Solution Manual for Database Systems Design Implementation and Management 11th Edition by Coronel and Morris ISBN 1285196147 9781285196145

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 Q2.7. (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 Q2.7 The Chen and Crow’s Foot ERDs for Question 7

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)?

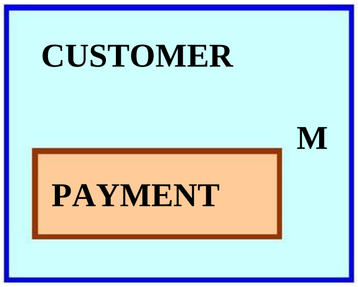
An object is an instance of a specific class. It is useful to point out that the object is a run-time concept, while the class is a more static description.

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 7's ERD is shown in Figure Q1.10:

Figure Q2.10 The OODM Model for Question 10



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.

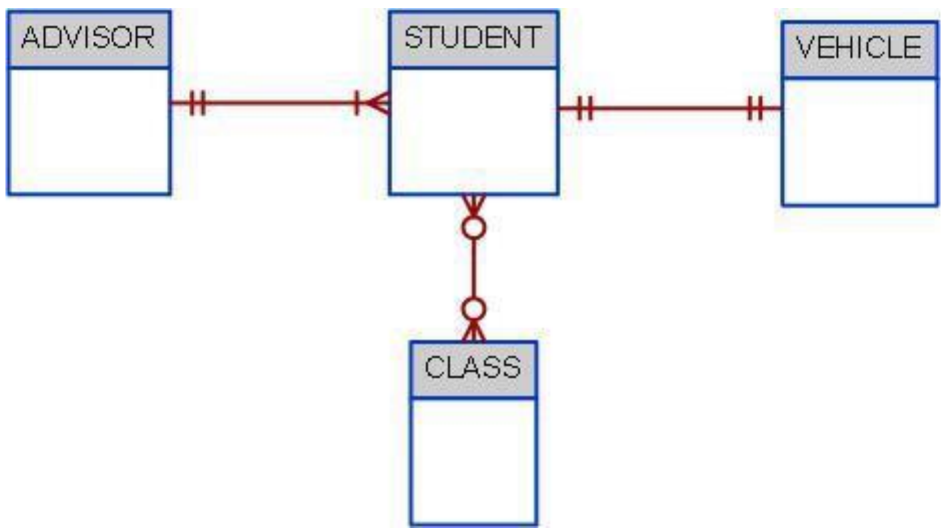
Tables are especially useful from the modeling and implementation perspecectives. Because tables are used to describe the entities they represent, they provide ane asy way to summarize entity characteristics and relationships among entities. And, because they are purely conceptual constructs, the designer 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.

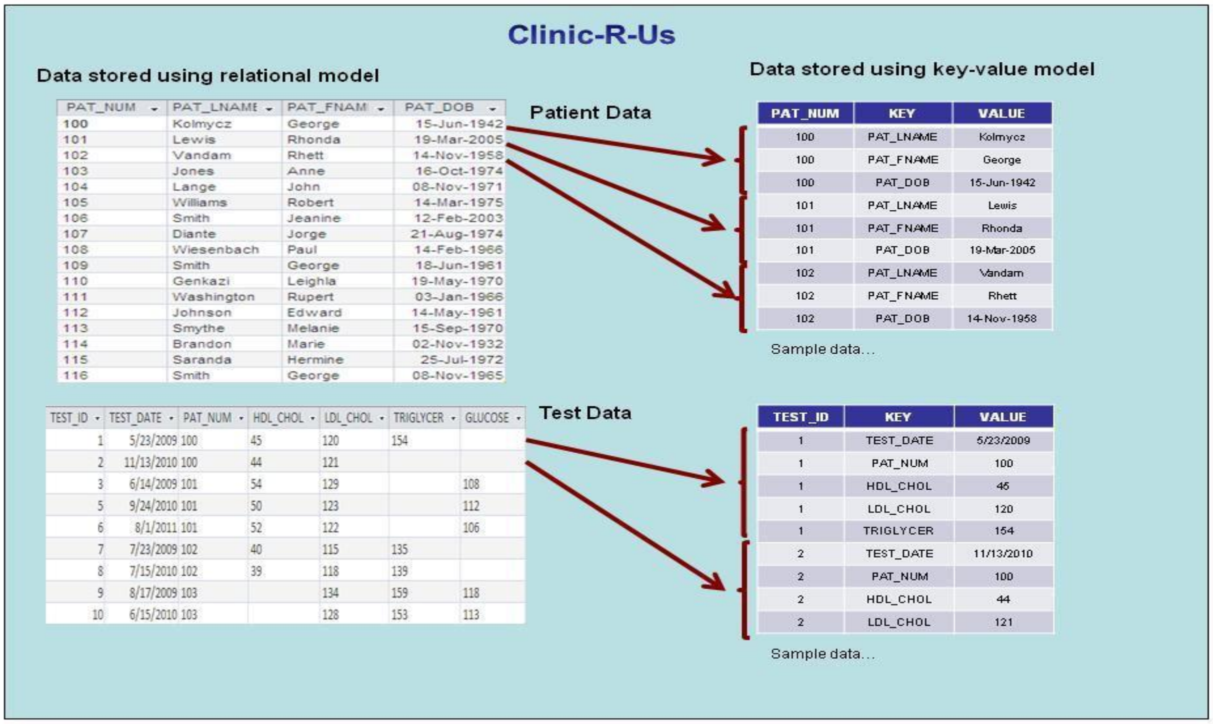
Because NoSQL databases use the key-value data model, they are suited to handle very high volumes of sparse data; that is for cases where the number of attributes is very large but the number of actual data instances is low.

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.



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.

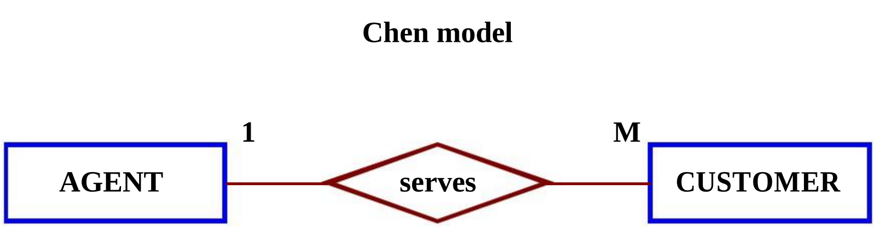
2. Given the business rule(s) you wrote in Problem 1, create the basic Crow's Foot ERD.

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.

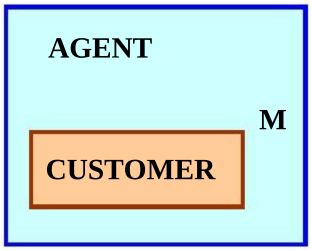
Figure P2.2b The Chen ERD for Problem 2



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 P2.3 The OO 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.

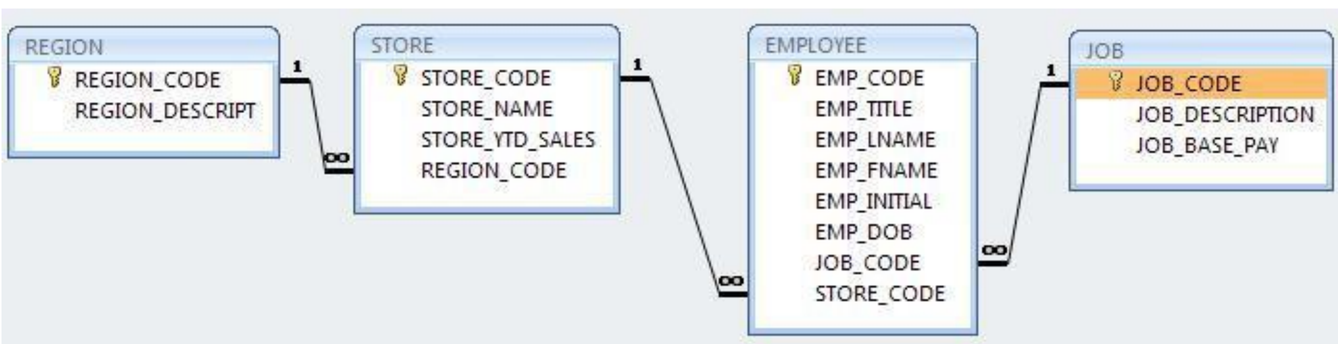


Figure P2.4 The DealCo relational diagram

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.

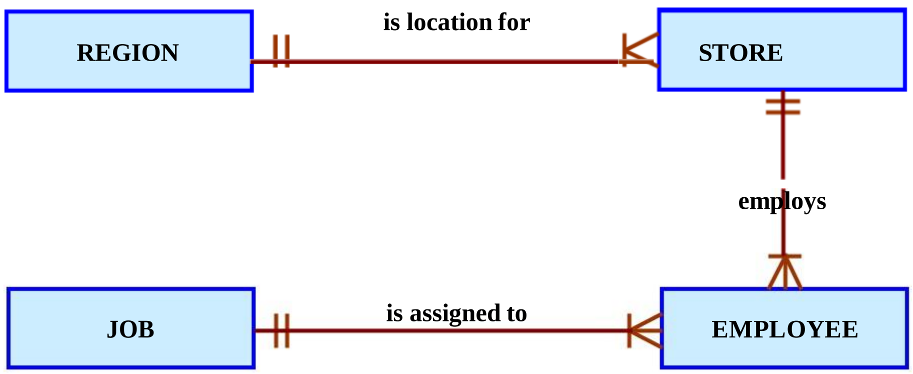
Each store employs one or more employees. Each employee is employed by one store. (In this case, we are assuming that the business rule specifies that an employee cannot work in more than one store at a time.) Therefore, the relationship between STORE and EMPLOYEE is 1:M.

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.

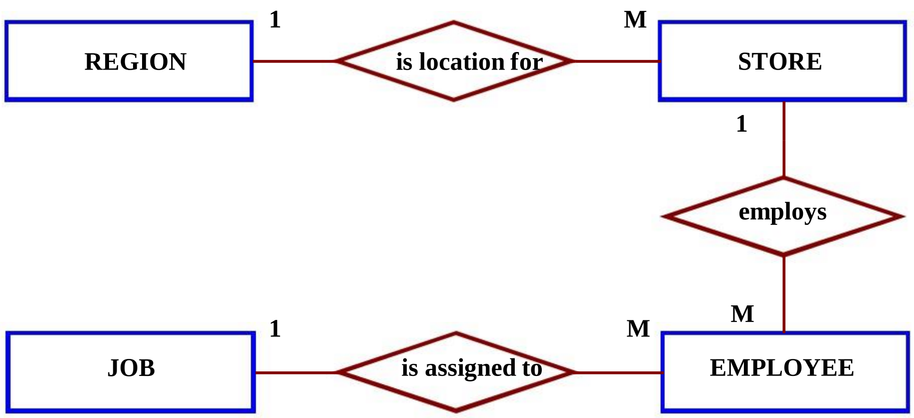
The Crow's Foot ERD is shown in Figure P2.5a.

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.

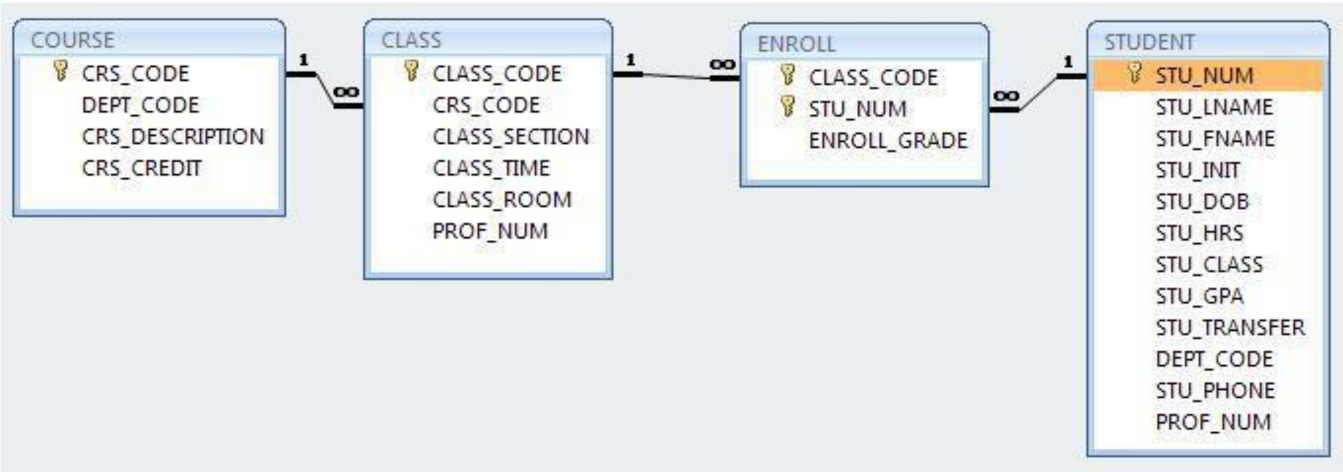


Figure P2.6 The Tiny College relational diagram

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.

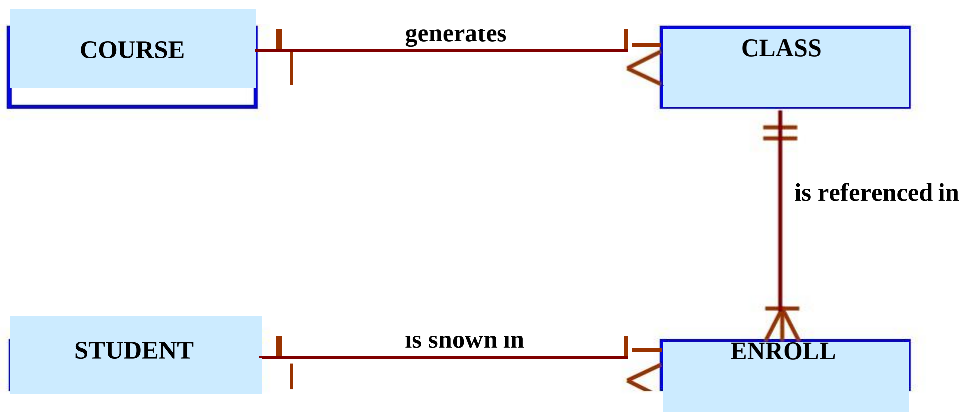
CLASS is referenced in ENROLL. One class can be referenced in enrollment many times. Each individual enrollment references one class. Note that the ENROLL entity is also related to STUDENT. Each entry in the ENROLL entity references one student and the class for which that student has enrolled. A student 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.

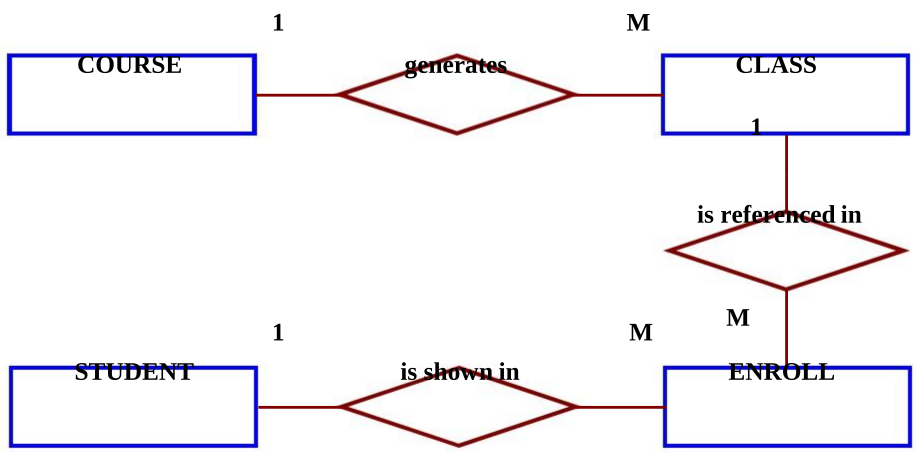
The Crow's Foot model is shown in Figure P2.7a.

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



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

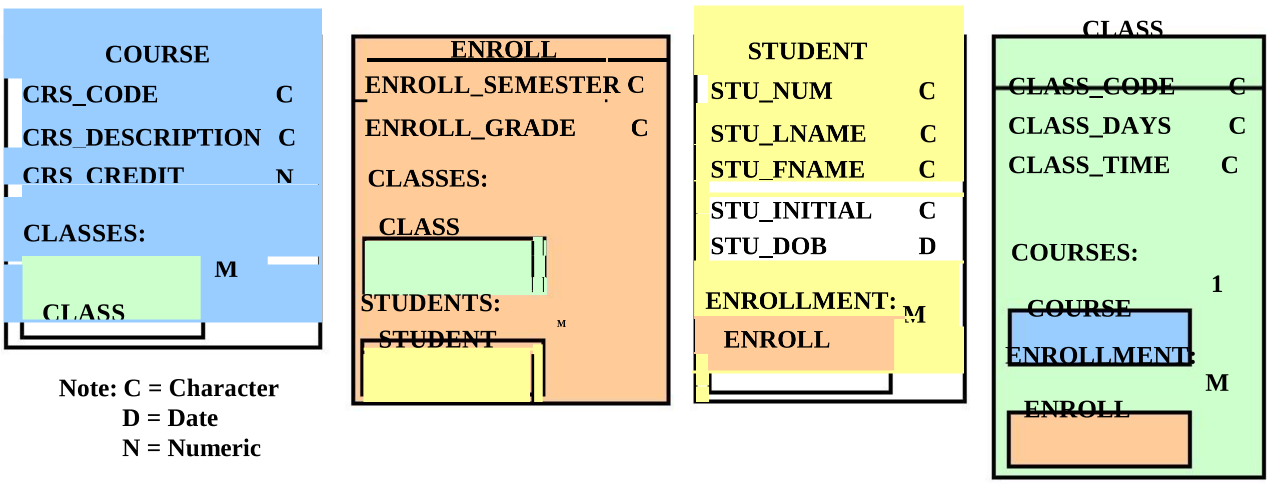
Figure P2.7b The Chen Model for Tiny College



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 P2.8 The OO 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.)

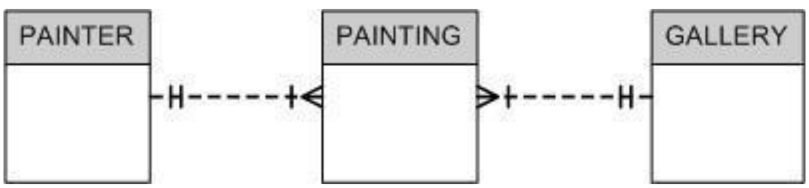
Each painting is exhibited in only one gallery.

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 Diagram



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 OODM solution is shown in Figure P2.12.

FIGURE P2.12 The OODM for Problem 12

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

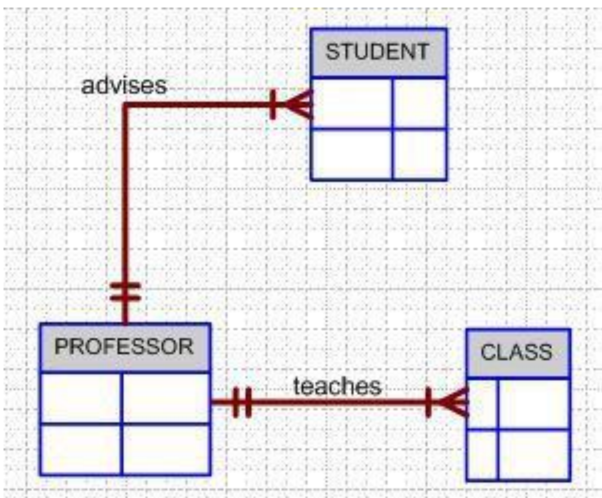


Figure P2.13 The Crow's Foot ERD for Problem 13

The business rules may be written as follows:

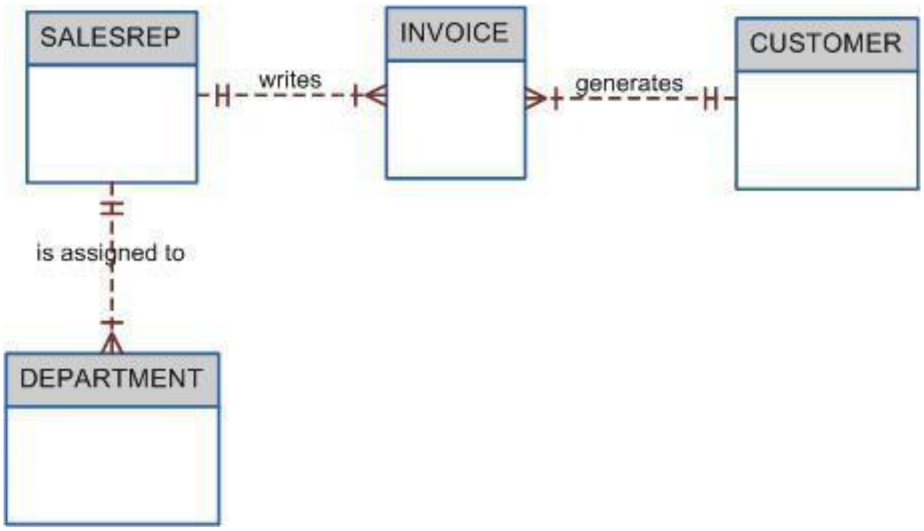
A professor can teach many classes.
Each class is taught by one professor.
A professor can advise many students.
Each student is advised by one professor.

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.**

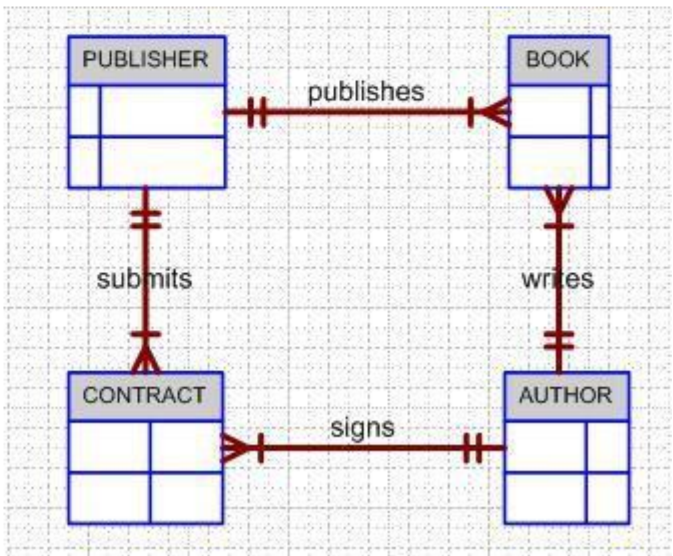
The Crow's Foot ERD is shown in Figure P2.23. Note that a 1:M relationship is always read from the one (1) to the many (M) side. Therefore, the customer-invoice relationship is read as "one customer generates many invoices."

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.

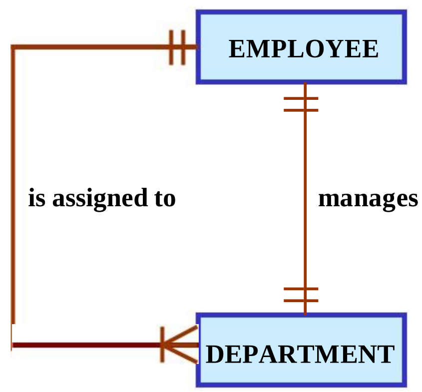
This ERD will be a good basis for a discussion about what happens when more realistic assumptions are made. For example, a book – such as this one – may be written by more than one author. Therefore, a contract may be signed by more than one author. Your students will learn how to model such relationships 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 rent many videotapes from the BigVid store. Each of the BigVid's videotapes can be rented to many customers during that period of time.

The solution is presented in Figure P2.16b. Note the M:N relationship between CUSTOMER and VIDEO. 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 RENTAL entity to create a 1:M relationship between CUSTOMER and RENTAL and a 1:M relationship between VIDEO and RENTAL. (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

Initial M:N Solution

Implementable Solution

We have created a small **Ch02_Airline** database to let you explore the implementation of the model. (Check your Instructor's CD.) The tables and the relational diagram are shown in the following two figures.

FIGURE P2.16c The Airline Database Tables

Table name: AIRCRAFT   Database name: Ch02_Airline

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

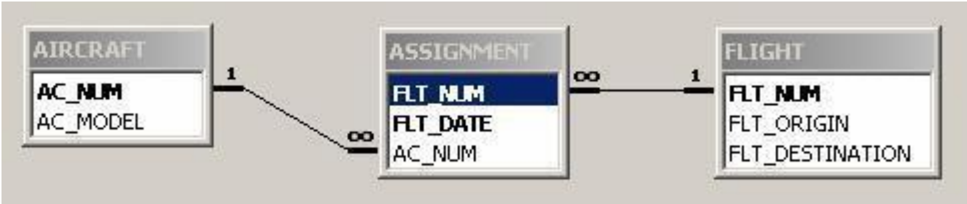
Table name: ASSIGNMENT

FLT_NUM	FLT_DATE	AC_NUM
101	14-Jan-08	375G
101	15-Jan-08	375G
101	16-Jan-08	411H
101	17-Jan-08	375G
101	18-Jan-08	123U
102	14-Jan-08	375G
102	15-Jan-08	375G
102	16-Jan-08	411H
102	17-Jan-08	375G
102	18-Jan-08	123U

Table name: FLIGHT

FLT_NUM	FLT_ORIGIN	FLT_DESTINATION
101	MEM	ATL
102	ATL	MEM

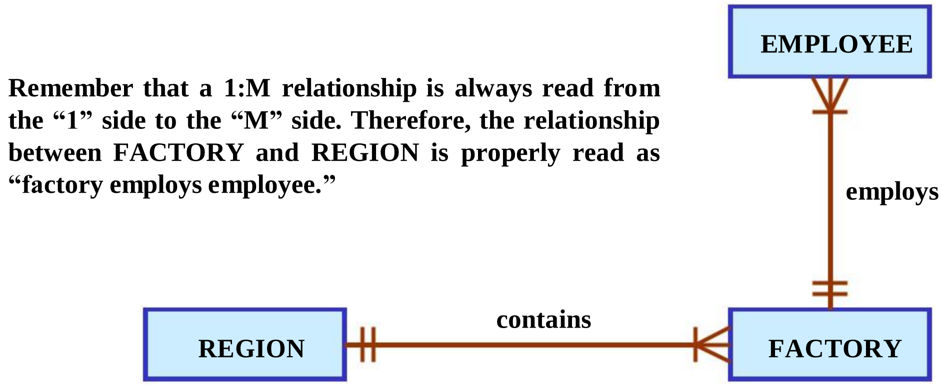
FIGURE P2.16c The Airline Relational Diagram



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



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.