Chapter 1 Questions:

1.1 Give four examples of database systems other than those listed in Section 1.1.

1.2 Name five tasks performed by the DBMS.

1.3 List three functions that you can perform with a database that you cannot perform with a spreadsheet.

1.4 Distinguish between a database and a database management system.

1.5 List five advantages of a database system and give an example of each.

1.6 List three responsibilities of the DBA.

1.7 Give an example of an end user and describe a typical task that a user can perform on a database.

1.8 Provide an example of an application besides payroll that might use sequential batch processing, and draw a diagram similar to Figure 1.6.

1.9 Briefly define each of the following terms used in database systems:

a. integrated database
b. enterprise
c. metadata
d. concurrent use
e. query
f. end user
g. data redundancy
h. data consistency
i. integrity constraint
j. data encryption
k. economy of scale
l. backup
m. recovery log
n. user view
o. semantic model
p. SQL
q. XML
r. data mining
s. data warehouse
t. big data
u. SQL database
v. the 5 V's of big data

Chapter 1 Answers:

1.1 Examples of other database systems can be found in many organizations including banks, state and local government agencies, real estate companies, information services, volunteer organizations, and so on.

1.2 Some tasks performed by the DBMS include setting up database structures, loading data, accepting user and program requests, hiding data, handling updates, allowing concurrent use of data, and performing backup and recovery procedures.

1.3 Functions provided by a DBMS but not by a spreadsheet include querying, report-writing, and indexing of data for faster retrieval.

1.4 A database is the actual stored data, while a database management system is the software that manages it.

1.5 Advantages of a database system are sharing of data, control of redundancy, data consistency, improved data standards, better data security, improved data integrity, balancing of conflicting requirements, faster development of new applications, better data accessibility, economy of scale, more control over concurrency, better backup and recovery procedures, and others.

1.6 The major responsibilities of the DBA include designing, creating, and maintaining the database.

1.7 An end user might be an order-taker for a catalog company, who enters data from telephone orders on a screen form that is used as input for the database. Another example is a manager who receives reports from the database and makes decisions based on them.

1.8 A magazine subscription file could be maintained in order by subscriber ID and updated each month as subscriptions are added or as they expire. The updates(new or expiring subscriptions) form the transaction file and the existing continuing ubscriptions form the master file in Figure 1.6

1.9 All of these terms are described in the textbook.

Chapter 2 Questions:

2.1 Name four resources of a typical business organization.

2.2 Distinguish between data and information.

2.3 Identify the four levels of abstraction in discussing data. For each, give an example of an item that appears on that level.

2.4 Distinguish between an entity set and an entity instance.

2.5 Distinguish between a record type and a record occurrence.

2.6 What level of data abstraction is represented in the data dictionary? Give examples of the types of entries that would be stored there.

2.7 Explain why a staged design approach is appropriate for database design.

2.8 Name five desirable characteristics of a conceptual model of an enterprise.

2.9 Name the eight major design stages in staged database design, along with the activities and possible outcomes of each.

2.10 Explain what is meant by saying staged database design is a "top-down" method.

2.11 Explain how a CASE package can be used in database design.

2.12 Name two advantages and two disadvantages of the following:
a. integrated data dictionaries
b. freestanding data dictionaries

2.13 Explain why users should not have access to the data dictionary.

2.14 Name eight uses for a data dictionary.

2.15 What types of skills must a database administrator possess? For what tasks are they needed?

2.16 List the major functions of the DBA.

2.17 Define each of the following terms:
a. operational data
b. corporate resource
c. metadata
d. entity
e. attribute
f. data item
g. data aggregate
h. data record
i. data file
j. data sublanguage

k. prototype
l. system tuning
m. CASE
n. integrated data dictionary
o. data dictionary synonym
p. data dictionary homonym
q. data standards
r. DBA

2.18 Describe the three-level architecture for databases.

2.19 Describe the two parts of data sublanguages.

2.20 Give five reasons why it is desirable to separate the physical representation from the external structure of a database.

2.21 Distinguish between the following: user interface, logical record interface, stored record interface, physical record interface.

2.22 Describe each of the following: external schema, logical schema, internal schema.

2.23 Explain the purpose of external/logical mapping and of logical/internal mapping.

2.24 Distinguish between logical and physical data independence, and give examples of possible changes they allow.

2.25 Explain why the logical model is called "the heart of the database."

2.26 Describe abstraction and explain how it is used in database design.

2.27 Distinguish between the intension and extension of a database.

2.28 Explain how record-based data models differ from semantic models.

Chapter 2 Answers:

2.1 Resources of a typical business organization include capital equipment, financial assets, personnel, data, and others.

2.2 Data is unprocessed facts, while information is data processed to be useful for decision making.

2.3 Four levels of abstraction in discussing data are the level of the enterprise in reality (including the mini-world); the conceptual model; the logical model, and data instances. The first level includes the mini-world, that portion of the entire enterprise in its environment focusing, on those aspects that are of interest, such as the human resources aspects of a software company. The conceptual level contains the entities, the attributes, and the relationships that are to be captured in the database, such as the employees and their jobs. The logical level includes the logical model consisting of record types and data item types, such as employee records having data items empId, lastName, and so on. The data instance level consists of record occurrences, containing actual data about objects, such as 'E101', 'Smith', and so on.

2.4 An entity set is a group of similar entities. An example of an entity set is the set of an organization's customers. An entity instance is a particular member of an entity set. An individual customer is an entity instance.

2.5 A record type refers to the structure of similar records. For example, the Customer record type has a specified length and consists of specific data items, such as custID, custName, custPhone, and creditStatus. A record occurrence means the actual data record of a particular entity. An example is the record of a particular customer. It may have entries 'C101', 'Acme, Inc.', '2129876543', and 'good' as values of the data items.

2.6 The data dictionary contains metadata, and represents the logical model, or the third level. It contains the name and structure of all record types and names and descriptions of all data item types. It might have an entry such as custId char(5).

2.7 Staged database design allows the database to change as needed, because the conceptual model is developed independently of the implementation. The conceptual model can evolve as changes are made to the miniworld. Even if the DBMS chosen is changed, the conceptual model can survive.

2.8 A conceptual model should mirrors the operations of the organization, be flexible enough to allow changes as new information needs arise, support many different user views, be independent of physical implementation, and be independent of the model used by a particular database management system.

2.9 Stages in staged database design are: analyze the user environment, develop the conceptual data model, choose a DBMS, develop the logical model by mapping the conceptual model to the DBMS, develop a physical model, evaluate the physical model, perform tuning as indicated, and implement the physical model. Their activities and outcomes are described in Section 2.3.

2.10 Staged database design is a "top-down" method because it begins with a general statement of needs and progresses to more detailed consideration of problems. It allows the designer to consider different problems at different phases.

2.11 CASE packages usually include system analysis and software engineering tools that can be useful for conceptual design. These include diagram generators and data dictionaries. Some packages include E-R diagrams, EE-Rdiagrams, and UML diagrams. They may also contain project management software.

2.12(a) Two advantages of integrated data dictionaries are that they can be maintained by the system when changes are made to the database structure, and that they can provide access control and audit information. Two disadvantages of an integrated data dictionary are that they are unavailable in the early design stages before the DBMS is chosen, and that once the DBMS is chosen the designer may have no choice but to implement its dictionary, even though it may not have the most desirable features.

(b) Two advantages of a freestanding data dictionary are that it is available for the early design states and that there is a wide variety of them available. Two disadvantages are that they require maintenance and they represent an extra cost to the designer.

2.13 Users should not be permitted access to the data dictionary because not all users should be aware of data that is stored for others.

2.14 Uses of a data dictionary include collecting information about data in central location, securing agreement on meanings of items, communicating with users, identifying inconsistencies – synonyms and homonyms, keeping track of changes to DB structure, determining impact of changes to DB structure, identifying sources of/responsibility for items, recording external/logical/physical models and mappings, recording access control information, and providing audit information

2.15 A DBA needs technical competence, managerial skills, diplomatic skills, and communication skills. Technical competence is required to understand the system, managerial skills to direct a staff, diplomatic skills to work out agreements with users, and communication skills to make the project understandable to others.

2.16 Major functions of the DBA are for planning, designing, developing and managing the operating database.

2.17 All of these terms are described in the chapter. The definitions here should be short.

2.18 The three-level database architecture consists of the external, logical, and internal levels. The external level contains the user views. A user view is the structure of the database as it appears to an individual user. The logical level is the entire information content of the database as seen by the DBA. It contains all the data that is available, including all record types, all data item types, all relationships, constraints, semantic information, and security and integrity information. The internal level includes the data structures and file organizations that are used to store the data in the database.

2.19 Two parts of a data sublanguage are the data definition language (DDL), used to describe the database and create its structure, and the data manipulation language (DML), used to process the data.

2.20 Separation of the physical structure from the external is desirable because different users need different views of the same data, users' needs may change over time, users should not have to deal with complex database structures, the DBA should be able to make changes to the logical structure without affecting users, changes to data structures and file structures should not affect users, and changes to physical storage devices should not affect users.

2.21 The user interface is the boundary below which the user is not permitted to see. The logical record interface is the boundary for the logical level. The stored record interface is the boundary between the database's internal level and the operating system's physical level. The physical record interface is the boundary created by the operating system, below which physical storage details are hidden.

2.22 An external schema is a complete description of a user's view, including each type of external record that appears in the view. A logical schema is a complete description of the information content of the database, including every record type and its complete structure. An internal schema is a complete description of the internal model, including data about all stored objects and their physical representation.

2.23 The external/logical mapping gives the correspondence between objects in user views and objects in the logical model. The logical/internal mapping gives the correspondence between logical level objects and their internal representations.

2.24 Logical data independence is the immunity of the external models to changes in the logical model. Changes such as the addition of a new data item should not affect users who do not need the new item. Physical data independence means the immunity of the logical model to changes in the internal model. Changing the sequencing of records is an example of the type of internal change that should not affect the logical level.

2.25 The logical model is the heart of the database because it dominates the other two levels. The external models depend on it to provide the data they require. The internal model is merely the representation of the logical model.

2.26 Abstraction is the process of identifying common properties of a set of objects rather than focusing on details. It allows us to categorize objects, and to simplify concepts and hide complexity. In database design, abstraction is used to identify the objects to be represented in the database, without considering how they are to be represented.

2.27 The intension of the database is the permanent logical structure of the database. An extension is an instance of the database, with all the data that appears in it at any given moment.

2.28 Record-based models focus on the logical level but provide some information about the internal level as well. They specify not only the logical structure but some implementation details. Semantic models focus on the external and logical levels, and are independent of internal representation. They attempt to incorporate semantic aspects or meanings of data.

Chapter 3 Questions:

3.1 Define each of the following terms.

a. entity type
b. entity set
c. well-defined set
d. intension of an entity
e. extension of an entity
f. attribute
g. domain of an attribute

h. null value
i. superkey
j. candidate key
k. composite key
l. primary key
m. alternate key
n. secondary key
o. relationship type
p. relationship set
q. binary relationship
r. ternary relationship
s. n-ary relationship
t. cardinality of a relationship
u. recursive relationship
v. existence dependency
w. weak entity
x. specialization
y. generalization
z. union
aa. superset
ab. subset
ac. isa relationship
ad. disjointness constraint
ae. completeness constraint
af. attribute-defined specialization
ag. multiple inheritance
ah. total category

3.2 A private book collector is designing a database to keep track of her purchases and book holdings. Consider the entity set Book with attributes title, author, publisher, pubDate, numberPages, condition, cost, and datePurchased.

a. Show how the entity set and its attributes would be represented on an E-R diagram.
b. Describe the domain of the cost attribute, making assumptions as needed.
c. Identify a superkey for the Book entity set.
d. Identify all candidate keys for the entity set.
e. Identify a primary key for the entity set and underline it on the E-R diagram.

3.3 a. Assume that in the same scenario as in Exercise 3.2, there is an entity set called Purchase with attributes purchaseDate, totalAmount, and any others you wish to

add to describe book purchases. A purchase may include several books. Show how this entity set and its relationship to Book would be represented on the E-R diagram.

b. Stating any necessary assumptions, make a decision about the cardinality and participation constraints of the relationship, and add appropriate symbols to the E-R diagram.

c. Assume there is another entity called Seller to be added to the diagram. The book collector makes purchases from many sellers, but each purchase is made from one seller. Making up attributes as needed, add this entity and appropriate relationship(s) to the diagram.

3.4 Design a database to keep data about college students, their academic advisors, the clubs they belong to, the moderators of the clubs, and the activities that the clubs sponsor. Assume each student is assigned to one academic advisor, but an advisor counsels many students. Advisors do not have to be faculty members. Each student can belong to any number of clubs, and the clubs can sponsor any number of activities. The club must have some student members in order to exist. Each activity is sponsored by exactly one club, but there might be several activities scheduled for one day. Each club has one moderator, who might or might not be a faculty member. Draw a complete E-R diagram for this example. Include all constraints.

3.5 A dentist's office needs to keep information about patients, the number of visits they make to the office, work that must be performed, procedures performed during visits, charges and payments for treatment, and laboratory supplies and services. Assume there is only one dentist, so there is no need to store information about the dentist in the database. There are several hundred patients. Patients make many visits, and the database should store information about the services performed during each visit and the charges for each of the services. There is a standard list of charges, kept outside the database. The office uses three dental laboratories that provide supplies and services, such as fabricating dentures. Draw a complete E-R diagram for this example.

3.6 An interior design firm would like to have a database to represent its operations. A client (customer) requests that the firm perform a job such as decorating a new home, redecorating rooms, locating and purchasing furniture, and so forth. One of the firm's decorators is placed in charge of each job. For each job, the firm provides an estimate of the amount of time and money required for the entire job. Some of the work for a job, such as planning furniture placement, is done by the decorator in charge of the job. In addition, the firm might hire contractors to work on a daily or hourly basis on a particular job. A job might also include several activities, such as painting, installing floor covering, fabricating draperies, wallpapering, constructing, installing cabinets, and so on. These activities are done by contractors hired by the firm. The contractor provides an estimate for each activity. An activity or job might also require materials such as paint or lumber, and the firm has to keep track of the cost of materials for each activity or job in order to bill the client. The database should store the estimated costs and actual costs of all activities and all jobs. Draw a complete E-R diagram for this example.

3.7 An automobile body repair shop needs to keep information about its operations. Customers initially bring their cars to the shop for an estimate of repairs. A mechanic looks at the car and estimates the cost and time required for the entire job. If the customer accepts the estimate, a job number is assigned and the customer's name and contact information; the car's license plate number, make, model, and year; and a list of the repairs needed are

recorded. The customer then makes an appointment to bring in the car on a specified date. When the car is brought in for repairs, the work begins. The shop keeps track of the charges for parts and labor as they accumulate. Only one mechanic works on the car for the entire job. A job might include several repairs (e.g., replacing the left fender, painting the passenger door). The time actually spent for each repair is recorded and used to calculate the cost of labor, using a fixed hourly rate. Draw a complete E-R diagram for this example.

3.8 A database is needed to keep track of the operations of a physical therapy center. Every patient must be referred by a physician and have a prescription for physical therapy in order to receive treatments. A patient may have different physicians at different times. The database keeps all information about prescriptions and treatments, both past and current. When appointments are made, the information about scheduled date and time is recorded. No patient is ever scheduled for two visits on one day. The center has several physical therapists, and a patient may be treated by different physical therapists on different visits. When a patient makes a visit at an appointed time, the name of the therapist, the treatment, the date, the time, and the equipment used are all recorded for that visit. Each of these has only one value for the visit. This information will be used later for insurance billing, which is not part of this database. Draw a complete E-R diagram for this example.

3.9 a. Assume the Seller entity set in Exercise 3.3 is specialized into Private and Dealer. Show how the specialization is represented on an EE-R diagram, making up attributes as needed and stating any assumptions you need to make.

b. Given the specialization in Part (a), add relationships and/or additional diagram elements that show:

(i) All sellers can have a rating assigned by the collector.

(ii) Private sellers are usually recommended by a source such as a friend or another collector.

(iii) Most dealers have a newsletter or a webpage that they use to advertise their offerings

c. Using (min,max) notation, add constraints for the relationships, stating any additional assumptions you need to make.

3.10 Develop an EE-R diagram for the college student and activities described in Exercise 3.4, but expand it to cover graduate students as well as undergraduates. Graduate students can participate in the clubs as members, or they can serve as moderators of the clubs. A graduate student, like an undergraduate, has one academic advisor, but may also have a thesis advisor, who oversees his or her thesis research.

3.11 Develop an EE-R diagram for the dental group described in Exercise 3.5, but expand it by assuming that some of the dentists are surgeons, who can perform oral surgery. Also assume some of the professionals are dental hygienists, who can perform routine work such as cleaning and taking X-rays. A patient typically begins an ordinary visit by starting with a dental hygienist who performs routine work, and then a dentist who does a checkup during the same initial visit. The work needed is usually determined at this visit. If follow-up visits are required, each of the next visits will be with one dentist. If a patient requires oral surgery, at least one of his or her visits must be with one of the surgeons, even though the rest of the work needed might be performed by his or her regular dentist. When surgery is performed, there must be a qualified assistant present. Only some of the hygienists are qualified to assist

in surgeries. Assume this database does not keep track of appointments, only of actual visits, work, billing, and payments.

3.12 Develop an EE-R diagram for the interior design firm described in Exercise 3.6, but expand it by assuming the clients can either be individuals or corporations. Also assume the contractors can be licensed professionals with various specialties (such as plumbers, electricians, or carpenters) or hourly laborers.

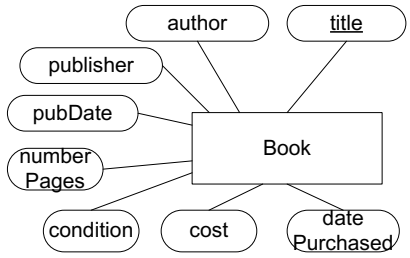
3.13 Develop an EE-R diagram for the automobile body repair shop described in Exercise 3.7, but assume the shop is expanding to perform mechanical repairs as well as body work. Body work can be done by a technician, but all mechanical repairs require a licensed mechanic. Jobs now may require one technician to handle the body work (if any) and a licensed mechanic to do the mechanical repairs (if any).

3.14 Develop an EE-R diagram for the physical therapy center described in Exercise 3.8, but expand it to assume that some visits require the use of specialized equipment for electrotherapy, ultrasound, hydrotherapy, and so on, and the patient is charged for each use of such equipment. Patients may also be supplied with special devices to take home, such as exercise bands, neck pillows, and so on, and the patient is charged for these special supplies.

Chapter 3 Answers:

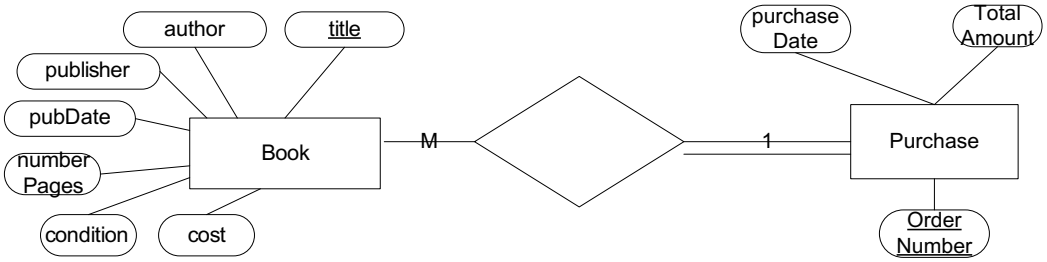
3.1 See text for definitions

3.2 (a)



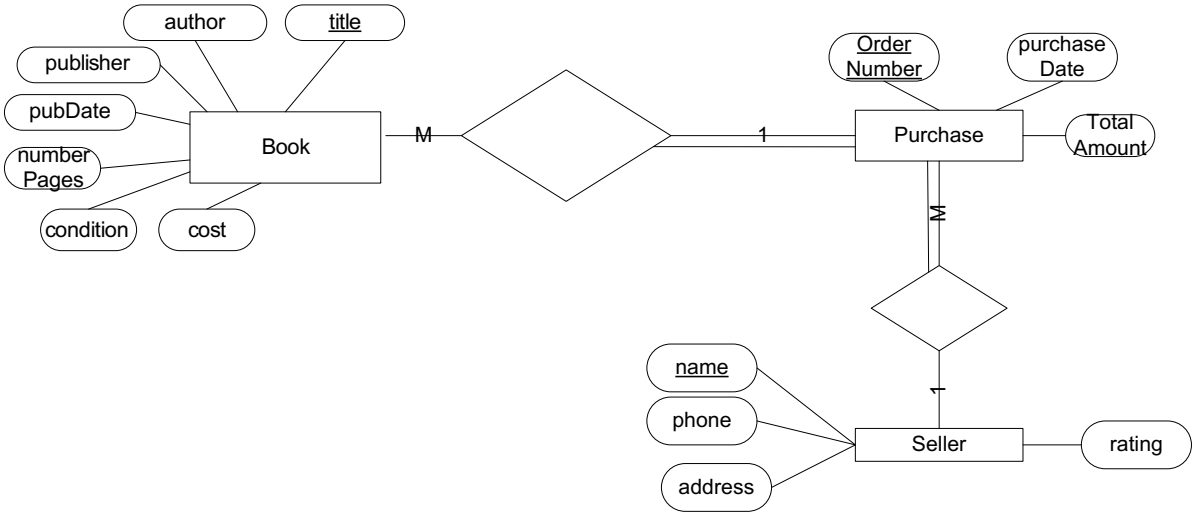
(b) The domain of the cost attribute is some subset of the set of real numbers. It may be stored in the format real(8,2), currency, or some similar form. The set may have a lower bound of 0.00 and upper bound 999999.99, or some other range of values.
(c) If we assume the collector never has two copies of the same book, never has two books with the same title, and stores only the first author's name if the book has multiple authors, then title is a superkey, and so is {title, author}.There are many others. If the assumptions about title are not valid, some combination such as {title, author}, or {title, author, datePurchased} may be needed as a superkey.
(d) Under our initial assumptions in part (c), title is a candidate key.
(e) The primary key is title, if that is the only candidate key.

3.3 (a)

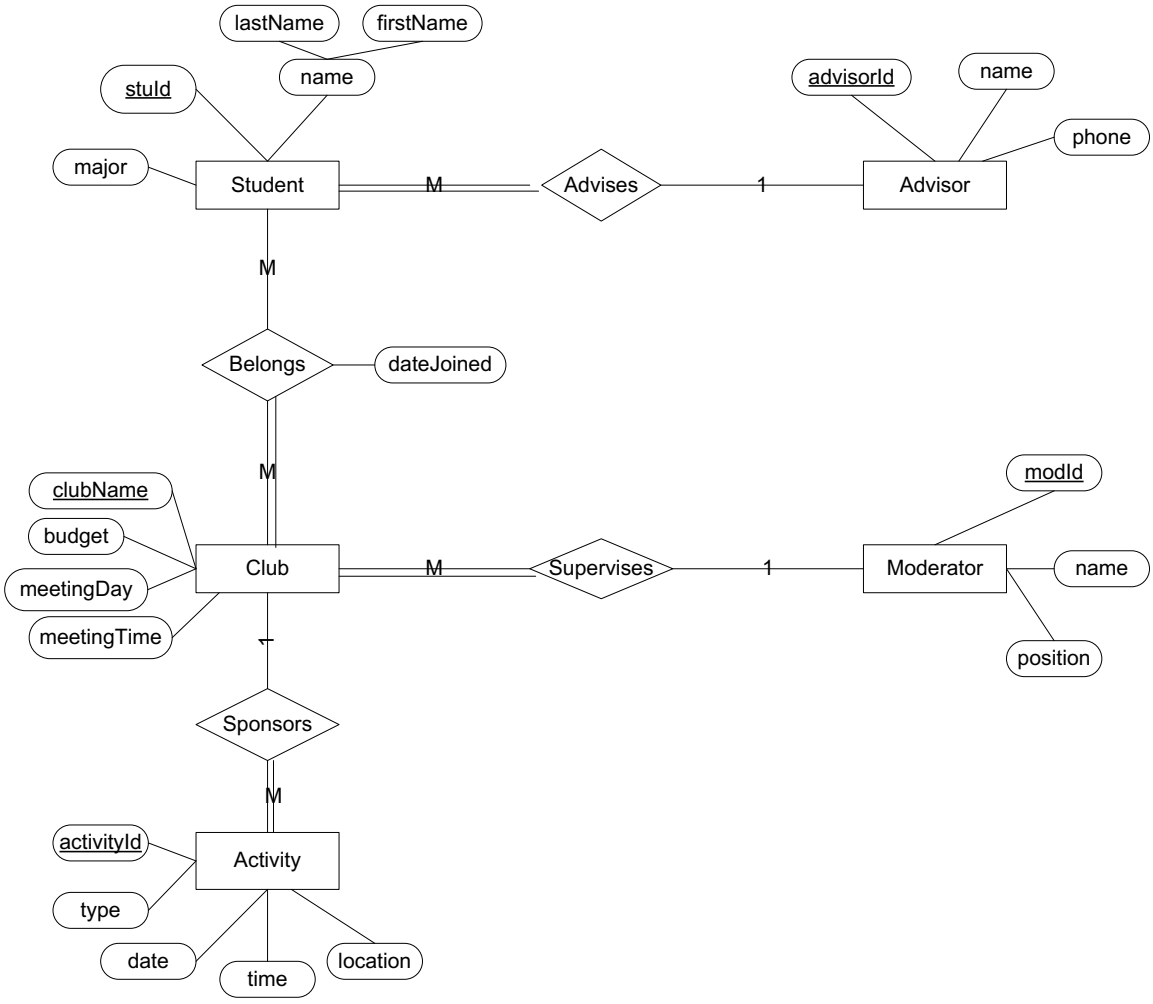


(b) We assume that each purchase must include at least one book, and may include several. Not all books are the result of a purchase, since the collector can acquire books through gifts, inheritance, barter, or other means. A book is purchased at most once by the collector. The 1 and M symbols and the single line and double line express these constratints.

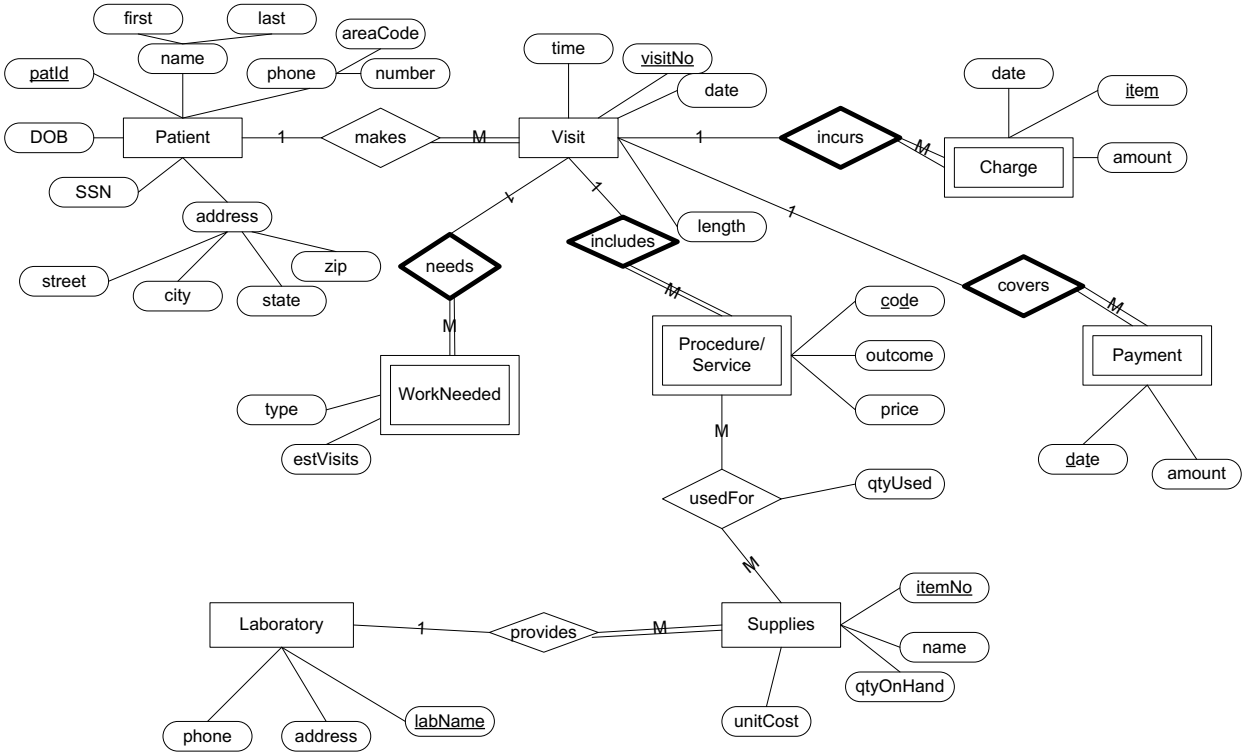
(c)



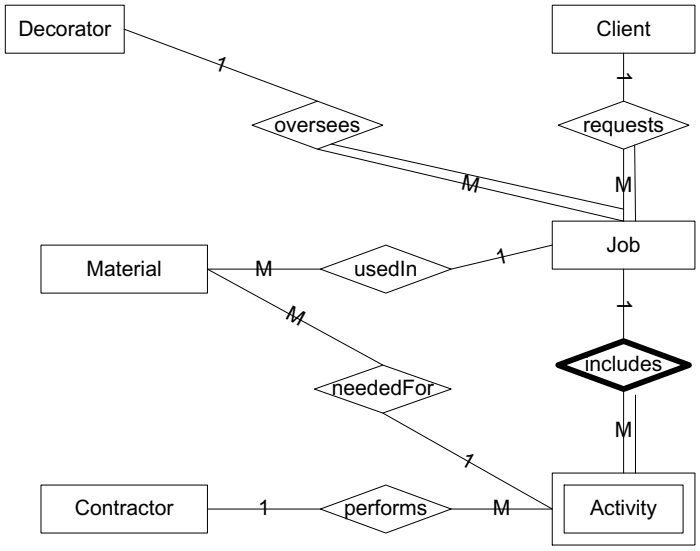
3.4 A possible solution is shown here, but variations can be correct as well.


lastName
firstName
name
stuId
major
Student
M
Advises
1
Advisor
advisorId
name
phone
M
Belongs
dateJoined
M
clubName
budget
meetingDay
meetingTime
Club
M
Supervises
1
Moderator
modId
name
position
1
Sponsors
M
activityId
type
date
time
location
Activity

3.5 A possible solution is shown here. (Note: heavy outline on a relationship diamond represents relationship for weak entity.)



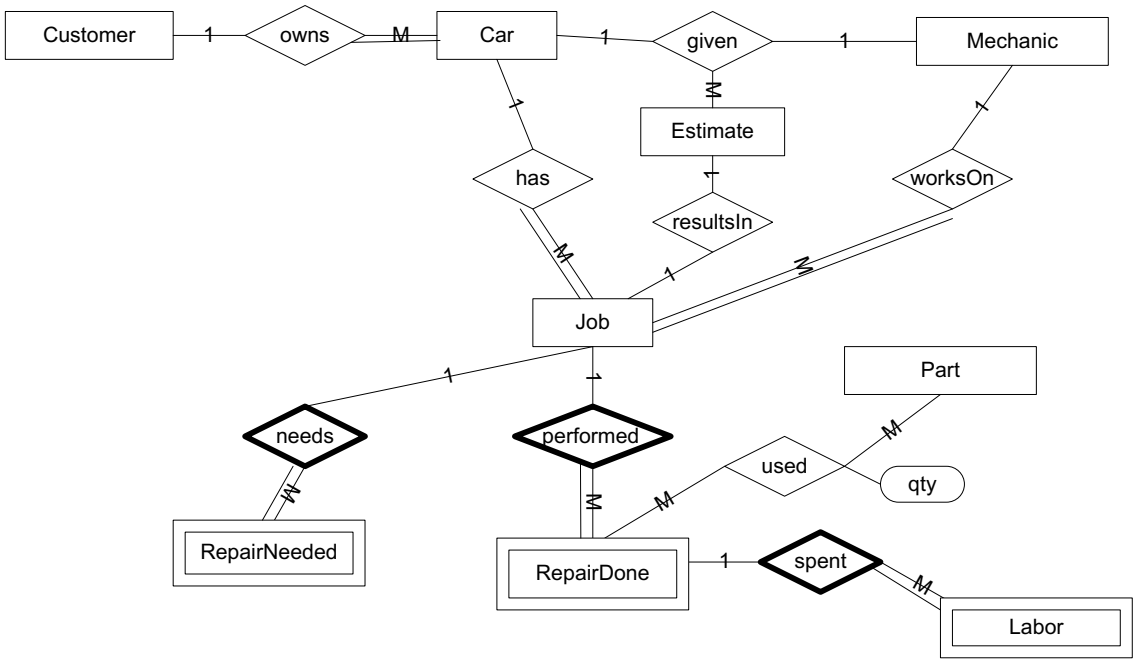
3.6 A possible solution is sketched here.



Attributes are omitted to save space. They could include
Decorator: empId, name(first, last), address(street, city, state, zip), phone(areaCode, number), mobilePhone, dateHired, salary

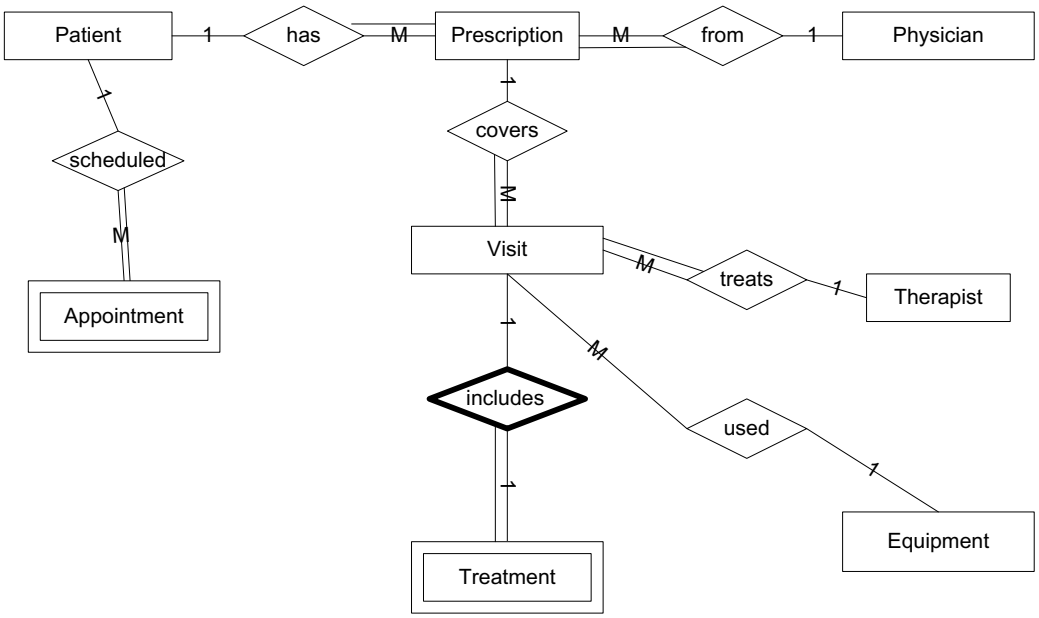
Client: clientId, name(first, last), address(street, city, state, zip), phone(areaCode, number), referredBy
Job: jobNo, description, estTime, estCost, plannedStartDate, plannedEndDate, actualStartDate, actualEndDate, actualTime, actualCost
Activity: activityName, description, estCost, estTime, plannedStartDate, plannedEndDate, actualStartDate, actualEndDate, actualTime, actualCost
Contractor: licNo, name(first, last), address(street, city, state, zip), phone(areacode, number), rating, specialty
Material: itemNo, manufacturer, description, supplier, quantity, cost

3.7 A possible solution is shown here.



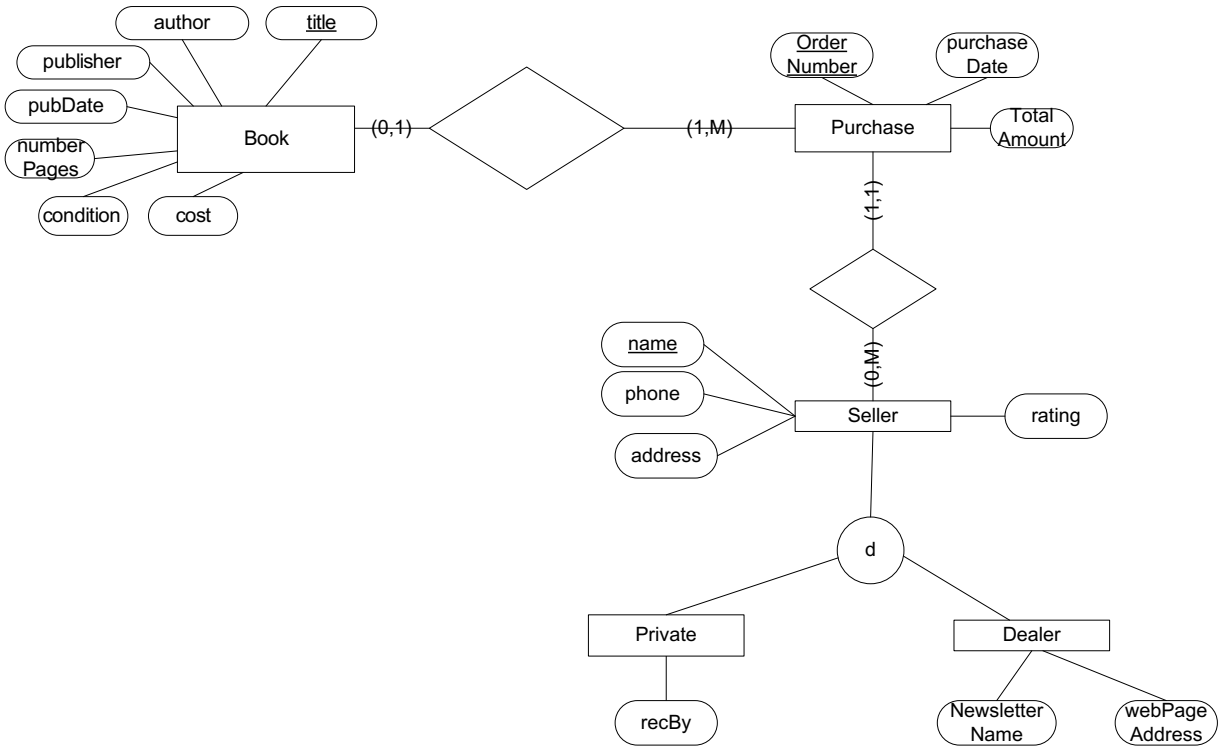
Attributes are omitted to save space. They could include
Customer: driverLicNo(state, number) , name(first, last), address(street, city, state, zip), phone(areacode, number), referredBy
Car: licPlateNo(state, number), VIN, make, model, year, color, mileage, cylinders
Mechanic: licNo, licDate, name(first, last), address(street, city, state, zip), phone(areacode, number), specialty
Estimate: estNo, dateGiven, timeNeeded, estCost
Job: jobNo, description, dateStarted, totalHoursSpent, totalCostOfParts, totalCostLabor, dateCompleted, totalCost
RepairNeeded: type, description, timeNeeded
RepairDone: type, description, dateStarted, dateCompleted, timeSpent
Part:partNo, partName, manufacturer, supplier, unitCost
Labor: type, numberHours, hourlyCost

3.8 A possible solution is shown here.

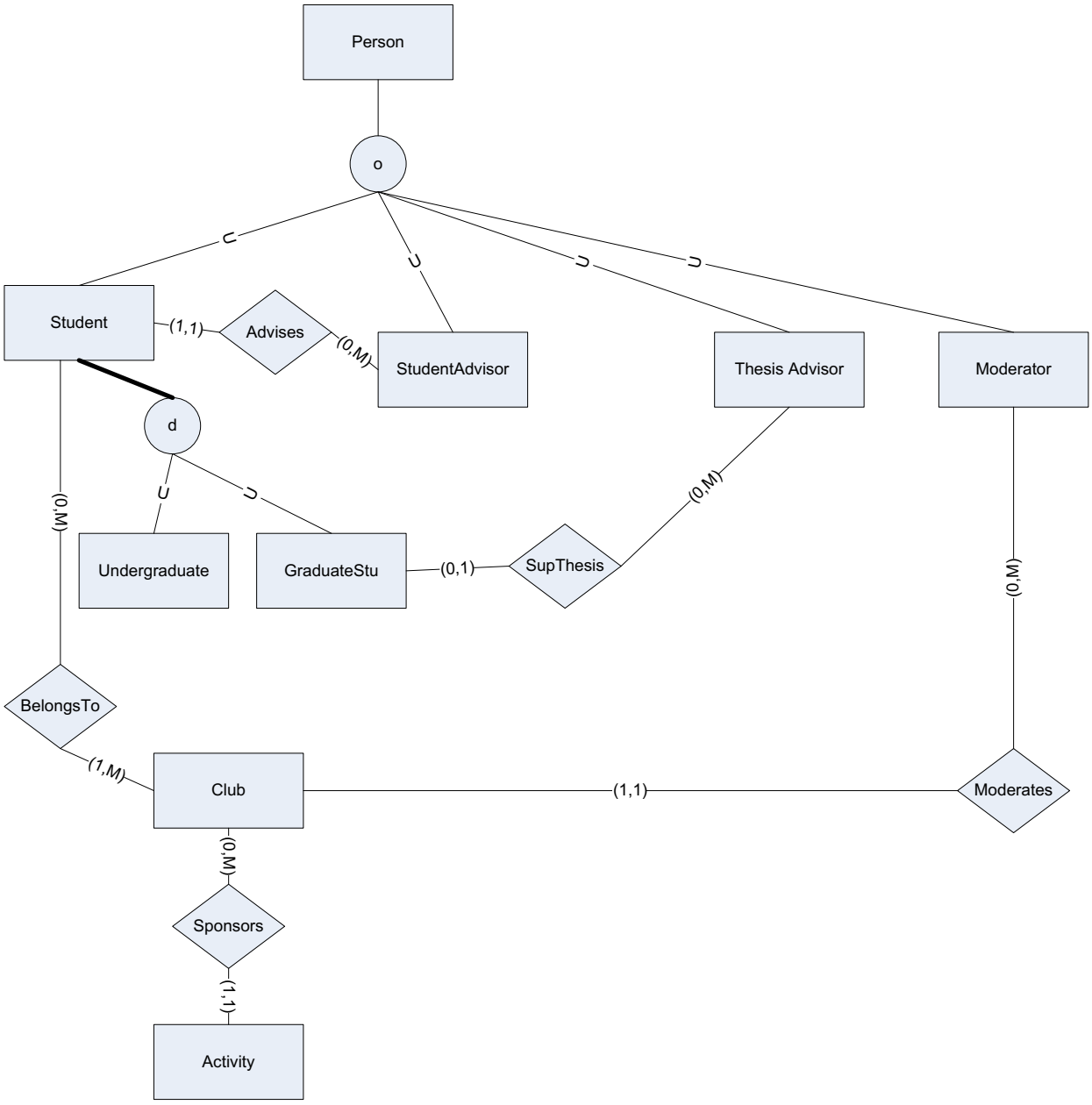


Attributes could be
Patient: patId, SSN, name(first, last), address(street, city, state, zip), phone(areaCode, number), DOB
Physician: NPI (National Provider Identifier, unique identifier in the US), licNo, name(first, last), office(street, city,state, zip), phone(areaCode,number), specialty
Prescription: RXNo, diagnosisCode, recTreatment, number, frequency, dateWritten, validStartDate,validEndDate
Appointment: date, time, duration
Visit: visitId, date, time, duration
Therapist: NPI, licNo, name(first, last), homeAddress(street, city,state, zip), homePhone(areaCode,number), specialty, dateHired)
Treatment: treatCode, type, duration, outcome
Equipment: equipmentId, type, manufacturer, datePurchased, price

3.9


author
title
publisher
pubDate
number Pages
condition
cost
Book
(0,1)
(1,M)
Order Number
purchase Date
Purchase
Total Amount
(1,1)
(0,M)
name
phone
address
Seller
rating
d
Private
recBy
Dealer
Newsletter Name
webPage Address

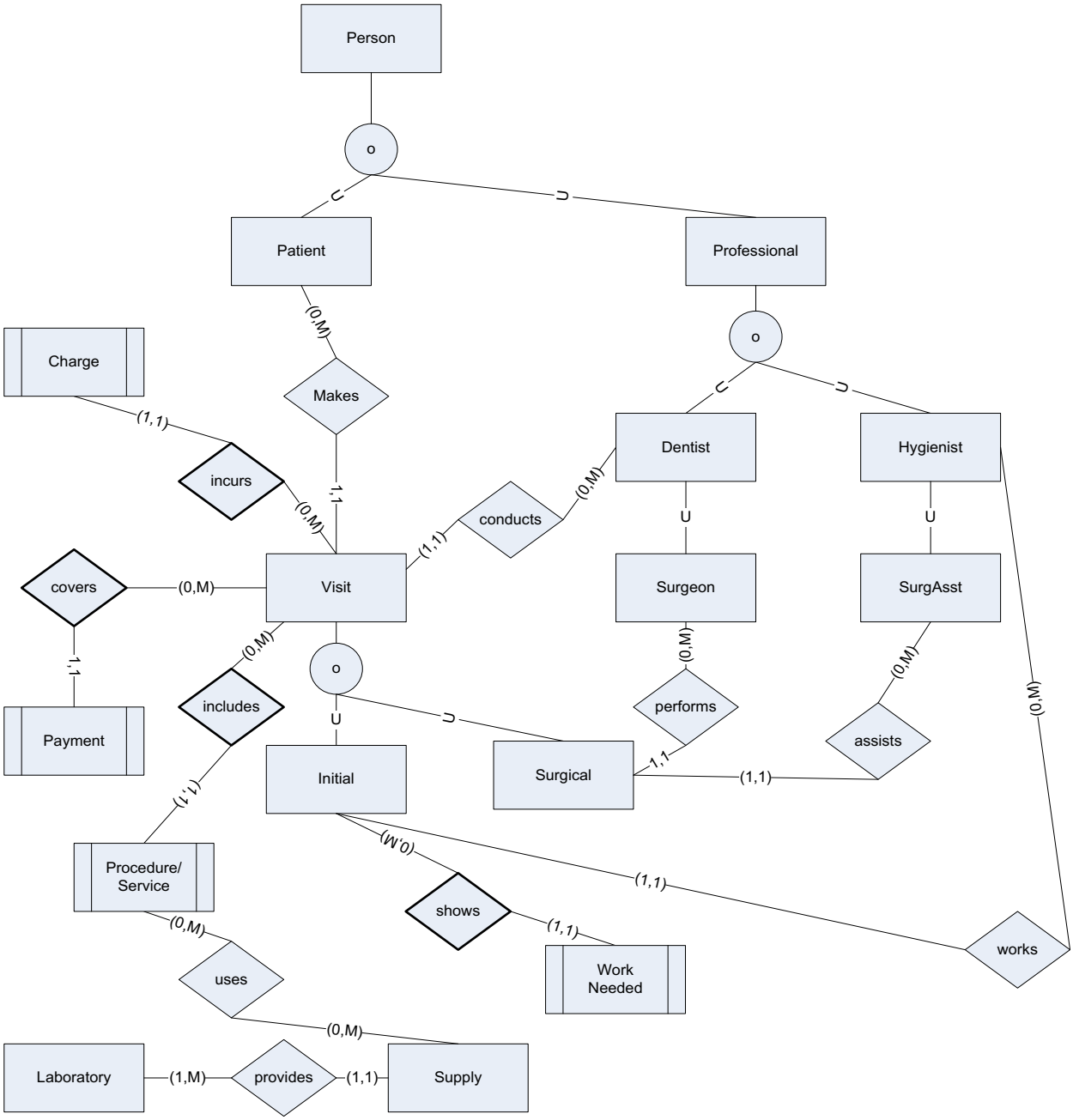
3.10



Attributes could be
Person: pId, name, address, phone
Student: credits, yearStarted
Undergraduate: major
GraduateStu: program
Club: clubName, budget, meetingDay
Activity: activityId, type, date, time, location
StudentAdvisor: department, title

ThesisAdvisor: researchArea
Moderator: position
BelongsTo relation: dateJoined

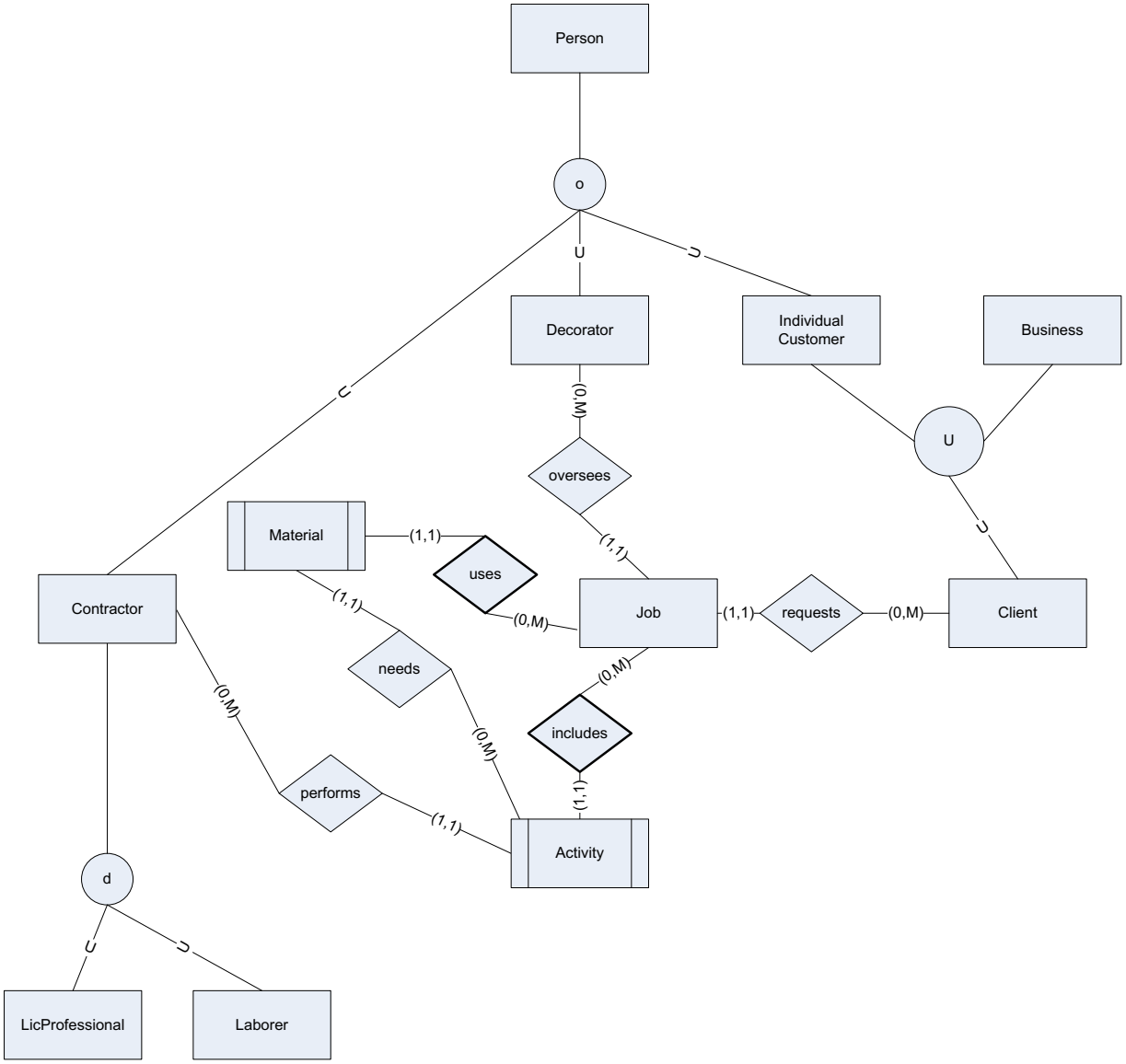
Note that since the Person specialization is overlapping, a graduate student can also be a moderator.
3.11



Attributes
Person: <u>pId</u>, SSN, name, address, phone
Patient: DOB
Professional: NPI, licNo, dateLic, type

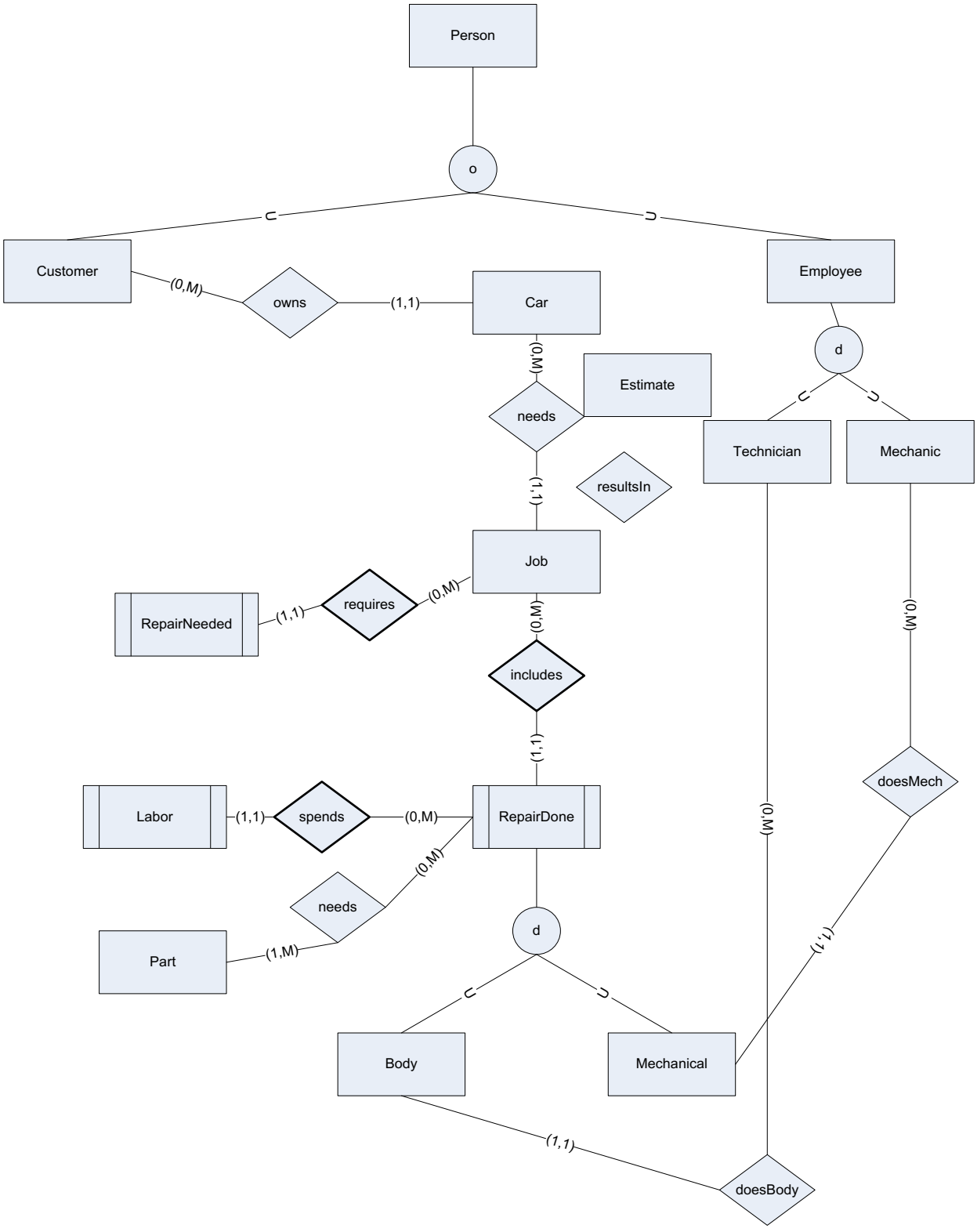
Dentist: specialty
Surgeon: certificationDate
Hygienist: no special attributes, only relationships
Surgical Assistant: yearsExperience
Visit: visitNo, date, time, length
Initial: no special attributes, only relationships
Surgical: no special attributes, only relationships
Charge: item, date, amount
Payment: date, amount
Procedure/Service: code, price, outcome
Supply: itemNo, name, qtyOnHand, unitCost
Laboratory: labName, address, phone
WorkNeeded: type, estVisits
Uses relationship: qtyUsed

3.12



Person: pId, name(first, last), address(street, city, state, zip), phone(areaCode, number)
Decorator: dateHired, salary, mobilePhone
Client: clientNo, type
IndividualCustomer:referredBy
Business: businessName, address, phone, contactPersonName, contactPersonPhone
Contractor: rating,specialty
LicProfessional: licNo, licState
Laborer: hourlyRate
Job: jobNo, description, estTime, estCost, plannedStartDate, plannedEndDate, actualStartDate, actualEndDate, actualTime, actualCost
Activity: activityName, description, estCost, estTime, plannedStartDate, plannedEndDate, actualStartDate, actualEndDate, actualTime, actualCost
Material: itemNo, manufacturer, description, supplier, quantity, cost

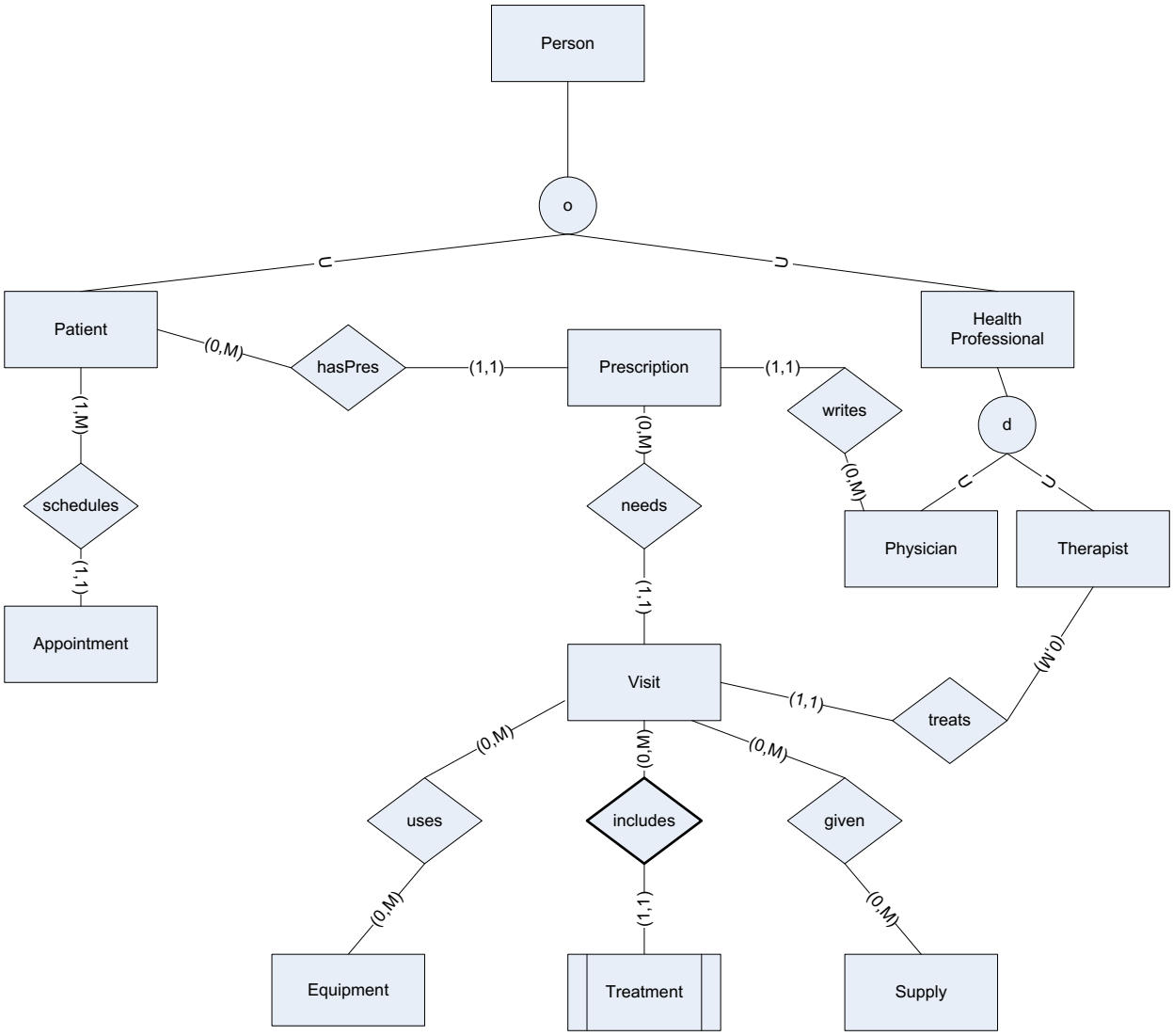
3.13.



Attributes are omitted to save space. They could include
Person: name(first, last), address(street, city, state, zip), phone(areacode, number))

Customer: driverLicNo(state, number), referredBy
Employee: empId,jobTitle, dateHired
Technician: no attributes, just relationship
Mechanic: mechaniclicNo
Car: licPlateNo(state, number), VIN, make, model, year, color, mileage, cylinders
Estimate: estNo, dateGiven, timeNeeded, estCost
Job: jobNo, description, dateStarted, totalHoursSpent, totalCostOfParts, dateCompleted, totalCost
RepairNeeded: type, description, timeNeeded
RepairDone: type, description, dateStarted, dateCompleted
BodyRepair: partRepaired
MechanicalRepair: systemRepaired
Part:partNo, partName, manufacturer, supplier, unitCost
Labor: type, numberHours, hourlyCost

3.14



Person: pId, name(first, last), address(street, city, state, zip), phone(areaCode, number)
Patient: SSN,DOB
HealthProfessional: NPI
Physician: specialty
Therapist: specialty, dateHired
Prescription: RXNo, diagnosisCode, recTreatment, number, frequency, dateWritten, validStartDate,validEndDate
Appointment: patientPId, date, time, duration
Visit: visitId, date, time, duration
Treatment: type, duration, outcome
Equipment: equipmentId, type, manufacturer, datePurchased, price
Treatment: type, duration, units, unitCost
Supply: itemName, itemDescription, supplierName, supplierAddress, price

Chapter 4 Questions:

4.1 Let S = {red, yellow, green} and T = {plaid, stripe, dot}. Find the Cartesian product of S and T.

4.2 Let Q = {Tom, Mary, Jim} and R = {walking, running}. Create a relation with Q and R as domains.

4.3 Consider the relation schema containing book data for a bookstore: Book (title, author, isbn, publisher, pubDate, pubCity, qtyOnHand)

a. Write out the table for an instance of this relation.

b. Identify a superkey, a candidate key, and the primary key, writing out any assumptions you need to make to justify your choice.

4.4 Consider the following database instance, which contains information about employees and the projects to which they are assigned:

Emp

empId	lastName
E101	Smith
E105	Jones
E110	Adams
E115	Smith

Assign

empId	projNo	hours
E101	P10	200
E101	P15	300
E105	P10	400
E110	P15	700
E110	P20	350
E115	P10	300
E115	P20	400

Proj

projNo	projName	budget
P10	Hudson	500000
P15	Columbia	350000
P20	Wabash	350000
P23	Arkansas	600000

Show ALL the tables (including the intermediate ones) that would be produced by each of the following relational algebra commands:

a. SELECT Emp WHERE lastName = 'Adams' GIVING T1 T1 JOIN Assign GIVING T2

Symbolically this is

$(\sigma_{\text{lastName='Adams'}}(\text{Emp}))$ |X| Assign

b. SELECT Proj WHERE budget > 400000 GIVING T1

T1 JOIN Assign GIVING T2

PROJECT T2 OVER empId GIVING T3

Symbolically this is

$\Pi_{\text{empId}}((\sigma_{\text{budget} > 400000}(\text{Proj}))|\text{X}|\ \text{Assign})$

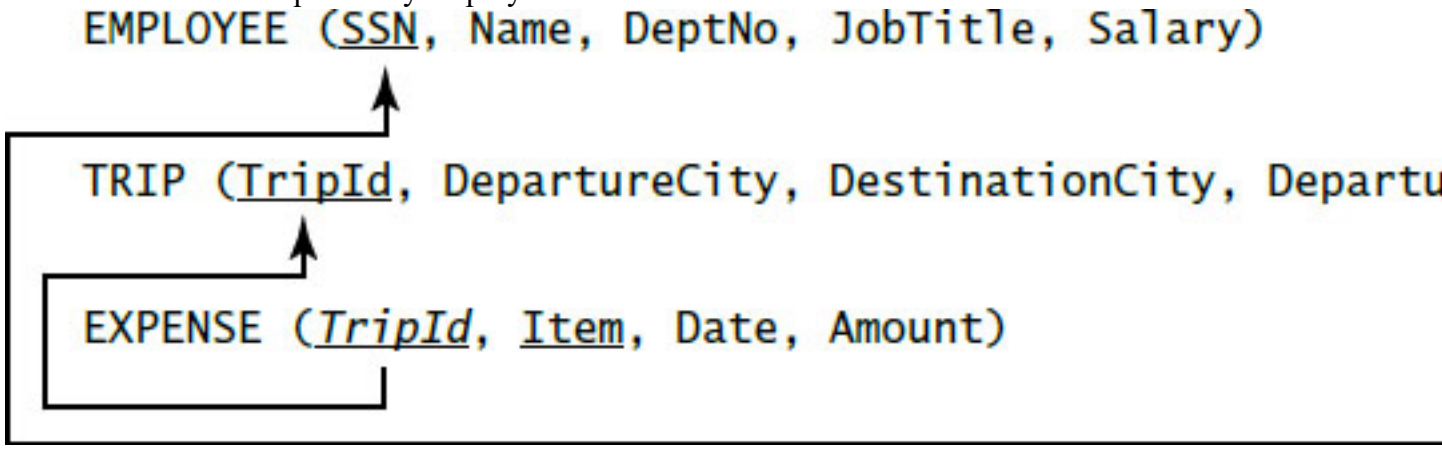
c. PROJECT Assign OVER projNo GIVING T1

T1 JOIN Proj GIVING T2

PROJECT T2 OVER budget GIVING T3

Symbolically this is
Π_{budget} (Π_{projNo} (Assign) |X| Proj)

4.5 Consider the following schema for a database that keeps information about business trips and their associated expenses by employees:



Write relational algebra queries for each of the following:

a. Get a list of all the different destination cities where the employees have taken trips.

b. Find all the employee information for employees who work in Department 10.

c. Get complete trip records (but not their associated expenses) for all trips with departure dates after January 1 of the current year.

d. Find the names of all employees who have departed on trips from London.

e. Find the SSN of all employees who have any single expense item of more than $1000 for any trip.

f. Find the names of all employees who have any expense item with value “Entertainment.”

g. Find the destination cities of all trips taken by employees who have the job title of “Consultant.”

h. Find the names and departments of all employees who have a single expense item of over $1000 since January 1 of this year.

i. Find the items and amounts of all expenses for a trip to Cairo beginning on January 3 of this year taken by employee Jones.

j. Find the names, departments, and job titles of all employees who have any expense item with value “Service Charge” for trips taken to Melbourne last year, along with the date and amount of the expense.

4.6 Design a relational database schema corresponding to the Customer and Order E-R diagram shown in

4.7 Design a relational database schema for the data about the book collection described in Exercises 3.2 and 3.3.

4.8 Design a relational database schema for the data about college students, academic advisors, and clubs described in Exercise 3.4.

4.9 Design a relational database schema for the data about the dental practice described in Exercise 3.5.

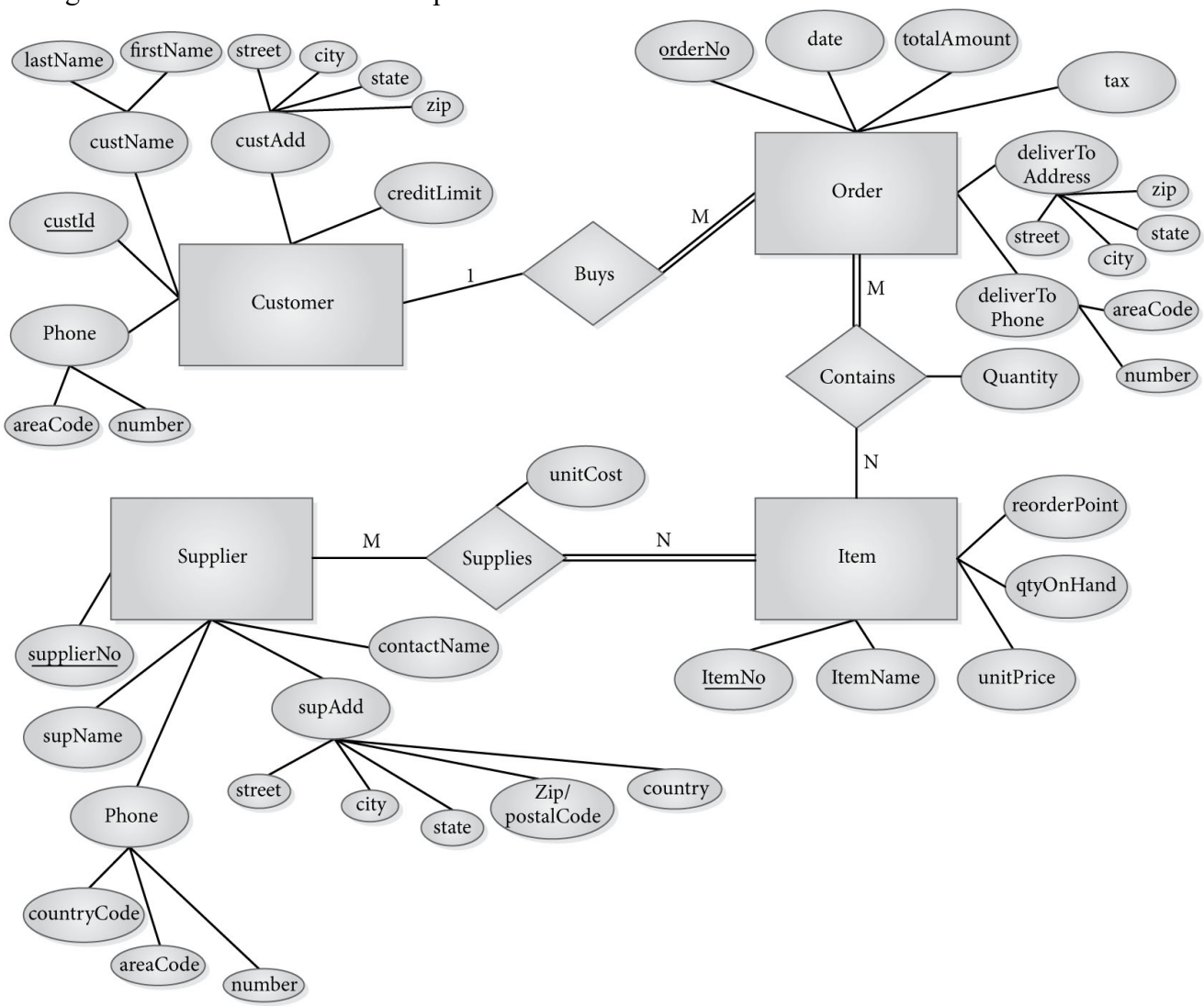
4.10 Design a relational database schema for the data about the interior design firm described in Exercise 3.6.

4.11 Design a relational database schema for the data about the automobile repair shop described in Exercise 3.7.

4.12 Design a relational database schema for the data about the physical therapy center described in Exercise 3.8.

4.13 Design a relational database schema for the data described in Exercise 3.9, including the specialization of book sellers.

FIGURE 4.9
E-R Diagram for CustomerOrder Example



4.14 Design a relational database schema for the data described in Exercise 3.10, including the specialization of students.

4.15 Design a relational database schema for the data described in Exercise 3.11, including the specialization of dentists.

4.16 Design a relational database schema for the data described in Exercise 3.12, including the specialization of clients.

4.17 Design a relational database schema for the data described in Exercise 3.13, including different types of work.

4.18 Design a relational database schema for the data described in Exercise 3.14, including equipment and treatments.

Chapter 4 Answers:

4.1 The Cartesian product is
S X T = {(red,plaid), (red,stripe), (red,dot), (yellow,plaid), (yellow,stripe), (yellow,dot), (green,plaid), (green,stripe), (green,dot)}. The other possible Cartesian product, T X S, consists of the nine ordered pairs formed by reversing each of these.

4.2 T ={(Tom, walking), (Mary, walking), (Jim, walking)} is one example.

4.3(a) A sample instance is the following
Book

title	author	isbn	publisher	pubDate	city	qtyOnHand
DB Illuminated	Ricardo&Urban	9781284056945	Jones&Bartlett	2017	Sudbury,MA	50
Java Prog	Smith	0123456789191	BigBooks	2016	New York	30
Software Eng	Jones	0987654321020	SmallBooks	2017	London	25

(b) We assume that every book has an ISBN that uniquely identifies the book, but not each copy of the book. We assume that a new ISBN is issued when a new edition of a book is published. ISBN is a superkey, a candidate key, and the primary key. Any combination of ISBN with other attributes is also a superkey. We assume that if a book has more than one author, only the one listed first is stored. The combination {title,author} is a superkey and a candidate key, if we assume that no author writes two books with the same title and that there is only one edition of each book available in the bookstore. Any combination of these two attributes with any other attributes is a superkey. If we assume that no publisher publishes two different books with the same title the combination {publisher, title}is also a superkey and a candidate key. Any combination these two attributes with any other attributes is a superkey.

4.4 (a) T1

empId	lastName
E110	Adams

T2

empId	lastName	projNo	hours
E110	Adams	P15	700
E110	Adams	P20	350

4.4(b)
T1

projNo	projName	budget
P10	Hudson	500000

P23	Arkansas	600000

T2

projNo	projName	budget	empId	hours
P10	Hudson	500000	E101	200
P10	Hudson	500000	E105	400
P10	Hudson	500000	E115	300

T3

empId
E101
E105
E115

4.4(c)

T1

projNo
P10
P15
P20

T2

projNo	projName	budget
P10	Hudson	500000
P15	Columbia	350000
P20	Wabash	350000

T3

budget
500000
350000

4.5 Note: There are often several ways to write the commands. Alternative answers should be considered.

(a) $\Pi_{destinationCity}(TRIP)$

(b) $\sigma_{DeptNo=10}(EMPLOYEE)$

(c) $\sigma_{DepartureDate>'01\text{-}Jan\text{-}2016'}(TRIP)$ Note: Adjust date for the current year

(d) $\Pi_{Name}(((\sigma_{DepartureCity='London'}(TRIP)) \mid x \mid EMPLOYEE))$

(e) $\Pi_{SSN}((\sigma_{Amount>1000}(EXPENSE)) \mid X \mid TRIP)$

(f) $\Pi_{Name}(((\sigma_{Item='Entertainment'}(EXPENSE)) \mid X \mid TRIP \mid X \mid) EMPLOYEE)$

(g) $\Pi_{DestinationCity}((\sigma_{JobTitle='Consultant''}(EMPLOYEE) \mid X \mid TRIP)$

(h) $\Pi_{Name,\ DeptNo}(((\sigma_{Amount>1000\ AND\ Date>'01\text{-}Jan\text{-}2016'}(EXPENSE)) \mid X \mid TRIP) \mid X \mid EMPLOYEE)$

(i) $\Pi_{Item,\ Amount}((\sigma_{DestinationCity='Cairo'\ AND\ departureDate='03\text{-}Jan\text{-}2016'}(TRIP)) \mid X \mid (\sigma_{Name='Jones'}(EMPLOYEE)))$

(j) $\Pi_{\text{Name, DeptNo, JobTitle, Date,Amount}}((\sigma_{\text{Item='ServiceCharge'}}(\text{ITEM})) \text{ |X| } (\sigma_{\text{DestinationCity='Melbourne' AND DepartureDate >'01-Jan-2003'}}(\text{TRIP})) \text{ |X| EMPLOYEE})$

4.6
Customer(custId, firstName, lastName, street, city, state, zip, areaCode, number, creditLimit)
Supplier(supplierNo, supName, countryCode, areaCode, number, street, city, state, zipOrPostal, country, contactName)
Order(orderNo, date, totalAmount, tax, delToStreet, delToCity, delToState, delToZip, delToAreaCode, delToNumber, *custId*)
Item(itemNo, itemName, unitPrice, qtyOnHand, reorderPoint)
Contains*(orderNo*, *itemNo*,quantity)
Supplies(*supplierNo*, *itemNo*, unitCost)

4.7 For the E-R diagram shown as a solution for Exercise 3.3(c), the following is a possible schema
Book(title, author, publisher, pubDate, numberPages, condition, cost, *orderNumber*)
Seller(name, address, phone, rating)
Purchase(orderNumber, purchaseDate, totalAmount, *sellerName*)

4.8 For the E-R diagram shown as a solution for Exercise 3.4, the following is a possible schema
Advisor (advisorId, name, phone)
Student(stuId, lastName, firstName, major, *advisorId*)
Moderator(modId, name, position)
Club(clubName, budget, meetingDay, *modId*)
Activity(activityId, type, date, time, location, *clubName*)
Belongs(*stuId, clubName*, dateJoined)

4.9 For the E-R diagram shown as a solution for Exercise 3.5, the following is a possible schema
Patient(patId, firstName, lastName, DOB, SSN, street, city, state, zip, phone)
Visit(visitNo, date, time, length, *patId*)
Laboratory(labName, address, phone)
Supplies(itemNo, name, qtyOnHand, unitCost, *labName*)
WorkNeeded(*visitNoDiagnosed*, type*, estVisits*)
ProcedureOrService(*visitNo*, code, price, outcome)
Charge(*visitNo*, item, date, amount)
Payment(*visitNo*, date, amount)
UsedFor(*visitNo*, *code*, *itemNo*, qtyUsed)

4.10 For the E-R diagram shown as a solution for Exercise 3.6, the following is a possible schema
Decorator(empId, firstName, lastName, address, phone, mobilePhone, dateHired, salary)
Client(clientId, firstName, lastName, street, city, state, zip, areacode, phoneNumber, referredBy)
Job(jobNo, description, estTime, estCost, plannedStartDate, plannedEndDate, actualStartDate, actualEndDate, actualTime, actualCost, *clientId, decEmpId*)
Contractor(licNo, firstName, lastName, street, city, state, zip, areacode, phoneNumber, rating, specialty)

Activity(*jobNo*, activityName, description, estCost, estTime, plannedStartDate, plannedEndDate, actualStartDate, actualEndDate, actualTime, actualCost*, conLicNo*)
Material(itemNo, manufacturer, description, supplier, quantity, cost, *jobNo, activityName*)

You might want to note that in Activity, conLicNo could be null, if no contractor is used.

4.11 For the E-R diagram shown as a solution for Exercise 3.7, the following is a possible schema
Customer(driverLicState,driverLicNumber, firstName, lastName, street, city, state, zip, areaCode, phoneNumber, referredBy)
Car(licPlateState, licPlateNumber, VIN, make, model, year, color, mileage, cylinders, *driverLicState, driverLicNo*)
Mechanic(mechLicNo, licDate, firstName, lastName, street, city, state, zip, areacode, phoneNumber, specialty)
Estimate(estNo, dateGiven, timeNeeded, estCost, *licPlateState, licPlateNumber, mechLicNo)*
Job(jobNo, description, dateStarted, totalHoursSpent, totalCostOfParts, totalCostLabor, dateCompleted, totalCost, *licPlateState, licPlateNo, workMechLicNo, estNo*)
RepairNeeded(*jobNo*, type, description, timeNeeded)
RepairDone(*jobNo*, type, description, dateStarted, dateCompleted, timeSpent)
Part(partNo, partName, manufacturer, supplier, unitCost)
PartUsed(*jobNo, repairType*, *partNo*, qty)
Labor(*jobNo*, *type,* numberHours, hourlyCost)

4.12 For the E-R diagram shown as a solution for Exercise 3.8, the following is a possible schema

Patient(patId, SSN, firstName, lastName, street, city, state, zip, areaCode, phoneNumber, DOB)
Physician(NPI, licNo, firstName, lastName, officeStreet, officeCity,officeState, officeZip, areaCode, phoneNumber, specialty
Prescription(RXNo, diagnosisCode, recTratment, number, frequency, dateWritten, validStartDate, validEndDate, *patId, physicianNPI*)
Appointment(patId, date, time, duration)
Therapist(NPI, licNo,firstName, lastName, homeStreet, homeCity, homeState, homeZip, homeAreaCode, homePhoneNumber, specialty, dateHired)
Visit(visitId, date, time, duration, *RXNo, therapistNPI, equipmentId*, equiptimeIn, equiptimeOut))
Treatment(*visitId*, treatmentCode, type, duration, outcome)
Equipment(equipmentId, type, manufacturer, datePurchased, price)

4.13 For the E-R diagram shown as a solution for Exercise 3.9, the following is a possible schema. For the Seller specialization, we used a single table with all attributes.
Book(title, author, publisher, pubDate, numberPages, condition, cost, *orderNumber*)
Seller(name, address, phone, rating, recBy, newsletterName, webAddress)
Purchase(orderNumber, purchaseDate, totalAmount, *sellerName*)

4.14 For the E-R diagram shown as a solution for Exercise 3.10, the following is a possible schema. For this example, we chose to create a table for the Person base type and individual tables for the four Person subtypes, using foreign keys to connect them. We created a single Student table for all students.

Person(pId, name, address, phone)
StudentAdvisor (*pid*, department, title)
ThesisAdvisor(*pid*, researchArea)
Moderator(*pId*, position)
Student(*pid*, credits, yearStarted, major, program, *stuAdisorId, thesisAdvisorId*)
Club(clubName, budget, meetingDay, *modId*)
Activity(activityId, type, date, time, location, *clubName*)
Belongs(*stuId, clubName*, dateJoined)

4.15 For the E-R diagram shown as a solution for Exercise 3.11, the following is a possible schema. For this example, we chose separate tables for Patients, Dentists, and Hygienists, but included the subsets Surgeon and SurgAssistant with their parent types.
Patient(patId, firstName, lastName, DOB, SSN, street, city, state, zip, phone)
Dentist(NPI, licNo, dateLic, SSN, name, address, phone, specialty, surgCertDate)
Hygienist(NPI, licNo, dateLic, SSN, name, address, phone,yearsSurgExp)
Visit(visitNo, date, time, length, *patId, dentistNPI, hygienistNPI*)
Laboratory(labName, address, phone)
Supply(itemNo, name, qtyOnHand, unitCost, *labName*)
WorkNeeded(*visitNoDiagnosed*, type, estVisits)
ProcedureOrService(*visitNo*, code, price, outcome)
Charge(*visitNo*, item, date, amount)
Payment(*visitNo*, payDate, amount)
UsedFor(*visitNo*, code, itemNo, qtyUsed)

4.16 For the E-R diagram shown as a solution for Exercise 3.12, the following is a possible schema. For this example, we chose not to create a Person table, but individual tables for Clients, Decorators, and Contractors. We used two tables for Clients, one for individuals and one for businesses, ignoring the union. This decision affects the Job table, which should have clientId as a foreign key. Because we must specify which table the foreign key refers to, we create two foreign keys, one to the individual client table, and one to the business client table. We use only one table for Contractors, both licensed professionals and laborers.

Decorator(empId, firstName, lastName, address, phone, mobilePhone, dateHired, salary)
IndividualClient(clientId, firstName, lastName, street, city, state, zip, areacode, phoneNumber, referredBy)
BusinessClient(clientId,street, city, state, zip, areacode, phoneNumber,contactName,contactPhone)
Contractor(contid, firstName, lastName, street, city, state, zip, areacode, phoneNumber, rating, specialty,licNo, licState, hourlyRate)
Job(jobNo, description, estTime, estCost, plannedStartDate, plannedEndDate, actualStartDate, actualEndDate, actualTime, actualCost, *indivclientId, busclientId, decEmpId*)
Activity(*jobNo*, activityName, description, estCost, estTime, plannedStartDate, plannedEndDate, actualStartDate, actualEndDate, actualTime, actualCost, *contId*)
Material(itemNo, manufacturer, description, supplier, quantity, cost, *jobNo, activityName*)

4.17 For the E-R diagram shown as a solution for Exercise 3.13, the following is a possible schema. For this example, we chose to create a single Employee table that holds both technicians and licensed mechanics. For RepairDone, we created a table for the entire repair, with subtables for BodyRepair and MechanicalRepair.

Customer(driverLicState,driverLicNumber, firstName, lastName, street, city, state, zip, areaCode, phoneNumber, referredBy)
Car(licPlateState, licPlateNumber, VIN, make, model, year, color, mileage, cylinders, *driverLicState, driverLicNo*)
Employee(empId, firstName, lastName, street, city, state, zip, areacode, phoneNumber, date hired, specialty, mechLicNo licDate)
Estimate(estNo, dateGiven, timeNeeded, estCost, *licPlateState, licPlateNumber, mechLicNo)*
Job(jobNo, description, dateStarted, totalHoursSpent, totalCostOfParts, totalCostLabor, dateCompleted, totalCost, *licPlateState, licPlateNo, workMechLicNo, estNo*)
RepairNeeded(*jobNo*, type, description, timeNeeded)
RepairDone(*jobNo*, type, description, dateStarted, dateCompleted, timeSpent)
BodyRepairDone(*jobNo, type,technicianEmpId*)
MechanicalRepariDone(*jobNo,type, mechanicEmpId*)
Part(partNo, partName, manufacturer, supplier, unitCost)
PartUsed(*jobNo, repairType*, *partNo*, qty)
Labor(*jobNo*, *type,* numberHours, hourlyCost)

4.18 For the E-R diagram shown as a solution for Exercise 3.14, the following is a possible schema. For this example, we chose to keep separate tables for Patient, Physician, and Therapist, ignoring the Person and the Health Professional generalizations. The schema is the same as the one for Exercise 4.12.

Patient(patId, SSN, firstName, lastName, street, city, state, zip, areaCode, phoneNumber, DOB)
Physician(NPI, licNo, firstName, lastName, officeStreet, officeCity,officeState, officeZip, areaCode, phoneNumber, specialty
Prescription(RXNo, diagnosisCode, recTratment, number, frequency, dateWritten, validStartDate, validEndDate, *patId, physicianNPI*)
Appointment(patId, date, time, duration)
Therapist(NPI, licNo,firstName, lastName, homeStreet, homeCity, homeState, homeZip, homeAreaCode, homePhoneNumber, specialty, dateHired)
Visit(visitId, date, time, duration, *RXNo, therapistNPI, equipmentId*, equiptimeIn, equiptimeOut))
Treatment(*visitId*, treatmentCode, type, duration, outcome)
Equipment(equipmentId, type, manufacturer, datePurchased, price)

Chapter 6 Questions:

6.1 Consider the following universal relation that holds information about the inventory of books in a bookstore:

Books (title, isbn, author, publisherName, publisherAdd, totalCopiesOrdered, copiesInStock, publicationDate, category, sellingPrice, cost)

Assume:

- The isbn uniquely identifies a book. (It does not identify each copy of the book, however.)
- A book may have more than one author.
- An author may have more than one book.
- Each publisher name is unique. Each publisher has one unique address—the address of the firm's national headquarters.
- Titles are not unique.
- totalCopiesOrdered is the number of copies of a particular book that the bookstore has ever ordered, while copiesInStock is the number still unsold in the bookstore.
- Each book has only one publication date. A revision of a book is given a new ISBN.
- The category may be biography, science fiction, poetry, and so on. The title alone is not sufficient to determine the category.
- The sellingPrice, which is the amount the bookstore charges for a book, is always 20 percent above the cost, which is the amount the bookstore pays the publisher or distributor for the book.

a. Using these assumptions and stating any others you need to make, list all the non-trivial functional dependencies for this relation.
b. What are the candidate keys for this relation? Identify the primary key.
c. Is the relation in third normal form? If not, find a 3NF lossless join decomposition of Books that preserves dependencies.
d. Is the relation or resulting set of relations in Boyce-Codd Normal Form? If not, find a lossless join decomposition that is in BCNF. Identify any functional dependencies that are not preserved.

6.2 Consider the following relation that stores information about students living in dormitories at a college:

College (lastName, stuId, homeAdd, homePhone, dormRoom, roommateName, dormAdd, status, mealPlan, roomCharge, mealPlanCharge)

Assume:

- Each student is assigned to one dormitory room and may have several roommates.
- Names of students are not unique.
- The college has several dorms. dormRoom contains a code for the dorm and the number of the particular room assigned to the student. For example, A221 means Adams Hall, room 221. Dorm names are unique.
- The dormAdd is the address of the dorm building. Each building has its own unique address. For example, Adams Hall may be 123 Main Street, Anytown, NY 10001.
- status tells the student's status: Freshman, Sophomore, Junior, Senior, or Graduate Student.
- mealPlan tells how many meals per week the student has chosen as part of his or her meal plan. Each meal plan has a single mealPlanCharge associated with it.

› The roomCharge is different for different dorms, but all students in the same dorm pay the same amount.

Answer questions (a)–(d) as in Exercise 6.1 for this example.

6.3 Consider the following attributes for tables in a relational model designed to keep track of information for a moving company that moves residential customers, usually from one home or apartment to another:

customerID, customerName, customerCurrentAddress, customerCurrentPhone, customerNewAddress, customerNewPhone,
pickupLocation, dropOffLocation, dateOfMove, startingTime,
estimatedWeight, estimatedCost, truck#Assigned, driverName,
driverLicNo, actualCost, amountOfDamages, truckCapacity,
costOfTolls, tax, finalAmount, invoiceNumber, amountPaid,
datePaid, checkNumber, amountDue

Assume:

› Although in most cases the pickupLocation is the customer's old address and the dropOffLocation is the new address, there are exceptions, such as when furniture is moved to or from storage.
› An estimate is provided before the move using a pre-printed invoice containing a unique invoice number. The actual cost is recorded on the same form once the move is complete. The actual cost may differ from the estimated cost. The final amount includes the actual cost of the move, plus tax and tolls.
› In most cases, the customer pays the final amount immediately, but sometimes he or she pays only part of the bill, especially if the amount of damages is high. In those cases, there is an amount due, which is the difference between the final amount and the amount paid.

(a)–(d) Answer questions (a)–(d) as in Exercise 6.1 for this example.
(e) What tables would you actually implement? Explain any denormalization, omissions, or additions of attributes.

6.4 The owner of a catering company wants a database to keep track of various aspects of the business. Clients may be individuals or businesses who make arrangements for events such as weddings, dances, corporate dinners, fundraisers, etc. The company owns a catering hall in which only one event can take place at a time. In addition, clients can use the services of the company to cater events at their homes, places of business, or places that the client rents, such as historic mansions. Several of these off-site events can take place at the same time. Clients who rent the catering hall must guarantee a minimum of 150 guests, but the hall can accommodate up to 200. Reservations for the hall are accepted up to two years in advance. Clients who provide their own space can have any number of guests. Reservations for off-site events are usually made several months in advance. The firm provides the food, table settings (dishes, silverware, and glassware), linens, waiters, and bartenders for the event, regardless of whether it is in the catering hall or elsewhere. Customers can choose the color of the linen. The firm has set menus that are identified by number. For a given menu number, there is one appetizer, one salad, one main course, and one dessert. (For example, menu number 10 may include shrimp cocktail, Caesar salad, prime rib, and chocolate mousse.) The firm quotes a total price for its services, based on the number of guests, the menu, the location, and intangible factors such as how busy they are on that date. The firm can also

contract for floral arrangements, musicians, entertainers, and photographers, if the client requests. The client pays the catering company, which then pays the contractor. The price charged the customer for each of these is always 10% above the cost to the catering company. Assume names are unique. Also assume musicians and entertainers provide only one type of music or entertainment.

Assume the following attributes have been identified:

clientLName, cFName, cPhone, cStreet, cCity, cState,
cZip, eventDate, eventStartTime, eventDuration,
eventType, numberGuests, locationName, locationStreet,
locationCity, locationState, locationZip,
linenColorRequested, numberWaiters, numberBartenders,
totalPrice, floristName, floristPhone, floristCost,
floristPrice, musicContact, musicContactPhone,
musicType, musicCost, musicPrice, entertainerName,
entertainerPhone, entertainerType, entertainerCost,
entertainerPrice, photographerName, photographerPhone,
photographerCost, photographerPrice, menuNumberChosen,
menuAppetizer, menuSalad, menuMain, menuDessert

(a)–(d) Answer questions (a)–(d) as in Exercise 6.1 for this example. (e) What tables would you actually implement? Explain any denormalization, omissions, or additions of attributes.

6.5 Consider the following relations, and identify the highest normal form for each, as given, stating any assumptions you need to make.

a. Work1 (empId, empName, dateHired, jobTitle, jobLevel)
b. Work2 (empId, empName, jobTitle, ratingDate, raterName, rating)
c. Work3 (empId, empName, projectNo, projectName, projBudget, empManager, hoursAssigned)
d. Work4 (empId, empName, schoolAttended, degreeAwarded, graduationDate)
e. Work5 (empId, empName, socialSecurityNumber, dependentName, dependentAddress, relationToEmp)

6.6 For each of the relations in Exercise 6.5, identify a primary key and:

a. if the relation is not in third normal form, find a 3NF lossless join decomposition that preserves dependencies.
b. if the relation or resulting set of relations is not in Boyce-Codd Normal Form, find a lossless join decomposition that is in BCNF. Identify any functional dependencies that are not preserved.

6.7 Normalize the relational database schema developed for Exercise 4.6, which was based on Figure 4.9.

6.8 Normalize the relational database schema developed for Exercise 4.7.

6.9 Normalize the relational database schema developed for Exercise 4.8.

6.10 Normalize the relational database schema developed for Exercise 4.9.

6.11 Normalize the relational database schema developed for Exercise 4.10.

6.12 Normalize the relational database schema developed for Exercise 4.11.

Chapter 6 Answers:

6.1 (a) Using only the stated assumptions, nontrivial functional dependencies are

isbn → {title, publisherName, publisherAdd, totalCopiesOrdered, copiesInStock, publicationDate, category, sellingPrice, cost}
publisherName → publisherAdd
cost →sellingPrice
sellingPrice → cost
{isbn, author} → all attributes
If no author writes two books with the same title, then
{title, author} → all attributes
If no publisher publishes two books with the same title, then
{title, publisher_name} → all attributes

(b) Candidate keys are {isbn, author} and {title, author}. We choose {isbn, author} as the primary key.
(c) The relation is not in third normal form because of the transitive dependencies involving publisherName and cost. It may be decomposed into the following schema that preserves dependencies

Book (isbn, author, title, *publisherName*, totalCopiesOrdered, copiesInStock, publicationDate, category, *cost*)
Pub (publisherName, publisherAddress)
Cash (cost, sellingPrice)

(d) These relations are already BCNF because every determinant ({isbn, author}, publisherName, cost, sellingPrice, {title, author}, { title ,publisherName }) is a candidate key in its own relation.

6.2 (a) Using the stated assumptions, adding that no two roommates have the same name, nontrivial functional dependencies are

{stuId, roommateName} → all attributes
stuId → lastName, homeAdd, homePhone, dormRoom, dormAdd, status, mealPlan, roomCharge, mealPlanCharge
dormRoom → dormAdd
mealPlan → mealPlanCharge
dormRoom → roomCharge
dormAdd → roomCharge

It is also possible to justify the following FDs

{stuName, homeAdd} → all attributes, assuming that no two students have the same name and home address.

homePhone → homeAdd, assuming each home telephone number is assigned to only one address. This gives us {stuName, roommateName, homePhone} → all attributes
(b) Candidate keys are {stuid, roommateName} and {stuName, roommateName, homePhone}. We will use {stuid, roommateName} as the primary key.
(c) Using the primary key chosen, the relation is not third normal form because of partial and transitive dependencies. We can decompose it as follows

Student (stuid, lastName, *homePhone, dormRoom*, status, *mealPlan*)
Dorm (dormRoom, dormAdd)
Meals (mealPlan, mealPlanCharge)
Dormcharge (dormAdd, roomCharge)
Phone (homePhone, homeAdd)
Roommates(*stuid*, roommateName)

In practice, a designer would be likely to ignore the dependency involving homePhone and keep the telephone number with the address in the Student relation, rather than keeping a table containing just home addresses and home telephone numbers. However, that decision is based on other considerations, and violates third normal form.
(d) Each of the relations is now BCNF. However, we have lost some dependency information. We no longer see stuId, roommateName and homePhone in the same relation, so we lose the fact that it is a candidate key for College.

6.3 (a) FDs are
invoiceNumber → all attributes
customerID → {customerName, customerCurrentAddress, customerCurrentPhone, customerNewAddress, customerNewPhone}
truck#Assigned → truckCapacity
driverLicNo → driverName
{actualCost, costOfTolls, tax} → finalAmount
{finalAmount, amountPaid} → amountDue // any two of these attributes determine the remaining one
{customerId, checkNumber} → amountPaid //Assuming a customer's check numbers are unique to that person

(b) The attribute invoiceNumber is a candidate key, and the primary key.

(c) The relation is not in third normal form because of transitive dependencies. We project it into
Invoice(invoiceNumber, pickupLocation, dropOffLocation, dateOfMove, startingTime, estimatedWeight, estimatedCost, *truck#Assigned*, *driverLicNo*, actualCost, amountOfDamages, costOfTolls, tax, finalAmount, invoiceNumber, datePaid, checkNumber, *customerID)*
Customer(customerID ,customerName, customerCurrentAddress, customerCurrentPhone, customerNewAddress, customerNewPhone)
Truck(truck#Assigned, truckCapacity)
Driver(driverLicNo, driverName)
Cost(actualCost, costOfTolls,tax, finalAmount)
Payment(finalAmount, amountPaid amountDue)

Check(customerID, checkNumber, amountPaid)

(d) This schema is in BCNF.
(e) It is not necessary to keep the Cost table, since finalAmount is easily calculated. The same is true of the Payment table, since amountDue is easily calculated. The Check table removes the amountPaid from the new Invoice table, which seems like a piece of information we would like to keep with that table. The checkNumber is less important and removing that eliminates the transitive dependency. Therefore we will remove checkNumber completely, put amountPaid back in the Invoice table, and drop the Check table. The final schema is

Invoice(invoiceNumber, pickupLocation, dropOffLocation, dateOfMove, startingTime, estimatedWeight, estimatedCost, *truck#Assigned*, *driverLicNo*, actualCost, amountOfDamages, costOfTolls, tax, finalAmount, invoiceNumber, datePaid, checkNumber, *customerID*, amountPaid*)*
Customer(customerID ,customerName, customerCurrentAddress, customerCurrentPhone, customerNewAddress, customerNewPhone)
Truck(truck#Assigned, truckCapacity)
Driver(driverLicNo, driverName)

6.4 (a) FDs include
{clientLName, cFName} → cPhone, cStreet, cCity, cState, cZip
Zip → cCity, cState
{clientLName, cFName, eventDate, eventStartTime} → all attributes (We assume no client ever has more than one event at the same time – if not true, add locationName to determinant)
menuNumberChosen → menuAppetizer, menuSalad, menuMain, menuDessert
locationName → locationStreet, locationCity, locationState, locationZip
locationZip → locationStreet, locationCity, locationState
floristName → floristPhone
floristCost → floristPrice
musicContact → musicContactPhone, musicType
musicCost → musicPrice
entertainerName → entertainerPhone, entertainerType
entertainerCost → entertainerPrice
photographerName → photographerPhone
photographerCost → photographerPrice

(b) A candidate key is {clientLName, cFName, eventDate, eventStartTime}. We will use that as the primary key.
(c) The relation is not in 3NF because of partial FDs on the key, and transitive dependencies. A 3NF representation is
Client(clientLName, cFName, cPhone, cStreet, *cZip*)
CZips(cZip, cCity, cState)
Event(*clientLName, cFName*, eventDate, eventStartTime, eventDuration, eventType, numberGuests, *locationName*, linenColorRequested, numberWaiters, numberBartenders, totalPrice, *floristName, floristCost*, *musicContact, musicCost, entertainerName, entertainerCost, photographerName, photographerCost, menuNumberChosen*)
Menu(menuNumberChosen, menuAppetizer, menuSalad, menuMain, menuDessert)

Location(locationName,locationStreet, *locationZip*)
LZips(locationZip, locationCity, locationState)
Florist(floristName, floristPhone)
Music(musicContact, musicContactPhone, musicType)
Entertainer(entertainerName, entertainerPhone, entertainerType)
Photographer(photographerName, photographerPhone)
Costs(cost, price) // applies to florist, music, entertainer and photographer costs and prices
(d) This schema is BCNF
(e) The two Zips tables should be combined. The Costs table can be dropped, since the price of outside services is easily calculated from cost. We might consider creating a clientID instead of using the name, and creating an ID for the event, since the key in the Event table is cumbersome.

6.5 (a) In Work1, we assume empName is not unique, so empId is the only candidate key. We also assume that jobTitle does not functionally determine or functionally depend on jobLevel, although the two may be related in some way. We assume the same about dateHired and jobLevel. Using empId as the primary key, the relation is in BCNF since the only determinant is the primary key. It has no multivalued dependencies, so it is 4NF. Since the only lossless decomposition would have to involve empId, it would be trivial, and the relation is 5NF. We cannot state whether it is DKNF without more information about the relationship, if any, between jobTitle and jobLevel.
(b) In Work2, we assume that empId alone uniquely identifies the employee. We assume that only the present jobTitle is stored, and that each employee has only one job title at any given time. If we also assume that only the latest rating information is stored, then ratingDate, raterName, and rating have only one value each, assuming only one person (e.g. the manager) rates an employee. In that case, the relation is BCNF, 4NF, and 5NF, as in part (a). However, if more than one rating can be stored, the relation is not 1NF if empId is used as the primary key. Let us assume ratings are stored for some fixed period, say two years. Then an employee may be rated several times during that period, and all the ratings may be stored, giving multiple values for ratingDate, raterName, and rating for the same empId value.
(c) In Work3, we assume empId uniquely determines the employee. If an employee can only be assigned to work on one project, the relation is 1NF and 2NF, but not 3NF, since projectNo determines projectName and projectBudget. If an employee can be assigned to work on more than one project, the relation is not 1NF, since there would be multiple values for projectNo for a given empId value. In that case, the example is similar to the one presented in Section 6.5
(d) In Work4, empId uniquely identifies the employee. If we assume the database stores only the highest or latest degree, the relation is 5NF. However, if we store all the degrees an employee has, the relation is not 1NF.
(e) In Work5, we assume both empId and socialSecurityNumber uniquely identify the employee. An employee may have several dependents, so the relation, as written, is not 1NF assuming either empId or socialSecurityNumber is used as the key. Note that the presence of another candidate key here is not a problem.

6.6 (a) The primary key of work1 is empid. The relation is already 3NF.

The primary key of Work2 is empId if only one rating can be stored, and the relation is 3NF in that case. However, if multiple ratings can be stored for each employee, we can make the relation 1NF by using {empId,rating_date} as the key, if we assume only one person rates an employee on a

given day. The resulting relation is not 2NF, since empId alone determines empName and jobTitle, so we have a non-full functional dependency. We need to decompose the relation into the following set of 3NF relations:
Work2a (empId, empName, jobTitle)
Work2b (*empId*, ratingDate, raterName, rating)

The primary key of Work3 is empId if an employee can only be assigned to one project. Then, the relation is 2NF but not 3NF. To make it 3NF, we decompose the relation into the following set
Work3a (empId, empName, *projectNo*, empManager, hoursAssigned)
Work3b (projectNo, projectName, projBudget).

If an employee can be assigned to more than one project, the relation can be made 1NF by using {empId, projectNo} as the primary key, but the resulting relation is not 2NF. We use the following 3NF decomposition
Work3c (empId, empName, empManager)
Work3d (projectNo, projectName, projBudget)
Work3e (*empId,projectNo*, hoursAssigned)

The primary key of Work4 is empId if only one degree is stored. In that case, the relation is already 3NF. However, if more than one degree can be stored, the combination {empId, degree} is the primary key. We assume an employee never earns the same degree twice. Note that we could use graduationDate in place of degree if an employee never receives two degrees on the same date. We do not choose {empId, schoolAttended} since employees may receive more than one degree from a school. Using {empId, degree}, the relation is not 2NF, since empId alone determines empName. We therefore use the following decomposition
Work4a (empId, empName)
Work4b (empId,degree, schoolAttended, graduationDate)
These relations are 3NF.

Work5 is not 1NF because an employee may have several dependents. Note that the combination of empId and dependentName may not be unique, since, for example, a spouse and a child of the employee may have the same name. Adding dependentAddress to this combination does not solve the problem, since both of the dependents may reside at the same address. We can choose {empId, dependentName, relationToEmp} as the primary key provided no two people who have the same name would have the same relationship to a particular employee (e.g. no two children in the same nuclear family have the same name). The relation is not 2NF since empName and socialSecurityNumber depend on empId alone. We can decompose the relation into a set of 3NF relations as follows
Work5a (empId, empName, socialSecurityNumber)
Work5b (empId,dependentName,relationToEmp, dependentAddress)

(b) Work1 is already BCNF.

Work2a and Work2b are both BCNF and preserve dependencies.

Work3a and Work3b are BCNF and preserve dependencies. Using the assumption that a worker can be assigned to more than one project, Work3c, Work3d, and Work3e are also BCNF and dependency-preserving.

Work4a and Work4b are BCNF and preserve dependencies.

Work5a and Work5b are BCNF. However, we have lost the information that the combination {socialSecurityNumber, dependentName, relationToEmp} is a candidate key, since this combination does not appear in a single relation. The decomposition is lossless, but not dependency-preserving.

6.7 Our solution to Exercise 4.6 was
Customer(<u>custId</u>, firstName, lastName, street, city, state, zip, areaCode, number, creditLimit)
Supplier(<u>supplierNo</u>, supName, countryCode, areaCode, number, street, city, state, zipOrPostal, country, contactName)
Order(<u>orderNo</u>, date, totalAmount, tax, delToStreet, delToCity, delToState, delToZip, delToAreaCode, delToNumber, *custId*)
Item(<u>itemNo</u>, itemName, unitPrice, qtyOnHand, reorderPoint)
Contains(<u>orderNo, itemNo,quantity)</u>
Supplies(*supplierNo*, *itemNo*, unitCost)

In Customer
Custid → all attributes
Zip → city, state

In Supplier
supplierNo → all attributes
zipOrPostal → city, state

In Order
orderNo → all attributes
deltoZip → deltoCity, deltoState

In Item
itemNo → all attributes

In Contains
{orderNo,itemNo} → all attributes

In supplies
{supplierNo, itemNo} → all attributes

The relations are almost BCNF already. The exception involves the zip code, which we assume always determines city and state. To normalize, remove city and state from each relation and place

them in a table along with zip code, which remains in the original table as well. The resulting schema is

Customer(custId, firstName, lastName, street, zip, areaCode, number, creditLimit)
Supplier(supplierNo, supName, countryCode, areaCode, number, street, *zipOrPostal*, country, contactName)
Order(orderNo, date, totalAmount, tax, delToStreet, *delToZip*, delToAreaCode, delToNumber, *custId*)
Item(itemNo, itemName, unitPrice, qtyOnHand, reorderPoint)
Contains(orderNo, itemNo,quantity)
Supplies(*supplierNo*, *itemNo*, unitCost)
Zips(zipOrPostal, city, state)

This is a good opportunity to point out that the ER diagram, along with the standard mapping to a relational schema, usually produces a relational schema that is already fairly well normalized.

6.8 Our solution to Exercise 4.7 was
For the E-R diagram shown as a solution for Exercise 3.3(c), the following is a possible schema
Department(deptNo, deptName, mgrName)
Employee(empId, socSecNo, empName, jobTitle, salary,*deptNo*)
Project(projName, startDate, endDate, budget)
Assign(*empId, projName*, hours)

In Department, we have
deptNo → all attributes
If we assume deptName is also unique, we have
deptName → all attributes
If we assume mgrName not unique, those are the only two FDs in Department. Since both determinants are superkeys, the relation is already BCNF. Note that this would be true even if mgrName were not unique, since it too would be a superkey in that case.

In Employee, we have
Empid → all attributes
socSecNo → all attributes
We assume jobtitle does not determine salary, since there may be two programmers, for example, who do not have the same salary. Since both determinants are superkeys, the relation is already BCNF.

In Project
projName → all attributes
There are no other FDs. We assume endDate is later than startDate, but that is not an FD, since two projects might have the same start date but different endDate values. Since the primary key is the only determint,this relation is already BCNF.

In Assign
{empId, projName} → all attributes

There are no other FDs. Again, the relation is already BCNF.

6.9 Our solution to Exercise 4.8 was
For the E-R diagram shown as a solution for Exercise 3.4, the following is a possible schema
Advisor (advisorId, name, phone)
Student(stuId, lastName, firstName, major, *advisorId*)
Moderator(modId, name, position)
Club(clubName, budget, meetingDay, *modId*)
Activity(activityId, type, date, time, location, *clubName*)
Belongs(*stuId, clubName*, dateJoined)

If we assume names are not unique, in each relation the primary key is the only determinant. Therefore each relation is already BCNF. If names are unique, the name is a superkey in Advisor and Moderator, so each relation is still BCNF.

6.10 Our solution to Exercise 4.9 was
For the E-R diagram shown as a solution for Exercise 3.5, the following is a possible schema
Patient(patId, firstName, lastName, DOB, SSN, street, city, state, zip)
Visit(visitNo, date, time, length, *patId*)
Laboratory(labName, address, phone)
Supplies(itemNo, name, qtyOnHand, cost, *labName*)
WorkNeeded(*visitNoDiagnosed*, type, *estVisits*)
ProcedureOrService(*visitNo*, code, price, outcome)
Charge(*visitNo*, date, item, amount)
Payment(*visitNo*, date, amount)
UsedFor(*visitNo*, code, itemNo, qtyUsed)

In Patient, we have
Patid → all attributes
SSN → all attributes
Zip → city, state
We change Patient to
Patient1(patId, firstName, lastName, DOB, SSN, street, zip)
Zips(zip, city, state)
Both of these are BCNF.

Visit is already BCNF, since visitNo, the primary key, is the only determinant.

In Laboratory, we have
labName → all attributes
phone → all attributes
The address may or may not be unique, depending on our assumptions. If unique, all three determinants are superkeys, so the relation is BCNF. If not unique, the two determinants are superkeys, so we still have BCNF.

In Supplies, we have

itemNo → all attributes
If item names are unique, we also have
name → all attributes
Both of these are superkeys, so the relation is already BCNF. If item names are not unique, the second FD is not true, so name is not a determinant, and the relation is BCNF.

In WordNeeded, we have
visitNoDiagnosed, type → all attributes
If we assume the same number of visits are always required for a particular problem (e.g. a tooth cap always requires two visits)
Type → estVisits
This is a partial FD, so we must use a projection, giving us
WorkNeeded1(visitNoDiagnosed, type)
VisitsNeeded(type, estVisits)

For ProcedureOrService, we have similar partial FD, if we assume
Code → price
We use projection to get
ProcedureOrService1(visitNo, code, outcome)
CodePrices(code, price)

For Charge, we have a similar partial FD, if we assume
Item → amount
Again, we use projection to get
Charge1(visitNo, date, item)
ItemPrices(item, amount)

In Payment and UsedFor, the only FD is the primary key, so both are already BCNF.
The final normalized schema consists of the relations in boldface.

6.11 Our solution to Exercise 4.10 was
For the E-R diagram shown as a solution for Exercise 3.6, the following is a possible schema
Decorator(empId, firstName, lastName, phone, dateHired, mobilePhone, salary)
Client(clientId, firstName, lastName, street, city, state, zip, areacode, phoneNumber, referredBy)
Job(jobNo, description, estTime, estCost, plannedStartDate, plannedEndDate, actualStartDate, actualEndDate, actualTime, actualCost, *clientId, empId*)
Contractor(licNo, firstName, lastName, street, city, state, zip, areacode, phoneNumber, rating, specialty)
Activity(jobNo, activityName, description, estCost, estTime, plannedStartDate, plannedEndDate, actualStartDate, actualEndDate, actualTime, actualCost*, licNo*)
Material(itemNo, manufacturer, description, supplier, quantity, cost, *jobNo, activityName*)

In Decorator, if phone is unique, we have
Empid → all attributes
Phone → all attributes
mobilePhone → all attributes

Since all three determinants are superkeys, the relation is already BCNF.
In Client, we have
clientId → all attributes
If phoneNumber is unique, we have
phoneNumber → all attributes
We also have the transitive dependency
zip → city, state
Since both clientId and phoneNumber are superkeys, we can leave them in the relation. However, we must eliminate the transitive dependency. Using projection, we change the relation to
Client(clientId, firstName, lastName, street, *zip*, areacode, phoneNumber, referredBy)
Zips(zip, city, state)

In Job, the only FD is the one involving the primary key, jobNo. The relation is already BCNF.

In Contractor, we have the transitive dependency
zip → city, state

We remove the city and state from the relation, and use the same Zips table constructed for Client, so we replace the Contractor table by

Contractor(licNo, firstName, lastName, street, zip, areacode, phoneNumber, rating, specialty)

In Activity, depending on our assumptions we may have a partial FD.
activityName → description, estCost, estTime

However, the activity name may not truly be a determinant, if it is a generic name such as "paint a room" or "install flooring". For example, in these cases it would be necessary to know how big the room is or what condition the walls or floor are in before giving the value of the other attributes. Therefore, we will assume the jobNo is required along with the activityName to determine description, estCost and estTime. We therefore assume that is not partial FD here and keep the original relation.

Activity(jobNo, activityName, description, estCost, estTime, plannedStartDate, plannedEndDate, actualStartDate, actualEndDate, actualTime, actualCost*, licNo*)

In Material, the only FD is the one involving the primary key, so it is already BCNF.
The final normalized schema consists of the relations in boldface.

6.12 Our solution for Exercise 4.11 was
For the E-R diagram shown as a solution for Exercise 3.7, the following is a possible schema
Customer(driverLicState,driverLicNumber, firstName, lastName, street, city, state, zip, areaCode, phoneNumber, referredBy)
Car(licPlateState, licPlateNumber, VIN, make, model, year, color, mileage, cylinders, *driverLicState, driverLicNo*)

Mechanic(mechLicNo, firstName, lastName, street, city, state, zip, areacode, phoneNumber, specialty)
Estimate(estNo, dateGiven, timeNeeded, estCost, *licPlateState, licPlateNumber, mechLicNo)*
Job(jobNo, description, dateStarted, totalHoursSpent, totalCostOfParts, dateCompleted, totalCost, *licPlateState, licPlateNo, mechLicNo*)
RepairNeeded(*jobNo*, type, description, timeNeeded)
RepairDone(*jobNo*, type, description, dateStarted, timeSpent, dateCompleted)
Part(partNo, partName, manufacturer, supplier, unitCost)
PartUsed(*jobNo, repairType*, *partNo*, qty)
Labor(*jobNo*, *repairType,* numberHours)

For Customer, we have
driverLicState, driverLicNumber → all attributes
If phoneNumber is unique, we may have
phoneNumber → all attributes
If that is the case, the determinant is a superkey, so we can leave it in the relation.

We also have the transitive dependency
zip → city, state

We therefore use projection to give us the following relations, both in BCNF
Customer(driverLicState,driverLicNumber, firstName, lastName, street, *zip*, areaCode, phoneNumber, referredBy)
Zips (zip, city, state)

In Car, in addition to the FD on the key, we have
VIN → all attributes
Since this is a superkey, we can leave it in the relation, so the relation is already BCNF
Car(licPlateState, licPlateNumber, VIN, make, model, year, color, mileage, cylinders, *driverLicState, driverLicNo*)

In Mechanic, we remove the city and state, leaving the zip. We will use the Zips table from Customer. We assume phoneNumber is a superkey, so we leave it in the relation.
Mechanic(mechLicNo, firstName, lastName, street, *zip*, areacode, phoneNumber, specialty)

Estimate is already BCNF, since the primary key is the only determinant.
Estimate(estNo, dateGiven, timeNeeded, estCost, *licPlateState, licPlateNumber, mechLicNo)*

In JobNo, in addition to the primary key FD, since we assume the total cost of a job is the sum of the cost of parts and the cost of labor, which is a fixed rate per hour, we have

{totalHoursSpent, totalCostOfParts} → totalCost
Job(jobNo, description, dateStarted, totalHoursSpent, totalCostOfParts, dateCompleted, totalCost, *licPlateState, licPlateNo, mechLicNo*)

We change the relation to
Job(jobNo, description, dateStarted, totalHoursSpent, totalCostOfParts, dateCompleted, *licPlateState, licPlateNo, mechLicNo*)
Since the total cost is easily computed, we do not store a table for it.

For Repair Needed, we have the usual FD for the primary key. Depending on our assumptions, we may also have
Type → description, time Needed

If this FD were true, we would take the dependent attributes out and create the two tables
RepairNeeded1(*jobNo*, type)
RepairMenu(type, description, timeNeeded)

However, we will assume the type of repair is generic enough so that different descriptions and time needed might exist for the same repair type, so we will keep the original relation.
RepairNeeded(*jobNo*, type, description, timeNeeded)

For the RepairDone table, we assume that type is generic, so the only FD is the one involving the primary key, leaving us

RepairDone(*jobNo*, type, description, dateStarted, timeSpent, dateCompleted)

For Part, we assume
partNo → all attributes
partName → all attributes
Since both determinants are superkeys, we leave the relation intact.

Part(partNo, partName, manufacturer, supplier, unitCost)

In both PartUsed and Labor, the only determinants are the primary keys, so they are already BCNF.
PartUsed(*jobNo, repairType, partNo*, qty)
Labor(*jobNo, repairType*, numberHours)

The final normalized schema consists of the relations in boldface.

6.13 Our solution for Exercise 4.12 was
For the E-R diagram shown as a solution for Exercise 3.8, the following is a possible schema

Patient(patId, SSN, firstName, lastName, street, city, state, zip, areaCode, phoneNumber, DOB)
Physician(NPI, licNo, firstName, lastName, officeStreet, officeCity,officeState, officeZip, areaCode, phoneNumber, specialty
Prescription(RXNo, procedure, number, frequency, dateWritten, *patId, physicianNPI*)
Appointment(patId, date, time, duration)
Therapist(NPI, licNo,firstName, lastName, homeStreet, homeCity, homeState, homeZip, homeAreaCode, homePhoneNumber, specialty, dateHired)

Visit(visitId, date, time, duration, *RXNo, therapistNPI, _equipmentId*, equiptimeIn, equiptimeOut))
Treatment(*visitId*, type, duration, outcome)
Equipment(equipmentId, type, manufacturer, datePurchased, price)

Because of the FD
Zip → city, state

we remove the city and state from all addresses, and create the table
Zips(zip, city, state)

In Patient, both patId and SSN are superkeys, so both can stay in the relation. The combination of {firstName, lastName, DOB} may also be a superkey, so we can leave it in. We note that we do not assume the telephone number is a determinant, since members of the same family may be patients, and have the same phone number (landline).

Patient1(patId, SSN, firstName, lastName, street, zip, areaCode, phoneNumber, DOB)

For Physician, we can assume the phoneNumber is a determinant for the relation if the phone number includes the extension for individual physicians in any group practices. We have the three superkeys, NPI, licNo, and phoneNumber. If we assume physician names are unique, we have the FD
{firstName,lastName} → all attributes
Since the determinant is a superkey, we can leave it in the relation.
If this FD is not true, there is no issue with transitive dependency, so we can still leave the name in the relation.

Physician(NPI, licNo, firstName, lastName, officeStreet, officeZip, areaCode, phoneNumber, specialty)

Prescription and Appointment have only the primary key as a determinant, so they are already BCNF

Using the same reasoning as we did for Physician, Therapist should be changed to
Therapist(NPI, licNo,firstName, lastName, homeStreet, homeZip, homeAreaCode, homePhoneNumber, specialty, dateHired)

In Visit, if we assume equipmentId refers to a single piece of equipment, we have some FDs. For a particular piece of equipment, assuming only one person can use it at a time, we have

equipmentId, date, time →all attributes
equipmentId, date, timeIn →all attributes
equipmentId, date, timeOut →all attributes

However, this makes the determinant a superkey, so we can leave it in the relation.

Visit(visitId, date, time, duration, *RXNo, therapistNPI, _equipmentId*, equiptimeIn, equiptimeOut)

Both Treatment and Equipment have only the primary key as a determinant, so they are both BCNF.

The final normalized schema consists of the relations in boldface.