Designing Databases

C

Introduction

Businesses rely on databases for accurate, up-to-date information. Without access to mission critical data, most businesses are unable to perform their normal daily functions, much less create queries and reports that help make strategic decisions. For those decisions to be useful, the database must have data that are accurate, complete, consistent, timely, and unique. However, without a good underlying database design, decisions will be inaccurate and inconstant.

A ***database*** maintains information about various types of objects (inventory), events (transactions), people (employees), and places (warehouses). A ***database management system (DBMS)*** creates, reads, updates, and deletes data in a database while controlling access and security. A DBMS provides a way to create, update, delete, store, and retrieve data in the database.

Using a data model offers a method for designing a database correctly that will help in meeting the needs of the users in a DBMS environment.

LEARNING OUTCOMES

C.1 Describe the purpose of the relational database model in a database management system.

C.2 List the relational database model's basic components.

C.3 Describe why entities and attributes are organized into tables and fields.

C.4 Describe how data redundancy is handled in the relational database model.

C.5 Explain the need for an entity-relationship diagram in a database management system.

C.6 Describe the Chen model symbols used in entity-relationship modeling.

The Relational Database Model

Numerous elements in a business environment need to store data, and those elements are related to one another in a variety of ways. In fact, a database must contain not only the data but also information about the relationships between those data. Designing a database properly is fundamental to establishing a solid foundation in which to base business decisions. This is done by using a ***data model,*** or the logical data structures that detail the relationships among data elements using graphics or pictures. A ***relational database model*** stores information in the form of logically related two-dimensional tables. Tables, or entities as they are formally referred to, will be discussed later.

In developing the relational database model to design a database, an entity-relationship diagram is used. An ***entity-relationship diagram (ERD)*** is a technique for documenting the entities and relationships in a database environment. Before describing the notation used for developing an ERD, it is important to understand what entities and attributes are.

ENTITIES AND ATTRIBUTES

An ***entity*** stores information about a person, place, thing, transaction, or event. A customer is an entity, as is a product and an appointment. An ***attribute*** is the data elements associated with an entity. For example, consider Mega-Video, a physical and online retail store that sells movie DVDs. The company would need to store information on its customers (especially for online purchases) by creating an entity called *CUSTOMER* that contained many attributes such as *Customer Number, First Name, Last Name, Street, City, State, Zip Code, Phone Number,* and *Email* (refer to Figure C.1).

figure C.1 **Entities and Attributes Examples**

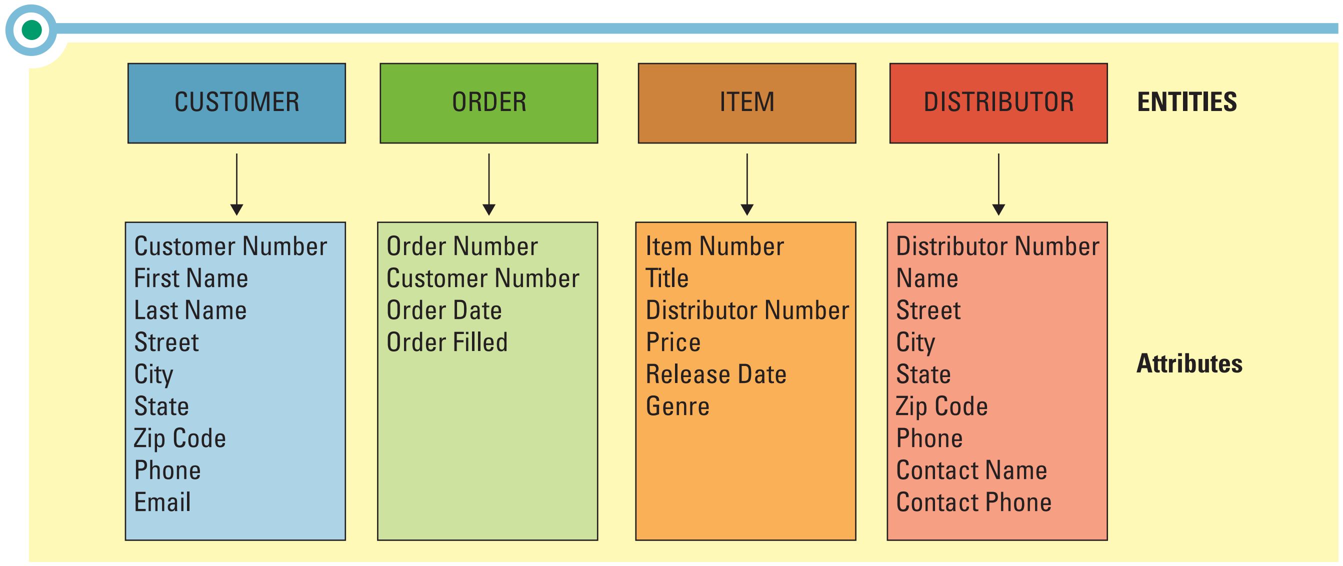
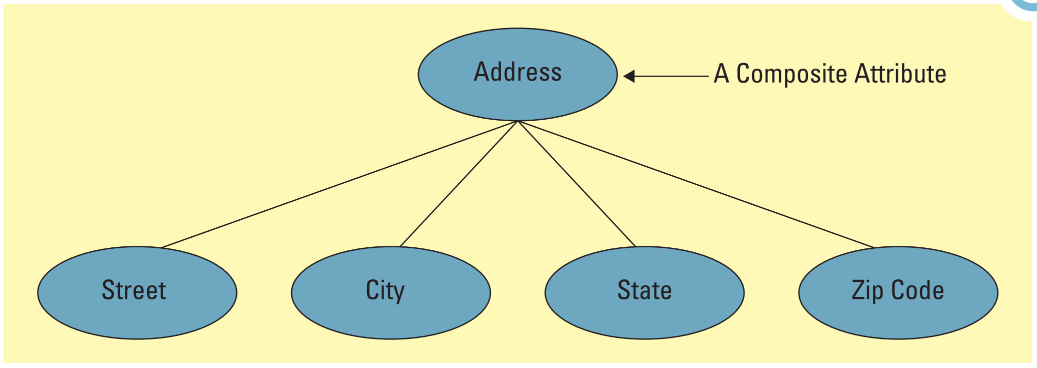


figure C.2 Composite Attributes



Type of Attributes There are several types of attributes, including:

- **Simple versus composite.** A simple attribute cannot be broken down into a smaller component. For example, a customer's first name and last name are simple. A composite attribute can be divided into smaller components, which represent more basic attributes that have their own meanings. A common example of a composite attribute is *Address* (see Figure C.2). *Address* can be broken down into a number of subparts, such as *Street, City, State, Zip Code.*
- **Single-valued versus multi-valued.** When creating a relational database, the attributes in the data model must be single-valued. ***Single-valued attribute*** means having only a single value of each attribute of an entity. A person's age is an example of a single-valued attribute because a person cannot have more than one age. ***Multi-valued attribute*** means having the potential to contain more than one value for an attribute. For example, an educational degree of a person is a multi-valued attribute because a person can have more than one degree. An entity in a relational database cannot have multi-valued attributes; those attributes must be handled by creating another entity to hold them. Therefore, in the example given previously, in designing the database there would be two entities, one called *PERSON* (or something similar) and one called *DEGREE.* If a multi-valued attribute has been identified, it typically is a clue that another entity is needed.
- **Stored versus derived.** If an attribute can be calculated using the value of another attribute, it is called a derived attribute. The attribute that is used to derive the attribute is called a stored attribute. Derived attributes are not stored in the file, but can be derived when needed from the stored attributes. One example of a derived and stored attribute is a person's age. If the database has a stored attribute such as the person's *Date of Birth,* then a derived attribute called *Age* can be created from taking the *Current Date* (this is retrieved from the DBMS) and subtracting the *Date of Birth* to get the age.
- **Null-valued.** There are cases where an attribute does not have an applicable value for an attribute. For these situations, the null-valued attribute is created. ***Null-valued attribute*** is assigned to an attribute when no other value applies or when a value is unknown. A person who does not have a mobile phone would have null stored for the value for the *Mobile Phone Number* attribute. Null can also be used in situations where the attribute value is unknown such as *Hair Color.* Every person has a hair color, but the information may be missing.

Business Rules

The "correct" design for a specific business depends on the business rules; what is correct for one organization may not be correct for another. A ***business rule*** is a statement that defines an aspect of a business. It is intended to convey the behavior and rules of a business. The following statements are examples of possible business rules for Mega-Video:

- A customer can purchase many DVDs.
- DVDs can be purchased by many customers.
- A DVD title can have many copies.

A typical business may have hundreds of business rules. Each business rule will have entities and sometimes even attributes in the statements. For instance, in the first example above, *CUSTOMER* and *DVD* would be entities according to this business rule. Identifying the business rules will help to create a more accurate and complete database design. In addition, the business rules also assist with identifying relationships between entities. This is very useful in creating ERDs.

Documenting Entity Relationship Diagrams

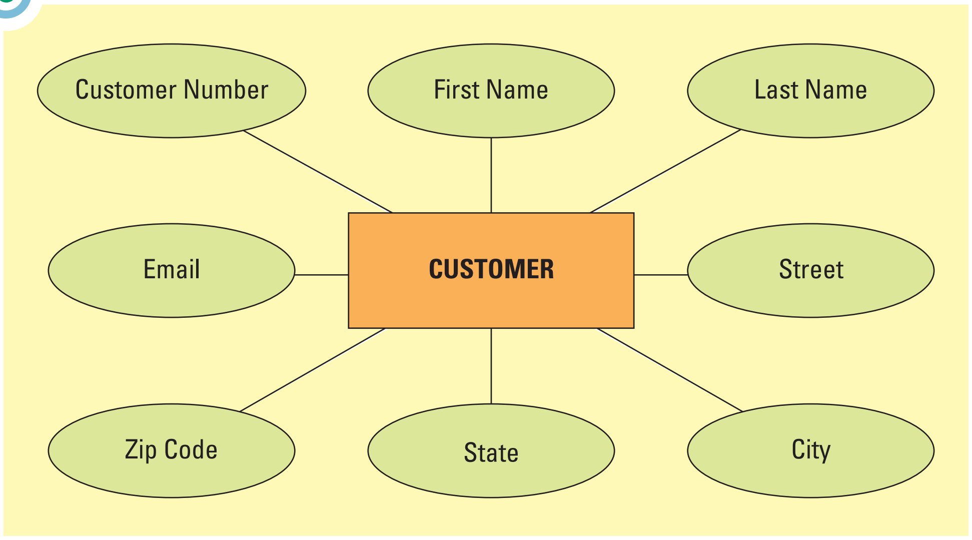
Once entities, attributes, and business rules have been identified, the ERD can be documented. The two most commonly used models of ERD documentation are Chen, named after the originator of entity-relationship modeling, Dr. Peter Chen, and information engineering, which grew out of work by James Martin and Clive Finkelstein. It does not matter which is used, as long as everyone who is using the diagram understands the notation. For purposes of simplicity, only the Chen model will be described here.

The Chen model notation uses very specific symbols in representing entities and attributes. Rectangles are used to represent entities. Each entity's name appears in the rectangle, is expressed in the singular, and is capitalized, as in *CUSTOMER.* Originally, attributes were not part of the Chen model; however, many database designers have extended it to include the attributes in ovals as illustrated in Figure C.3.

BASIC ENTITY RELATIONSHIPS

One of the main reasons for creating an ERD is to identify and represent the relationships between entities. If the business rules for Mega-Video state that a customer can order many videos (in this case, an item), then a relationship needs to be created between *CUSTOMER, ORDER,* and *ITEM.* This is a purely conceptual representation of what the database will look like and is completely unrelated to the physical storage of the data. Again, what the ERD is doing is creating a model in which to design the database.

figure C.3
Chen Model with Attributes



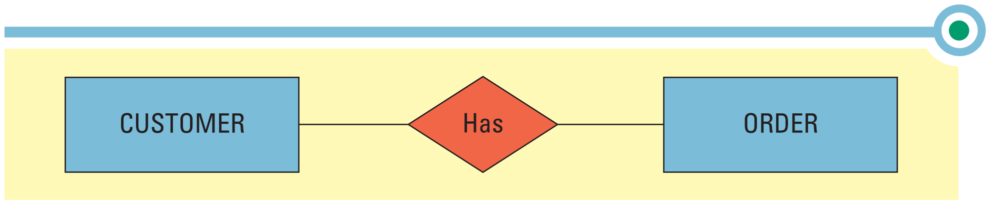


figure C.4
Chen Method with Relationship

The Chen model uses diamonds for relationships and lines to connect relationships between entities. Figure C.4 displays the relationship between a Mega-Video *CUSTOMER* and *ORDER* using this notation. The word within the relationship gives some indication of the meaning of the relationship.

Once the basic entities and their attributes have been defined, the next task is to identify the relationships among those entities. There are three basic types of relationships: (1) one-to-one, (2) one-to-many, and (3) many-to-many.

One-to-One Relationship A ***one-to-one relationship (1:1)*** is between two entities in which an instance of one entity can be related to only one instance of a related entity. Consider Mega-Video, which has many different stores with several employees and one manager. According to the company's business rules, the manager, who is an employee, can manage only one store. The relationship then becomes 1:1 between *EMPLOYEE* and *STORE*. Using the Chen model notation, as shown in Figure C.5, the relationships between the two instances can then be expressed as "An employee can manage one store and one store has one manager." The number "1" next to the *EMPLOYEE* and *STORE* entities indicates that only one *EMPLOYEE* manages one *STORE*.

One-to-Many Relationship Most relational databases are constructed from one-to-many relationships. A ***one-to-many relationship (1:M)*** is between two entities in which an instance of one entity can be related to many instances of a related entity. For example, Mega-Video receives many *ITEM*(s) from one *DISTRIBUTOR* and each *DISTRIBUTOR* supplies many *ITEM*(s) as Figure C.6 illustrates. Similarly, a *CUSTOMER* can

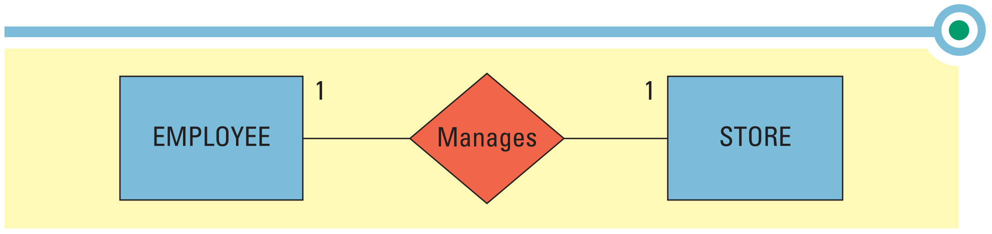
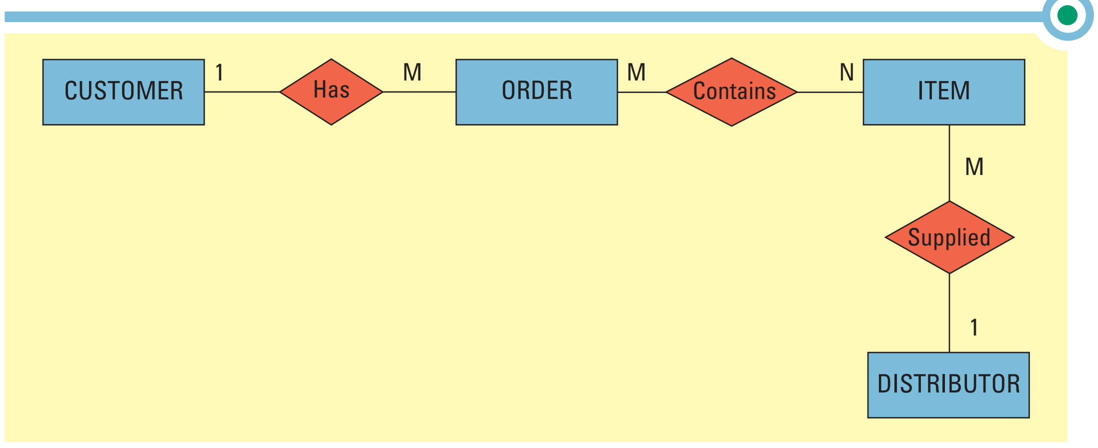


figure C.5
A One-to-One Relationship

figure C.6 A One-to-Many Relationship



have many *ORDER*(s), but an *ORDER* has only one *CUSTOMER.* These are both examples of a one-to-many relationship. The "M" next to the *ORDER* entity indicates that a *CUSTOMER* can place one or more *ORDER*(s). That notation is also used with *ITEM,* as an *ORDER* can contain one or more *ITEM*(s).

Many-to-Many Relationship Identifying and removing many-to-many relationships will help to create an accurate and consistent database. A ***many-to-many relationship (M:N)*** is between two entities in which an instance of one entity is related to many instances of another and one instance of the other can be related to many instances of the first entity. There is a many-to-many relationship between a Mega-Video *ORDER* and *ITEM* (refer back to Figure C.6). An *ORDER* can contain many *ITEM*(s) and each *ITEM* can be on many *ORDER*(s). The letter "N" next to *ITEM* in Figure C.6 indicates the many-to-many relationship between *ORDER* and *ITEM.*

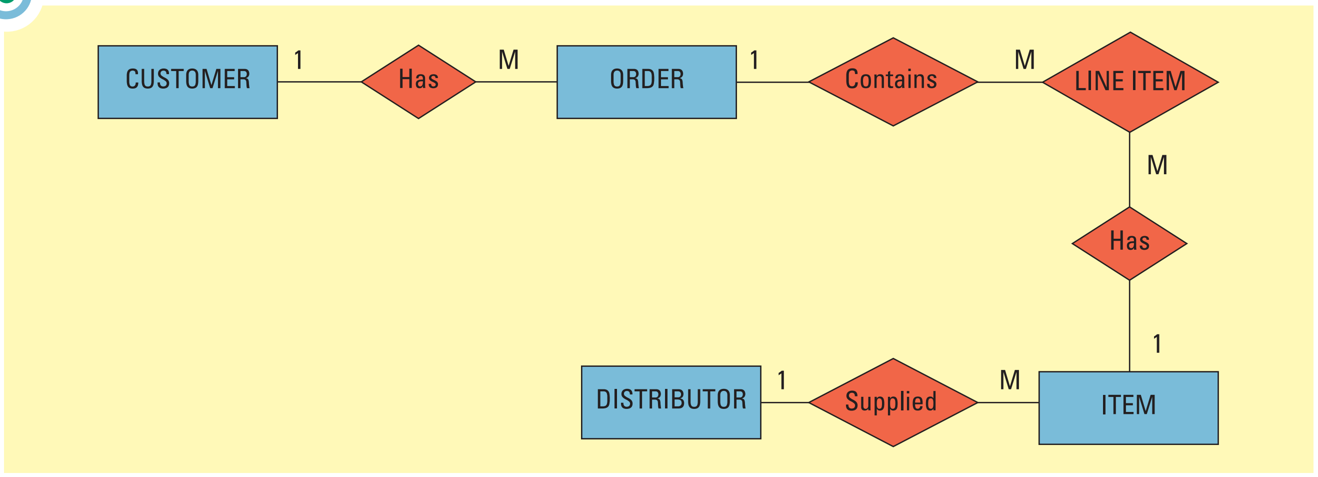
However, there are problems with many-to-many relationships. First, the relational data model was not designed to handle many-to-many relationships. This means they need to be replaced with a one-to-many relationship to be used in a relational DBMS. Second, many-to-many relationships will create redundancy in the data that are stored. This then has a negative impact on the accuracy and consistency that a database needs. To better understand this problem, consider the relationship between *ITEM* and *ORDER.* There is a many-to-many relationship between the *ORDER* and the *ITEM* because each *ORDER* can contain many *ITEM*(s) and, over time, each *ITEM* will be on many *ORDER*(s). Whenever a *CUSTOMER* places an *ORDER* for an *ITEM,* the number of *ITEM*(s) varies, depending on how many DVDs the *CUSTOMER* is buying. To break the many-to-many relationship, a composite entity is needed.

Entities that exist to represent the relationship between two other entities are known as ***composite entities.*** Our above example needs another entity that breaks up the many-to-many relationship between *ORDER* and *ITEM.* Figure C.7 displays the new relationship.

Creating a composite entity called *LINE ITEM* (think of it as a line item on an invoice slip) breaks up the many-to-many relationship between *ORDER* and *ITEM,* which then eliminates redundancy and other anomalies when deleting or updating information. Using the Chen model, composite entities are documented with a combination of a rectangle and a diamond.

Given the new ERD in Figure C.7, each *ORDER* can contain many *LINE ITEM*(s), but a *LINE ITEM* can belong to only one *ORDER.* As a result, the relationship between *ORDER* and *LINE ITEM* is one-to-many (one order has one or more line items) and the relationship between *LINE ITEM* and *ITEM* is one-to-many (one item can be in many line items). The composite entity has removed the original many-to-many relationship and turned it into two one-to-many relationships.

figure C.7 A Composite Entity



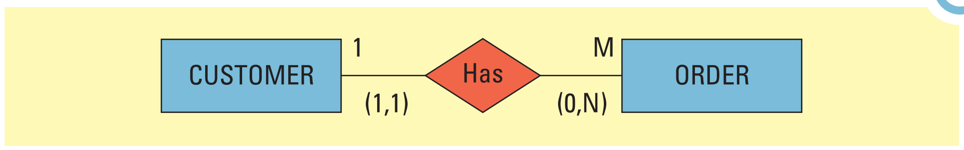


figure C.8
Example of Cardinalities

RELATIONSHIP CARDINALITY

Cardinality expresses the specific number of instances in an entity. In the Chen model, the cardinality is indicated by placing numbers beside the entities in the format of (x, y). The first number in the cardinality represents the minimum value and the second number is for the maximum value.

Mega-Video can store data about a *CUSTOMER* in its database before the *CUSTOMER* places an *ORDER*. An instance of the *CUSTOMER* entity does not have to be related to any instance of the *ORDER* entity, meaning there is an optional cardinality.

However, the reverse is not true for the Mega-Video database, an *ORDER must* be related to a *CUSTOMER*. Without a *CUSTOMER,* an *ORDER* cannot exist. An instance of the *CUSTOMER* entity can be related to zero, one, or more *ORDER*(s) using the cardinality notation (0,N). An instance of the *ORDER* entity must be related to one and only one *CUSTOMER,* having a cardinality of (1,1). The relationship between an instance of *ORDER* and *CUSTOMER* is a mandatory relationship. Figure C.8 illustrates these cardinalities.

Relational Data Model and the Database

Once the ERD is completed, it can be translated from a conceptual logical model into the formal data model required by the DBMS. The relational data model is the result of the work of Edgar (E. F.) Codd, a mathematician. During the 1960s, Dr. Codd was working with existing data models when he noticed that data relationships were very inefficient. Using his experience and knowledge in mathematics, he created the relational data model. Most databases, such as Access 2010 and SQL Server 2010, are based on the relational data model.

FROM ENTITIES TO TABLES

In creating an ERD for the conceptual model, the focus was on identifying entities and attributes. For the logical relational data model, the attention is on tables and fields. Using the ERD the entities become tables and attributes turn into fields. A ***table*** is composed of rows and columns that represent an entity. A ***field*** is a characteristic of a table. A ***record*** is a collection of related data elements. The columns in the table definition represent the field, while a row is a record.

At first glance, a table along with the fields and records looks much like information in a spreadsheet, such as that displayed in Figure C.9 of a *CUSTOMER* table.

figure C.9
A Sample Customer Table

CUSTOMER

Customer Number	First Name	Last Name	Phone Number
0001	Bill	Miller	777-777-7777
0505	Jane	Cook	444-444-4444
1111	Sam	Smith	555-555-5555
1212	John	Doe	666-666-6666

Fields Figure C.9 has four fields, *Customer Number, First Name, Last Name,* and *Phone Number.* Two or more tables within the same relational data model may have fields with the same names, but a single table must have unique field names. Using the relational data model notation, the table names are capitalized (e.g., *CUSTOMER*) and all columns are in title case (e.g., Customer Number), as in:

CUSTOMER (Customer Number, First Name, Last Name, Phone Number)

Records A record in a table has the following properties:

- A table cannot have multi-valued attributes (as mentioned previously); therefore, only one value is allowed at the intersection of a field and record.
- Each record needs to be unique; there are no duplicate records in a table.
- A record must have an entity identifier, or ***primary key,*** which is a field (or group of fields), that uniquely identifies a given record in a table.

Primary Key A primary key makes it possible to uniquely identify every record in a table. The primary key is important to retrieve data accurately from the database.

Using a *Customer Number* as a primary key states that no two customers will ever have the same number. The primary key will be used to identify records associated with it. For example, if someone was searching the Mega-Video database for all the *ITEMS* that a *CUSTOMER* with a *Customer Number* of "112299" bought, he would retrieve only those records and not those associated with another customer.

Along with being unique, a primary key must not contain the value *null.* Recall that null is a special value meaning unknown; however, it is not the same as a field being blank or set to the value of zero. If one record has a null primary key, then the data structure is not in violation. But once a second null value for another record is introduced, the uniqueness of the primary key is lost. Therefore, nulls are forbidden when establishing primary keys.

The proper notation to use when documenting the primary key is to underline it, such as:

CUSTOMER (<u>Customer Number</u>, First Name, Last Name, Phone Number)

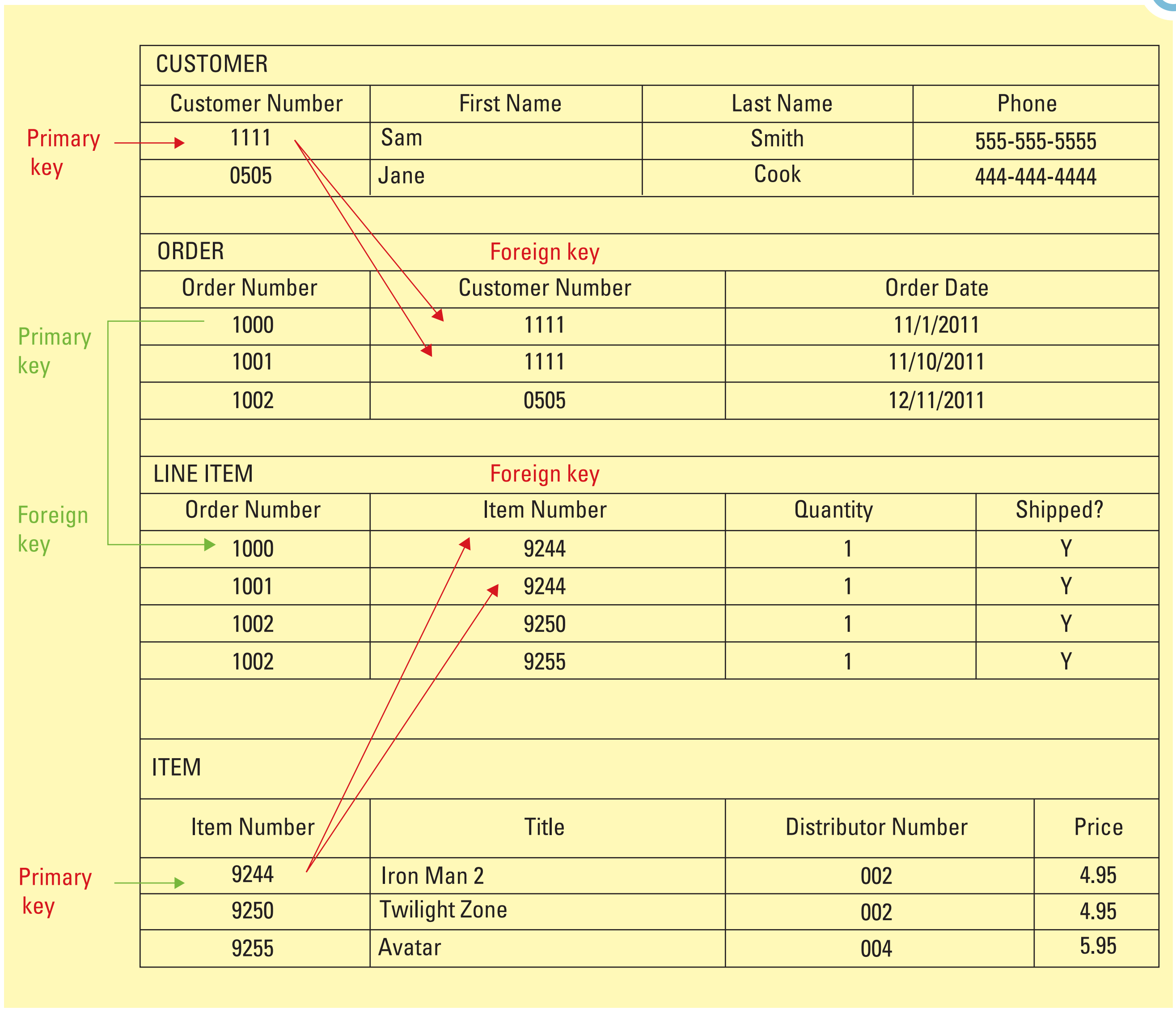
LOGICALLY RELATING TABLES

Once the primary key has been defined, tables can be logically related. Each table in Figure C.10 is directly analogous to the entities of the same name in the Mega-Video ERD presented in Figure C.8, excluding the *DISTRIBUTOR.* The *CUSTOMER* table is identified by a Customer Number, a randomly generated unique primary key. The *ORDER* table is identified by an Order Number, another arbitrary unique primary key assigned by Mega-Video. The table *ORDER LINE* tells the company which *ITEM*(s) are part of which *ORDER.* This table requires a concatenated primary key (that is to say, joining two fields that act as one primary key) because multiple *ITEM*(s) can appear on multiple *ORDER*(s). The selection of this primary key, however, has more significance than simply identifying each record; it also represents a relationship between the *ORDER LINES,* the *ORDER* on which they appear, and the *ITEM*(s) being ordered. The primary key for the *ITEM* table is identified by the *Item Number.*

The *Item Number* field in the *ORDER LINE* table is the same as the primary key in the *ITEM* table. This indicates a one-to-many relationship between the two tables. Similarly, there is also a one-to-many relationship between the *ORDER* and *ORDER LINE* tables because the *Order Number* column in the *ORDER LINE* table is the same as the primary key of the *ORDER* table.

When a table contains a field that is the same as the primary key of another table, it is called a foreign key. A ***foreign key*** is a primary key of one table that appears as an attribute in another table and acts to provide a logical relationship between the two tables. The matching of foreign keys to primary keys represents data relationships in a relational database.

figure C.10 Logically Relating Tables



CUSTOMER

Customer Number	First Name	Last Name	Phone
1111	Sam	Smith	555-555-5555
0505	Jane	Cook	444-444-4444

ORDER

Order Number	Customer Number	Order Date
1000	1111	11/1/2011
1001	1111	11/10/2011
1002	0505	12/11/2011

LINE ITEM

Order Number	Item Number	Quantity	Shipped?
1000	9244	1	Y
1001	9244	1	Y
1002	9250	1	Y
1002	9255	1	Y

ITEM

Item Number	Title	Distributor Number	Price
9244	Iron Man 2	002	4.95
9250	Twilight Zone	002	4.95
9255	Avatar	004	5.95

Foreign keys may be a part of a concatenated primary key, as is the case in the *LINE ITEM* table in Figure C.10. By concatenating, or combining, both *Order Number* and *Item Number* in the *LINE ITEM* table as foreign keys, they then become primary keys. However, most foreign keys are not part of the table's primary key. Consider the relation between Mega-Video's *CUSTOMER* and *ORDER* in Figure C.10. The *Customer Number* field in the *ORDER* table is a foreign key that matches the primary key of the *CUSTOMER* table. It represents the one-to-many relationship between *CUSTOMER* and *ORDER*. However, the *Customer Number* is not part of the primary key of the *ORDER* table; it is simply used to create a relationship between the two tables, *CUSTOMER* and *ORDER*.

A relational database uses the relationships indicated by matching data between primary and foreign keys. Assume that a Mega-Video employee wanted to see what *Titles* had been ordered with *Order Number* 1002. First, the database identifies the records in the *LINE ITEM* table that contain an *Order Number* of 1002. Then, it matches them to the *Item Number*(s) in the *ITEM* table. The results are those that match the records from each table.

Key Terms

Attributes, C.2
Business rule, C.3
Cardinality, C.7
Composite entity, C.6
Data model, C.2
Database, C.1
Database management system (DBMS), C.1
Entity, C.2
Entity-relationship diagram (ERD), C.2
Field, C.7
Foreign key, C.8
Many-to-many relationship (M:N), C.6
Multi-valued attribute, C.3
Null-valued attribute, C.3
One-to-many relationship (1:M), C.5
One-to-one relationship (1:1), C.5
Primary key, C.8
Record, C.7
Relational database model, C.2
Single-valued attribute, C.3
Table, C.7

Apply Your Knowledge

1. SportTech Events

SportTech Events puts on athletic events for local high school athletes. The company needs a database designed to keep track of the sponsor for the event and where the event is located. Each event needs a description, date, and cost. Separate costs are negotiated for each event. The company would also like to have a list of potential sponsors that includes each sponsor's contact information, such as the name, phone number, and address.

Each event will have a single sponsor, but a particular sponsor may sponsor more than one event. Each location will need an ID, contact person, and phone number. A particular event will use only one location, but a location may be used for multiple events. SportTech asks you to create an ERD from the information described here.

2. Course and Student Schedules

Paul Bauer, the chair for the information technology (IT) department at the University of Denver, needs to create a database to keep track of all the courses offered by the department. In addition to the course information, Bauer would like the database to include each instructor's basic contact information, such as ID number, name, office location, and phone number. Currently, the department has nine instructors (seven full-time faculty and two adjuncts). For each course, Bauer would like to keep track of the course ID, title, and number of credit hours. When courses are offered, the section of the course receives an ID number, and with that number, the department keeps track of which instructor is teaching the course. There is only one instructor per course.

Finally, Bauer needs to be able to keep track of the IT students and to know which courses each student has taken. The information he would like to know about each student includes ID number, name, and phone number. He also needs to know what grade the student receives in each course.

He has asked you to create an ERD from the information described here using the Chen model.

3. Foothills Athletics

Foothills Athletics is an athletic facility offering services in Highlands Ranch, Colorado. All property owners living in Highlands Ranch are members of the Highlands Ranch Community Association (HRCA), which has partnered with Foothills Athletics to provide recreation facilities for its residents. Foothills Athletics has been using a spreadsheet to keep track of its personnel, facilities, equipment, and the HRCA members. The spreadsheet has created many redundancies along with several anomalies in adding, modifying, and deleting information. One of the HRCA members has suggested that the athletic facility should create a database to improve data collection that will also remove many of the difficulties that the spreadsheet is creating.

Foothills Athletics primary business operations are based on the following:

- **Personnel:** Foothills Athletics has a number of employees, primarily fitness instructors and administrative personnel. Records are kept on each employee, detailing employee name, address, phone number, data of hire, position, and status as either a current or former employee. Employees are assigned a unique four-digit employee ID number when they are hired.

- **Members:** When joining Foothills Athletics, HRCA members are assigned a unique four-digit member ID number. This information along with their name, address, phone number, gender, birth date, and date of membership are recorded. At the time of enrollment, each member decides on one of three available membership types along with a fixed membership fee: Platinum ($400), Gold ($300), and Silver ($200). This is a one-time fee that establishes a lifetime membership.
- **Facilities and equipment:** Foothills Athletics has a variety of facilities and equipment choices. Each facility has a unique room number and a size limitation associated with it. Some of the rooms contain a variety of exercise equipment; all have a serial number that is used for inventory and maintenance purposes. In addition, for each piece of equipment, the purchase date and the date of its last maintenance are recorded. Each piece of equipment belongs to a specific equipment type, such as elliptical machine, and is assigned a unique three-digit identification number. The description, the manufacturer's model number, and the recommended maintenance interval for that model of equipment are also kept on file. Each equipment type is associated with a single manufacturer that is referenced by a unique two-digit manufacturer ID number.

Project Focus:

You have been hired to assist Foothills Athletics to create an ERD from the information described here using the Chen model.

4. Slopeside Ski Rentals

Vail Resort in Vail, Colorado, is internationally known as one of the best places in North America for skiing. Since 1973, Slopeside Ski Rentals has been a tradition in the area. At Slopeside Ski Rentals, customers will find the largest selection of skis, boots, snowboards, clothing, helmets, eyewear, and a variety of other accessories needed for the slopes.

You have been employed for the past three winters by the company. Recently, there has been a surge in business, and the owners need a more accurate way to manage the rental business. You have decided to create a database to help the owners keep track of the ski rentals, who the customers are, amount paid, and any damage to the skis when they are rented. The skis and snowboards vary in type, size, and bindings. When customers rent equipment, they are required to leave their driver's license number and to give a home address, phone number, and credit card number.

A few business rules that you are aware of include:

- A customer can rent one or more skis or snowboards at one time.
- Skis and snowboards can be rented by many customers.
- A ski or snowboard need not be assigned to any customer.

Project Focus:

Your job is to develop an ERD from the business rules mentioned here.