Chapter 1 Questions and Exercises 1.1, 1.3, 1.6, 1.8, 1.9, 1.12 Chapter 2 Questions and Exercises 2.2, 2.3, 2.4, 2.5 Chapter 3 Questions and Exercises 3.3, 3.4, 3.7, 3.8, 3.12, 3.13, 3.23 Chapter 4 Questions and Exercise 4.2, 4.17.

1.1

- Data – known facts that can be recorded and that have implicit meaning. I.e. names, telephone numbers, etc.
- Database – a collection of related data which represents some aspect of the real world called the mini world. A logically coherent collection of data with some inherent meaning. It is designed, built, and populated with data for a specific purpose, has an intended group of users, and a preconceived notion of applications for use.
- DBMS – a collection of programs and modules that enables users to create and maintain a database. It is a general purpose software system that facilitates the process of defining, constructing, manipulating, and sharing databases among users and applications.
- Database system – Contains both the database and the DBMS together as one system.
- Database catalog – a dictionary that defines or describes the database structure and constraints.
- Program-data independence – The ability to change the structure of data files stored in the DBMS catalog and not have to change all applications that access the data files.
- User view – a perspective of the database shown to a user. A view may contain a subset of data in the database or virtual data or derived data.
- DBA – a database administrator. The person responsible for overseeing maintenance of a database. They are responsible for authorizing access, coordinating, monitoring, and acquiring software and hardware for the database.
- End user – users whose jobs require access to the database for querying, updating, and generating reports. The database primarily exists for their use.
- Canned transaction – standard types of queries and updates that have been carefully programmed and tested.
- Deductive database system – provides capabilities for defining deduction rules for inferencing new information from the stored database facts.
- Persistent object – an object whose information is stored and can be accessed even after the program that created it exits.
- Meta-data – The information stored in the DBMS catalog.
- Transaction processing application – an application that processes the transactions with the database. A DBMS is an example.

1.3

- The main difference between a database vs. a file system approach is that a database approach is organized. The main characteristics that exemplify this are:
    - 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

1.6

- Capabilities that should be provided by the DBMS are
  - Controlling redundancy – don't keep more than one copy of data around
  - Restricting unauthorized access – only allow authorized individuals to access parts of the database
  - Providing persistent storage for program objects – allow applications to store large objects in a file so they persist after the program exits
  - Providing storage structures for efficient query processing – store data so that it can be quickly accessed by queries
  - Providing backup and recovery – allow easy backup and recovery of data.
  - Providing multiple user interfaces – allow multiple user interfaces to the database; multiple types of users will use the database and customizing the interface for each use allows easier access for them.
  - Representing complex relationships among data – represent the relationships among the data.
  - Enforcing integrity constraints – enforce data types and range restrictions to maintain the integrity of the database.
  - Permitting inferencing and actions using rules – allow the data to be used to inference new data
  - Potential for enforcing standards
  - Reduced application development time – since application developers won't have to develop their own data storage solutions
  - Flexibility
  - Availability of up-to-date information – real-time access to updated data for users.
  - Economies of scale – consolidation of data means less hardware needed to run it.

1.8

- Some informal queries for figure 1.2 could include
  - List the names of students
  - List the names of instructors
  - List course names
  - Get a transcript for a specified student
  - Get a list of students in a specified course
  - List prerequisites
  - Add a new grade for a specified student in a specified course
  - Add a new course
  - List the departments

1.9

- Controlled redundancy is used sometimes in databases to more efficiently organize the data for queries. For example, we store Student_name and Course_number redundantly in a GRADE_REPORT file (Figure 1.6(a)) because whenever a user wants to see a grade report, they usually also want to know the student name and course number that go with the grade. Whenever a new Student_name and Course_number are inserted into the GRADE_REPORT records, the DBMS checks the name and number against the already stored names and numbers to make sure they are in the database. This controls inconsistency and redundancy. Uncontrolled redundancy does not perform these checks. It would be akin to letting the accounting and the registration departments keep their own versions of the data on grades and students. The data may be inconsistent due to entrance error and the data would definitely be redundant and waste storage space.

1.12

- Some integrity constraints for figure 1.2 might include
    - Data type integrity.
        - Name may only contain alphabetic characters.
        - Student_number may only contain numeric characters.
        - Class may only contain numeric characters.
        - Major may only be alphabetic characters.
        - Course_name may only contain alphabetic characters.
        - Etc.
    - Data consistency integrity
        - A student must have a major
        - A student's major may only be from the listed departments
        - A course must have a number of credit hours less than 9 and must belong to a department
        - A section must have a course number and section identifier (which must be unique) and a semester, year, and instructor. The course number must match a course number from the course records.
        - A grade report must have a student number listed in the student records.
        - Etc.

2.2

- There are three main categories of data models
    - High-level (conceptual) – provides concepts close to the way many users perceive data. An example is the ER model which graphically represents the relationship between different parts of the data and the data itself.
    - Representational (implementation) – provides concepts that may be understood by end users but which are not too far removed from the way data is organized within the

computer. These hide some details of data storage but can be implemented on a computer directly.

- Low-level (physical) – provides concepts that describe the details of how data is stored on the computer. These are usually meant for computer specialists, not for typical end users. These could represent the data structures used to organize the data in the database.

2.3

- A database schema is different from a database state. A schema is the blue print or description of the database. It can be visualized as the headers to tables in the database. The database state is a snapshot of the database at any point in time including the schema and all data populated in the database. Each schema construct has its own set of instances in the database state.

2.4

- The three schema architecture is designed to give us data independence. There are three "levels" which are three different schemas.
  - External level (view level) contains multiple external schemas (user views). Each schema here describes some part of the database that a particular user or user group is interested in and hides the rest of the data from that user group. Each external schema is typically implemented using a representational data model.
  - Conceptual level contains a conceptual schema. This describes the structure of the whole database for a community of users. This hides the details of physical storage and concentrates on describing entities, data types, relationships, user operations, and constraints. Usually a relational model is used to describe the conceptual schema and this is used for the implementation schema.
  - Internal level contains an internal schema. This describes the physical storage structures of the database. This typically uses a physical data model and completely describes the details and access paths for the database.
- We need mappings between schema levels so that the levels remain insulated from each other. This allows one level's schema to change without the need to change other levels' schemas.
- Different schema definition languages help support this architecture because it gives each level its own way to describe the database. The language can be updated and changed as necessary and it will not affect other levels' schemas.

2.5

- The main difference between logical and physical data independence is that logical independence is used at the application level and physical independence is used at the physical level. Physical independence allows us to restructure or change how the data is stored physically without the need to change the application. This may be needed because of file restructuring or addition of data structures to improve performance. Logical data independence requires that

applications referring to unchanged parts of the database still work properly. It also requires that changes to constraints not require application rewrites.

3.3

- Entity – a thing in the real world with an independent existence. May be an object with a physical or conceptual existence.
- Attribute – the particular properties that describe an entity. A particular entity will have a value for each attribute.
- Attribute value – the actual value an attribute takes on when it describes an entity. This could be like assigning blue to the color attribute.
- Relationship instance – an instance of a relationship. The relationship actually takes on a value.
- Composite attribute – an attribute composed of many sub-attributes. Like an address has a street address, city, state, and zip code.
- Multivalued attribute – an attribute that an entity can have multiples of. If an entity has a color attribute it could be both red and blue.
- Derived attribute – an attribute that can be derived from known data. Age is an example; given the birth date of a person and today's date, we can derive their age.
- Complex attribute – an attribute composed of arbitrarily nested attributes. A residence (made of an address as described above) may have multiple phone numbers (which would be composed of an area code followed by the 7 digits).
- Key attribute – an attribute that can be used to uniquely identify an entity it is describing. Example: social security number for a person.
- Value set (domain) – the set of values a particular attribute can take on. Example: age we can restrict to be > 0.

3.4

- Entity – described in previous question.
- Entity type – defines a collection (or set) of entities that have the same attributes. Each entity type in a database is described by its name and attributes.
- Entity set – the instanced data for a type.

3.7

- A participation role is the label assigned to the link between an entity and another entity. It is necessary to use role names when a recursive relationship to the same entity is created. Example: when we have Employee <-> Supervision. It is necessary to have two links, one stating supervises and one stating managed by.

3.8

1. Cardinality ratios – These are listed as a single number on either side of a relationship. These apply only to binary relationships. The four ratios are 1:1, 1:N, N:1, and N:M. These are used in

conjunction with a single line or double line along the relationship link to indicate total or optional participation in a relationship.

2. MinMax Notation – uses a single (min, max) label for each relationship link to represent the minimum and maximum number of times a particular entity may participate in the relationship. Only single lines are used, for total participation we use (1, x) where x is the max number to indicate that each entity must participate at least once in the relationship. One feature of this versus the cardinality ratios is that it can be used with n-ary relationships unlike cardinality ratios.

3.12

- The concept of a weak entity is used whenever an entity is linked with another entity and it cannot be described uniquely using its own data.
- Owner entity type – the entity the weak entity is owned by, whose data must be used in conjunction with the weak entity type to uniquely identify the weak entity type.
- Weak entity type – entities that do not have a key attribute of their own.
- Identifying relationship type – the type of the relationship linking the owner entity type to the weak entity type.
- Partial key – a set of attributes for a weak entity type that uniquely identify weak entities that are related to the same owner entity.

3.13

- Yes it is possible. A weak entity type may have more than one owner type. Take a supply for example. It has no partial key for itself and its owners Supplier, Part, and Project must be used to identify it.

3.23

a. BANK, ACCOUNT, LOAN, CUSTOMER
b. Yes, BANK_BRANCH. The partial key is Branch_no. The identifying relationship is BRANCHES.
c. It specifies that a BANK_BRANCH is only uniquely identifiable by using its Branch_no and the parent BANK's Code. Each BANK_BRANCH participates in only one BANK but a BANK may have more than one BANK_BRANCH.
d. The relationship types are
    i. BRANCHES
        1. BANK (1,N) BRANCHES (1,1) BRANCH – A bank must have one or more branches, and a branch must belong to only one bank.
    ii. ACCTS
        1. BANK_BRANCH (0, N) ACCTS (1,1) ACCOUNT – A bank branch may have zero or more accounts, and an account must belong to only one bank branch.

iii. LOANS
    1. BANK_BRANCH (0, N) LOANS (1,1) BANK_BRANCH – A bank branch may have zero or more loans, and a loan must belong to only one bank branch.
iv. A_C
    1. ACCOUNT (1, N) A_C (0, M) CUSTOMER – An account must belong to one or more customers, and a customer may have zero or more accounts.
v. L_C
    1. LOAN (1,N) L_C (0,M) CUSTOMER – A loan must belong to one or more customers, and a customer may have zero or more loans.

e. Some user requirements are probably
    i. A bank has a Name and Address
    ii. A bank branch has an address and a branch number
    iii. An account has an account number, balance, and type
    iv. A loan has a loan number, amount, and type
    v. A customer has a social security number, a phone number, a name, and an address.
    vi. Structural constraints apply as in part (d) above.
f. This would alter the relationships in part (d) above to:
    i. CUSTOMER (1, N) A_C (1, M) ACCOUNT
    ii. CUSTOMER (0,2) A_L (1, M) LOAN
    iii. BANK_BRANCH (0, 1000) LOANS (1,N) BANK_BRANCH

4.2

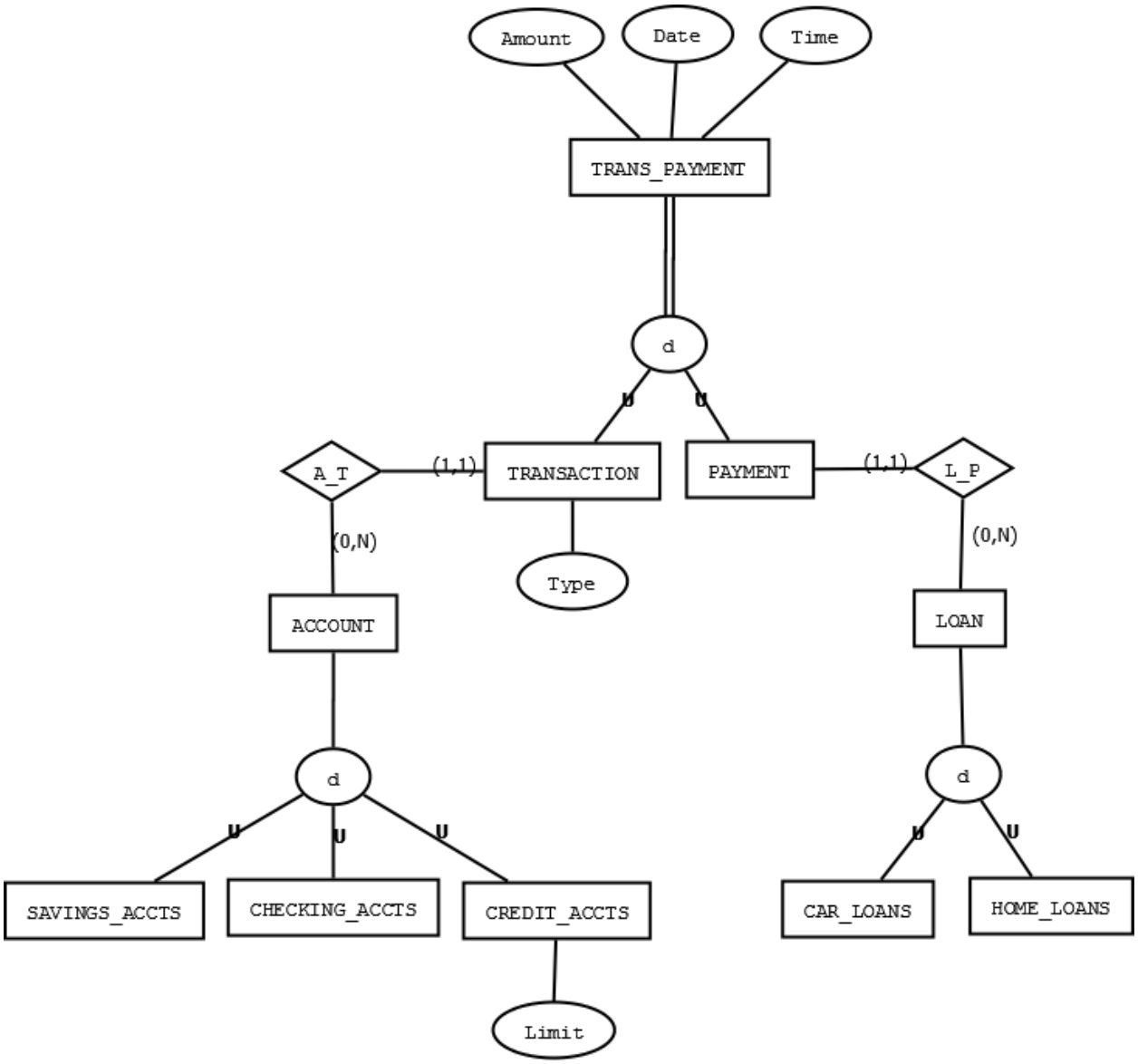
- Superclass of a subclass – The parent or base class. This is more general than the sub class.
- Superclass/subclass relationship – A concept from object oriented programming. Objects are modeled as classes and the can have a super/sub relationship. A super class is more general than a sub class and the sub class is a specialization of the superclass. The sub class inherits all of the attributes and characteristics of the super class. It can be thought of as the sub class is a specific kind of super class.
- Is-a relationship – A relationship between objects/classes. The is-a relationship specifically applies to super/sublcasses. A sub class “is-a” specific kind of the super class. So if the super class is ANIMAL the sub class could be DOG. A DOG “is-a” ANIMAL.
- Specialization – The process of classifying a class of objects into specialized subclasses.
- Generalization – The inversion of specialization.
- Category – A collection of objects that is a subset of the UNION of distinct entity types.
- Specific (local) attributes – attributes specific to a subclass. These are known only to the subclass and not the superclass.
- Specific relationships – A relationship between a subclass and another entity that isn’t the superclass.

4.17

Alter the ER diagram to become an EER diagram. Then add the entities and relationships in the following schema representation, while keeping all of the diagram as listed in the book.

Assumptions:

- Every kind of cash inflow to the bank with respect to an ACCOUNT or LOAN is a TRANSACTION or a PAYMENT. It cannot be both.
- TRANSACTIONs additionally have a Type which can take values like deposit, withdrawal, and check.
- A TRANSACTION must belong to only one account, but an ACCOUNT may have more than one TRANSACTION.
- A PAYMENT must belong to only one LOAN, but a LOAN may have more than one PAYMENT.
- There exist other types of ACCOUNTs other than SAVING_ACCTS, CHECKING_ACCTS, and CREDIT_ACCTS.
- There exist other types of LOANs other than CAR_LOANS and HOME_LOANS.
- Dia (a drawing program) doesn't directly support the arc shape for sub classes, hence the U shape is used instead.


Amount
Date
Time
TRANS_PAYMENT
d
U
U
A_T
(1,1)
TRANSACTION
PAYMENT
(1,1)
L_P
(0,N)
Type
(0,N)
ACCOUNT
LOAN
d
d
U
U
U
U
U
SAVINGS_ACCTS
CHECKING_ACCTS
CREDIT_ACCTS
CAR_LOANS
HOME_LOANS
Limit