Solution Manual for Database Systems Design Implementation and Management 12th Edition by Coronel and Morris ISBN 1305627482 9781305627482

Chapter 2

Data Models

Discussion Focus

Although all of the topics covered in this chapter are important, our students have given us consistent feedback: *If you can write precise business rules from a description of operations, database design is not that difficult*. Therefore, once data modeling (Sections 2-1, "Data Modeling and Data Models", Section 2-2 "The Importance of Data Models," and 2-3, "Data Model Basic Building Blocks,") has been examined in detail, Section 2-4, "Business Rules," should receive a lot of class time and attention. Perhaps it is useful to argue that the answers to questions 2 and 3 in the **Review Questions** section are the key to successful design. That's why we have found it particularly important to focus on business rules and their impact on the database design process.

What are business rules, what is their source, and why are they crucial?

Business rules are precisely written and unambiguous statements that are derived from a detailed description of an organization's operations. ***When written properly***, business rules define one or more of the following modeling components:

entities
relationships
attributes
connectivities
cardinalities – these will be examined in detail in Chapter 3, "The Relational Database Model." Basically, the cardinalities yield the minimum and maximum number of entity occurrences in an entity. For example, the relationship decribed by "a professor teaches one or more classes" means that the PROFESSOR entity is referenced at least once and no more than four times in the CLASS entity.
constraints

Because the business rules form the basis of the data modeling process, their precise statement is crucial to the success of the database design. And, because the business rules are derived from a precise description of operations, much of the design's success depends on the accuracy of the description of operations.

Examples of business rules are:

An invoice contains one or more invoice lines.

Each invoice line is associated with a single
invoice. A store employs many employees.
Each employee is employed by only one store.
A college has many departments.
Each department belongs to a single college. (This business rule reflects a university that has multiple colleges such as Business, Liberal Arts, Education, Engineering, etc.)

A driver may be assigned to drive many different vehicles.

Each vehicle can be driven by many drivers. (Note: Keep in mind that this business rule reflects the assignment of drivers during some period of time.)

A client may sign many contracts.

Each contract is signed by only one client.

A sales representative may write many contracts.

Each contract is written by one sales representative.

Note that each relationship definition requires the definition of two business rules. For example, the relationship between the INVOICE and (invoice) LINE entities is defined by the first two business rules in the bulleted list. This two-way requirement exists because there is always a two-way relationship between any two related entities. (This two-way relationship description also reflects the implementation by many of the available database design tools.)

Keep in mind that the ER diagrams cannot always reflect all of the business rules. For example, examine the following business rule:

> A customer cannot be given a credit line over $10,000 unless that customer has maintained a satisfactory credit history (as determined by the credit manager) during the past two years.

This business rule describes a constraint that cannot be shown in the ER diagram. The business rule reflected in this constraint would be handled at the applications software level through the use of a trigger or a stored procedure. (Your students will learn about triggers and stored procedures in Chapter 8, “Advanced SQL.”)

Given their importance to successful design, we cannot overstate the importance of business rules and their derivation from properly written description of operations. It is not too early to start asking students to write business rules for simple descriptions of operations. Begin by using familiar operational scenarios, such as buying a book at the book store, registering for a class, paying a parking ticket, or renting a DVD.

Also, try reversing the process: Give the students a chance to write the business rules from a basic data model such as the one represented by the text’s Figure 2.1 and 2.2. Ask your students to write the business rules that are the foundation of the relational diagram in Figure 2.2 and then point their attention to the relational tables in Figure 2.1 to indicate that an AGENT occurrence can occur multiple times in the CUSTOMER entity, thus illustrating the implementation impact of the business rules

An agent can serve many customers.

Each customer is served by one agent.

Answers to Review Questions

1. Discuss the importance of data modeling.

A data model is a relatively simple representation, usually graphical, of a more complex real world object event. The data model's main function is to help us understand the complexities of the real-world environment. The database designer uses data models to facilitate the interaction among designers, application programmers, and end users. In short, a good data model is a communications device that helps eliminate (or at least substantially reduce) discrepancies between the database design's components and the real world data environment. The development of data models, bolstered by powerful database design tools, has made it possible to substantially diminish the database design error potential. (Review Section 2.1 in detail.)

2. What is a business rule, and what is its purpose in data modeling?

A business rule is a brief, precise, and unambigous description of a policy, procedure, or principle within a specific organization's environment. In a sense, business rules are misnamed: they apply to *any* organization -- a business, a government unit, a religious group, or a research laboratory; large or small -- that stores and uses data to generate information.

Business rules are derived from a *description of operations*. As its name implies, a description of operations is a detailed narrative that describes the operational environment of an organization. Such a description requires great precision and detail. If the description of operations is incorrect or inomplete, the business rules derived from it will not reflect the real world data environment accurately, thus leading to poorly defined data models, which lead to poor database designs. In turn, poor database designs lead to poor applications, thus setting the stage for poor decision making – which may ultimately lead to the demise of the organization.

Note especially that business rules help to create and enforce actions within that organization's environment. Business rules must be rendered in writing and updated to reflect any change in the organization's operational environment.

Properly written business rules are used to define entities, attributes, relationships, and constraints. Because these components form the basis for a database design, the careful derivation and definition of business rules is crucial to good database design.

3. How do you translate business rules into data model components?

As a general rule, a noun in a business rule will translate into an entity in the model, and a verb (active or passive) associating nouns will translate into a relationship among the entities. For example, the business rule "a customer may generate many invoices" contains two nouns (customer and invoice) and a verb ("generate") that associates them.

4. **Describe the basic features of the relational data model and discuss their importance to the end user and the designer.**

   A relational database is a single data repository that provides both structural and data independence while maintaining conceptual simplicity.

   The relational database model is perceived by the user to be a collection of tables in which data are stored. Each table resembles a matrix composed of row and columns. Tables are related to each other by sharing a common value in one of their columns.

   The relational model represents a breakthrough for users and designers because it lets them operate in a simpler conceptual environment. End users find it easier to visualize their data as a collection of data organized as a matrix. Designers find it easier to deal with *conceptual* data representation, freeing them from the complexities associated with physical data representation.

5. **Explain how the entity relationship (ER) model helped produce a more structured relational database design environment.**

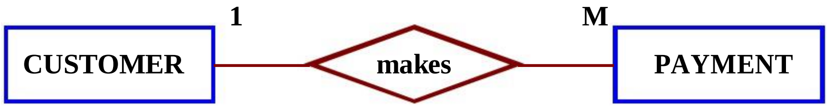
   An entity relationship model, also known as an ERM, helps identify the database's main entities and their relationships. Because the ERM components are graphically represented, their role is more easily understood. Using the ER diagram, it’s easy to map the ERM to the relational database model’s tables and attributes. This mapping process uses a series of well-defined steps to generate all the required database structures. (This structures mapping approach is augmented by a process known as normalization, which is covered in detail in Chapter 6 “Normalization of Database Tables.”)

6. **Consider the scenario described by the statement “A customer can make many payments, but each payment is made by only one customer” as the basis for an entity relationship diagram (ERD) representation.**

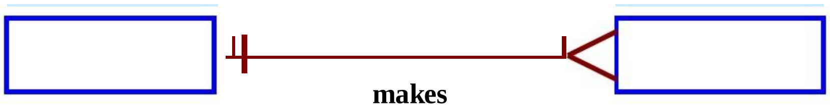
   This scenario yields the ERDs shown in Figure Q2.6. (Note the use of the PowerPoint Crow’s Foot template. We will start using the Visio Professional-generated Crow’s Foot ERDs in Chapter 3, but you can, of course, continue to use the template if you do not have access to Visio Professional.)

Figure Q2.6 The Chen and Crow’s Foot ERDs for Question 6

Chen model



Crow's Foot model



NOTE

Remind your students again that we have not (yet) illustrated the effect of optional relationships on the ERD's presentation. Optional relationships and their treatment are covered in detail in Chapter 4, "Entity Relationship (ER) Modeling."

7. Why is an object said to have greater semantic content than an entity?

An object has greater semantic content because it embodies both data and behavior. That is, the object contains, in addition to data, also the description of the operations that may be performed by the object.

8. What is the difference between an object and a class in the object oriented data model (OODM)?

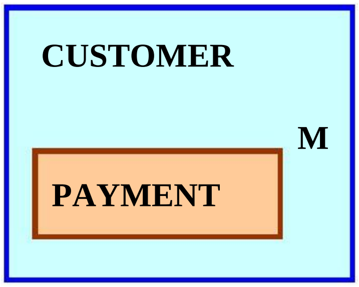
An object is an instance of a specific class. It is useful to point out that the object is a run-time concept, while the class is a more static description.

Objects that share similar characteristics are grouped in classes. A class is a collection of similar objects with shared structure (attributes) and behavior (methods.) Therefore, a class resembles an entity set. However, a class also includes a set of procedures known as methods.

9. How would you model Question 6 with an OODM? (Use Figure 2.4 as your guide.)

The OODM that corresponds to question 6's ERD is shown in Figure Q1.9:

Figure Q2.9 The OODM Model for Question 9



10. What is an ERDM, and what role does it play in the modern (production) database environment?

The Extended Relational Data Model (ERDM) is the relational data model's response to the Object Oriented Data Model (OODM.) Most current RDBMSes support at least a few of the ERDM's extensions. For example, support for large binary objects (BLOBs) is now common.

Although the "ERDM" label has frequently been used in the database literature to describe the relational database model's response to the OODM's challenges, C. J. Date objects to the ERDM label for the following reasons: [1]

> The useful contribution of "the object model" is its ability to let users define their own -- and often very complex -- data types. However, mathematical structures known as "domains" in the relational model also provide this ability. Therefore, a relational DBMS that properly supports such domains greatly diminishes the reason for using the object model. Given proper support for domains, relational database models are quite capable of handling the complex data encountered in time series, engineering design, office automation, financial modeling, and so on. Because the relational model can support complex data types, the notion of an "extended relational database model" or ERDM is "extremely inappropriate and inaccurate" and "it should be firmly resisted." (The capability that is supposedly being extended is already there!)
>
> Even the label **object/relational model** (**O/RDM**) is not quite accurate, because the relational database model's domain is not an object model structure. However, there are already quite a few O/R products -- also known as **Universal Database Servers** -- on the market. Therefore, Date concedes that we are probably stuck with the O/R label. In fact, Date believes that "an O/R system is in everyone's future." More precisely, Date argues that a true O/R system would be "nothing more nor less than a true relational system -- which is to say, a system that supports the relational model, with all that such support entails."

[1] C. J. Date, "Back To the Relational Future", http://www.dbpd.com/vault/9808date.html

C. J. Date concludes his discussion by observing that "We need do nothing to the relational model achieve object functionality. (Nothing, that is, except implement it, something that doesn't yet seem to have been tried in the commercial world.)"

11. What is a relationship, and what three types of relationships exist?

A relationship is an association among (two or more) entities. Three types of relationships exist: one-to-one (1:1), one-to-many (1:M), and many-to-many (M:N or M:M.)

12. Give an example of each of the three types of relationships.

1:1
An academic department is chaired by one professor; a professor may chair only one academic department.

1:M
A customer may generate many invoices; each invoice is generated by one customer.

M:N
An employee may have earned many degrees; a degree may have been earned by many employees.

13. What is a table, and what role does it play in the relational model?

Strictly speaking, the relational data model bases data storage on *relations*. These relations are based on algebraic set theory. However, the user perceives the relations to be tables. In the relational database environment, designers and users *perceive* a table to be a matrix consisting of a series of row/column intersections.Tables, also called relations, are related to each other by sharing a common entity characteristic. For example, an INVOICE table would contain a customer number that points to that same number in the CUSTOMER table. This feature enables the RDBMS to link invoices to the customers who generated them.

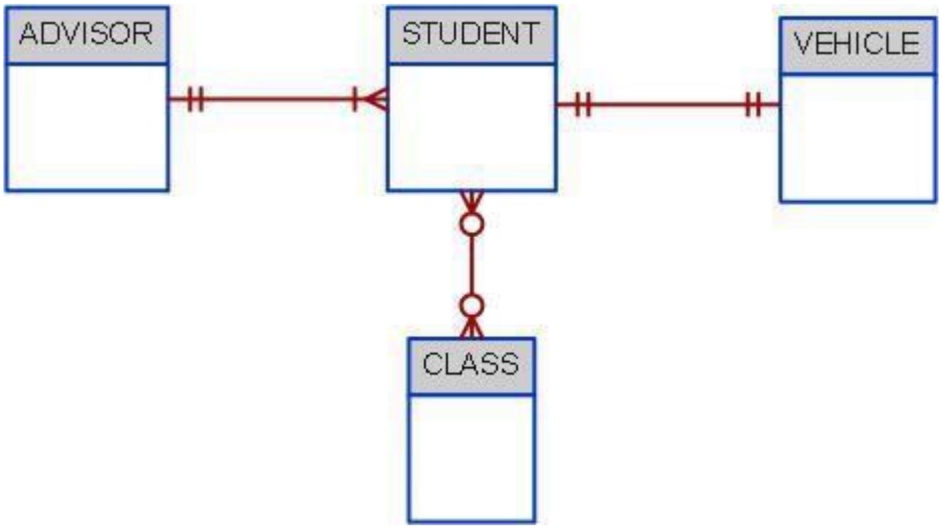
Tables are especially useful from the modeling and implementation perspecectives. Because tables are used to describe the entities they represent, they provide ane asy way to summarize entity characteristics and relationships among entities. And, because they are purely conceptual constructs, the designer does not need to be concerned about the physical implementation aspects of the database design.

14. What is a relational diagram? Give an example.

A relational diagram is a visual representation of the relational database's entities, the attributes within those entities, and the relationships between those entities. Therefore, it is easy to see what the entities represent and to see what types of relationships (1:1, 1:M, M:N) exist among the entities and how those relationships are implemented. An example of a relational diagram is found in the text's Figure 2.2.

15. What is connectivity? (Use a Crow's Foot ERD to illustrate connectivity.)

Connectivity is the relational term to describe the types of relationships (1:1, 1:M, M:N).



In the figure, the businesss rule that an advisor can advise many students and a student has only one assigned advisor is shown with in a relationship with a connectivity of 1:M. The business rule that a student can register only one vehicle to park on campus and a vehicle can be registered by only one student is shown with a relationship with a connectivity of 1:1. Finally, the rule that a student can register for many classes, and a class can be registered for by many students, is shown by the relationship with a connectivity of M:N.

16. Describe the Big Data phenomenon.

Over the last few years, a new wave of data has "emerged" to the limelight. Such data have alsways exsisted but did not recive the attention that is receiving today. These data are characterized for being high volume (petabyte size and beyond), high frequency (data are generated almost constantly), and mostly semi-structured. These data come from multiple and vatied sources such as web site logs, web site posts in social sites, and machine generated information (GPS, sensors, etc.) Such data; have been accumulated over the years and companies are now awakining to the fact that it contains a lot of hidden information that could help the day-to-day business (such as browsing patterns, purchasing preferences, behaivor patterns, etc.) The need to manage and leverage this data has triggered a phenomenon labeled "Big Data". **Big Data** refers to a movement to find new and better ways to manage large amounts of web-generated data and derive business insight from it, while, at the same time, providing high performance and scalability at a reasonable cost.

17. What does the term “3 vs” refers to?

The term “3 Vs” refers to the 3 basic characteristics of Big Data databases, they are:

Volume: Refers to the amounts of data being stored. With the adoption and growth of the Internet and social media, companies have multiplied the ways to reach customers. Over the years, and with the benefit of technological advances, data for millions of e-transactions were being stored daily on company databases. Furthermore, organizations are using multiple technologies to interact with end users and those technologies are generating mountains of data. This ever-growing volume of data quickly reached petabytes in size and it's still growing.

Velocity: Refers not only to the speed with which data grows but also to the need to process these data quickly in order to generate information and insight. With the advent of the Internet and social media, business responses times have shrunk considerably. Organizations need not only to store large volumes of quickly accumulating data, but also need to process such data quickly. The velocity of data growth is also due to the increase in the number of different data streams from which data is being piped to the organization (via the web, e-commerce, Tweets, Facebook posts, emails, sensors, GPS, and so on).

Variety: Refers to the fact that the data being collected comes in multiple different data formats. A great portion of these data comes in formats not suitable to be handled by the typical operational databases based on the relational model.

The 3 Vs framework illustrates what companies now know, that the amount of data being collected in their databases has been growing exponentially in size and complexity. Traditional relational databases are good at managing structured data but are not well suited to managing and processing the amounts and types of data being collected in today's business environment.

18. What is Haddop and what are its basic components?

In order to create value from their previously unused Big Data stores, companies are using new Big Data technologies. These emerging technologies allow organizations to process massive data stores of multiple formats in cost-effective ways. Some of the most frequently used Big Data technologies are Hadoop and MapReduce.

Hadoop is a Java based, open source, high speed, fault-tolerant distributed storage and computational framework. Hadoop uses low-cost hardware to create clusters of thousands of computer nodes to store and process data. Hadoop originated from Google's work on distributed file systems and parallel processing and is currently supported by the Apache Software Foundation.[2]Hadoop has several modules, but the two main components are Hadoop Distributed File System (HDFS) and MapReduce.

Hadoop Distributed File System (HDFS) is a highly distributed, fault-tolerant file storage system designed to manage large amounts of data at high speeds. In order to achieve high throughput, HDFS uses the write-once, read many model. This means that once the data is written, it cannot be modified. HDFS uses three types of nodes: a name node that stores all the metadata about the file system; a data node that stores fixed-size data blocks (that could be replicated to other data nodes) and a client node that acts as the interface between the user application and the HDFS.

[2] For more information about Hadoop visit hadoop.apache.org.

MapReduce is an open source application programming interface (API) that provides fast data analytics services. MapReduce distributes the processing of the data among thousands of nodes in parallel. MapReduce works with structured and nonstructured data. The MapReduce framework provides two main functions, Map and Reduce. In general terms, the Map function takes a job and divides it into smaller units of work; the Reduce function collects all the output results generated from the nodes and integrates them into a single result set.

19. What is sparse data? Give an example.

Sparse data refers to cases in which the number of attributes are very large, but the numbers but the actual number of distinct value instances is relatively small. For example, if you are modeling census data, you will have an entitty called person. This entity person can have hundred of attributes, some of those attributes would be first name, last name, degree, employer, income, veteran status, foreign born, etc. Although, there would be many millions of rows of data for each person, there will be many attributes that will be left blank, for example, not all persons will have a degree, an income or an employer. Even fewer persons will be veterans or foreign born. Every time that you have an data entity that has many columns but the data instances for the columns are very low (many empty attribute occurrences) it is said that you have sparse data.

There is another related terminoligy, data sparcity that refers to the number of different values a fiven columns could have. In this case, a column such as “gender” although it will have values for all rows, it has a low data sparcity because the number of different values is ony two: male or female. A column such as name and birthdate will have high data sparcity because the number of different values is high.

20. Define and describe the basic characteristics of a NoSQL database.

Every time you search for a product on Amazon, send messages to friends in Facebook, watch a video in YouTube or search for directions in Google Maps, you are using a NoSQL database. **NoSQL** refers to a new generation of databases that address the very specific challenges of the “big data” era and have the following general characteristics:

Not based on the relational model.

These databases are generally based on a variation of the key-value data model rather than in the relational model, hence the NoSQL name. The ***key-value*** data model is based on a structure composed of two data elements: a key and a value; in which for every key there is a corresponding value (or a set of values). The key-value data model is also referred to as the attribute-value or associative data model. In the key-value data model, each row represents one attribute of one entity instance. The “key” column points to an attribute and the “value” column contains the actual value for the attribute. The data type of the “value” column is generally a long string to accommodate the variety of actual data types of the values that are placed in the column.

Support distributed database architectures.

One of the big advantages of NoSQL databases is that they generally use a distributed architecture. In fact, several of them (Cassandra, Big Table) are designed to use low cost commodity servers to form a complex network of distributed database nodes

Provide high scalability, high availability and fault tolerance.

NoSQL databases are designed to support the ability to add capacity (add database nodes to the distributed database) when the demand is high and to do it transparently and without downtime. Fault tolerant means that if one of the nodes in the distributed database fails, the database will keep operating as normal.

Support very large amounts of sparse data.

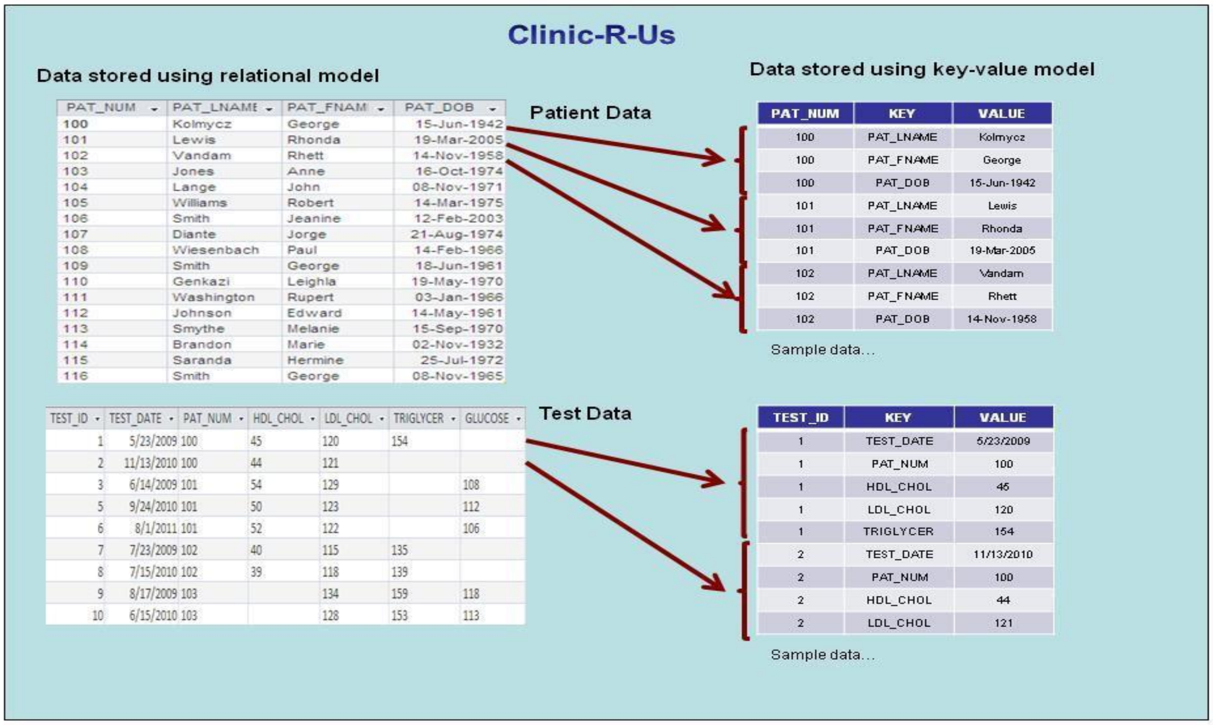
Because NoSQL databases use the key-value data model, they are suited to handle very high volumes of sparse data; that is for cases where the number of attributes is very large but the number of actual data instances is low.

Geared toward performance rather than transaction consistency.

One of the biggest problems of very large distributed databases is to enforce data consistency. Distributed databases automatically make copies of data elements at multiple nodes – to ensure high availability and fault tolerance. If the node with the requested data goes down, the request can be served from any other node with a copy of the data. However, what happen if the network goes down during a data update? In a relational database, transaction updates are guaranteed to be consistent or the transaction is rolled back. NoSQL databases sacrifice consistency in order to attain high levels of performance. NoSQL databases provide eventual consistency. **Eventual consistency** is a feature of NoSQL databases that indicates that data are not guaranteed to be consistent immediately after an update (across all copies of the data) but rather, that updates will propagate through the system and eventually all data copies will be consistent.

21. Using the example of a medical clinic with patients and tests, provide a simple representation of how to model this example using the relational model and how it wold be represented using the key-value data modeling technique.

As you can see in Figure Q2.21, the relational model stores data in a tabular format in which each row represents a "record" for a given patient. While, the key-value data model uses three differnet fields to represent each data element in the record. Therefore, for each patient row, there are three rows in the key-value model.



Clinic-R-Us

Data stored using relational model

PAT_NUM	PAT_LNAME	PAT_FNAM	PAT_DOB
100	Kolmycz	George	15-Jun-1942
101	Lewis	Rhonda	19-Mar-2005
102	Vandam	Rhett	14-Nov-1958
103	Jones	Anne	16-Oct-1974
104	Lange	John	08-Nov-1971
105	Williams	Robert	14-Mar-1975
106	Smith	Jeanine	12-Feb-2003
107	Diante	Jorge	21-Aug-1974
108	Wiesenbach	Paul	14-Feb-1966
109	Smith	George	18-Jun-1961
110	Genkazi	Leighla	19-May-1970
111	Washington	Rupert	03-Jan-1966
112	Johnson	Edward	14-May-1961
113	Smythe	Melanie	15-Sep-1970
114	Brandon	Marie	02-Nov-1932
115	Saranda	Hermine	25-Jul-1972
116	Smith	George	08-Nov-1965

TEST_ID	TEST_DATE	PAT_NUM	HDL_CHOL	LDL_CHOL	TRIGLYCER	GLUCOSE
1	5/23/2009	100	45	120	154	
2	11/13/2010	100	44	121		
3	6/14/2009	101	54	129		108
5	9/24/2010	101	50	123		112
6	8/1/2011	101	52	122		106
7	7/23/2009	102	40	115	135	
8	7/15/2010	102	39	118	139	
9	8/17/2009	103		134	159	118
10	6/15/2010	103		128	153	113

Patient Data

Test Data

Data stored using key-value model

PAT_NUM	KEY	VALUE
100	PAT_LNAME	Kolmycz
100	PAT_FNAME	George
100	PAT_DOB	15-Jun-1942
101	PAT_LNAME	Lewis
101	PAT_FNAME	Rhonda
101	PAT_DOB	19-Mar-2005
102	PAT_LNAME	Vandam
102	PAT_FNAME	Rhett
102	PAT_DOB	14-Nov-1958

Sample data...

TEST_ID	KEY	VALUE
1	TEST_DATE	5/23/2009
1	PAT_NUM	100
1	HDL_CHOL	45
1	LDL_CHOL	120
1	TRIGLYCER	154
2	TEST_DATE	11/13/2010
2	PAT_NUM	100
2	HDL_CHOL	44
2	LDL_CHOL	121

Sample data...

22. What is logical independence?

Logical independence exists when you can change the internal model without affecting the conceptual model.

When you discuss logical and other types of independence, it’s worthwhile to discuss and review some basic modeling concepts and terminology:

In general terms, a *model* is an abstraction of a more complex real-world object or event. A model’s main function is to help you understand the complexities of the real-world environment. Within the database environment, a data model represents data structures and their characteristics, relations, constraints, and transformations. As its name implies, a purely *conceptual* model stands at the highest level of abstraction and focuses on the basic ideas (concepts) that are explored in the model, without specifying the details that will enable the designer to *implement* the model. For example, a conceptual model would include entities and their relationships and it may even include at least some of the attributes that define the entities, but it would not include attribute details such as the nature of the attributes (text, numeric, etc.) or the physical storage requirements of those atttributes.

The terms *data model* and *database model* are often used interchangeably. In the text, the term *database model* is be used to refer to the implementation of a *data model* in a specific database system.

Data models (relatively simple representations, usually graphical, of more complex real-world data structures), bolstered by powerful database design tools, have made it possible to substantially diminish the potential for errors in database design.

The **internal model** is the representation of the database as “seen” by the DBMS. In other words, the internal model requires the designer to match the conceptual model’s characteristics and constraints to those of the selected implementation model.

An **internal schema** depicts a specific representation of an internal model, using the database constructs supported by the chosen database.

The **external model** is the end users’ view of the data environment.

23. What is physical independence?

You have **physical independence** when you can change the *physical model* without affecting the *internal model*. Therefore, a change in storage devices or methods and even a change in operating system will not affect the internal model.

The terms physical model and internal model may require a bit of additional discussion:

The **physical model** operates at the lowest level of abstraction, describing the way data are saved on storage media such as disks or tapes. The physical model requires the definition of both the physical storage devices and the (physical) access methods required to reach the data within those storage devices, making it both software- and hardware-dependent. The storage structures used are dependent on the software (DBMS, operating system) and on the type of storage devices that the computer can handle. The precision required in the physical model’s definition demands that database designers who work at this level have a detailed knowledge of the hardware and software used to implement the database design.

The **internal model** is the representation of the database as “seen” by the DBMS. In other words, the internal model requires the designer to match the conceptual model’s characteristics and constraints to those of the selected implementation model. An **internal schema** depicts a specific representation of an internal model, using the database constructs supported by the chosen database.

Problem Solutions

Use the contents of Figure 2.1 to work problems 1-3.

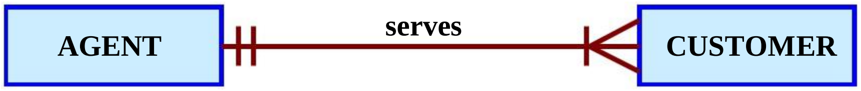
1. Write the business rule(s) that governs the relationship between AGENT and CUSTOMER.

Given the data in the two tables, you can see that an AGENT – through AGENT_CODE -- can occur many times in the CUSTOMER table. But each customer has only one agent. Therefore, the business rules may be written as follows:
One agent can have many customers.
Each customer has only one agent.
Given these business rules, you can conclude that there is a 1:M relationship between AGENT and CUSTOMER.

2. Given the business rule(s) you wrote in Problem 1, create the basic Crow's Foot ERD.

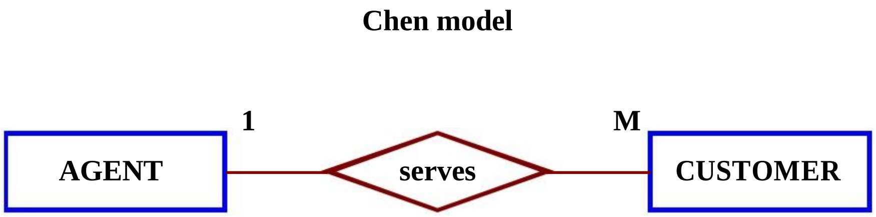
The Crow's Foot ERD is shown in Figure P2.2a.

Figure P2.2a The Crow's Foot ERD for Problem 3



For discussion purposes, you might use the Chen model shown in Figure P2.2b. Compare the two representations of the business rules by noting the different ways in which connectivities (1,M) are represented. The Chen ERD is shown in Figure P2.2b.

Figure P2.2b The Chen ERD for Problem 2



3. Using the ERD you drew in Problem 2, create the equivalent Object representation and UML class diagram. (Use Figure 2.4 as your guide.)

The OO model is shown in Figure P2.3.

Figure P2.3a The OO Model for Problem 3

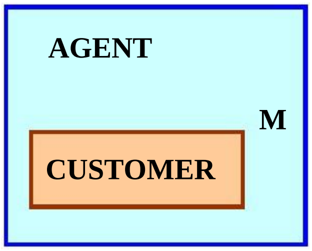
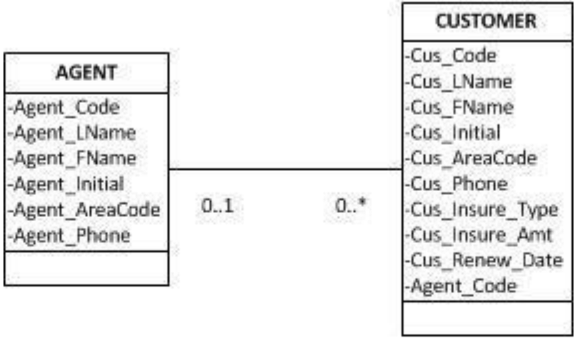


Figure P.3b The UML Model for Problem 3



Using Figure P2.4 as your guide, work Problems 4–5. The DealCo relational diagram shows the initial entities and attributes for the DealCo stores, located in two regions of the country.

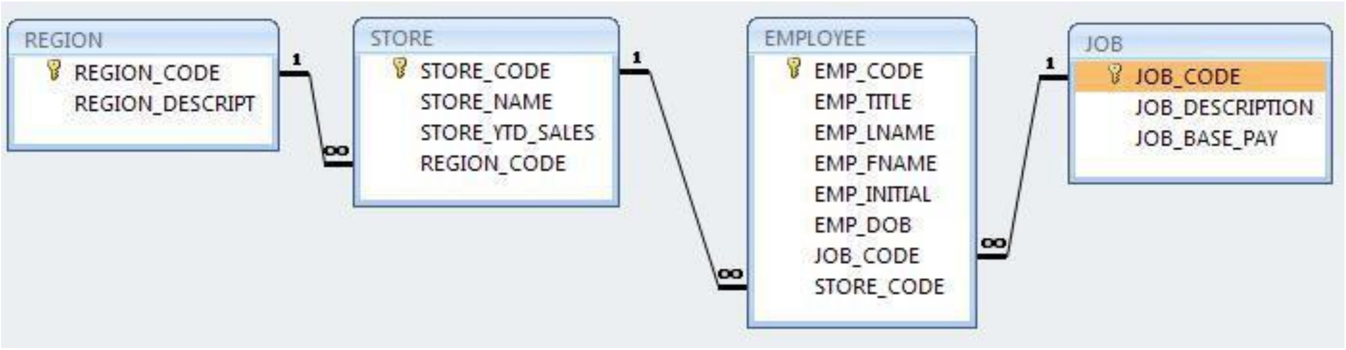


Figure P2.4 The DealCo relational diagram

4. Identify each relationship type and write all of the business rules.

One region can be the location for many stores. Each store is located in only one region. Therefore, the relationship between REGION and STORE is 1:M.

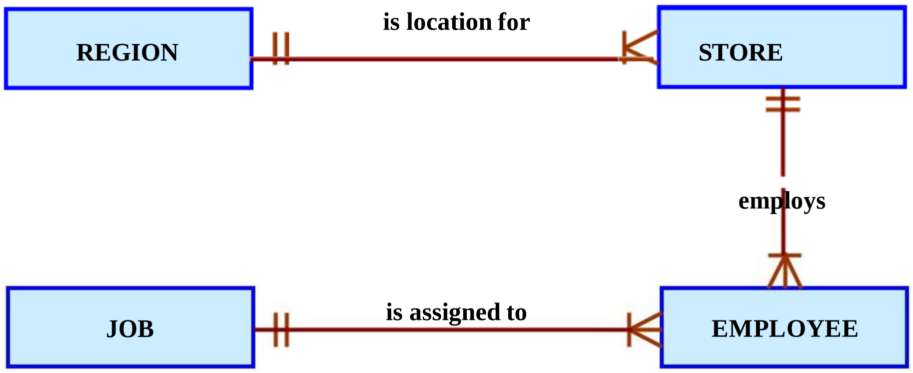
Each store employs one or more employees. Each employee is employed by one store. (In this case, we are assuming that the business rule specifies that an employee cannot work in more than one store at a time.) Therefore, the relationship between STORE and EMPLOYEE is 1:M.

A job – such as accountant or sales representative -- can be assigned to many employees. (For example, one would reasonably assume that a store can have more than one sales representative. Therefore, the job title "Sales Representative" can be assigned to more than one employee at a time.) Each employee can have only one job assignment. (In this case, we are assuming that the business rule specifies that an employee cannot have more than one job assignment at a time.) Therefore, the relationship between JOB and EMPLOYEE is 1:M.

5. Create the basic Crow's Foot ERD for DealCo.

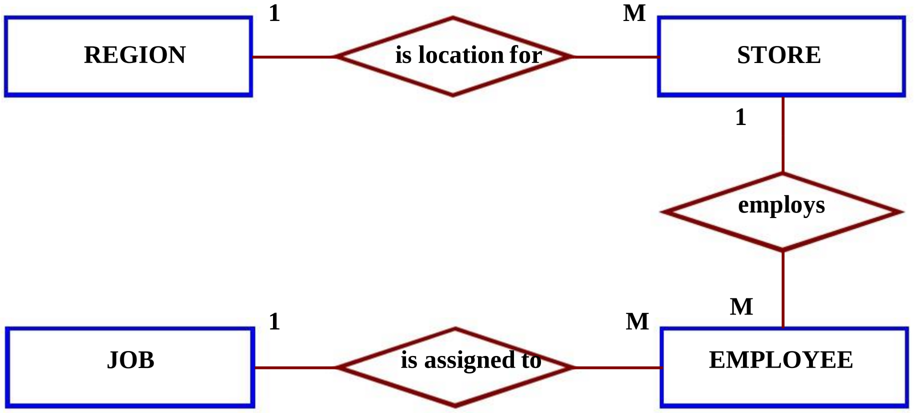
The Crow's Foot ERD is shown in Figure P2.5a.

Figure P2.5a The Crow's Foot ERD for DealCo



The Chen model is shown in Figure P2.5b. (Note that you always read the relationship from the "1" to the "M" side.)

Figure P2.5b The Chen ERD for DealCo



Using Figure P2.6 as your guide, work Problems 6−8 The Tiny College relational diagram shows the initial entities and attributes for Tiny College.

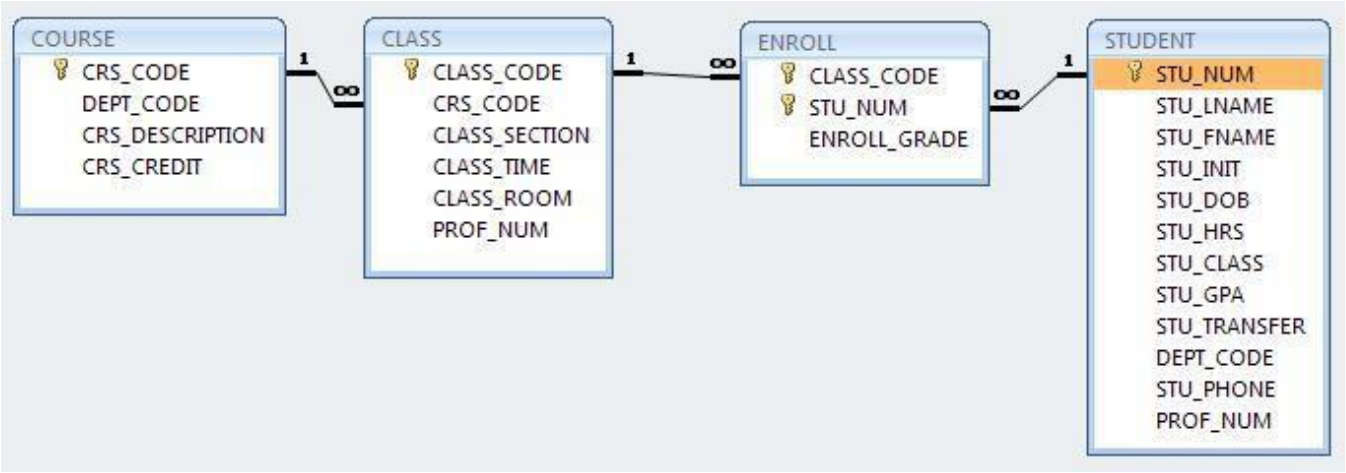


Figure P2.6 The Tiny College relational diagram

6. Identify each relationship type and write all of the business rules.

The simplest way to illustrate the relationship between ENROLL, CLASS, and STUDENT is to discuss the data shown in Table P2.6. As you examine the Table P2.6 contents and compare the attributes to relational schema shown in Figure P2.6, note these features:

We have added an attribute, ENROLL_SEMESTER, to identify the enrollment period.

Naturally, no grade has yet been assigned when the student is first enrolled, so we have entered a default value "NA" for "Not Applicable." The letter grade – A, B, C, D, F, I (Incomplete), or W (Withdrawal) -- will be entered at the conclusion of the enrollment period, the SPRING-12 semester.

Student 11324 is enrolled in two classes; student 11892 is enrolled in three classes, and student 10345 is enrolled in one class.

Table P2.6 Sample Contents of an ENROLL Table

STU_NUM	CLASS_CODE	ENROLL_SEMESTER	ENROLL_GRADE
11324	MATH345-04	SPRING-14	NA
11324	ENG322-11	SPRING-14	NA
11892	CHEM218-05	SPRING-14	NA
11892	ENG322-11	SPRING-14	NA
11892	CIS431-01	SPRING-14	NA
10345	ENG322-07	SPRING-14	NA

All of the relationships are 1:M. The relationships may be written as follows:

COURSE generates CLASS. One course can generate many classes. Each class is generated by one course.

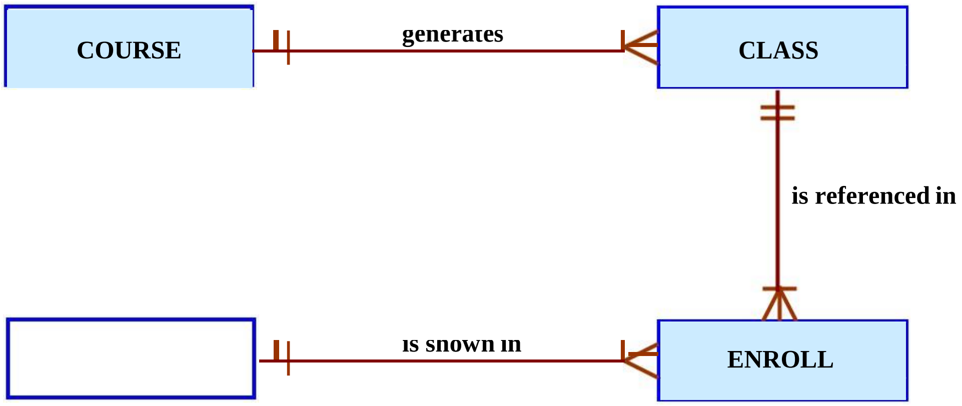
CLASS is referenced in ENROLL. One class can be referenced in enrollment many times. Each individual enrollment references one class. Note that the ENROLL entity is also related to STUDENT. Each entry in the ENROLL entity references one student and the class for which that student has enrolled. A student cannot enroll in the same class more than once. If a student enrolls in four classes, that student will appear in the ENROLL entity four times, each time for a different class.

STUDENT is shown in ENROLL. One student can be shown in enrollment many times. (In database design terms, “many” simply means “*more than once*.”) Each individual enrollment entry shows one student.

7. Create the basic Crow’s Foot ERD for Tiny College.

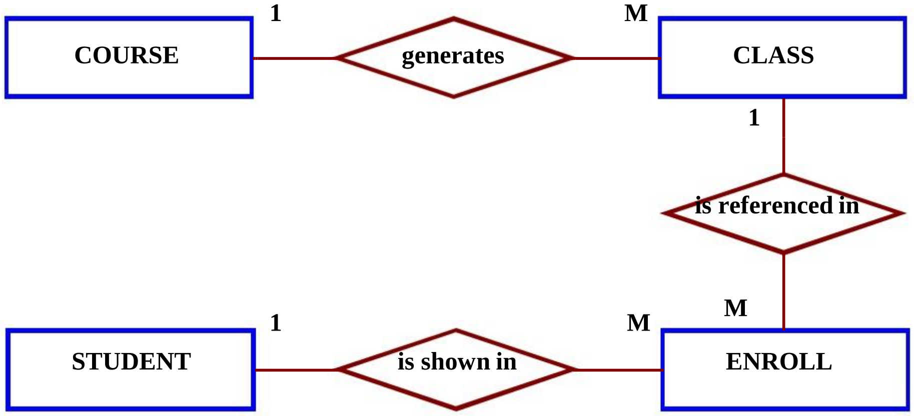
The Crow’s Foot model is shown in Figure P2.7a.

Figure P2.7a The Crow’s Foot Model for Tiny College



The Chen model is shown in Figure P2.7b.

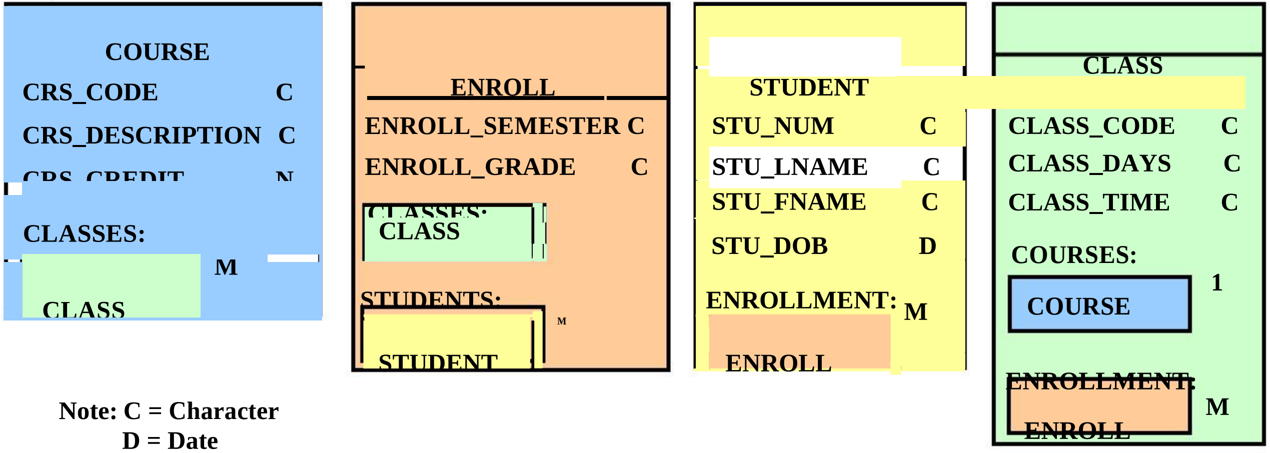
Figure P2.7b The Chen Model for Tiny College



8. **Create the UML class diagram that reflects the entities and relationships you identified in the relational diagram.**

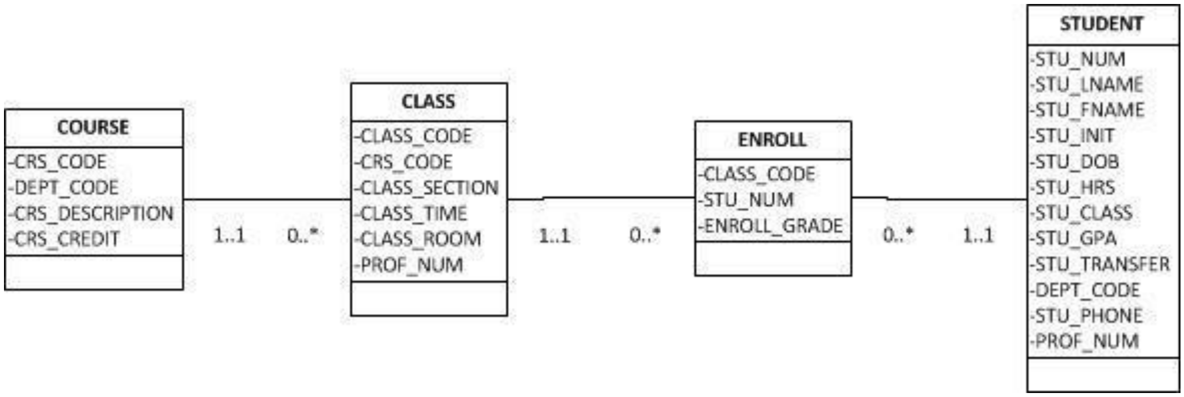
The OO model is shown in Figure P2.8.

Figure P2.8a The OO Model for Tiny College



Note: C = Character
D = Date
N = Numeric

Figure P2.8b The UML Model for Tiny College



9. Typically, a patient staying in a hospital receives medications that have been ordered by a particular doctor. Because the patient often receives several medications per day, there is a 1:M relationship between PATIENT and ORDER. Similarly, each order can include several medications, creating a 1:M relationship between ORDER and MEDICATION.

a. Identify the business rules for PATIENT, ORDER, and MEDICATION.

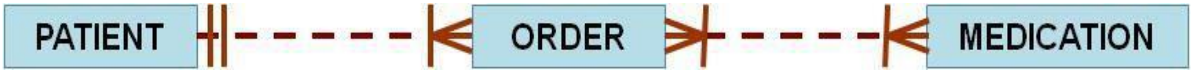
The business rules reflected in thePATIENT description are:
A patient can have many (medical) orders written for him or her. Each (medical) order is written for a single patient.

The business rules refected in the ORDER description are:
Each (medical) order can prescribe many medications.
Each medication can be prescribed in many orders.

The business rules refected in the MEDICATION description are: Each medication can be prescribed in many orders.
Each (medical) order can prescribe many medications.

b. Create a Crow's Foot ERD that depicts a relational database model to capture these business rules.

Figure P2.9 Crow's foot ERD for Problem 9



10. United Broke Artists (UBA) is a broker for not-so-famous painters. UBA maintains a small network database to track painters, paintings, and galleries. A painting is painted by a particular artist, and that painting is exhibited in a particular gallery. A gallery can exhibit many paintings, but each painting can be exhibited in only one gallery. Similarly, a painting is painted by a single painter, but each painter can paint many paintings. Using PAINTER, PAINTING, and GALLERY, in terms of a relational database:

a. What tables would you create, and what would the table components be?

We would create the three tables shown in Figure P2.10a. (Use the teacher’s **Ch02_UBA** database in your instructor's resources to illustrate the table contents.)

FIGURE P2.10a The UBA Database Tables

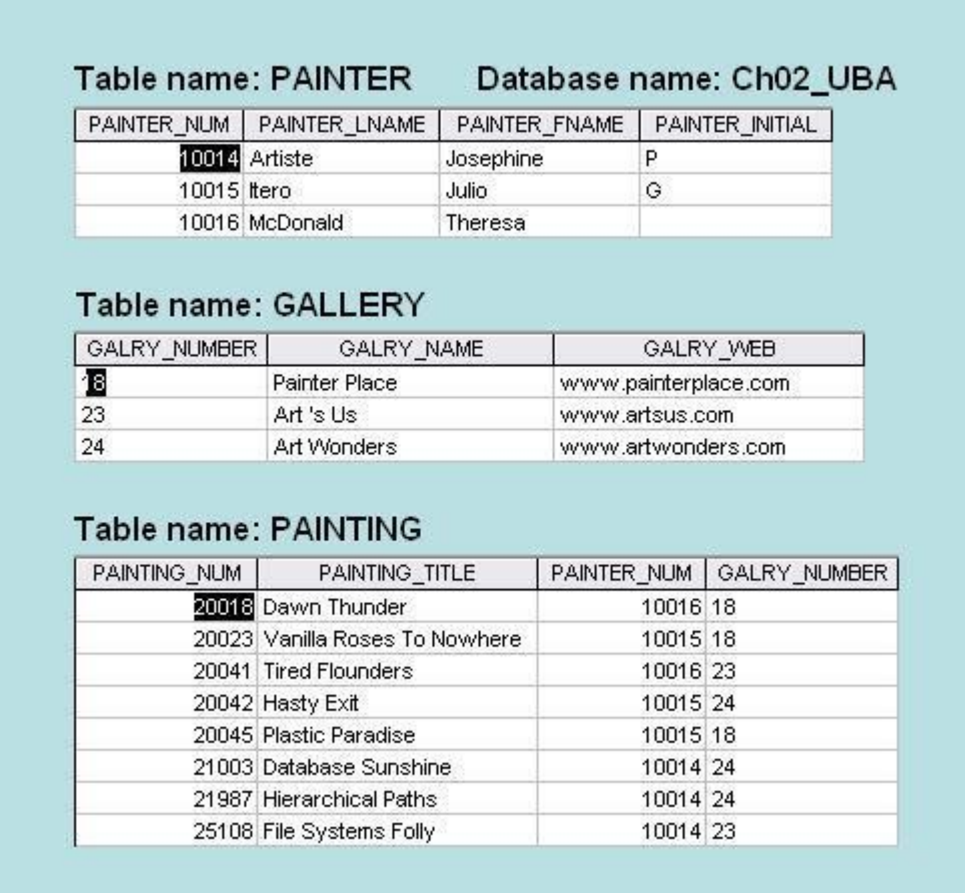


Table name: PAINTER   Database name: Ch02_UBA

PAINTER_NUM	PAINTER_LNAME	PAINTER_FNAME	PAINTER_INITIAL
10014	Artiste	Josephine	P
10015	Itero	Julio	G
10016	McDonald	Theresa	

Table name: GALLERY

GALRY_NUMBER	GALRY_NAME	GALRY_WEB
18	Painter Place	www.painterplace.com
23	Art 's Us	www.artsus.com
24	Art Wonders	www.artwonders.com

Table name: PAINTING

PAINTING_NUM	PAINTING_TITLE	PAINTER_NUM	GALRY_NUMBER
20018	Dawn Thunder	10016	18
20023	Vanilla Roses To Nowhere	10015	18
20041	Tired Flounders	10016	23
20042	Hasty Exit	10015	24
20045	Plastic Paradise	10015	18
21003	Database Sunshine	10014	24
21987	Hierarchical Paths	10014	24
25108	File Systems Folly	10014	23

As you discuss the UBA database contents, note in particular the following business rules that are reflected in the tables and their contents:

A painter can paint may paintings.

Each painting is painted by only one painter.

A gallery can exhibit many paintings.

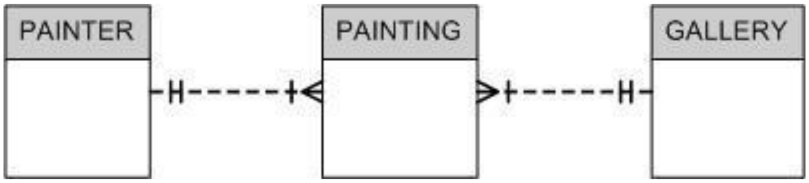
A painter can exhibit paintings at more than one gallery at a time. (For example, if a painter has painted six paintings, two may be exhibited in one gallery, one at another, and three at the third gallery. Naturally, if galleries specify exclusive contracts, the database must be changed to reflect that business rule.)

Each painting is exhibited in only one gallery.

The last business rule reflects the fact that a painting can be physically located in only one gallery at a time. If the painter decides to move a painting to a different gallery, the database must be updated to remove the painting from one gallery and add it to the different gallery.

b. **How might the (independent) tables be related to one another?** Figure P2.10b shows the relationships.

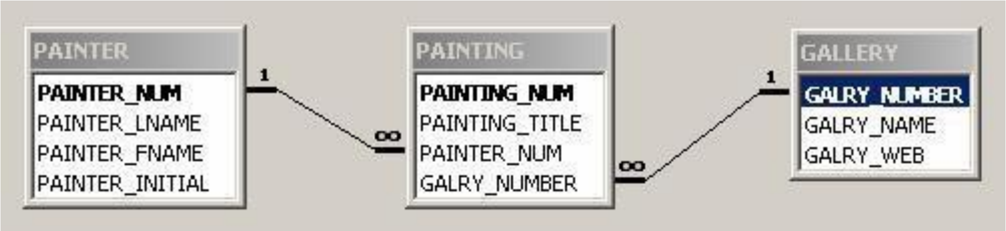
FIGURE P2.10b The UBA Relational Model



11. Using the ERD from Problem 10, create the relational schema. (Create an appropriate collection of attributes for each of the entities. Make sure you use the appropriate naming conventions to name the attributes.)

The relational diagram is shown in Figure P2.11.

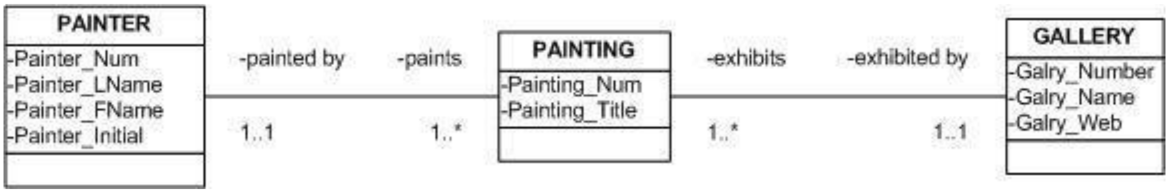
FIGURE P2.11 The Relational Diagram for Problem 11



12. Convert the ERD from Problem 10 into the corresponding UML class diagram.

The basic UML solution is shown in Figure P2.12.

FIGURE P2.12 The UML for Problem 12



13. Describe the relationships (identify the business rules) depicted in the Crow's Foot ERD shown in Figure P2.13.

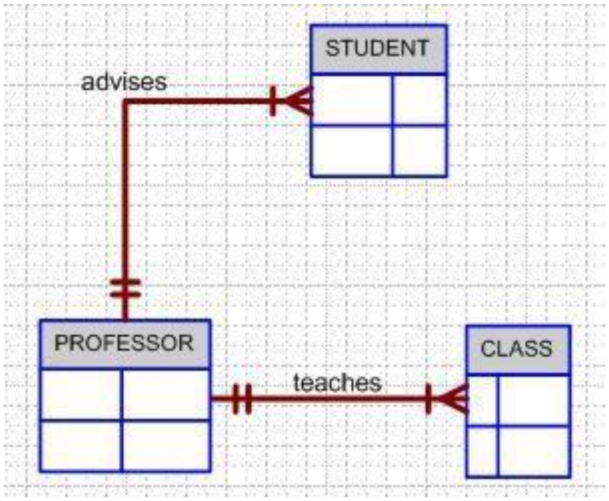


Figure P2.13 The Crow's Foot ERD for Problem 13

The business rules may be written as follows:

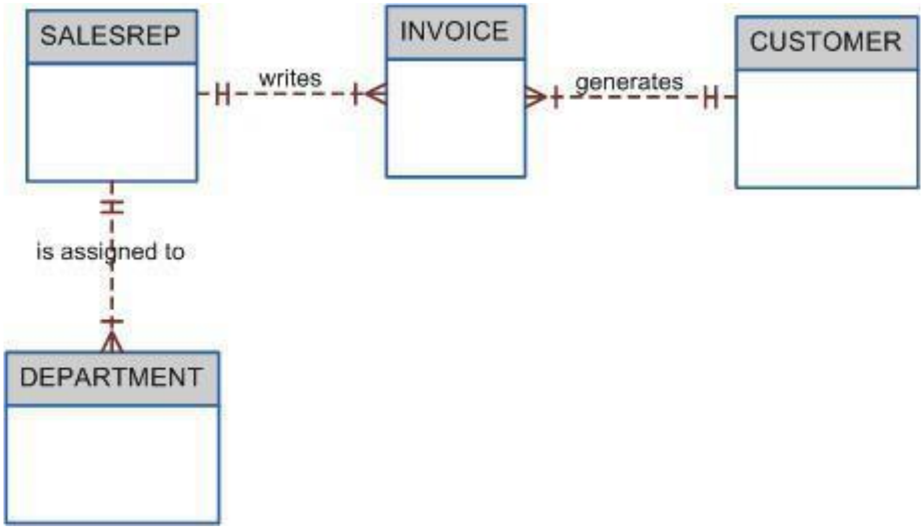
A professor can teach many classes.
Each class is taught by one professor.
A professor can advise many students.
Each student is advised by one professor.

14. Create a Crow's Foot ERD to include the following business rules for the ProdCo company:

- **a. Each sales representative writes many invoices.**
- **b. Each invoice is written by one sales representative.**
- **c. Each sales representative is assigned to one department.**
- **d. Each department has many sales representatives.**
- **e. Each customer can generate many invoices.**
- **f. Each invoice is generated by one customer.**

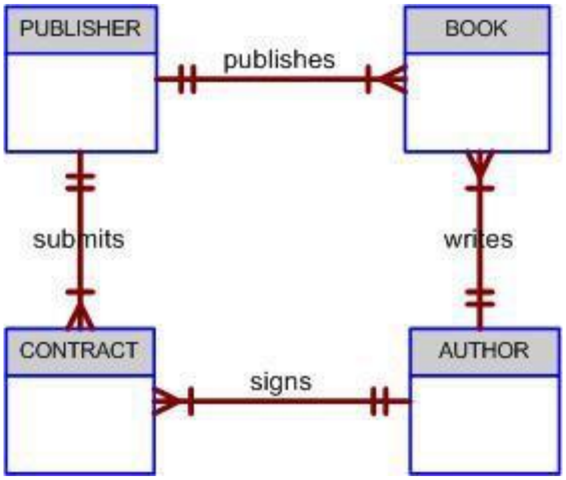
The Crow's Foot ERD is shown in Figure P2.23. Note that a 1:M relationship is always read from the one (1) to the many (M) side. Therefore, the customer-invoice relationship is read as "one customer generates many invoices."

Figure P2.14 Crow's Foot ERD for the ProdCo Company



15. Write the business rules that are reflected in the ERD shown in Figure P2.15. (Note that the ERD reflects some simplifying assumptions. For example, each book is written by only one author. Also, remember that the ERD is always read from the "1" to the "M" side, regardless of the orientation of the ERD components.)

FIGURE P2.15 The Crow's Foot ERD for Problem 15



The relationships are best described through a set of business rules:

One publisher can publish many books.
Each book is published by one publisher.
A publisher can submit many (book) contracts.
Each (book) contract is submitted by one publisher.
One author can sign many contracts.
Each contract is signed by one author.
One author can write many books.
Each book is written by one author.

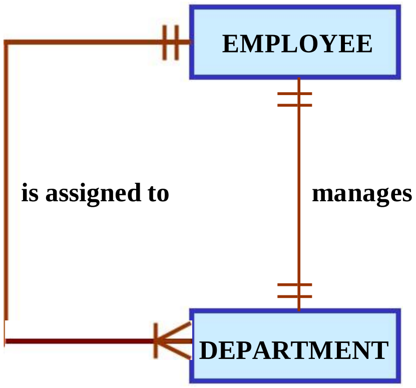
This ERD will be a good basis for a discussion about what happens when more realistic assumptions are made. For example, a book – such as this one – may be written by more than one author. Therefore, a contract may be signed by more than one author. Your students will learn how to model such relationships after they have become familiar with the material in Chapter 3.

16. Create a Crow's Foot ERD for each of the following descriptions. (*Note*: The word *many* merely means "more than one" in the database modeling environment.)

a. Each of the MegaCo Corporation's divisions is composed of many departments. Each of those departments has many employees assigned to it, but each employee works for only one department. Each department is managed by one employee, and each of those managers can manage only one department at a time.

The Crow's Foot ERD is shown in Figure P2.16a.

FIGURE P2.16a The MegaCo Crow's Foot ERD

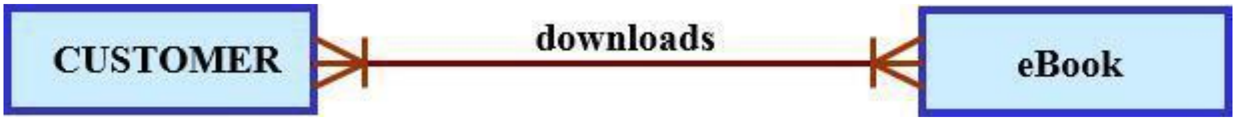


As you discuss the contents of Figure P2.16a, note the 1:1 relationship between the EMPLOYEE and the DEPARTMENT in the "manages" relationship and the 1:M relationship between the DEPARTMENT and the EMPLOYEE in the "is assigned to" relationship.

b. During some period of time, a customer can download many ebooks from BooksOnline. Each of the ebooks can be downloaded by many customers during that period of time.

The solution is presented in Figure P2.16b. Note the M:N relationship between CUSTOMER and EBOOK. Such a relationship is not implementable in a relational model.

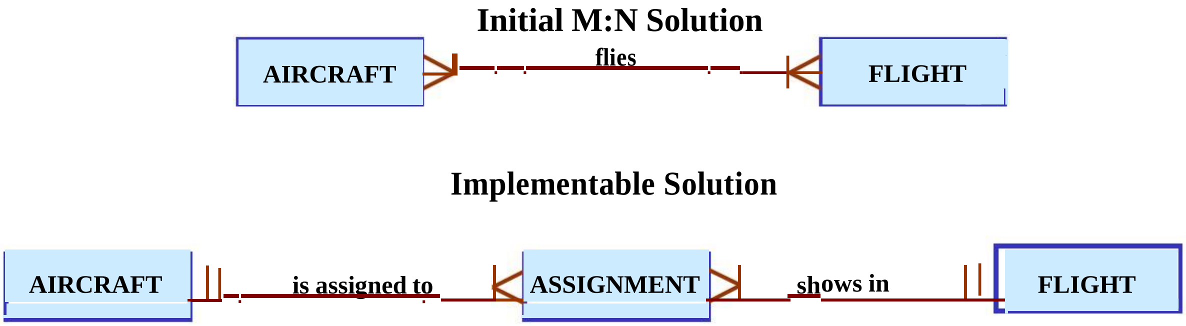
FIGURE P2.16b The BigVid Crow's Foot ERD



If you want to let the students convert Figure P2.16b's ERD into an implementable ERD, add a third DOWNLOAD entity to create a 1:M relationship between CUSTOMER and DOWNLOAD and a 1:M relationship between EBOOK and DOWNLOAD. (Note that such a conversion has been shown in the next problem solution.)

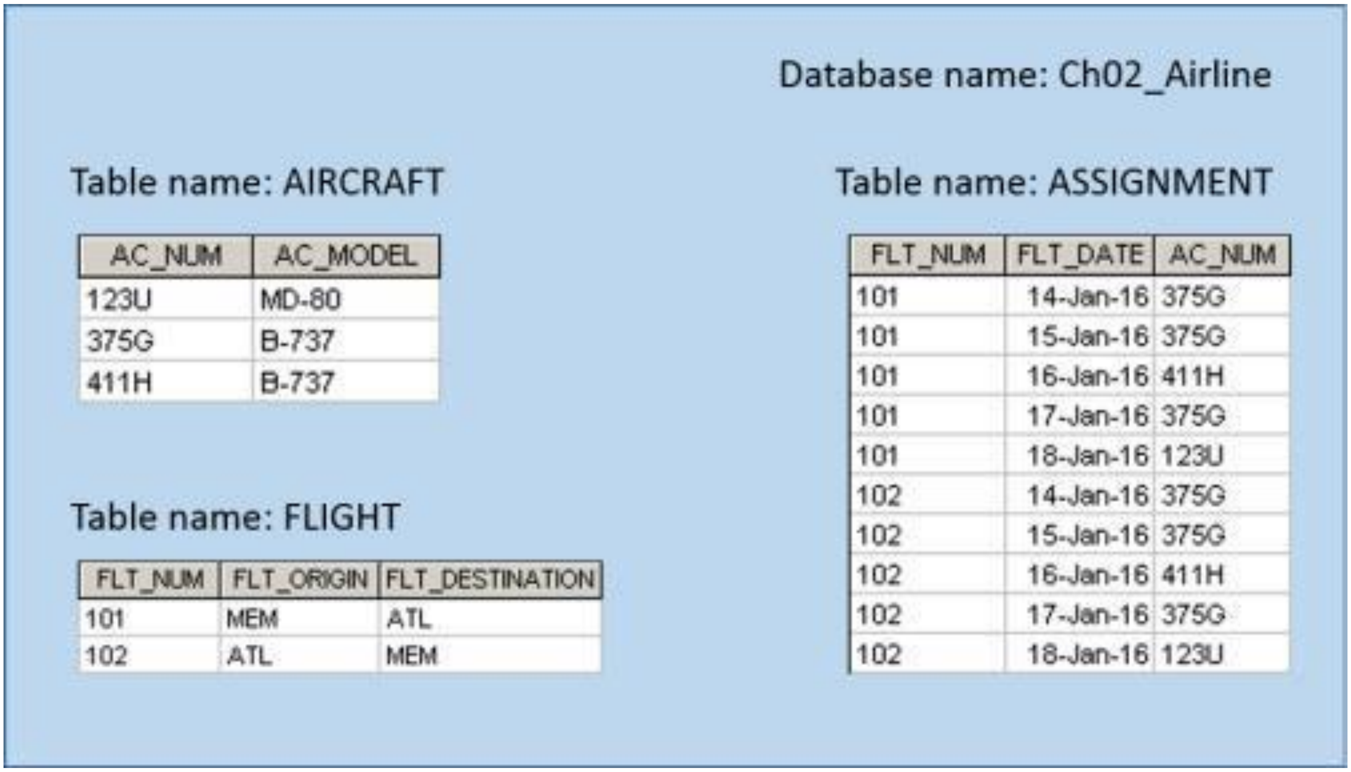
c. An airliner can be assigned to fly many flights, but each flight is flown by only one airliner.

FIGURE P2.16c The Airline Crow's Foot ERD



We have created a small **Ch02_Airline** database to let you explore the implementation of the model. (Check the data files available for Instructors at www.cengagebrain.com.) The tables and the relational diagram are shown in the following two figures.

FIGURE P2.16c The Airline Database Tables



Database name: Ch02_Airline

Table name: AIRCRAFT

AC_NUM	AC_MODEL
123U	MD-80
375G	B-737
411H	B-737

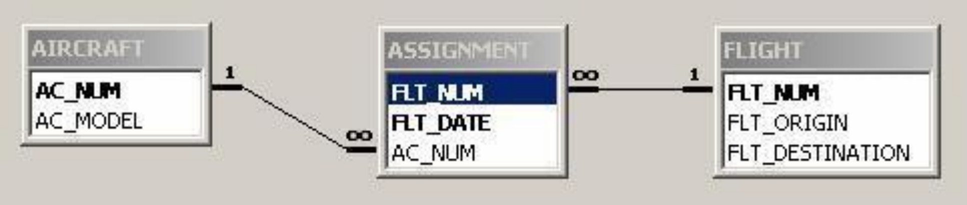
Table name: FLIGHT

FLT_NUM	FLT_ORIGIN	FLT_DESTINATION
101	MEM	ATL
102	ATL	MEM

Table name: ASSIGNMENT

FLT_NUM	FLT_DATE	AC_NUM
101	14-Jan-16	375G
101	15-Jan-16	375G
101	16-Jan-16	411H
101	17-Jan-16	375G
101	18-Jan-16	123U
102	14-Jan-16	375G
102	15-Jan-16	375G
102	16-Jan-16	411H
102	17-Jan-16	375G
102	18-Jan-16	123U

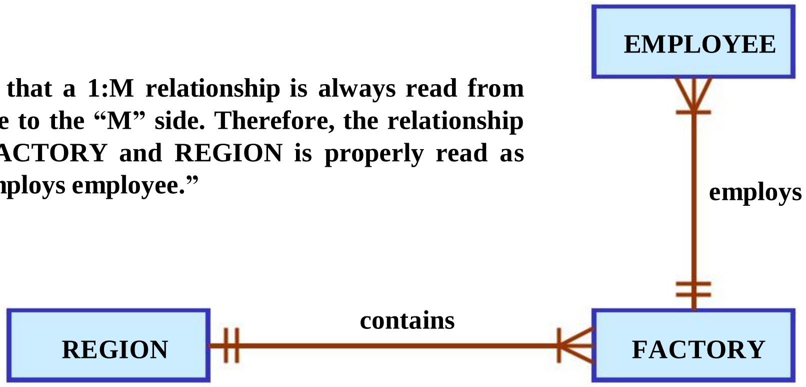
FIGURE P2.16c The Airline Relational Diagram



d. The KwikTite Corporation operates many factories. Each factory is located in a region. Each region can be "home" to many of KwikTite's factories. Each factory employs many employees, but each of those employees is employed by only one factory.

The solution is shown in Figure P2.16d.

FIGURE P2.16d The KwikTite Crow's Foot ERD

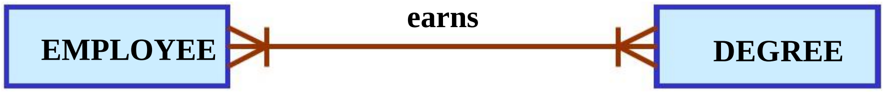


Remember that a 1:M relationship is always read from the "1" side to the "M" side. Therefore, the relationship between FACTORY and REGION is properly read as "factory employs employee."

e. An employee may have earned many degrees, and each degree may have been earned by many employees.

The solution is shown in Figure P2.16e.

FIGURE P2.16e The Earned Degree Crow's Foot ERD



Note that this M:N relationship must be broken up into two 1:M relationships before it can be implemented in a relational database. Use the Airline ERD's decomposition in Figure P2.16c as the focal point in your discussion.

17. **Write the business rules that are reflected in the ERD shown in Figure P2.17.** A theater show many movies.
A movie can be shown in many theaters.
A movie can receive many reviews. Each review is for a single movie.
A reviewer can write many reviews.
Each review is written by a single reviewer.

Note that the M:N relationship between theater and movie must be broken into two 1:M relationships using a bridge table before it can be implemented in a relational database.

Appendix B

The University Lab: Conceptual Design

Discussion Focus

What actions are taken during the database initial study, and why are those actions important to the database designer?

NOTE

illustrate how the database design map was used to generate the *initial* ER diagram.

The database initial study is essentially a process based on data gathering and analysis. Carefully conducted and systematic interviews usually constitute an important part of this process.

The initial study must take its cues from an organization's key end users. Therefore, one of the first initial study tasks is to establish who the organization's key end users are. Once the key end users are identified, the initial study must be conducted to establish the following outputs:

objectives
    organizational structure
    description of operations
    definition of problems and constraints
    description of system objectives
definition of system scope and boundaries

The database designer cannot expect to develop a usable design unless these outputs are carefully defined and delineated. The importance of having such a list of outputs is self-explanatory. For example, a database design is not likely to be useful unless and until it accomplishes specific objectives and helps solve an organization's problems. The inherent assumption is that those problems are usually based on the lack of useful and timely information.

The value of having such a list of required outputs is clear, too, because this list constitutes a checklist to be used by the database designer. The designer should not proceed with the database design until all the items on this list have been completed.

What is the purpose of the conceptual design phase, and what is its end product?

The conceptual design phase is the first of three database design phases: conceptual, logical, and physical. The purpose of the conceptual design phase is to develop the following outputs:

information sources and users

Information needs: user

requirements the initial ER model

the definition of attributes and domains

The conceptual design's end product is the initial ER diagram, which constitutes the preliminary database blueprint. It is very unlikely that useful logical and physical designs can be produced unless and until this blueprint has been completed.

Too much "design" activity takes place without the benefit of a carefully developed database blueprint. Implementing a database without a good database blueprint almost invariably produces a lack of data integrity based on various data anomalies. In fact, it may easily be argued that implementing a successful database without a good database blueprint is just as likely as writing a great book by stringing randomly selected words together.

Why is an initial ER model not likely to be the basis for the implementation of the database?

ER modeling is an iterative process. The initial ER model may establish many of the appropriate entities and relationships, but it may be impossible to implement such relationships. Also, given the nature of the ER modeling process, it is very likely that the end users begin to develop a greater understanding of their organization's operations, thus making it possible to establish additional entities and relationships. In fact, it may be argued that one very important benefit of ER modeling is based on the fact that it is an outstanding *communications tool.* In any case, before the ER model can be implemented, it must be carefully *verified* with respect to the business transactions and information requirements. (Note that students will learn how to develop the verification process in Appendix C.)

Clearly, unless and until the ER model *accurately* reflects an organization's operations and requirements, the development of logical and physical designs is premature. After all, the database implementation is only as good as the final ER blueprint allows it to be!

Answers to Review Questions

1. **What factors relevant to database design are uncovered during the initial study phase?**

The database initial study phase yields the information required to determine an organization's needs, as well as the factors that influence data generation, collection, and processing. Students must understand that this phase is generally concurrent with the planning phase of the SDLC and that, therefore, several of the initial study activities are common to both.

The most important discovery of the initial study phase is the set of the company's objectives. Once the designer has a clear understanding of the company's main goals and its mission, (s)he can use this as the guide to making all subsequent decisions concerning the analysis, design, and implementation of the database and the information system.

The initial study phase also establishes the company's organizational structure; the description of operations, problems and constraints, alternate solutions; system objectives; and the proposed system scope and boundaries.

The organizational structure and the description of operations are interdependent because operations are usually a function of the company's organizational structure. The determination of structure and operations allows the designer to analyze the existing system and to describe a set of problems, constraints, and possible solutions.

Naturally, the designer must find a feasible solution within the existing constraints. In most cases, the best solution is not necessarily the most feasible one. The constraints also force the designer to narrow the focus on very specific problems that must be solved.

In short, the combination of all the factors we have just discussed help the designer to put together a set of realistic, achievable, and measurable system objectives within the system's required scope and boundaries.

2. **Why is the organizational structure relevant to the database designer?**

The delivery of information must be timely, it must reach the right people, and the delivered information must be accurate. Since the proper use of timely and accurate information is the key factor in the success of any system, the reports and queries drawn from the database must reach the key decision makers within the organization. Clearly, understanding the organization structure helps the designer to define the organization's lines of communication and to establish reporting requirements.

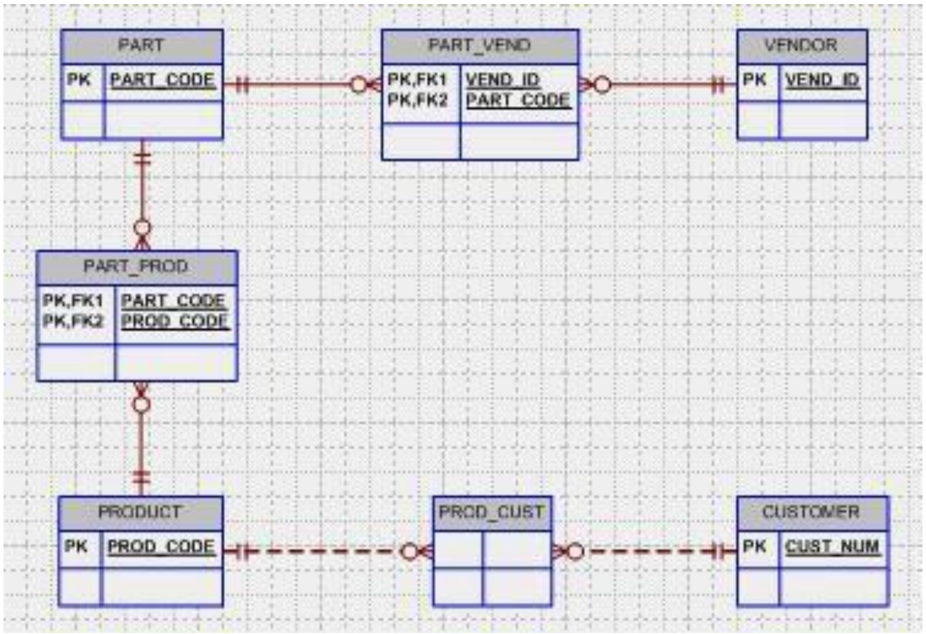
3. **What is the difference between the database design scope and its boundaries? Why is the scope and boundary statement so important to the database designer?**

The system's boundaries are the limits imposed on the database design by external constraints such as available budget and time, the current level of technology, end-user resistance to change, and so on. The scope of a database defines the extent of the database design coverage and reflects a conscious decision to include some things and exclude others. Note that the existence of boundaries usually has an effect on the system's scope.

For legal and practical design reasons, the designer cannot afford to work on an unbounded system. If the system's limits have not been adequately defined, the designer may be legally required to expand the system indefinitely. Moreover, an unbounded system will not contain the built-in constraints that make its use practical in a real-world environment. For example, a completely unbounded system will never be completed, nor may it ever be ready for reasonable use. Even a system with an "optimistic" set of bounds may drag the design out over many years and may cost too much. Keep in mind that company managers almost invariably want least-cost solutions to specific problems.

4. What business rule(s) and relationships can be described for the ERD in Figure QB.4?

Figure QB.4 The ERD for Question 4



The business rules and relationships are summarized in Table QB.4

Table QB.4 Business Rules and Relationships Summary

Business rules	Relationships
A supplier supplies many parts. Each part is supplied by many suppliers.	many to many PART - SUPPLIER
A part is used in many products. Each product is composed of many parts.	many to many PRODUCT - PART
A product is bought by many customers. Each customer buys many products.	many to many PRODUCT - CUSTOMER

Note that the ERD in Figure QB.4 uses the PART_PROD, PROD_VEND and PROD_CUST entities to convert the M:N relationships to a series of 1:M relationships. Also, note the use of two composite entities:

The PART_VEND entity's composite PK is VEND_ID + PART_CODE.
The PART_PROD entity's composite PK is, PART_CODE + PROD_CODE.

The use of these composite PKs means that the relationship between PART and PART_VEND is strong, as is the relationship between VENDOR and PART_VEND. These strong relationships are indicated through the use of a solid relationship line.

No PK has been indicated for the PROD_CUST entity, but the existence of weak relationships – note the dashed relationship lines – lets you assume that the PROD_CUST entity's PK is not a composite one. In this case, a revision of the ERD might include the establishment of a composite PK (PROD_CODE + CUST_NUM) for the PROD_CUST entity. (If you are using Microsoft Visio Professional, declaring the relationships between CUSTOMER and PROD_CUST and between PRODUCT and PROD_CUST to be strong will automatically generate the composite PK (PROD_CODE + CUST_NUM.)

5. **Write the connectivity and cardinality for each of the entities shown in Question 4.**

We have indicated the connectivities and cardinalities in Figure QB.5. (The Crow's Foot ERD combines the connectivity and cardinality depiction through the use of the relationship symbols. Therefore, the use of text boxes – we have created those with the Visio text tool -- to indicate connectivities and cardinalities is technically redundant.)

Figure QB.5 Connectivities and Cardinalities

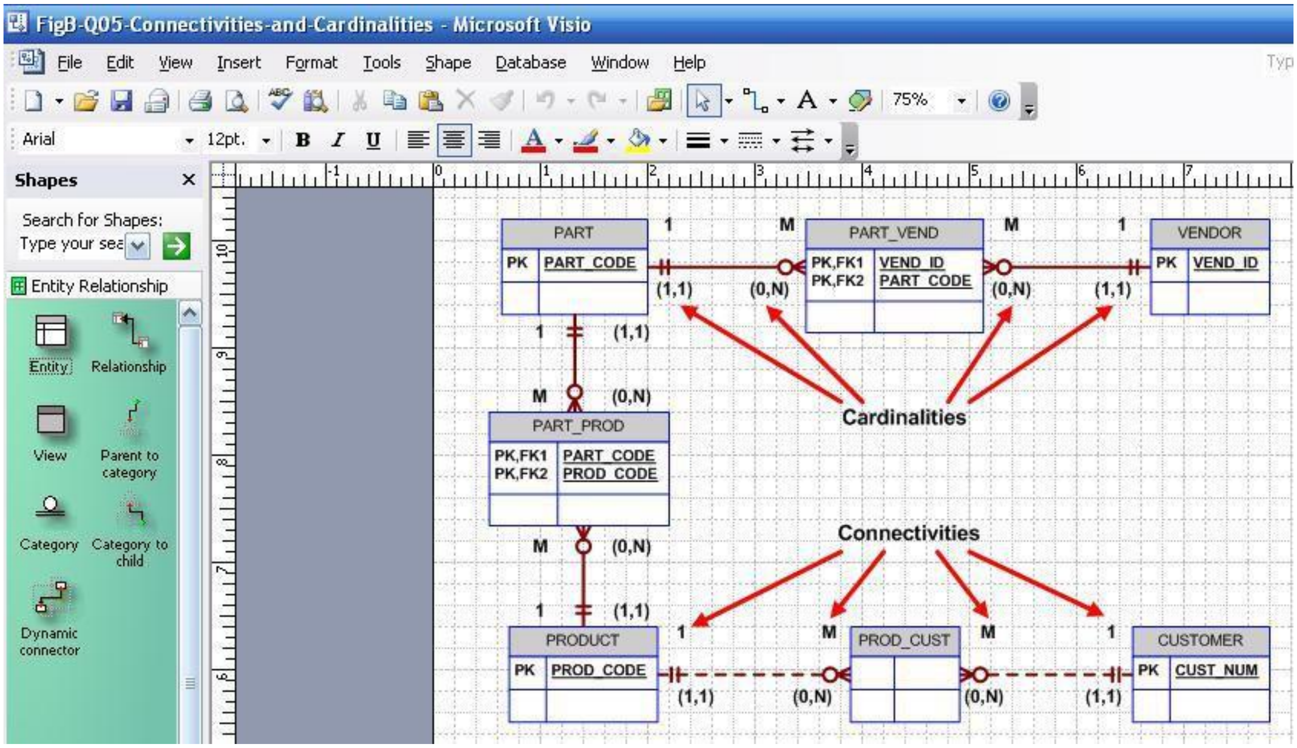


Figure QB.5's connectivities and cardinalities are reflected in the business rules:

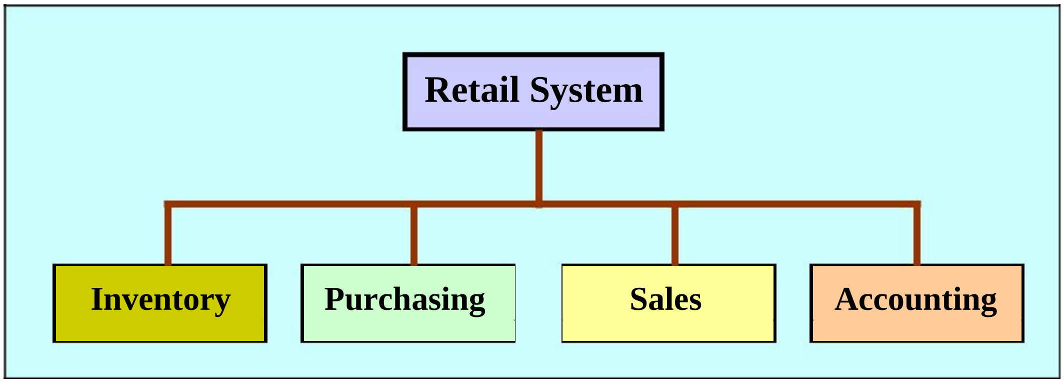
One part can be supplied by one or more suppliers, and a supplier can supply many parts. A product is made up of several parts, and a part can be a component of different products.

A product can be bought by several customers, and a customer can purchase several products.

6. What is a module, and what role does a module play within the system?

A module is a separate and independent collection of application programs that covers a given operational area within an information system. A module accomplishes a specific system function, and it is, therefore, a component of the overall system. For example, a system designed for a retail company may be composed of the modules shown in Figure QB.6.

Figure QB.6 The Retail Company System Modules



Within Figure QB.6's Retail System, each module addresses specific functions. For example:

The inventory module registers any new item, monitors quantity on hand, reorder quantity, location, etc.

The purchasing module registers the orders sent to the suppliers, any supplier information, order status, etc.

The sales module covers the sales of items to customers, generates the sales slips (invoices), credit sales checks, etc.

The accounting module covers accounts payable, accounts receivable, and generates appropriate financial status reports.

The example demonstrates that each module has a specific purpose and operates on a database subset (external view). Each external view represents the entities of interest for the specific module. However, an entity set may be shared by several modules.

7. What is a module interface, and what does it accomplish?

A module interface is the method through which modules are connected and by which they interact with one another to exchange data and status information. The definition of proper module interfaces is critical for systems development, because such interfaces establish an ordered way through which system components (modules) interchange information. If the module interfaces are not properly defined, even a collection of properly working modules will not yield a useful working system.

Problem Solutions

1. **Modify the initial ER diagram presented in Figure B.19 to include the following entity supertype and subtypes: The University Computer Lab USER may be a *student* or a *faculty member*.**

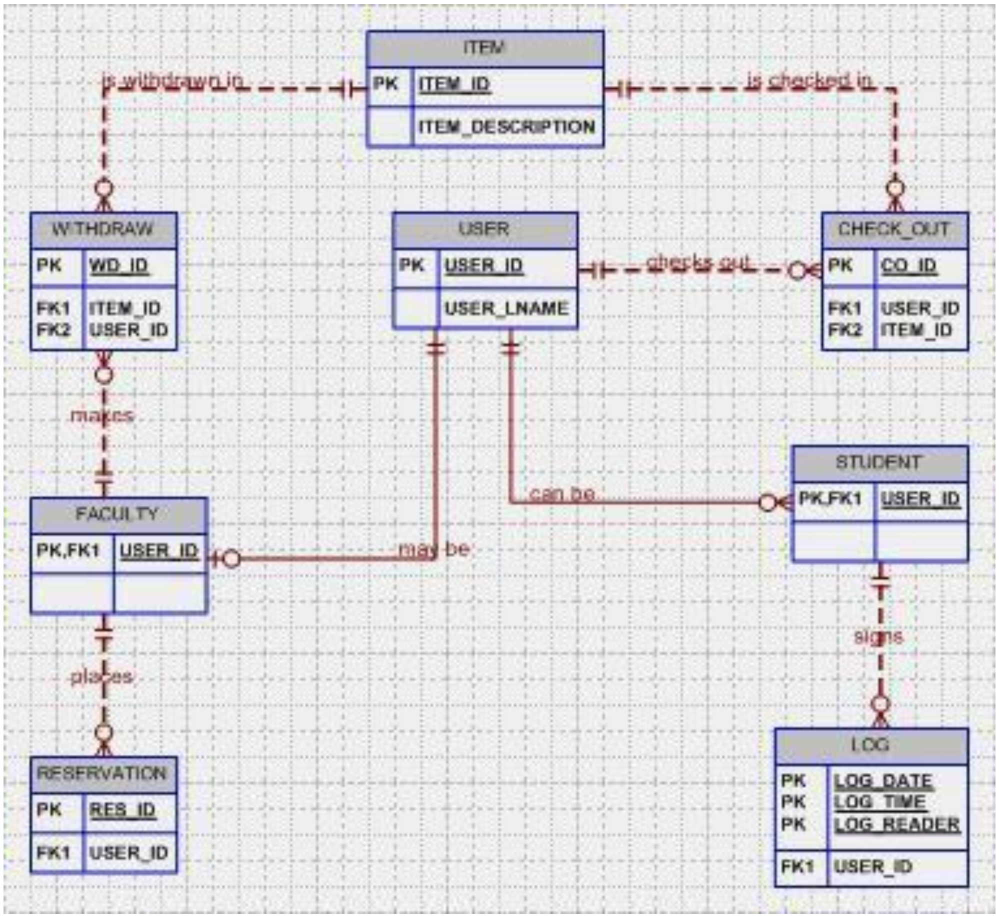
   The answer to problem 1 is included in the answer to problem 2.

2. **Using an ER diagram, illustrate how the change you made in problem 1 affects the relationship of the USER entity to the following entities:**

   LAB_USE_LOG
   RESERVATION
   CHECK_OUT
   WITHDRAW

   The new ER diagram segment will contain the supertype USER and the subtypes FACULTY and STUDENT. How the use of this supertype/subtype relationships affect the entities shown here is illustrated in the ER diagram shown in Figure PB.2a.

Figure PB.2a The Crow’s Foot ERD with Supertypes and Subtypes



The ER segment shown in Figure PB.2a reflects the following conditions:

Not all users are faculty members, so FACULTY is optional to USER. Not all users are students, so STUDENT is optional to USER.

The conditions in the first two bullets are typical of the supertype/subtype implementation.

Not all faculty members withdraw items, so a faculty member may not ever show up in the WITHDRAW table. Therefore, WITHDRAW is optional to FACULTY.

Not all items are necessarily withdrawn; some are never used. Therefore WITHDRAW is optional to ITEM. (An item that is never withdrawn will never show up in the WITHDRAW table.)

Not all items are checked out, so an ITEM may never show up in the CHECK_OUT table. Therefore, CHECK_OUT is optional to ITEM.

Not all users check out items, so it is possible that a USER – a faculty member or a student -- never shows up in the CHECK_OUT table. Therefore, CHECK_OUT is optional to USER.

Not all faculty members place reservations, so RESERVATION is optional to FACULTY.

Not all students use the lab, i.e., some students will never sign the log to check in. Therefore, LOG is optional to STUDENT.

Given the text’s initial development of the UCL Management System’s ERD, the USER entity was related to both the WITHDRAW and CHECK_OUT entities. Therefore, there was no way of knowing whether a STUDENT or a FACULTY member was related to WITHDRAW or CHECK_OUT. Although the business rules were quite specific about the relationships, the ER diagram did not reflect them.

By adding a new USER supertype and two STUDENT and FACULTY subtypes, the ERD more closely represents the business rules. The supertype/subtype relationship in Figure PB.2a lets us see that STUDENT is related to LOG, and that only FACULTY members can make a RESERVATION and WITHDRAW items. However, both STUDENT and FACULTY can CHECK_OUT items.

While this supertype/subtype solution conforms to the problem solution requirements, the design is far from complete. For example, one would suppose that FACULTY is already a subtype to EMPLOYEE. Also, can a faculty member also be a student? In other words, are the supertypes/subtypes overlapping or disjoint? In this initial ERD, we have assumed overlapping subtypes; that is, a user can be a faculty member and a student at the same time.

Another solution -- which would eliminate the USER/FACULTY and USER/STUDENT supertype/subtypes in the ERD – is to add an attribute, such as USER_TYPE, to the USER entity to identify the user as faculty or student. The application software can then be used to enforce the restrictions on various user types. Actually, that approach was used in the final (verified) **Computerlab.mdb** database on your CD. (The verified database is provided for Appendix C.)

3. **Create the initial ER diagram for a car dealership. The dealership sells both new and used cars, and it operates a service facility. Base your design on the following business rules:**
   a. **A salesperson can sell many cars but each car is sold by only one salesperson.**
   b. **A customer can buy many cars but each car is sold to only one customer.**
   c. **A salesperson writes a single invoice for each car sold.**
   d. **A customer gets an invoice for each car (s)he buys.**
   e. **A customer might come in just to have a car serviced; that is, one need not buy a car to be classified as a customer.**
   f. **When a customer takes one or more cars in for repair or service, one service ticket is written for each car.**
   g. **The car dealership maintains a service history for each car serviced. The service records are referenced by the car's serial number.**
   h. **A car brought in for service can be worked on by many mechanics, and each mechanic may work on many cars.**
   i. **A car that is serviced may or may not need parts. (For example, parts are not necessary to adjust a carburetor or to clean a fuel injector nozzle.)**

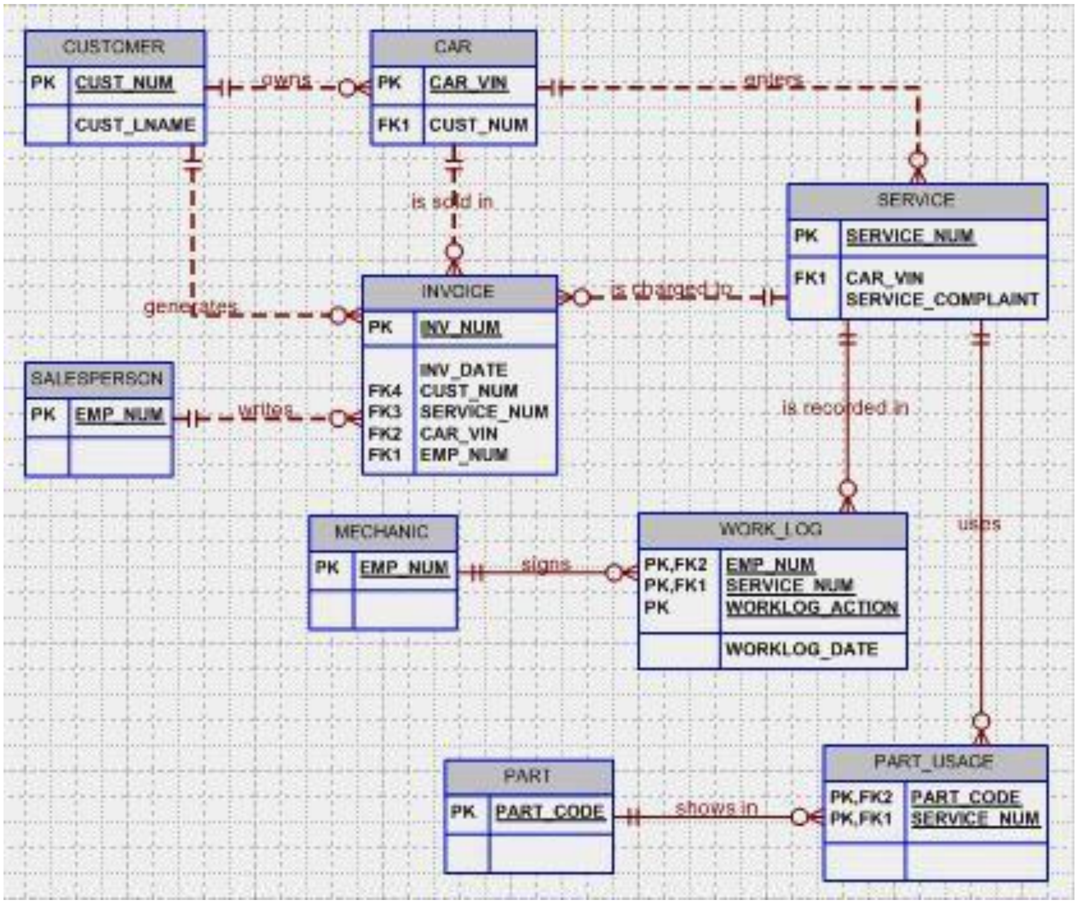
As you examine the initial ERD in Figure PB.3a, note that business rules (a) through (d) refer to the relationships of four main entities in the database: SALESPERSON, INVOICE, CUSTOMER, and CAR. Note also that an INVOICE requires a SALESPERSON, a CUSTOMER, and a CAR. Business rule (e) indicates that INVOICE is optional to CUSTOMER and CAR because a CAR is not necessarily sold to a CUSTOMER. (Some customers only have their cars serviced.)

The position of the CAR entity and its relationships to the CUSTOMER and INV_LINE entities is subject to discussion. If the dealer sells the CAR, the CAR entity is clearly related to the INV_LINE that is related to the INVOICE. (If the car is sold, it generates one invoice line on the invoice. However, the invoice is likely to contain additional invoice lines, such as a dealer preparation charge, destination charge, and so on.) At this point, the discussion can proceed in different directions:

The sold car can be linked to the customer through the invoice. Therefore, the relationship between CUSTOMER and CAR shown in Figures PB.3a and PB.3b is not necessary.

If the customer brings a car in for service – whether or not that car was bought at the dealer – the relationship between CUSTOMER and CAR is desirable. After all, when a service ticket is written in the SERVICE_LOG, it would be nice to be able to link the customer to the subsequent transaction. More important, it is the customer who gets the invoice for the service charge. However, if the CUSTOMER-CAR relationship is to be retained, it will be appropriate to make a distinction between the cars in the dealership's inventory – which are not related to a customer at that point – and the cars that are owned by customers. If no distinction is made between customer-owned cars and cars still in the dealership inventory, Figure PB.3a's CAR entity will either have a null CUST_NUM or the customer entity must contain a dummy record to indicate a "no customer – dealer-owned" condition.

Figure PB.3a The Car Dealership Initial Crow's Foot ERD



Regardless of which argument "wins" in the presentation of the various scenarios, remind the students that the ERD to be developed in this exercise is to reflect the *initial* design. More important, such discussions clearly indicate the need for very detailed descriptions of operations and the development of precisely written business rules. (It may be useful to review that business rules, which are derived from the description of operations, are written to help define entities, relationships, constraints, connectivities, and cardinalities.)

The dealer's service function is likely to be crucial to the dealer – good service helps generate future sales and the service function is very likely an important cash flow generator. Therefore, the CAR entity plays an important role. If a customer brings in a car for service and the car was not bought at the dealership, it must be added to the CAR table in order to enable the system to keep a record of the service. This is why we have depicted the CUSTOMER – *owns* - CAR relationship in Figures PB.3a and PB.3b. Also, note that the optionality next to CAR reflects the fact that not all cars are owned by a customer: Some cars belong to the dealership.

Because Figure PB.3a shows the *initial* ERD, that ERD will be subject to revision as the description of operations becomes more detailed and accurate, thus modifying some of the existing business rules and creating additional business rules. Therefore, additional entities and relationships are likely to be developed and some optional relationships may become mandatory, while some mandatory relationships may become optional. Additional changes are likely to be generated by normalization procedures. Finally, the initial design includes some features that require fine-tuning. For example, a SALESPERSON is just another kind of EMPLOYEE – perhaps the main difference between "general" employee and a sales person is that the latter requires tracking of sales performance for commission and/or bonus purposes. Therefore, EMPLOYEE would be the supertype and SALESPERSON the subtype. All these issues must be addressed in the verification and logical design phases addressed in Appendix E.

Incidentally, your students may ask why the design does not show a HISTORY entity. The reason for is absence is that the car's history can be traced through the SERVICE entity.

NOTE

provides ample evidence that the initial ERD is only a *starting* point for the design process.

As you discuss the design shown in Figure PB.3a, note that it is far from implementation-ready. For example,

The INVOICE is likely to contain multiple charges, yet it is only capable of handling one charge at a time at this point. The addition of an INV_LINE entity is clearly an excellent idea.

The SERVICE entity has some severe limitations caused by the lack of a SERVICE_LINE entity. (Note the previous point.) Given this design, it is impossible to store and track all the individual service (maintenance) procedures that are generated by a single service request. For example, a 50,000 mile check may involve multiple procedures such as belt replacements, tire rotation, tire balancing, brake service, and so on. Therefore, the SERVICE entity, like the INVOICE entity, must be related to service lines, each one of which details a specific maintenance procedure.

The PART_USAGE entity's function is rather limited. For example, its depiction as a composite entity does properly translate the notion that a part can be used in many service procedures and a service procedure can use many parts. Unfortunately, the lack of a SERVICE_LINE entity means that we cannot track the parts use to a particular maintenance procedure.

According to business rule (d), the relationship between CAR and INVOICE would be 1:1. However, if it is possible for the dealer to take the car in trade at a later date and subsequently sells it again, the same CAR_VIN value may appear in INVOICE more than once. We have depicted the latter scenario.

The initial design does have one very nice feature at this point: The existence of the WORK_LOG entity's WORKLOG_ACTION attribute makes it possible to record which mechanic started the service procedure and which one ended the procedure. (The WORKLOG_ACTION attribute has only two values, open and close.) Note that this feature eliminates the need for a null ending date in the SERVICE entity while the car is being serviced. Better yet, if we need to be able to track which mechanics opened and closed the service procedure, the WORK_LOG entity's presence eliminates the need for synonyms in the SERVICE entity. Note, for example, that the following few sample entries in the WORK_LOG table lets us conclude that service number 12345 was opened by mechanic 104 on 10-Mar-2014 and closed by the same mechanic on 11-Mar-2014.

Table PB.3 Sample Data Entries in the WORK_LOG Entity

EMP_NUM	SERVICE_NUM	WORKLOG_ACTION	WORKLOG_DATE
104	12345	OPEN	10-Mar-2014
107	12346	OPEN	10-Mar-2014
104	12345	CLOSE	11-Mar-2014
104	12346	CLOSE	11-Mar-2014
112	12347	OPEN	11-Mar-2014

The format you see in Table PB.3 is based on a standard we developed for aviation maintenance databases. Because almost all aspects of aviation are tightly regulated, accountability is always close to the top of the list of design requirements. (In this case, we must be able to find out who opened the maintenance procedure and who closed it.) You will discover in Chapter 9, "Database Design," that we will apply the accountability standard to other aspects of the design, too. (Who performed each maintenance procedure? Who signed out the part(s) used in each maintenance procedure? And so on.)

It is worth repeating that a discussion of the shortcomings of the initial design will set an excellent stage for the introduction of Appendix C's verification process.

Strict accountability standards are becoming the rule in many areas outside aviation. Such standards may be triggered by legislation or by company operations in an increasingly litigious environment.

4. **Create the initial ER diagram for a video rental shop. Use (at least) the following description of operations on which to base your business rules.**

The video rental shop classifies movie titles according to their type: Comedy, Western, Classical, Science Fiction, Cartoon, Action, Musical, and New Release. Each type contains many possible titles, and most titles within a type are available in multiple copies. For example, note the summary presented in Table PB.4:

Table PB.4 The Video Rental Type and Title Relationship

TYPE	TITLE	COPY
Musical	My Fair Lady	1
	My Fair Lady	2
	Oklahoma!	1
	Oklahoma!	2
	Oklahoma!	3
Cartoon	Dilly Dally & Chit Chat Cat	1
	Dilly Dally & Chit Chat Cat	2
	Dilly Dally & Chit Chat Cat	3
Action	Amazon Journey	1
	Amazon Journey	2

Keep the following conditions in mind as you design the video rental database:
The movie type classification is standard; not all types are necessarily in stock.

The movie list is updated as necessary; however, a movie on that list might not be ordered if the video shop owner decides that it the movie is not desirable for some reason.

The video rental shop does not necessarily order movies from all of the vendor list; some vendors on the vendor list are merely potential vendors from whom movies may be ordered in the future.

Movies classified as new releases are reclassified to an appropriate type after they have been in stock for more than 30 days. The video shop manager wants to have an end-of-period (week, month, year) report for the number of rentals by type.

If a customer requests a title, the clerk must be able to find it quickly. When a customer selects one or more titles, an invoice is written. Each invoice may thus contain charges for one or more titles. All customers pay in cash.

When the customer checks out a title, a record is kept of the checkout date and time and the expected return date and time. Upon the return of rented titles, the clerk must be able to check quickly whether the return is late and to assess the appropriate late return fee.

The video-store owner wants to be able to generate periodic revenue reports by title and by type. The owner also wants to be able to generate periodic inventory reports and to keep track of titles on order.

The video-store owner, who employs two (salaried) full-time and three (hourly) part-time employees, wants to keep track of all employee work time and payroll data. Part-time employees must arrange entries in a work schedule, while all employees sign in and out on a work log.

NOTE

The description of operations not only establishes the operational aspects of the business; it also establishes some specific system objectives we have listed next.

As you design this database, remember that transaction and information requirements help drive the design by defining required entities, relationships, and attributes. Also, keep in mind that the description provided by the problem leaves many possibilities for design differences. For example, consider the EMPLOYEE classification as full-time or part-time. If there are few distinguishing characteristics between the two, the situation may be handled by using an attribute EMP_CLASS (***whose values might be F or P***) in the EMPLOYEE table. If full-time employees earn a base salary and part-time employees earn only an hourly wage, that problem can be handled by having two attributes, EMP_HOURPAY and EMP_BASE_PAY, in EMPLOYEE. Using this approach, the HOUR_PAY would be $0.00 for the salaried full-time employees, while the EMP_BASE_PAY would be $0.00 for the part-time employees. (To ensure correct pay computations, the application software would select either F or P, depending on the employee classification.) On the other hand, if part-time employees are handled quite differently from full-time employees in terms of work scheduling, benefits, and so on, it would be better to use a supertype/subtype classification for FULL_TIME and PART_TIME employees. (The more unique variables exist, the more sense a supertype/subtype relationship makes.)

For discussion purposes, examine the following requirements:

The clerk must be able to find customer's requests quickly.

This requirement is met by creating an easy way to query the MOVIE data (by name, type, etc.) while entering the RENTAL data.

The clerk must be able to check quickly whether or not the return is late and to assess the appropriate “late return” fee. This requirement is met by adding attributes such as expected return date, actual return date, and late fees to the RENTAL entity. Note that there is no need to add a new entity, nor do we need to create an additional relationship. Keep in mind that some requirements are easily met by including the appropriate attributes in the tables and by combining those attributes through an application program that enforces the business rule. ***Remember that not all business rules can be represented in the database conceptual diagram.***

The (store owner) wants to be able to keep track of all employee work time and payroll data. Here we must create two new entities: WORK_SCHEDULE and WORK_LOG, which will show the employee's work schedule and the actual times worked, respectively. These entities will also help us generate the payroll report.

The description also specifies some of the expected reports:

End-of-period report for the number of rentals by type. This report will use the RENTAL, MOVIE, and TYPE entities to generate all rental data for some specified period of time.

Revenue report by title and by type. This report will use the RENTAL, MOVIE, and TYPE entities to generate all the rental data.

Periodic inventory reports. This report will use the MOVIE and TYPE entities.

Titles on order. This report will use the ORDER, MOVIE, and TYPE entities.

Employee work times and payroll data. This report will use the EMPLOYEE, WORK_SCHEDULE, and WORK_LOG entities.

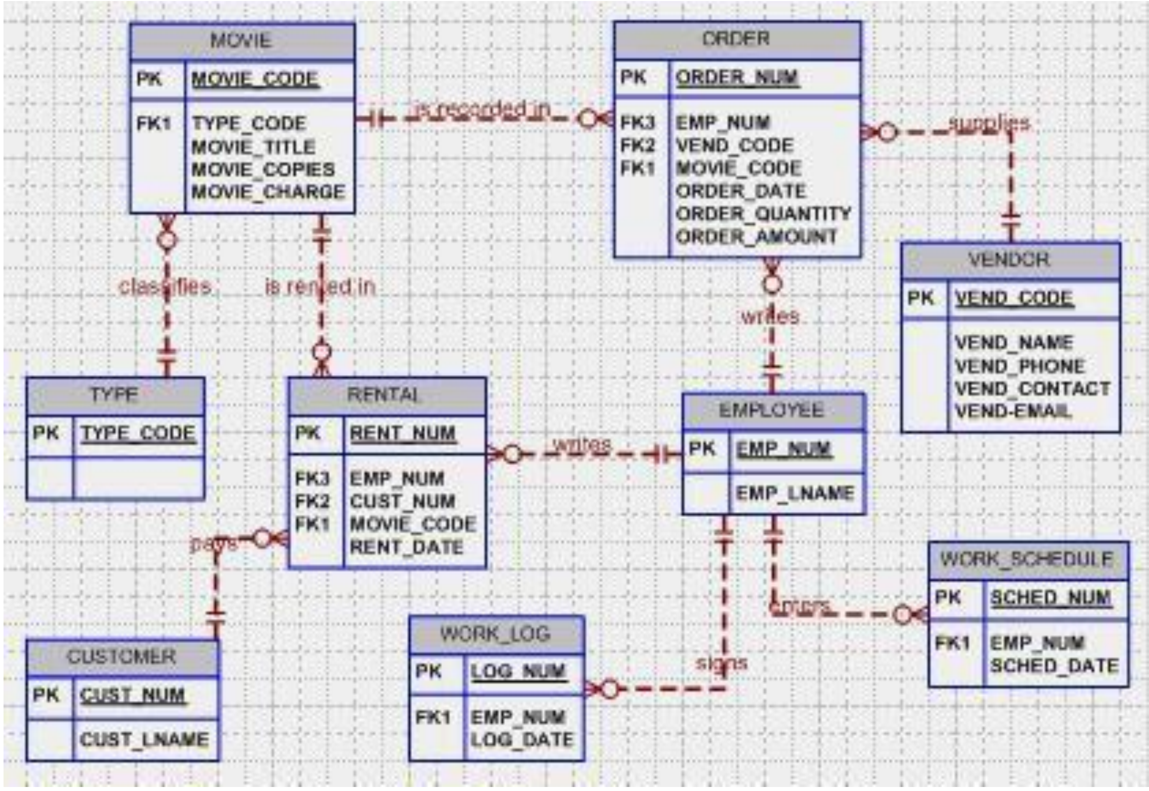
This summary sets the stage for the ERD shown in Figure PB.4a. Note that the WORK_SCHEDULE and WORK_LOG entities are optional to EMPLOYEE. The optionalities reflect the following conditions:

Only part-time employees have corresponding records in the work log table.
Only full-time employees have corresponding records in the work schedule table.

Although there is a temptation to create FULL_TIME and PART_TIME entities, which are then related to WORK_LOG and WORK_SCHEDULE, respectively, such a decision reflects a substitution of an entity for an attribute. It is far better to simply create an attribute, perhaps named EMP_TYPE, in the EMPLOYEE entity. The EMP_TYPE attribute values would then be P = part- time or F = full-time. The applications software can then be used to force an entry into the WORK_LOG and WORK_SCHEDULE entities, depending on the EMP_TYPE attribute value.

Student question: Using the argument just presented, what other entity might be replaced by an attribute? Answer: The TYPE entity can be represented by a TITLE_TYPE attribute in the TITLE entity. The TITLE_TYPE values would then be "Western", "Adventure", and so on. This approach works fine, as long as the type values don't require additional descriptive material. In the latter case, the TYPE would be better represented by an entity in order to avoid data redundancy problems.

Figure PB.4a The Initial Crow's Foot ERD for the Video Rental Store



Additional discussion: At this point, the ERD has not yet been verified against the transaction requirements. For example, there is no way to check which specific video has been rented by a customer. (If five customers rent copies of the same video, you don't know which customer has which copy.) Therefore, the design requires additional work triggered by the verification process.

In addition, the work log entity's LOG_DATE is incapable of tracking when the part-time employees logged in or out. Therefore, two dates must be used, perhaps named LOG_DATE_IN and LOG_DATE_OUT. In addition, if you want to determine the hours worked by each part-time employee, it will be necessary to record the time in and time out.

Similarly, the work schedule cannot yet be used to track the full-time employees' schedules. Who has worked and when? Clearly, the verification process discussed in Appendix C is not a luxury!

5. **Suppose a manufacturer produces three high-cost, low-volume products: P1, P2, and P3. Product P1 is assembled with components C1 and C2; product P2 is assembled with components C1, C3, and C4; and product P3 is assembled with components C2 and C3. Components may be purchased from several vendors, as shown in Table PB.5:**

Table PB.5 The Component/Vendor Summary

VENDOR	COMPONENTS SUPPLIED
V1	C1, C2
V2	C1, C2, C3, C4
V3	C1, C2, C4

Each product has a unique serial number, as does each component. To keep track of product performance, careful records are kept to ensure that each product's components can be traced to the component supplier.

Products are sold directly to final customers; that is, no wholesale operations are permitted. The sales records include the customer identification and the product serial number. Using the preceding information, do the following:

a. **Write the business rules governing the production and sale of the**

products. The business rules are summarized in Figure PB.5A.

Figure PB.5A The Business Rule Summary

PRODUCT	COMPONENTS			
P1	C1	C2		
P2	C1		C3	C4
P3		C2	C3	

Business Rule

1. A component can be part of several products, and a product is made up of several components.

VENDOR	COMPONENTS SUPPLIED			
V1	C1	C2		
V2	C1	C2	C3	C4
V3	C1	C2		C3

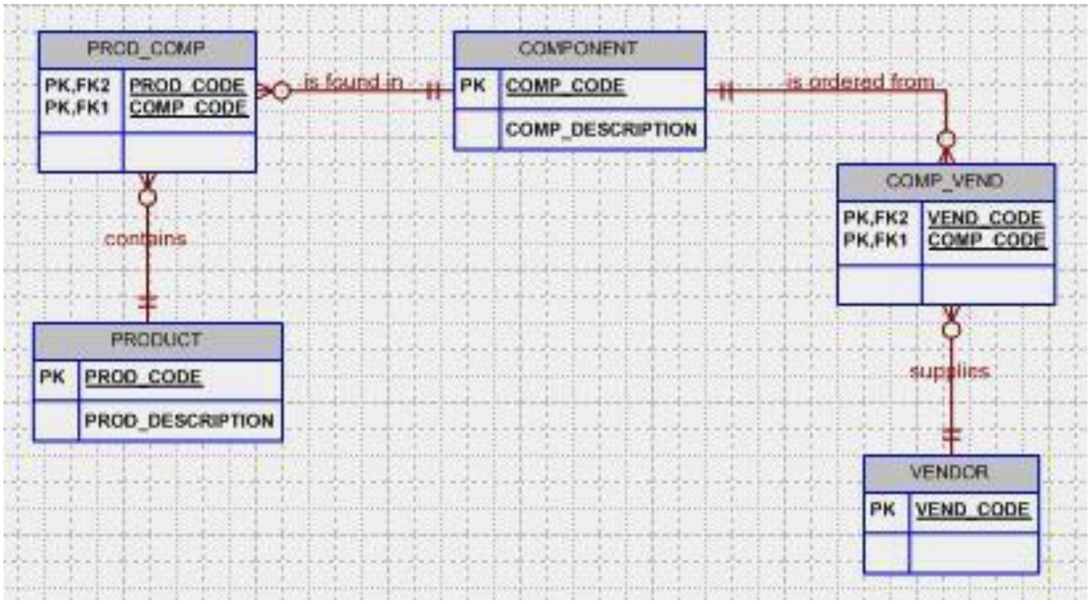
Business Rule

2. A component can be supplied by several vendors, and a vendor supplies several components.

b. Create an ER diagram capable of supporting the manufacturer's product/component tracking requirements.

The two business rules shown in Figure PB.5A allow the designer to generate the ERD Shown in Figure PB.5B1. (Note the M:N relationships between PRODUCT and COMPONENT and between COMPONENT and VENDOR that have been converted through the composite entities PROD_COMP and COMP_VENB.)

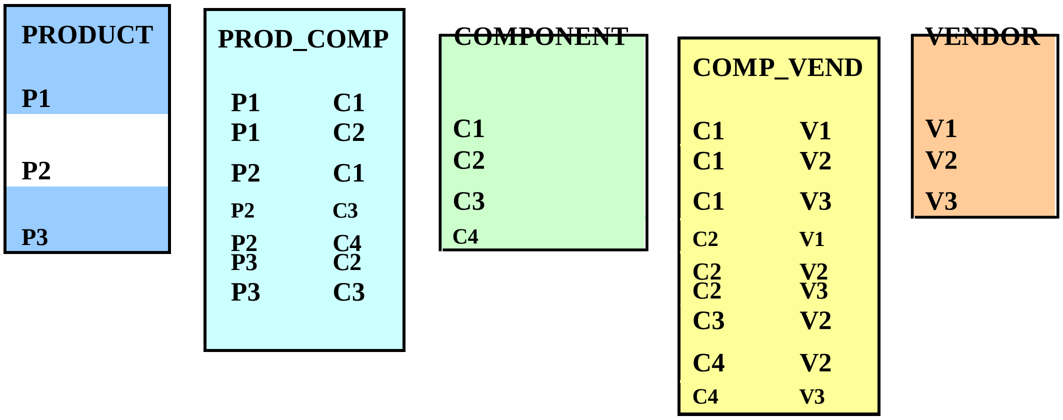
Figure PB.5B1 The Initial Crow's Foot ERD for Problem B.5B



As you examine Figure PD5.B1, note that we have use default optionalities in the composite entities named PROD_COMP and COMP_VENB. Naturally, these optionalities must be verified against the business rules before the design is implemented. However, at this point the optionalities make sense – after all, various version of a PRODUCT do not necessarily contain all available COMPONENTs, not do all VENDORs supply all COMPONENTs. Quite aside from the likely existence of the relationships we just pointed out, optionalities are generally desirable from an operational point of view – at least from the database management angle. Yet, no matter how “obvious” a relationship may appear to be, it is worth repeating that the existence of the optionalities must be verified. Designs that do not reflect the actual data environment are not likely to be useful at the end user level.

Given the ERDs in Figures PB.5B1 and PB.B2, you can see that each PRODUCT entry ***actually represents a product line, i.e., a collection of products belonging to the same product type or line, rather than a specific product occurrence with a unique serial number***. Therefore, this model will not enable us to identify the serial number for each component used in, for example, a product with serial number 348765. Therefore, this solution does not allow us to track the provider of a part that was used in a specific PRODUCT occurrence. (Note the example in Figure PB.5C.)

Figure PB.5C An Initial Implementation



PRODUCT
P1
P2
P3

PROD_COMP	
P1	C1
P1	C2
P2	C1
P2	C3
P2	C4
P3	C2
P3	C3

COMPONENT
C1
C2
C3
C4

COMP_VEND	
C1	V1
C1	V2
C1	V3
C2	V1
C2	V2
C2	V3
C3	V2
C4	V2
C4	V3

VENDOR
V1
V2
V3

As you examine Figure PB.5C, note that there are no serial numbers for the components, nor are there any for the products produced. In other words, we do not meet the requirements imposed by:

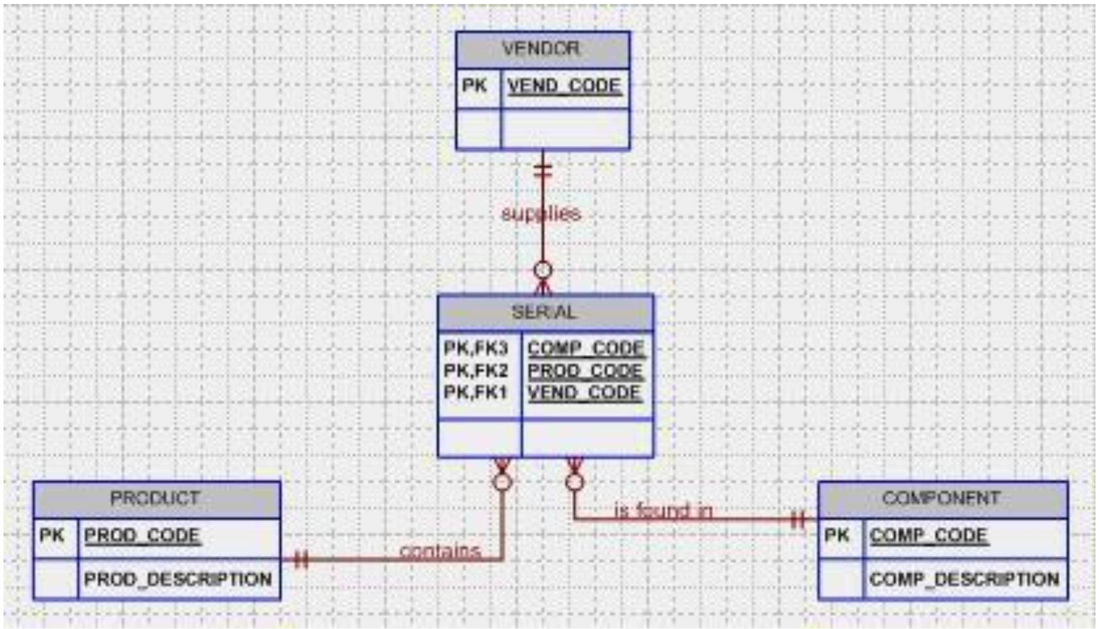
BUSINESS RULE 3

Each product has a unique serial number. For example, there will be several products P1, each with a unique serial number. Each unique product will be composed of several components, and each of those components has a unique serial number.

The implementation of business rule 3 will allow us to keep track of the supplier of each component.

One way to produce the tracking capability required by business rule 3 is to use a ternary relationship between PRODUCT, COMPONENT, and VENDOR, shown in Figure P5.5D1:

Figure PB.5D1 The Crow's Foot Ternary Relationship between PRODUCT, COMPONENT, and VENDOR



The ER diagram we have just shown represents a many-to-many-to-many TERNARY relationship, expressed by M:N:P. This ternary relationship indicates that:

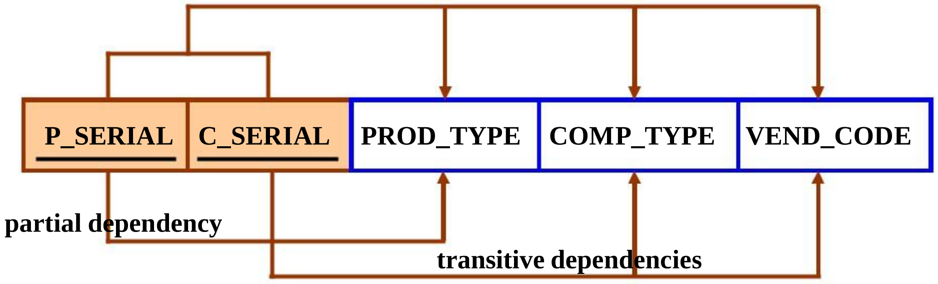
A product is composed of many components and a component appears in many products.

A component is provided by many vendors and a vendor provides many products.

A product contains components of many vendors and a vendor's components appear in many products.

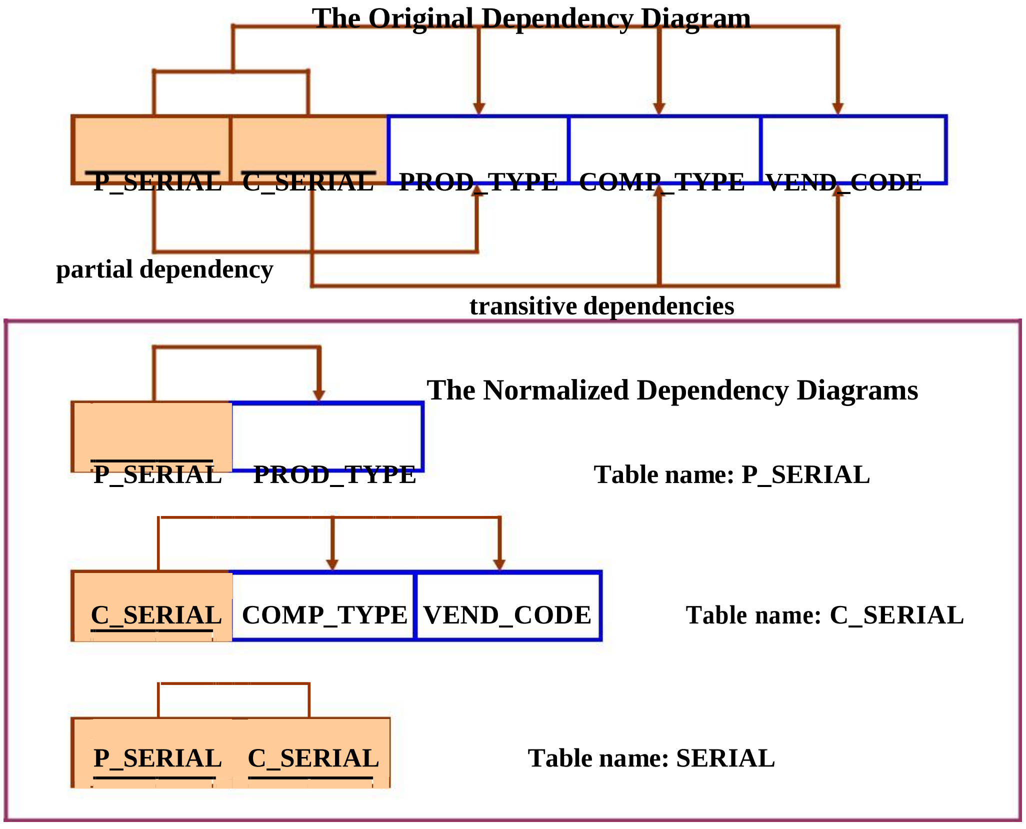
Assigning attributes to the SERIALS entity, we may draw the dependency diagram shown in Figure P7-5E.

Figure P7-5E The Initial Dependency Diagram



We may safely assume that all serial numbers are unique. If we make this assumption, we can conclude that the product serial number will identify the product type and that the component serial number will identify the component type and the vendor. Using the standard normalization procedures, we may thus decompose the entity as shown in the dependency diagrams in Figure PB.5F.

Figure PB.5F The Normalized Structure

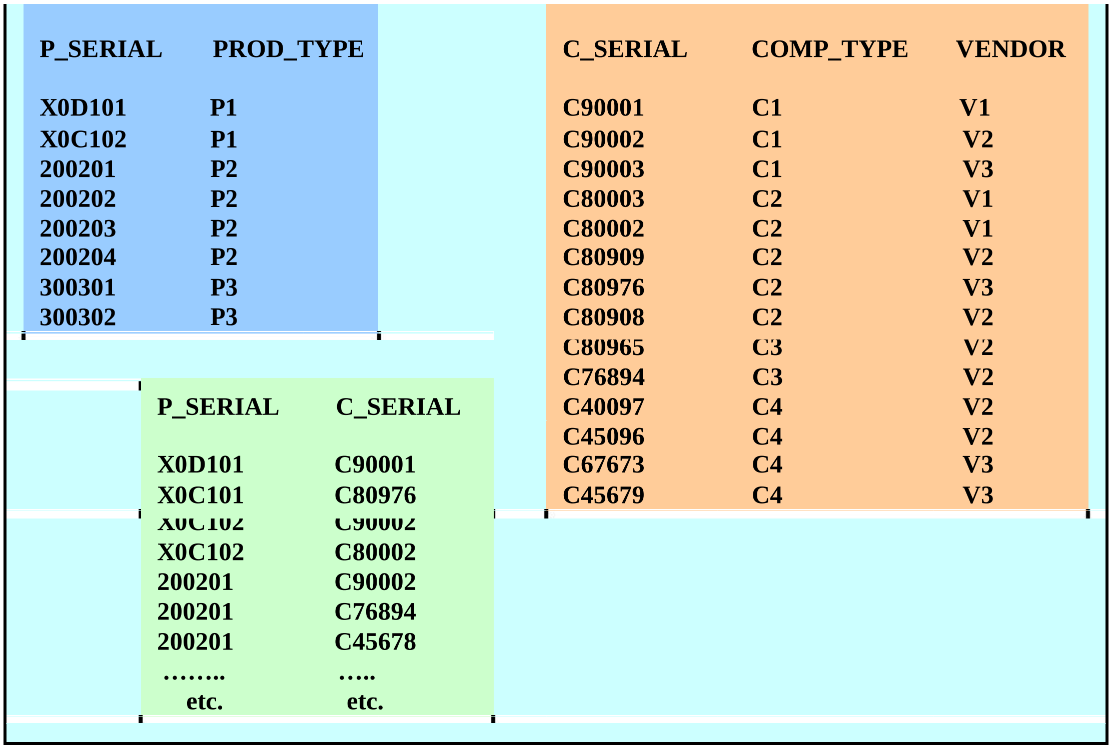


As you examine the dependency diagrams in Figure PB.5F, note the following:

P_SERIAL has a 1:M relationship with PRODUCT, because one product has many product serial numbers.
C_SERIAL has a 1:M relationship with COMPONENT, because one component has many component serial numbers.
SERIAL is the composite entity that connects P_SERIAL and C_SERIAL, thus reflecting the fact that one product has many components and a component can be found in many products.

To illustrate the relationships we have just described, let's take a look at some data in Figure P7-5G:

Figure PB.5G Sample Data



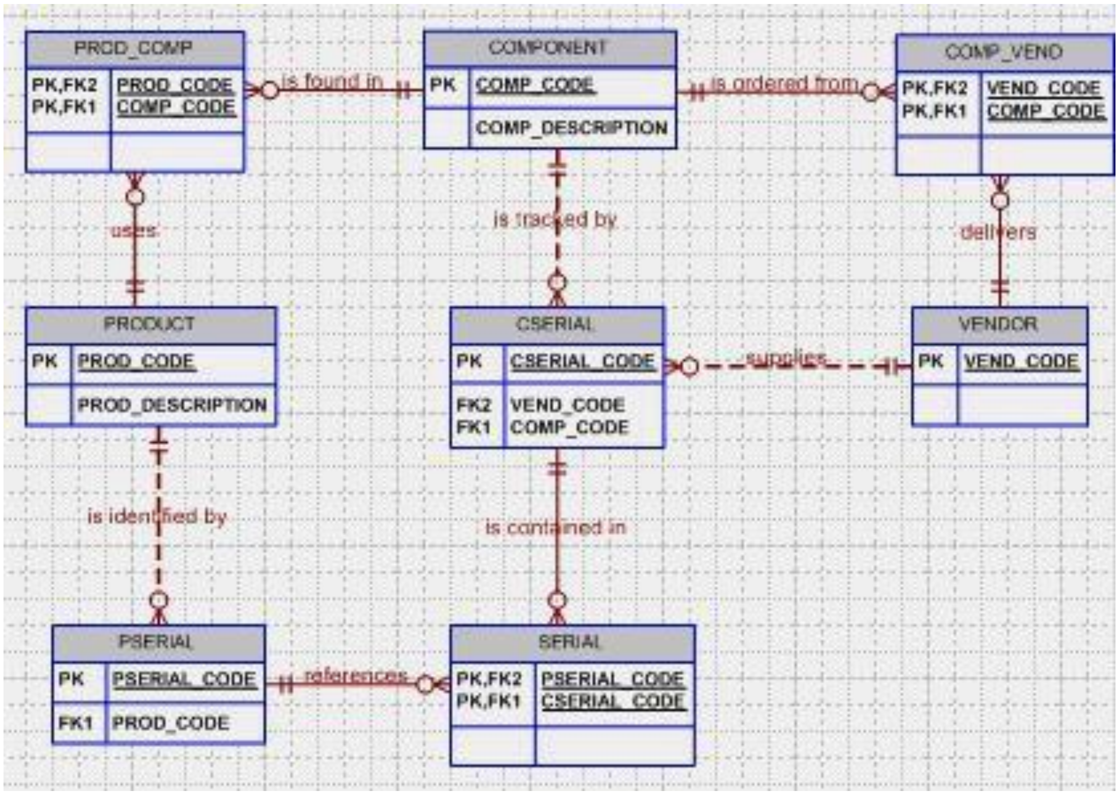
P_SERIAL	PROD_TYPE
X0D101	P1
X0C102	P1
200201	P2
200202	P2
200203	P2
200204	P2
300301	P3
300302	P3

C_SERIAL	COMP_TYPE	VENDOR
C90001	C1	V1
C90002	C1	V2
C90003	C1	V3
C80003	C2	V1
C80002	C2	V1
C80909	C2	V2
C80976	C2	V3
C80908	C2	V2
C80965	C3	V2
C76894	C3	V2
C40097	C4	V2
C45096	C4	V2
C67673	C4	V3
C45679	C4	V3

P_SERIAL	C_SERIAL
X0D101	C90001
X0C101	C80976
X0C102	C90002
X0C102	C80002
200201	C90002
200201	C76894
200201	C45678
……..	…..
etc.	etc.

The new ER diagram will enable us to identify the product by a unique serial number, and each of the product's components will have a unique serial number, too. Therefore, the new ER diagram will look like Figure PB.5H1.

Figure PB.5H1 The Revised (Final) Crow's Foot ERD



As you examine Figure PB.5H1's ERD, note that the COMP_VEND composite entity seems redundant, because the CSERIAL entity already depicts the many-to-many relationship between VENDOR and COMPONENT. However, COMP_VEND represents a more general relationship that enables us to determine who the *likely* providers of the general component are (what vendors supply component C1?), rather than letting us determine a specific component's vendor (which vendor supplied the component C1 with a serial number C90003?). The designer must confer with the end user to decide whether such a general relationship is necessary or if it can be removed from the database ***without affecting its semantic contents***.

6. **Create an ER diagram for a hardware store. Make sure that you cover (at least) store transactions, inventory, and personnel. Base your ER diagram on an appropriate set of business rules that you develop. (*Note:* It would be useful to visit a hardware store and conduct interviews to discover the type and extent of the store's operations.)**

Since the problem does not specify a set of business rules, we will create some that will enable us to develop an initial ER diagram.

NOTE

Please take into consideration that, depending on the assumptions made and on the selection of business rules, students are likely to create quite different solutions to this problem. You may find it quite useful to study each student solution and to incorporate the most interesting parts of each solution into a common ER diagram. We know that this is not an easy job, but your students will benefit because you will thus enable them to develop very important analytical skills. You should stress that:

A problem may be examined from many different angles.

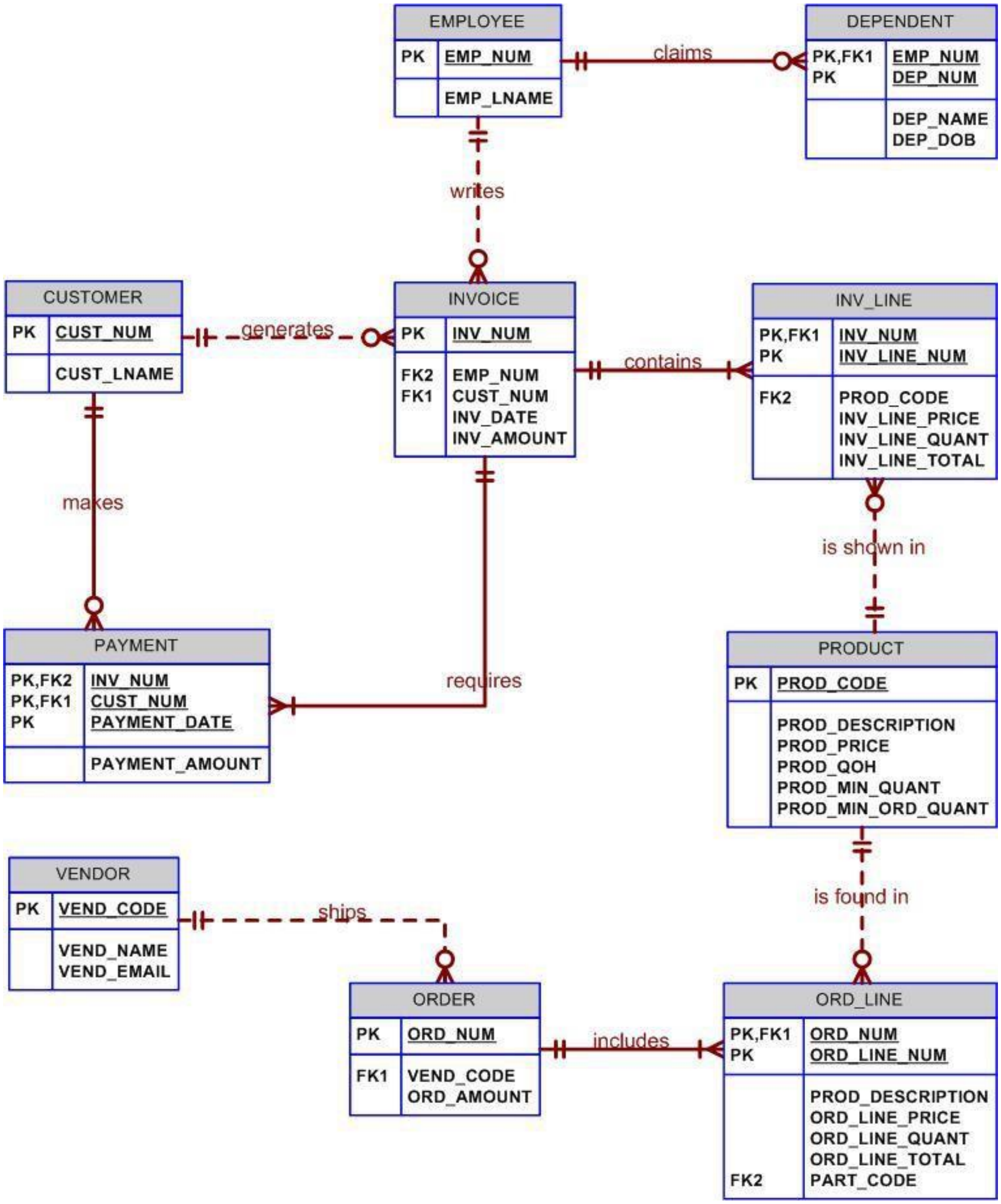
Similar organizations, using different business rules, will generate design problems that may be solved through the use of quite different solutions.

To get the class discussion started, we will assume these business rules:

1. A product is provided by many suppliers, and a supplier can provide several products.
2. An employee has many dependents, but a dependent can be claimed by only one employee.
3. An employee can write many invoices, but each invoice is written by only one employee.
4. Each invoice belongs to only one customer, and each customer owns many invoices.
5. A customer makes several payments, and each payment belongs to only one customer.
6. Each payment may be applied partially or totally to one or more invoices, and each invoice can be paid off in one or more payments.

Using these business rules, we may generate the ERD shown in Figure PB.6A.

Figure PB.6A The Crow's Foot ERD for Problem 6 (The Hardware Store)



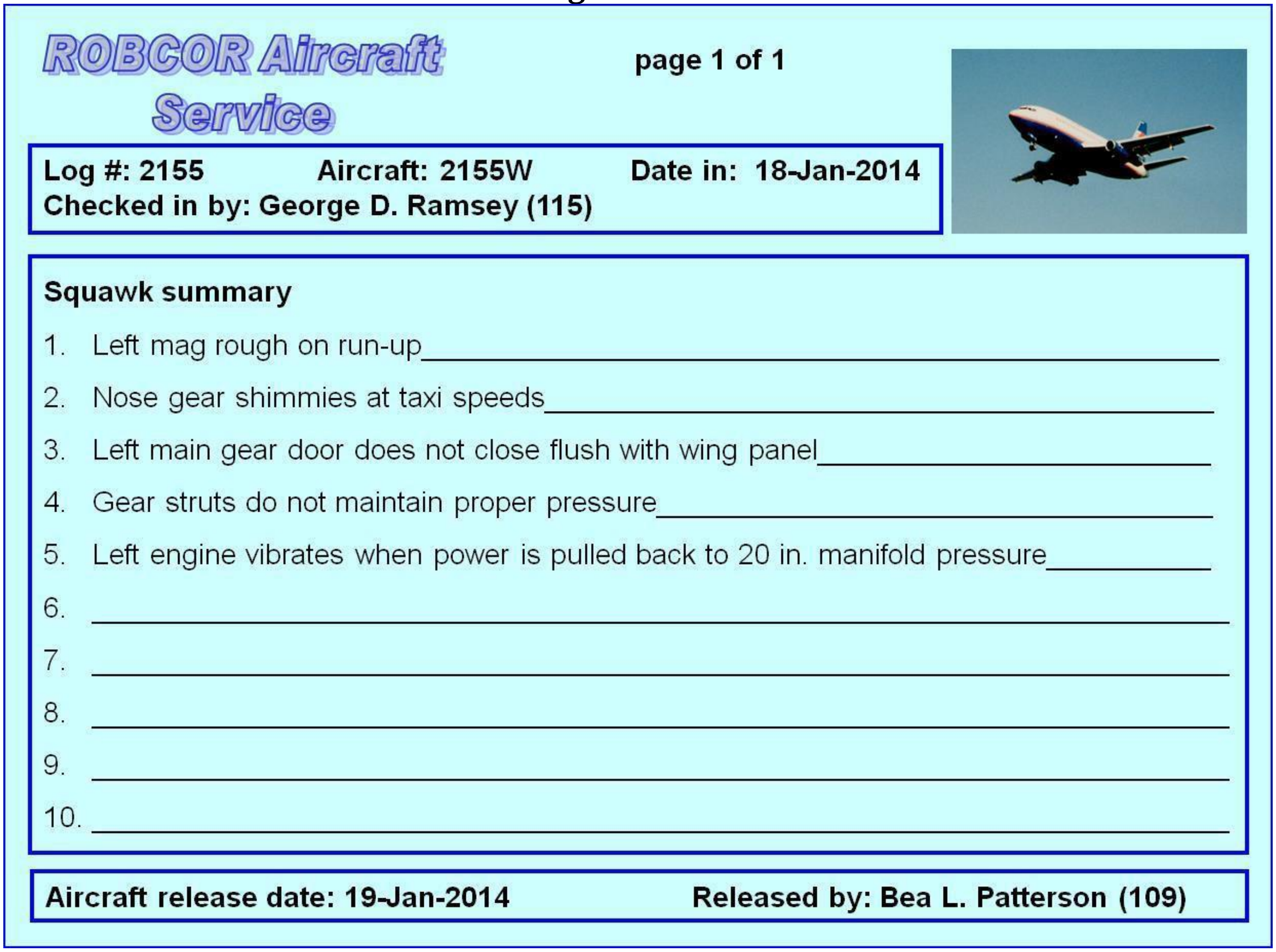
The ERD shown in Figure PB.6A requires less tweaking than the previous ERDs to get it ready for implementation. For example, given the presence of the INV_LINE entity, the customer can buy more than one product per invoice. Similarly, the ORD_LINE entity makes it possible for more than one product to be ordered per order.

However, as you examine the PAYMENT entity in Figure PB.6A, note that the current PK definition limits the payments for a given customer and invoice number to one per day. (Two payments by the same customer for the same invoice number on the same date would violate the entity integrity rules, because the two composite PK values would be identical in that scenario.) Therefore, the design shown in Figure PB.6A still requires additional work, to be completed during the verification process.

7. Use the following brief description of operations as the source for the next database design:

All aircraft owned by ROBCOR require periodic maintenance. When maintenance is required, a maintenance log form is used to enter the aircraft identification number, the general nature of the maintenance, and the maintenance starting date. A sample maintenance log form is shown in Figure PB.7A.

FIGURE PB.7A The Maintenance Log Form



ROBCOR Aircraft Service

page 1 of 1

Log #: 2155 Aircraft: 2155W Date in: 18-Jan-2014
Checked in by: George D. Ramsey (115)

Squawk summary

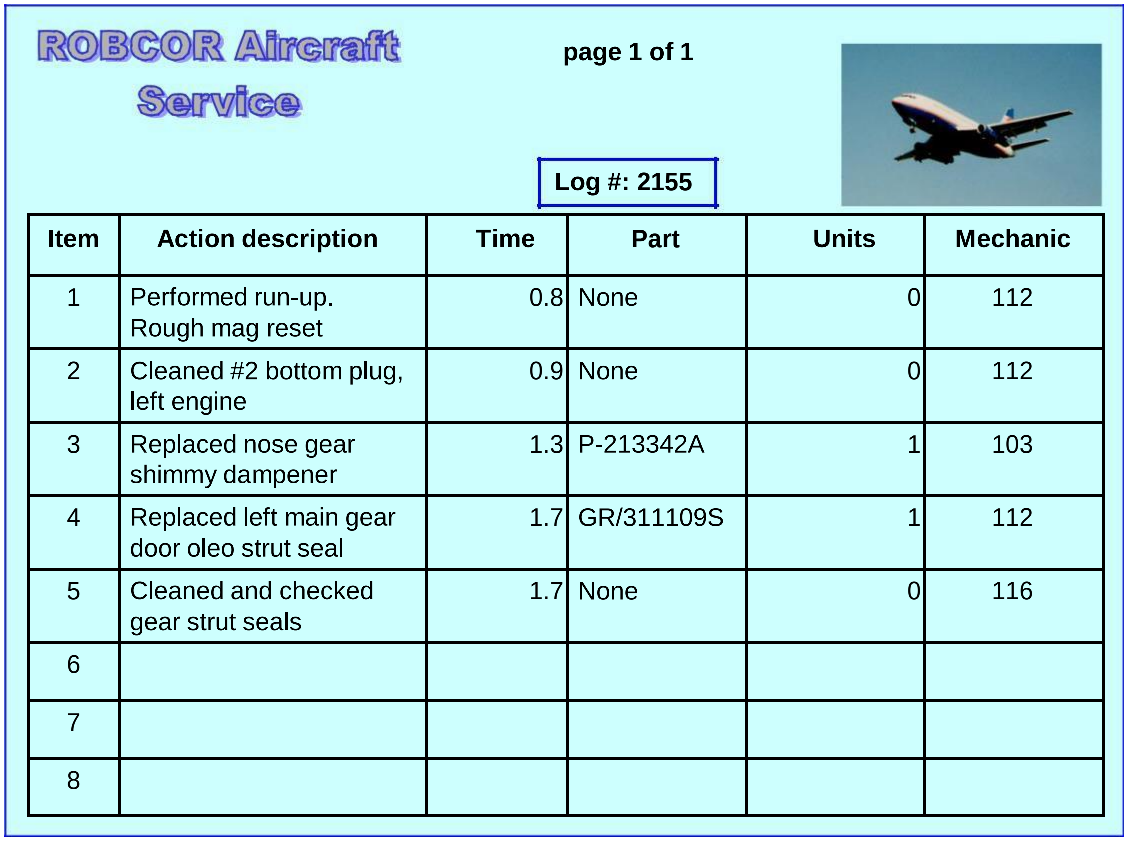
1. Left mag rough on run-up______________________________
2. Nose gear shimmies at taxi speeds__________________________
3. Left main gear door does not close flush with wing panel_______________
4. Gear struts do not maintain proper pressure______________________
5. Left engine vibrates when power is pulled back to 20 in. manifold pressure______
6. ____
7. ____
8. ____
9. ____
10. ____

Aircraft release date: 19-Jan-2014 Released by: Bea L. Patterson (109)

Note that the maintenance log form shown in Figure PB.7A contains a space used to enter the maintenance completion date and a signature space for the supervising mechanic who releases the aircraft back into service. Each maintenance log form is numbered sequentially. *Note*: A supervising mechanic is one who holds a special Federal Aviation Administration (FAA) Inspection Authorization (IA). Three of ROBCOR's ten mechanics hold such an IA.

Once the maintenance log form is initiated, the maintenance log form's number is written on a *maintenance specification sheet*, also known as a *maintenance line form*. When completed, the specification sheet contains the details of each maintenance action, the time required to complete the maintenance, parts (if any) used in the maintenance action, and the identification of the mechanic who performed the maintenance action. The maintenance specification sheet is the billing source (time and parts for each of the maintenance actions), and it is one of the sources through which parts use may be audited. A sample maintenance specification sheet (line form) is shown in Figure PB.7B.

FIGURE PB.7B The Maintenance Line Form



ROBCOR Aircraft Service

page 1 of 1

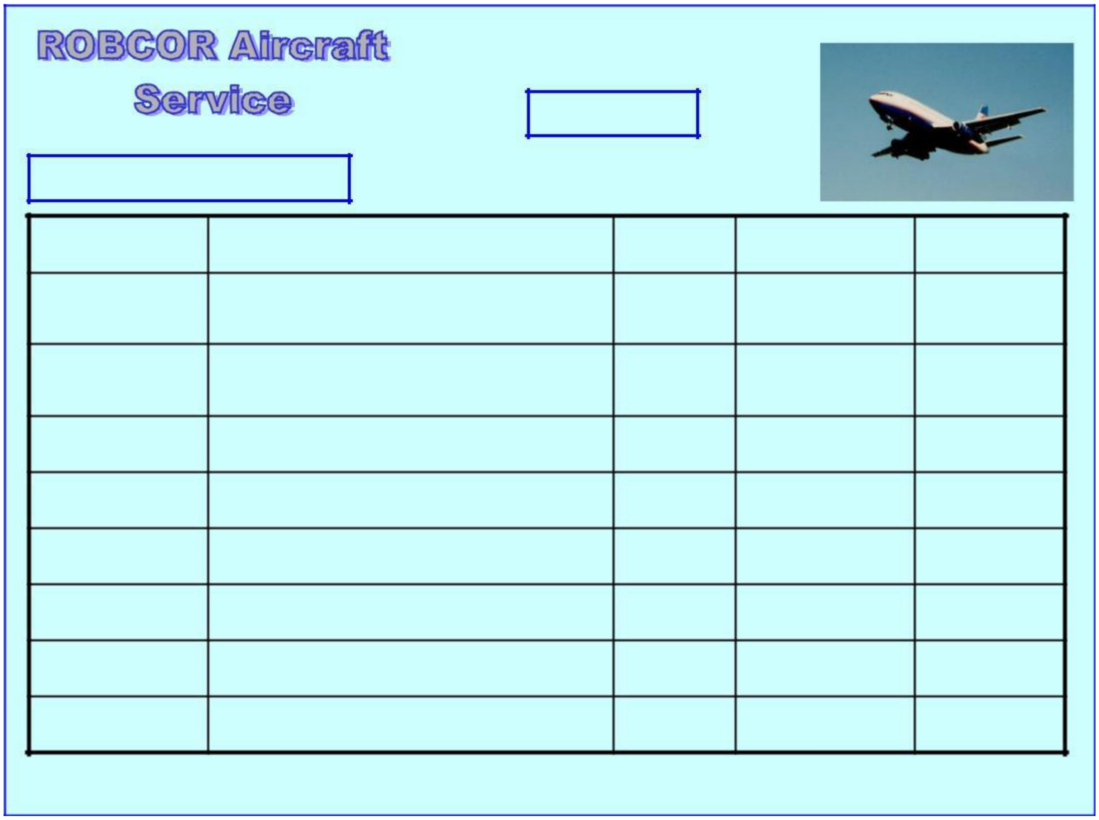
Log #: 2155

Item	Action description	Time	Part	Units	Mechanic
1	Performed run-up. Rough mag reset	0.8	None	0	112
2	Cleaned #2 bottom plug, left engine	0.9	None	0	112
3	Replaced nose gear shimmy dampener	1.3	P-213342A	1	103
4	Replaced left main gear door oleo strut seal	1.7	GR/311109S	1	112
5	Cleaned and checked gear strut seals	1.7	None	0	116
6					
7					
8					

Parts used in any maintenance action must be signed out by the mechanic who used them, thus allowing ROBCOR to track its parts inventory. Each sign-out form contains a listing of all the parts associated with a given maintenance log entry. Therefore, a parts sign-out form contains the maintenance log number against which the parts are charged. In addition, the parts sign-out procedure is used to update the ROBCOR parts inventory. A sample parts sign-out form is shown

in Figure PB.7C.

FIGURE PB.7C The Parts Sign-out Form



ROBCOR Aircraft
Service

Mechanics are highly specialized ROBCOR employees, and their qualifications are quite different from those of an accountant or a secretary, for example.

Given this brief description of operations, draw the fully labeled ER diagram. Make sure you include all the appropriate relationships, connectivities, and cardinalities.

Before drawing the ER diagram, note the following relationships:

Not all employees are mechanics, but all mechanics are employees. Therefore, the MECHANIC entity is optional to EMPLOYEE. The EMPLOYEE is the supertype to MECHANIC.

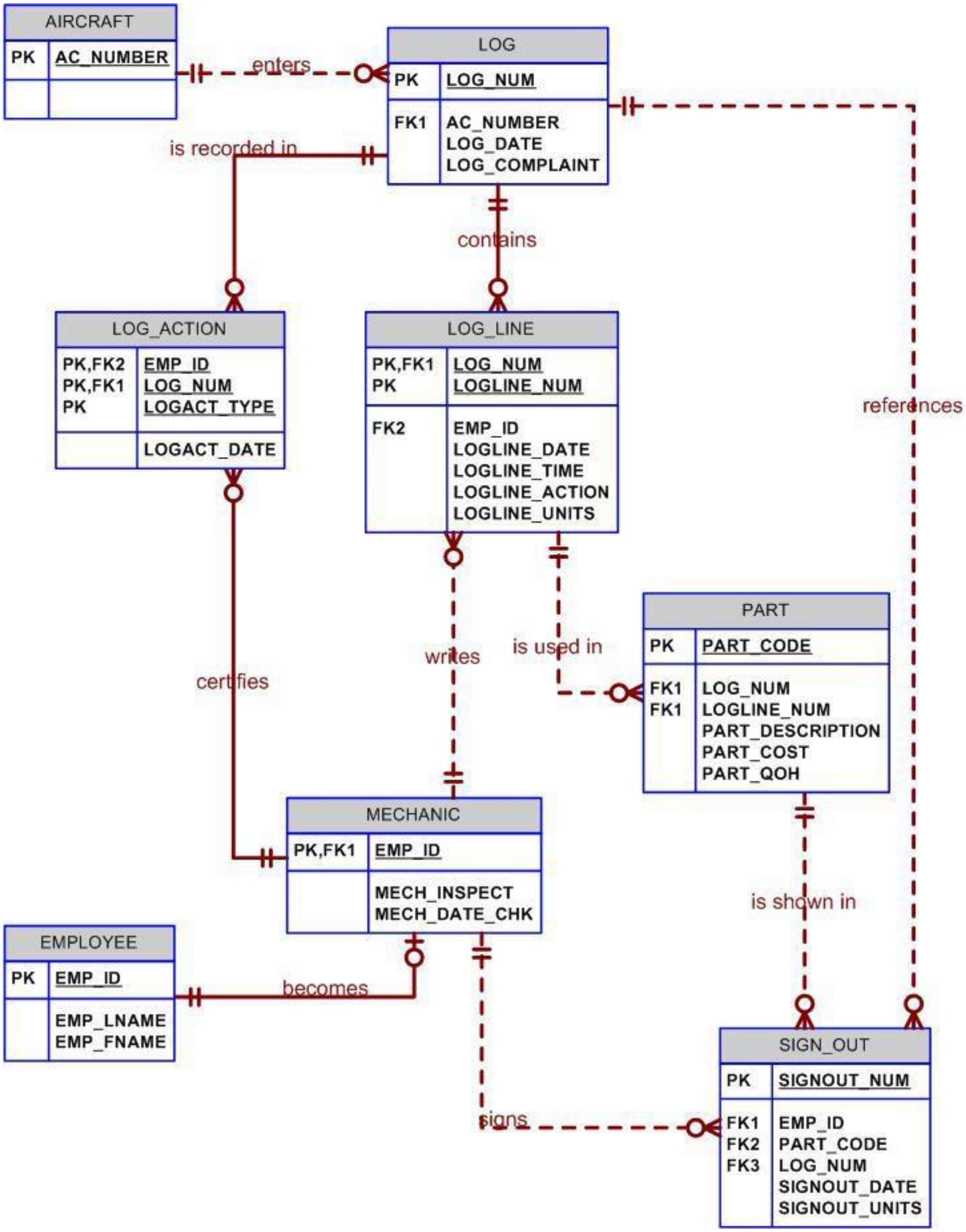
All mechanics must sign off work on the MAINTENANCE they performed and they must sign out for the PART(s) used.

Only some mechanics (the IAs) may sign off the LOG. Therefore, LOG is optional to MECHANIC.

Because not all MAINTENANCE entries are associated with a PART --- some maintenance doesn't require parts --- PART is optional to MAINTENANCE.

These relationships are all reflected in the ER diagrams shown in Figure PB.7.

Figure PB.7D1 The Initial Crow's Foot ERD for Problem 7 (ROBCOR Aircraft Service)



As you discuss the ERD shown in Figure PB.7D1, note its similarity to the car dealership's maintenance section of the ERD presented in Figure PB.3a. However, the ROBCOR Aircraft Service ERD has been developed at a much higher detail level, thus requiring fewer modifications during the verification process. Figure PB.7D1 shows that:

Each LOG entity occurrence will yield one *or more* maintenance procedures.

Each of the individual maintenance procedures will be listed in the LOG_LINE entity. A mechanic must sign off on each of the LOG_LINE entity occurrences.

The possible parts use in each LOG_LINE entity occurrence is now traceable.

A part can be accounted for from the moment it is signed out by the mechanic to the point at which it is installed during the maintenance procedure.

The "references" relationship between LOG and PART is subject to discussion. After all, you can always trace each part's use to the LOG through the LOG_LINE entity. Therefore, the relationship is redundant. Such redundancies are – or should be – picked up during the verification process.

We have shown the MECHANIC to be a subtype of the EMPLOYEE supertype. Whether the supertype/subtype relationship makes sense depends on the type and extent of the attributes that are to be associated with the MECHANIC entity. There may be externally imposed requirements – often imposed through the government's regulatory process -- that can best be met through a supertype/subtype relationship. However, in the absence of such externally imposed requirements, it is usually better to use an attribute in EMPLOYEE – such as the employee's primary job code – and link the employees to their various qualifications through a composite entity. The applications software will then be used to enforce the requirement that the person doing maintenance work is, in fact, a mechanic.

8. **You have just been employed by the ROBCOR Trucking Company to develop a database. To gain a sense of the database's intended functions, you have spent some time talking to ROBCOR's employees and you've examined some of the forms used to track driver assignments and truck maintenance. Your notes include the following observations:**

Some drivers are qualified to drive more than one type of truck operated by ROBCOR. A driver may, therefore, be assigned to drive more than one truck type during some period of time. ROBCOR operates several trucks of a given type. For example, ROBCOR operates two panel trucks, four half-ton pick-up trucks, two single-axle dump trucks, one double-axle truck, and one 16-wheel truck. A driver with a chauffeur's license is qualified to drive only a panel truck and a half-ton pick-up truck and, thus, may be assigned to drive any one of six trucks. A driver with a commercial license with an appropriate heavy equipment endorsement may be assigned to drive any of the nine trucks in the ROBCOR fleet. Each time a driver is assigned to drive a truck, an entry is made in a log containing the employee number, the truck identification, and the sign-out (departure) date. Upon the driver's return, the log is updated to include the sign-in (return) date and the number of driver duty hours.

If trucks require maintenance, a maintenance log is filled out. The maintenance log includes the date on which the truck was received by the maintenance crew. The truck cannot be released for service until the

maintenance log release date has been entered and the log has been signed off by an inspector.

All inspectors are qualified mechanics, but not all mechanics are qualified inspectors.

Once the maintenance log entry has been made, the maintenance log number is transferred to a service log in which all service log transactions are entered. A single maintenance log entry can give rise to multiple service log entries. For example, a truck might need an oil change as well as a fuel injector replacement, a brake adjustment, and a fender repair.

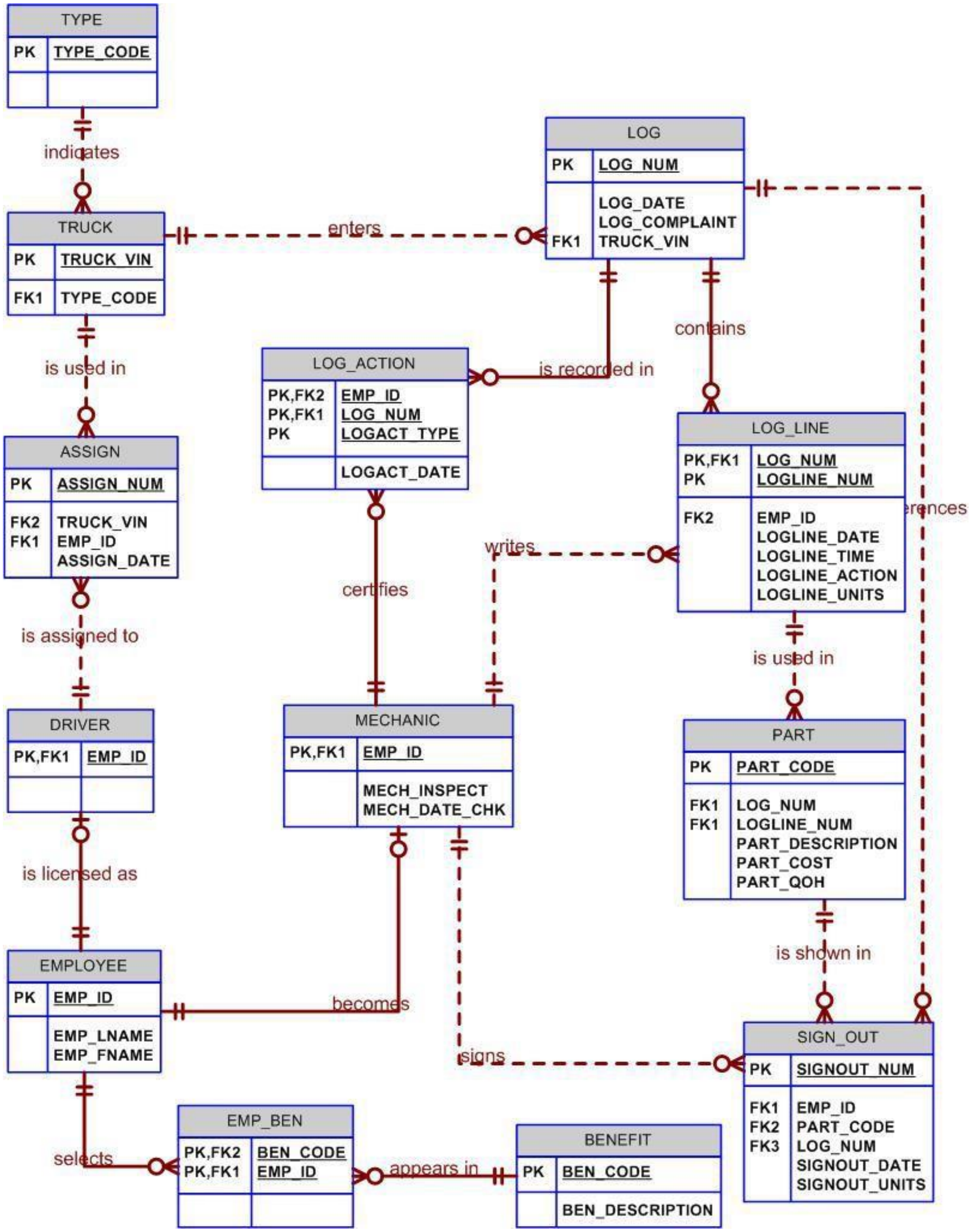
Each service log entry is signed off by the mechanic who performed the work. To track the maintenance costs for each truck, the service log entries include the parts used and the time spent to install the part or to perform the service. (Not all service transactions involve parts. For example, adjusting a throttle linkage does not require the use of a part.)

All employees are automatically covered by a standard health insurance policy. However, ROBCOR's benefits include optional co-paid term life insurance and disability insurance. Employees may select both options, one option, or no options.

Given those brief notes, create the ER diagram. Make sure you include all appropriate entities and relationships, and define all connectivities and cardinalities.

The ERD in Figure PB.8a contains a maintenance portion that has become our standard, given that it enables the end user to track all activities and parts for all vehicles. In fact, given its ability to support high accountability standards, we first developed the "basics" of this design for aviation maintenance tracking.

Figure PB.8a The Initial Crow's Foot ERD for the ROBCOR Trucking Service



As you examine the ERD in Figure PB.8a, note that the driver assignment to drive trucks is a M:N relationship: Given the passage of time, a driver can be assigned to drive a truck many times and a truck can be assigned to a driver many times. We have implemented this relationship through the use of a composite entity named ASSIGN.

The M:N relationship between EMPLOYEE and BENEFIT – that is, the insurance package mentioned in problem 8's last bullet -- has been implemented through the composite entity named EMP_BEN. (An employee can select many benefit packages and each insurance package may be selected by many employees.) The reason for the optionality is based on the fact that not all of the insurance packages are necessarily selected by the employee. For example, using the BENEFIT table contents shown in Table PB.8A, an employee may decide to select option 2 or options 2 and 3, or neither option. (The standard health insurance package is assigned automatically.)

Table PB.8A Table name: BENEFIT

BEN_CODE	BEN_DESCRIPTION	BEN_CHARGE
1	Standard health	$0.00
2	Co-paid term life insurance, $100,000	$35.00
3	Co-paid disability insurance	$42.50

Incidentally, we have used a BENEFIT entity, rather than an INSURANCE entity to anticipate the likelihood that benefits may include items other than insurance. For example, employees might be given a benefit such as an investment plan, a flextime option, child care, and so on.

The decomposition of M:N relationships continues to be a good subject for discussion. For example, we have shown many of the decompositions as composite entities. However, while such an approach is perfectly acceptable at the initial design stage, caution your students that composite PKs cannot be referenced easily by subsequent additions of entities that must reference those PKs. Therefore, we would note that the composite PK used in the LOG_ACTION entity -- EMP_ID + LOG_NUM + LOGACT_TYPE – should be replaced by an "artificial" single-attribute PK named LOGACT_NUM. The EMP_ID and LOG_NUM attributes would continue to be used as FKs to the MECHANIC and LOG entities. (Naturally, the EMP_ID and LOG_NUM attributes should be indexed to avoid duplication of records and to speed up queries.) A few sample entries are shown in Table

Table PB.8B Table name: LOG_ACTION

LOGACT_NUM	LOG_NUM	EMP_ID	LOGACT_TYPE	LOGACT_DATE
1000	5023	409	Open	14-May-2014
1001	5024	409	Open	15-May-2014
1002	5023	411	Close	15-May-2014
1003	5025	378	Open	15-May-2014
1004	5024	411	Close	15-May-2014
1005	5026	409	Open	16-May-2014

Finally, we have used supertype/subtype relationships between EMPLOYEE and DRIVER and MECHANIC. If drivers and mechanics are assumed to have many characteristics (such as special certifications at different levels) that are not common to EMPLOYEE, this approach eliminates nulls. However, keep in mind the discussion about the use of supertypes/subtypes in Problem B. (The use of the supertype/subtype approach may be dictated by external factors … but the use of supertypes and subtypes must be approached with some caution. For example, if drivers have multiple license types, it would be far better to create a LICENSE entity and relate it to DRIVER through a composite entity, perhaps named DRIVER_LICENSE. The composite entity may then be designed to include the date on which the license was earned and other pertinent facts pertaining to licenses. (Such flexibility is not available in a subtype, unless you are willing to tolerate the possible occurrence of nulls as more pertinent data about the (multiple) licenses are kept – if some of the drivers do not have all of those licenses.)