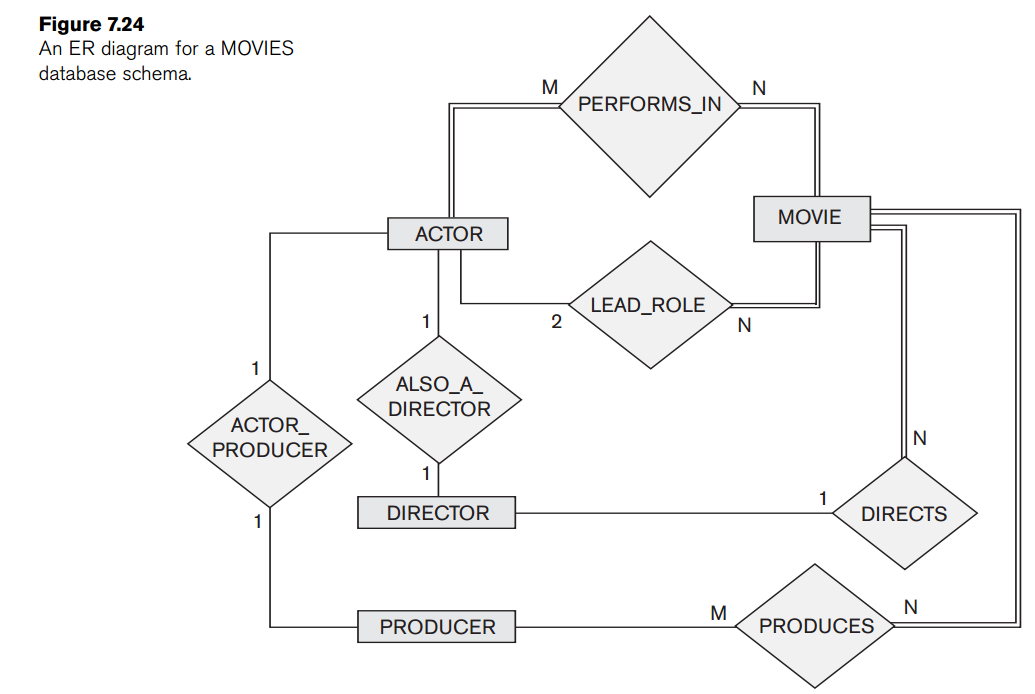
(d) Relationship Types: BRANCHES, ACCTS, LOANS, A-C, L-C. The (min, max) constraints are shown below.

(e) The requirements may be stated as follows: Each BANK has a unique Code, as well as a Name and Address. Each BANK is related to one or more BANK-BRANCHes, and the BranhNo is unique among each set of BANK-BRANCHes that are related to the same BANK. Each BANK-BRANCH has an Address. Each BANK-BRANCH has zero or more LOANS and zero or more ACCTS. Each ACCOUNT has an AcctNo (unique), Balance, and Type and is related to exactly one BANK-BRANCH and to at least one CUSTOMER. Each LOAN has a LoanNo (unique), Amount, and Type and is related to exactly one BANK-BRANCH and to at least one CUSTOMER. Each CUSTOMER has an SSN (unique), Name, Phone, and Address, and is related to zero or more ACCOUNTs and to zero or more LOANs.

(f) The (min, max) constraints would be changed as follows:



7.28. Consider the ER schema for the MOVIES database in Figure 7.24. Assume that MOVIES is a populated database. ACTOR is used as a generic term and includes actresses. Given the constraints shown in the ER schema, respond to the following statements with True, False, or Maybe. Assign a response of Maybe to statements that, while not explicitly shown to be True, cannot be proven False based on the schema as shown. Justify each answer.



a. There are no actors in this database that have been in no movies. True

b. There are some actors who have acted in more than ten movies. Maybe

c. Some actors have done a lead role in multiple movies. True

d. A movie can have only a maximum of one lead actor. True

e. Every director has been an actor in some movie. False

f. No producer has ever been an actor. False

g. A producer cannot be an actor in some other movie. False

h. There are movies with more than a dozen actors. Maybe

i. Some producers have been a director as well. False

j. Most movies have one director and one producer. False

k. Some movies have one director but several producers. True

l. There are some actors who have done a lead role, directed a movie, and produced some movie. Maybe

m. No movie has a director that also acted in that movie. False

Chapter 8

8.2. Define the following terms: superclass of a subclass, superclass/subclass relationship, IS-A relationship, specialization, generalization, category, specific (local) attributes, and specific relationships.

Superclass of a subclass: entity set or collection of entities of that type that exist in the database called subclass, and superclass is for each of these subclasses.

Superclass/subclass relationship: the relationship between a superclass and any one of its subclasses

IS-A relationship: 3A class/subclass relationship

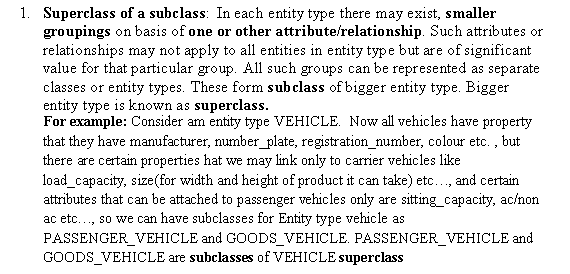
Specialization: Specialization is the process of defining a set of subclasses of an entity type

Generalization: process of defining a generalized entity type from the given entity types.

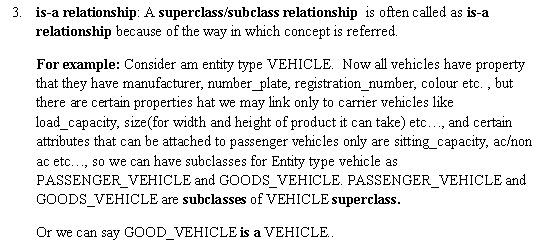
Category: a single superclass/subclass relationship with more than one superclass, where the superclasses represent different entity types. In this case, the subclass will represent a collection of objects that is a subset of the UNION of distinct entity types; we call such a subclass a union type or a category.

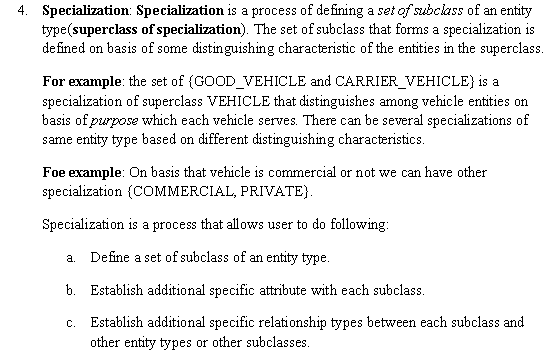
Specific (local) attributes: Attributes that apply only to entities of a particular subclass which are attached to the rectangle representing that subclass.

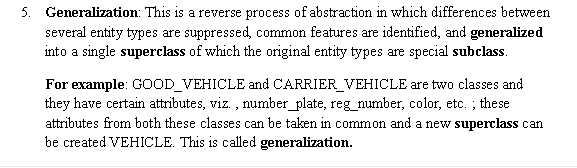
Specific relationships: Similarly, a subclass participated in relationship called specific relationship

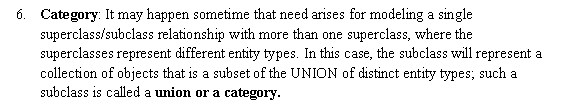


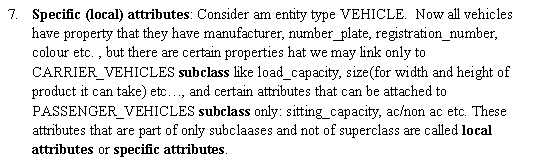


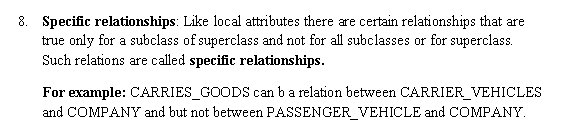






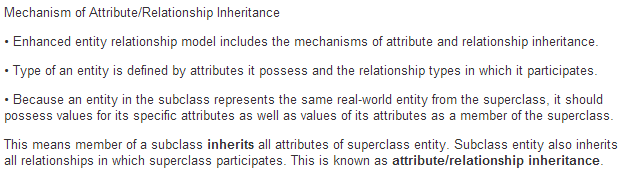


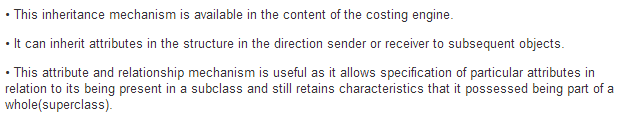




8.3. Discuss the mechanism of attribute/relationship inheritance. Why is it useful?

An important concept associated with subclasses (subtypes) is that of type inheritance. Recall that the type of an entity is defined by the attributes it possesses and the relationship types in which it participates. Because an entity in the subclass represents the same real-world entity from the superclass, it should possess values for its specific attributes as well as values of its attributes as a member of the superclass. We say that an entity that is a member of a subclass inherits all the attributes of the entity as a member of the superclass. The entity also inherits all the relationships in which the superclass participates. Notice that a subclass, with its own specific (or local) attributes and relationships together with all the attributes and relationships it inherits from the superclass, can be considered an entity type in its own right. It makes design a database much easier than before.





8.20. Design a database to keep track of information for an art museum. Assume that the following requirements were collected:

■ The museum has a collection of ART\_OBJECTS. Each ART\_OBJECT has a unique Id\_no, an Artist (if known), a Year(when it was created, if known), aTitle, and a Description. The art objects are categorized in several ways, as discussed below.

■ ART\_OBJECTS are categorized based on their type. There are three main types: PAINTING, SCULPTURE, and STATUE, plus another type called OTHER to accommodate objects that do not fall into one of the three main types.

■ APAINTING has a Paint\_type (oil, watercolor, etc.), material on which it is Drawn\_on (paper, canvas, wood, etc.), and Style(modern, abstract, etc.).

■ A SCULPTURE or a statue has a Material from which it was created (wood, stone, etc.), Height, Weight, and Style.

■ An art object in the OTHER category has a Type(print, photo, etc.) and Style.

■ ART\_OBJECTs are categorized as either PERMANENT\_COLLECTION (objects that are owned by the museum) and BORROWED. Information captured about objects in the PERMANENT\_COLLECTION includes Date\_acquired, Status (on display, on loan, or stored), and Cost. Information captured about BORROWED objects includes the Collection from which it was borrowed, Date\_borrowed, and Date\_returned.

■ Information describing the country or culture of Origin(Italian, Egyptian, American, Indian, and so forth) and Epoch(Renaissance, Modern, Ancient, and so forth) is captured for each ART\_OBJECT.

■ The museum keeps track of ARTIST information, if known: Name, DateBorn (if known),Date\_died (if not living), Country\_of\_origin, Epoch, Main\_style, and Description. The Nameis assumed to be unique.

■ Different EXHIBITIONS occur, each having a Name, Start\_date, and End\_date. EXHIBITIONS are related to all the art objects that were on display during the exhibition.

■ Information is kept on other COLLECTIONS with which the museum interacts, including Name(unique), Type (museum, personal, etc.), Description, Address, Phone, and current Contact\_person.

Draw an EER schema diagram for this application. Discuss any assumptions you make, and that justify your EER design choices.

