Scott Mankowitz

3.1 Information Technology Systems

CHAPTER OUTLINE

3.1.1 COMPUTER SYSTEMS

The study of informatics must begin at the most elemental level. Only by understanding computer systems can we progress to understanding how they work and what they can do.

3.1.1.1 Programming

Programming is the technique by which we tell computers what we expect of them. From a physical (electronic) standpoint, all information in the computer must be represented by a series circuits which are either "on" or "off". For convenience, we usually refer to these values as 1's and 0's (binary). By convention, the computer reads a series of binary numbers (usually a multiple of 8) which correlates to a specific instruction. Each of these instructions tells the computer to perform some simple task. Most of these instructions involve fetching data from memory, storing data into memory or performing some kind of mathematical operation. By combining thousands or millions of very simple operations, we can instruct the computer to perform useful and interesting tasks.

For example, the following is a very short program written in Intel x86 **machine code** which adds the numbers 23 and 44.

MACHINE CODE	WHAT IT MEANS
01100110101110000001011100000000	Load the number 23 into a register called ax
01100110101110110010110000000000	Load the number 44 into another register called bx
0110011000000000111011000	Add the two numbers in ax and bx

When written on paper, binary notation can be quite verbose. For convenience, the above program can be expressed in

Box 6-1: Hexadecimal Notation

When humans talk about numbers, we use base 10 (decimal). Computers store data in base 2 (binary). Each **B**inary digi**IT** is called a **bit**. Eight bits compose one **byte**. Programmers often use base 16 (**hexadecimal**) to express numbers because it is more compact than binary. Since 16 is a power of 2, an 8-digit binary number can be represented as a 2-digit hex number. Longer strings of bytes are measured in multiples of 1024 (2^{10}). For example, a string of 1024 bytes is called a **kilobyte**; 1024 kilobytes is a **megabyte**, and so on.

DECIMAL	BINARY	HEX
1	0000 0001	01
2	0000 0010	02
3	0000 0011	03
4	0000 0100	04
5	0000 0101	05
6	0000 0110	06
7	0000 0111	07
8	0000 1000	08
9	0000 1001	09
10	0000 1010	0A
11	0000 1011	0B
12	0000 1100	0C
13	0000 1101	0D
14	0000 1110	0E
15	0000 1111	0F
16	0001 0000	10

hexadecimal notation (base 16). Note that the information is exactly the same, only the presentation is different. See Box 6-1—Hexadecimal Notation.

```
66B8170066BB2C006601D8
```

Although the computer understands this code natively, it is very difficult for us humans to parse. In order to make programming easier, computer scientists created **assembly language** which uses words and abbreviations in place of the long strings of numbers. The programmer writes the program in assembly language while another program (called an **assembler**) translates it into machine code.

In the code snippet below, the left column represents the machine code while the right column is assembly language. To the far right are human-readable comments which make the code easier to understand (The assembler ignores the comments when it produces machine code.)

```
0:     66 b8 17 00     mov ax,0x17     ;move the number 23 (17 hex)into
                                        register ax
4:     66 bb 2c 00     mov bx,0x2c     ;move 44 (2c hex) in register bx
8:     66 01 d8        add ax,bx       ;add the two registers ax and bx
```

As the complexity of programs grew, it became impractical to write programs in assembly language, and higher-order languages were created. By the late 1950s, IBM had developed a language called **FORTRAN** (FORmula TRANslator). Compared to assembly language, FORTRAN was easier to read, easier to write, and much easier to debug. Instead of mapping the computer's machine code into short words and mnemonics, FORTRAN was a completely new language with its own syntax and grammar. Below is an example of an FORTRAN program in the f90 dialect which prints the result of an arithmetic problem.

```
program addnumbers
print *, 23*55+33*155
end program
```

That's much easier to read, even for non-programmers. Computers, however, don't natively understand Fortran. Instead, yet another program, called a **compiler**, is needed to convert the FORTRAN program (known as **source code**) into machine code. In compiled languages, the entire program has to be complete before the compiler can begin. Only after compilation is finished can the program be run.

It is important to note that while assembly language is simply a human-readable form of machine code, FORTRAN is its own language. There is no direct 1:1 mapping between FORTRAN and machine code as there is with assembly language. In fact, if two different FORTRAN compilers are given identical source code, they can produce entirely different machine code. In some cases, a good compiler can make up for a sloppy programmer by applying optimizations so that the code runs more quickly.

Sometimes, programmers would like to examine and change a program *while* it is running. This desire lead to the creation of the **interpreted language**, where each line of the program is passed through an **interpreter** for execution, and only the part of the program that is immediately needed will be translated. The program can be stopped and started whenever necessary. The cost of this flexibility is speed. Compiled languages tend to run much faster. Probably the most famous interpreted language is BASIC (Beginner's All-purpose Symbolic Instruction Code) which appeared on most home computers in the 1980s. Interestingly, Visual Basic, created by Microsoft, is a compiled language.

Today, there are more than 8000 established computer languages with widely varying purposes. There has been great progress in the development of both interpreters and compilers, especially with the advent of Just-In-Time (JIT) compilers, so that both speed of execution and real-time mutability can be had with either compiled or interpreted languages.

3.1.1.2 Data and Control Structures

Control Structures are the linguistic mechanisms that programmers use to tell a computer what to do in a given circumstance. The most simple control structure is the **if/then/else** block. Suppose you want the computer to address a female user using an age-appropriate greeting using the following rule: a person under age 10 is a girl, a person over age 40 is a lady and everyone else is a woman. In Visual Basic, this logic can be expressed as the following program. (Note: The command **Console.Write** prints text to the screen)

```
If age < 10 Then
    Console.Write("Girl")
ElseIf age > 40 Then
    Console.Write("Lady")
Else
    Console.Write("Woman")
End If
```

Another common construct is the **loop**, which executes code over and over. In the following Visual Basic program, the computer is instructed to print the word "cat" 100 times. In Basic, this kind of loop is called a **for/next** loop because it begins with the word **For** and ends with the word **Next**. All of the instructions between the words **For** and **Next** are repeated.

```
For counter = 1 To 100
    Console.Write("cat")
Next
```

This program starts by setting a counter to 1 and then increasing that counter until it gets to 100. Each time the loop is run is called an **iteration.** The counter, which changes its value during the operation of the program, is called a **variable**. When using a placeholder for a value that does not change, it is called a **constant** or sometimes an **invariant.** Constants are commonly used in programs for numbers or strings that are unlikely to change. Some constants are even built into the language. For example, in Javascript there is a constant called Math.PI which equals 3.141592653589793.

In some cases, the programmer does not know exactly how many times to run a piece of code. The **while loop** instructs the computer to repeat a process until some condition is met. In this case, all the instructions between **While** and **End While** are repeated. The for/next loop above could be rewritten as

```
counter = 1
While counter < 100
    Console.Write("cat")
    counter = counter + 1
End While
```

Note that the instruction **counter = counter + 1** tells the computer to increase the value of counter by one, just like the for/next loop. The loop runs over and over (i.e. iterates) until the value of **counter** reaches 100.

Control structures can be incredibly powerful. Consider the following code that might be used to control an insulin pump[1]:

```
If (glucose < 60) Then
    ' Danger: Hypoglycemia
    Activate_Alarm_Bell()
    Page_Covering_Physician()
ElseIf (glucose > 60 And glucose < 180) Then
    ' No action required
ElseIf (glucose > 180 And glucose < 400) Then
    ' Use a formula to determine insulin dosage
    ' Insulin dose = 3.2 units for each 100 mg/dl glucose over 140
    Inject_Insulin(3.2 * (glucose - 140)/100)
ElseIf (glucose > 400) Then
    ' Danger: Hyperglycemia
    Inject_Insulin(10)
    Activate_Alarm_Bell()
    Page_Covering_Physician()
End If
```

Functions (or methods) are used to encapsulate pieces of code that are used multiple times. In the example above, `Inject_Insulin` and `Activate_Alarm_Bell` are examples of functions. In general, functions can take a certain number of arguments which provide the function with more information on what to do. In the above program, `Inject_Insulin`

1 There are bugs in this "program." Can you find them? Hint: what would happen if the blood glucose were exactly 60?

takes the number of units of insulin as an argument, so that `Inject_Insulin(10)` would tell the program to inject 10 units of insulin. Functions can also provide a return value. Consider the following Visual Basic function:

```
Function Words_In_Book(pages As Integer, words_per_page As Integer) As Integer
    Return pages * words_per_page
End Function
```

This function may be useful to figure out how many words there are in a book, given the number of pages and the number of words per page. The keyword **return** means that the function gives that number back to the calling program. For example, suppose our book had 200 pages with 450 words per page. We could use our newly defined function like this:

```
Console.Write("The book has")
Console.Write(Words_In_Book(200, 450))
Console.Write("words")
```

Every language has its own version of control structures and syntax tailored to make certain tasks easier. While some languages are designated all-purpose (such as C, Java, Basic and others), others are dedicated to a particular purpose. For example, Pascal is a language designed specifically to teach programming; SQL is designed to interact with databases; and R is designed for statistics. In some cases, languages can be designed for one purpose and evolve as needs arise. Both Java and Javascript (ECMAscript) were developed to provide interactivity to web pages. Today, they play significant roles in server-side programming and embedded systems.

An introductory programming course is beyond the scope of this book. A truly great and free reference to learn programming (and many other things) is Khan Academy, www.khanacademy.org.

3.1.1.3 Software Development Methods

Small programs are often written by individual developers, but as application requirements become more complex, the coordination of software teams becomes vital. One of the earliest software development methods is called **waterfall**. This is a sequential method, where each phase has to be complete before proceeding to the next level.

The key components of the waterfall method are arranged in the shape of a waterfall. (See Fig. 6-1).

The waterfall method has its origins in manufacturing, where applying structural revisions later on in production could be difficult or impossible. Usually, up to 50% of the project resources are consumed in the Requirement and Design phases. Once those are thoroughly hashed out, the remainder are spent on implementation (i.e. writing code), verification (i.e. debugging) and maintenance. In fact, this is the central tenet of the waterfall model: a few hours spent thoroughly evaluating requirements and design can save many hundreds of

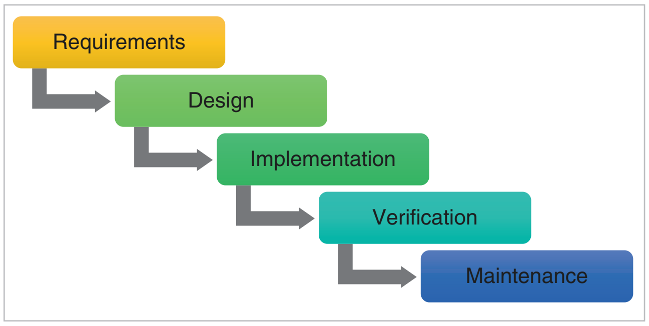


FIGURE 6-1

The waterfall development method

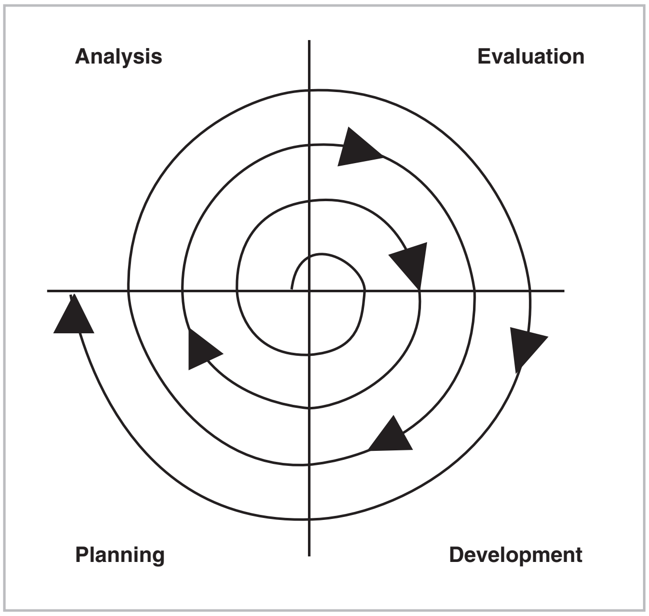


FIGURE 6-2

The spiral model of development emphasizes cyclical iterative risk assessment. Public domain image wikibooks.com. https://commons.wikimedia.org/wiki/File:Software_Development_Spiral.svg

hours later on. An additional benefit is that it provides recognizable and predictable milestones to ensure that a project is running on-time and within budget. Waterfall is often referred to as a **top-down** method because it starts at the design phase and works down towards implementation and maintenance.

Critics of the waterfall method point out that in the real world, project requirements rarely remain stationary.[2] For example, clients may not know what they really want until they have a working prototype in their hands.

This common occurrence lead to another method called **rapid prototyping** where the programmers create representative but incomplete parts of their program so that the client can see what it will look like in advance. Since the user is involved early on, future changes are less likely. Starting with the user interface and working backwards is often referred to as **bottom-up**.

In an effort to combine waterfall and rapid prototyping, Barry Boehm created the **spiral** model which emphasized cyclical iterative risk assessment. Each **cycle** or **iteration** represents only a part of the whole project. It begins with identification of the key stakeholders and their "win conditions" and ends with review and commitment. The cycles consist of four parts: (1) **Analysis** of objectives, alternatives and constraints; (2) **Evaluation** of alternatives and identification of risks; (3) **Development** and verification; (4) **Planning** for the next cycle (Fig. 6-2).

A similar method is **rapid application development (RAD)**. Like spiral, it attempts to reduce project risk by breaking the project into smaller chunks and adhering to strict timelines or **timeboxes**. If the project is not progressing as expected, the requirements are reduced to fit the timebox, instead of postponing the deadline. Emphasis is on keeping users involved in the development process and fulfilling business needs over technical excellence. In distinction to rapid prototyping, RAD involves building completely functional models as opposed to prototypes. RAD also favors computerized development tools such as code generators and object-oriented techniques.

Agile software development is another iterative method that advocates for a more lightweight and people-centric approach. It favors face-to-face, daily interactions between customers and programmers. It welcomes change, even late changes, as long as it doesn't disrupt the planned delivery date. Agile seeks to avoid being trapped by comprehensive documentation and strict planning, and instead stresses short cycles of continuously improving software. **Scrum** and **Extreme Programming (XP)** are variations of the agile theme.

2 Hence the old joke: Walking on water, like writing to a specification, is easy—as long as it's frozen.

In **Test-driven Development (TDD),** before any feature is added to a project, the programmer writes an automated test. After the feature is added and the test passes, it is added to the application's library of internal tests. As the application develops, the code is continuously re-tested to ensure that none of the previous tests fail with the addition of new features. Critics of this process point out that by the time the application is released, it is not uncommon for the testing code to be much larger than the application code.

3.1.1.4 System Integration

In an ideal world, there would be one computer system to manage information for an entire organization. In practice, however, most organizations employ a variety of different systems to accomplish different tasks. In a hospital, for example, the clinical laboratory may be running a Laboratory Information System (LIS), the radiology department may have a Radiology Information System (RIS) as well as a Picture Archiving and Communication System (PACS). Specialized units, such as the Emergency Department or Labor and Delivery may also have their own information systems tailored to their own particular workflow. The decision of what kind of system to use in each department is often both budgetary and political.

One of the greatest challenges in hospital information technology is to enable these disparate systems to communicate. In general, this process is called **System Integration.** In some cases, the integration is very weak and one system is barely aware of another. In tightly integrated systems, there is reliable and verifiable two-way communication. The software used to bind two systems is called an **interface.** In order to create an interface, the vendors from the two systems must agree on the volume and type of data to be transferred.

Vertical Integration is the process of integrating systems based on similar functionality. For example, a RIS may keep track of radiology orders and results, while a PACS may maintain the images. In a vertical integration example, these systems would be tightly integrated with each other, while other systems, such as dietary or patient billing would be more loosely integrated. The tightly integrated units are often called **silos** because they hold information in one place, but don't necessarily share it with other silos (Fig. 6-3).

In some cases, it is desirable for every single system to be able to communicate with every other system. With a small number of systems, this may be a reasonable option,

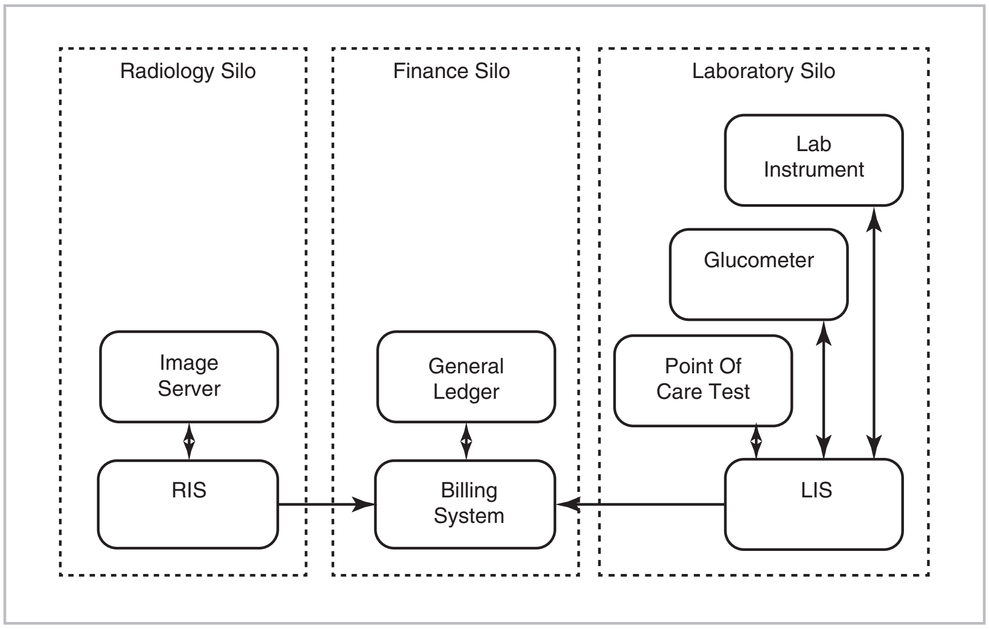


FIGURE 6-3

Vertical integration. Similar systems are located in silos (7 interfaces) but as the number of systems increases, the number of interfaces increases exponentially. This topology is sometimes referred to as **star integration** because the resulting map looks like an N-pointed star where N is the number of systems. Colloquially, it is called **spaghetti integration** because the links look more like a spilled plate of spaghetti. Although expensive, this method provides the best inter-system communication (Fig. 6-4).

A compromise between the high cost of star integration and the weak communication offered by vertical integration is **horizontal integration**. In this method, an entirely new system is created which is dedicated to providing communication between the other systems. Once this system is created, the maintenance and expansion costs are minimal because each new system only requires a single new interface. Moreover, adding new systems is completely transparent to the existing systems, since systems only consume data that they require. Of course, the cost of the system to coordinate this information (often called a **bus** or **enterprise service bus**) can be quite high (Fig. 6-5).

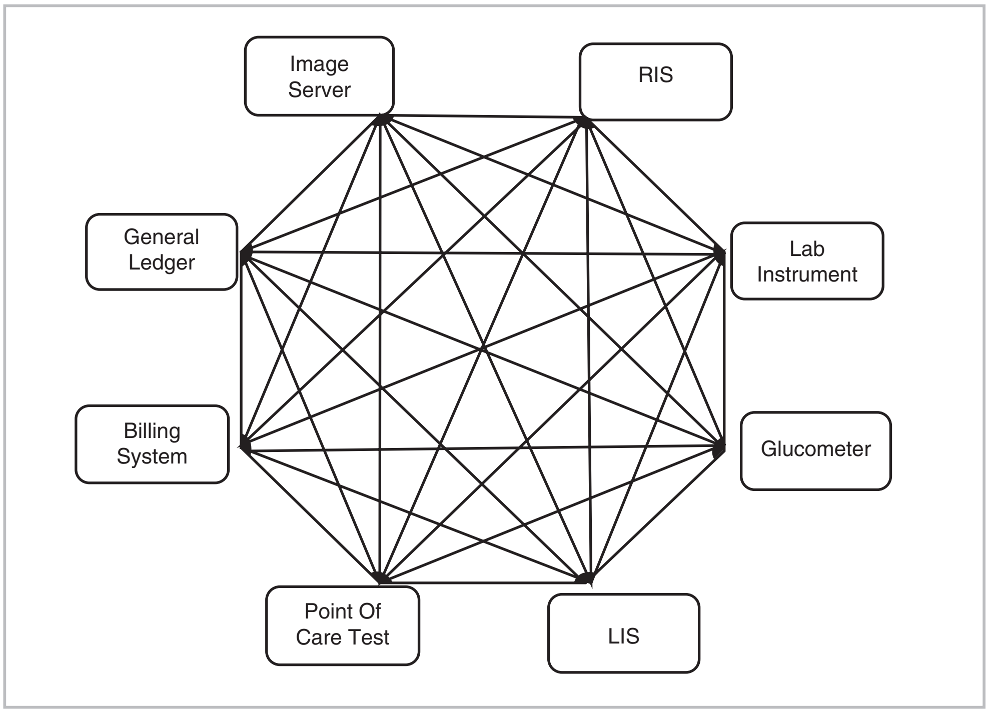


FIGURE 6-4

Star integration. Each system is connected to every other system (28 interfaces)

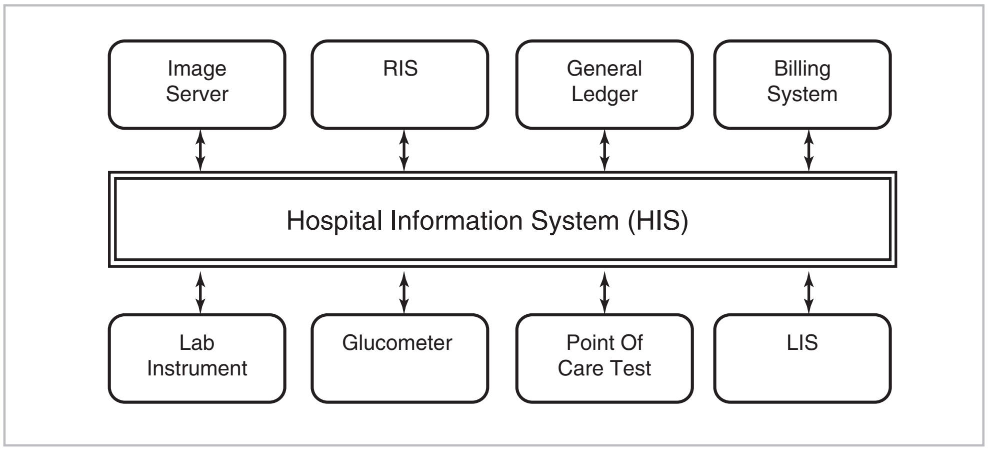


FIGURE 6-5

Horizontal integration. Each system is connected to a bus (8 interfaces +1 new system)

3.1.1.5 Quality

Quality in computer systems (as in medicine) is often difficult to define because of the many factors that need to be taken into account. Some define quality in terms of the usability or reliability of the final product. Others measure structural aspects of the code base. One attempt to describe quality is specified in the International Organization for Standards document 25010 (ISO-25010), which lists the following eight categories.

1. **Functional Suitability**—How well does the system meet the needs of the client? Does it provide correct and complete solutions to the user needs? Can it accomplish all the tasks it is supposed to do? For example, when a hospital or medical practice wishes to attest to meaningful use of a computer system, does it meet all the required specifications?
2. **Performance efficiency**—How well does this system function with a given amount of resources? Is the data store large enough to contain all the patient information required? Will it scale in capacity as the population grows? Does the system access that information quickly enough to be useful on a day-to-day basis?
3. **Compatibility**—Can this system communicate with other systems? (see Sect 3.1.1.4 **System Integration**, above) How expensive is it to build interfaces with other common computer architectures? Can this system coexist with other systems in the same environment?
4. **Usability**—Can an end-user figure this system out? Can he learn it easily? What is the cost of training users to operate this system? To what degree does it prevent errors, or at least prevent user errors from harming patients or other users? Once learned, how many steps are needed to accomplish common, simple tasks?
5. **Reliability**—How often must the system be taken down for maintenance? How often is this system expected to fail? When it does fail, how hard is it to recover?
6. **Security**—With the advent of the Health Insurance Portability and Accountability Act (HIPAA), security is a very important topic in assessing the value of a computer system. Does the system have a sufficient array of access levels so that users can be given exactly the permissions that they need and no others? Does the system prevent unauthorized access to data or computer programs? Is there a reliable authentication process to correctly identify users and prevent one user from masquerading as another? Are there detailed security policies?
7. **Maintainability**—This section is much more apparent to the software developer and may not be easily measured by potential customers. In many relationships, this measure can be boiled down to the question of how well the vendor supports its product. However, in order to make a supportable product, these questions are useful: Is the system composed of discrete components so that changing one aspect does not require modification of the whole system? Can aspects of the system be reused in other systems that require similar functionality? How difficult is it to test the system to make sure it is running correctly? To what extent does the system report internal errors? How are those errors tracked? How hard is it for a service engineer to analyze the errors to make improvements?
8. **Portability**—Can the system be moved ("**ported**") to another architecture or environment? When better hardware becomes available, will it be able to take advantage of the new technology? How difficult is the initial install? Does it use standard installation tools, or does it require special expertise? How well can it replace another product or be replaced by another product when needs change?

3.1.1.6 Information Systems Design and Analysis

When designing a data store, it is very important to make sure that your data architecture matches the data you intend to gather. Let's explore a simple example where a researcher is collecting information on patients for a study. She needs to know the patient's name, phone number, address and pulse rate at various times. We can organize the data in a table as follows (Table 6-1):

TABLE 6-1

PATIENT PULSE DATA

PATIENT ID	PATIENT NAME	PATIENT PHONE	ADDRESS	DATE/TIME	PULSE RATE
0002	Jon Harvey	(201) 555-1212	24 Front St.	11/12/16 11:39	78
0002	Jon Harvey	(201) 555-1212	24 Front St.	11/12/16 11:50	85
0001	Marc Jones	(201) 555-2342	90 Field Ave.	11/12/16 11:40	77
0001	Marc Jones	(201) 555-2342	90 Field Ave.	11/12/16 11:51	88
0003	Shara Bevs	(201) 555-6565	16 Broad St.	11/12/16 11:10	66
0002	Jon Harvey	(201) 555-1212	24 Front St.	11/12/16 10:35	90

Each **record**, or row, of the table contains information about a measurement. That data is stored in several columns, or **fields** of different types.

Whenever the researcher records another pulse rate, she adds another line in the table. For small amounts of data, this could be an Excel™ **spreadsheet**.

The problem with a spreadsheet is that it is not **scalable.** As the amount of data grows, a table might have a million or more rows, which can exceed the capacity of most spreadsheets. As the number of researchers grows, there may be many different people trying to access the data at the same time. Most spreadsheets can not accommodate multiple simultaneous users. Finally, spreadsheets are only able to provide a single representation of the data at a time. For example, suppose that one researcher wanted to sort the data by the patient's name in order to identify a trend. At the same time, another researcher is trying to enroll a new patient into the study. Spreadsheets lack the capacity to accomplish both tasks simultaneously. Despite these limitations, spreadsheets are used for a startling amount of scientific recordkeeping.

A more robust method of collecting and maintaining data is the **database.** A database is able to handle many simultaneous users and vast amounts of data. It usually contains a permission system so that specific users only have access to certain types of data.

The database is also able to enforce certain rules about what sort of data it will accept. These rules are often called a **schema**. For example, in our table, each column has a particular data type. The pulse rate column and the patient id column are always numbers; the Date/Time column always contains a date and time. The remainder of the columns are strings of different lengths. These rules, or **constraints,** ensure greater data integrity. Although constraints wouldn't prevent a user from entering a pulse rate that is inaccurate, they would prevent someone from inadvertently entering an address in the patient ID column. The database could also prevent someone from leaving one of the fields blank.

Standardized Query Language (SQL) is a language used to create and query databases. The code below can be used to create Table 6-1. Note that INT means an integer type, DATETIME stores a date and time, and VARCHAR stores a variable length character string with an optional limit. VARCHAR(25) is a string limited to 25 characters.

```
CREATE TABLE patient_pulse_data (
     patient_id      INT,
     patient_name    VARCHAR(25),
     patient_phone   VARCHAR(15),
     patient_address VARCHAR(45),
     pulse_date_time DATETIME,
     pulse_rate      INT
  );
```

SQL can be surprisingly easy to read for simple queries. The following **query** will select the patient name, patient address and pulse readings for a patient with ID of 0003 from the table called patient_pulse_data.

```
SELECT patient_name,
       patient_address,
       pulse_rate
FROM   patient_pulse_data
WHERE  patient_id = 0003
```

Normalization is the process of organizing the columns and tables in a database to reduce data redundancy and improve data integrity. How might we improve our Patient Pulse Data table, above? One way to do this is to remove repeating elements. For example, patient name, phone number and address is repeated multiple times. If we can assume that a patient will maintain the same demographic information throughout the length of the study, we could separate the table into two smaller tables, one which describes the patient and another which describes our measurements.

The corresponding SQL is

```
CREATE TABLE pulses (
     patient_id      INT,
     pulse_date_time DATETIME,
     pulse_rate      INT
  );
CREATE TABLE patients (
     patient_id      INT,
     patient_name    VARCHAR(25),
     patient_phone   VARCHAR(15),
     patient_address VARCHAR(45)
  );
```

In this representation, each row of the pulse table (Table 6-2) refers to a single pulse measurement. Each row in the patient information table (Table 6-3) refers to a single patient. Since a single patient can not be divided into smaller parts, these rows are said to be **atomic.**

The two-table format (i.e. Tables 6-2 and 6-3) is said to be **normalized** because each row of each table refers to one single indivisible idea (i.e. a pulse measurement or a patient, but not both). When a table is intentionally designed to keep redundant data for performance purposes, it is called **denormalized**.

There are two key benefits to normalized data. Firstly, since there is no duplicate information, it takes up less storage space and is generally easier to modify. The trade-off is that that if information is needed from both tables at the same time, the denormalized table is generally faster because the computer does not have to spend extra time matching up the patient ID values.

TABLE 6-2

PULSE INFORMATION (PULSES)

DATE/TIME	PATIENT ID	PULSE RATE
11/12/16 11:39	0002	78
11/12/16 11:50	0002	85
11/12/16 11:40	0001	77
11/12/16 11:51	0001	88
11/12/16 11:10	0003	66
11/13/16 06:00	0002	90

TABLE 6-3

PATIENT INFORMATION (PATIENTS)

PATIENT ID	PATIENT NAME	PATIENT PHONE	ADDRESS
0001	Marc Jones	(201) 555-2342	90 Field Ave.
0002	Jon Harvey	(201) 555-1212	24 Front St.
0003	Shara Bevs	(201) 555-6565	16 Broad St.

The second benefit of normalization is the prevention of **data anomalies**. If one needs to edit the address for Marc Jones in the normalized table (Table 6-3), it only needs to be edited in one place. In the denormalized table (Table 6-1), it has to be edited in three different places. If, for some reason, the address is only updated in one of those locations, the data becomes inconsistent. This is called an **update anomaly** because an incomplete update causes anomalous data.

Suppose a new patient was enrolled into the study, but did not have any pulse data acquired yet. In the one-table model, there would be no way to add the person to the table. (The pulse column can not be left blank because it would violate the constraint.) This problem is called an **insertion anomaly,** which is the inability to add new data to a table because of absence of other data.

The final anomaly is the **deletion anomaly,** where deletion of one piece of data causes unintentional loss of other data. In our example above, suppose the researcher made a mistake and had to delete Shara Brev's pulse reading for 11/12/14. Since the entire row would be deleted, all reference to this patient would disappear.

The two tables are linked (or **related**) by their common data element, namely the patient ID. If a user wants to know the pulse values for patient 0001, he would select those values from the pulse table. If he wants to get demographic information, he would look in the patient information table.

What if he needs data from both tables? Since the two tables are related by the patient ID, the computer will match up rows from the two tables to produce the necessary output. This process is called a **join**. The fact that tables can be related this way is the defining characteristic of the Relational DataBase Management System (**RDBMS**).

The following SQL command will select the patient name from the patients table and the pulse rate from the pulses table. The two tables will be joined using the common data point called patient_id.

```
SELECT patients.patient_name, pulses.pulse_rate
FROM   patients JOIN pulses ON (patients.patient_id = pulses.patient_id)
```

PATIENT_NAME	PULSE_RATE
Marc Jones	77
Marc Jones	88
Jon Harvey	78
Jon Harvey	78
Jon Harvey	85
Jon Harvey	90
Shara Bevs	66

3.1.2 ARCHITECTURE

The architecture of a health information system is every bit as important to its function as the physical architecture of a house. Understanding the different methods in which data is stored and retrieved is crucial to building a durable and usable system.

3.1.2.1 Systems (e.g., Distributed, Centralized, Relational, Object Oriented, Warehouses/ Data Marts)

Until this point, we have considered data that is rigidly assigned to tables, rows and columns. An alternative to the RDBMS is the **NoSQL** database, or the **object-oriented** (OO) database. In this format, there is no schema to define what data is required or what format it should be in. The primary feature of the OO database is that persistent objects in the database should be as transferrable and manipulatable as in-memory objects that are used in an object-oriented programming language.

In one common manifestation of the OO database called a *document store*, each section of the database (often called a **collection**) contains numerous arbitrary objects called **documents**. In the example above, each patient could be stored in a document which would include all the demographic information as well as the pulse readings.[3]

OO data is often represented in eXtensible Markup Language (XML) or JavaScript Object Notation (JSON). The following is an example of what a document for patient 0002, Jon Harvey might look like. Note that all the pulse data pertaining to this patient is stored together with the demographic information which makes it easy to fetch all the patient information at once.

```
{
   "patient_id":2,
   "patient_name":"Jon Harvey",
   "patient_phone":"(201) 555-1212",
   "patient_address":"24 Front St.",
   "pulses":[
      {
         "pulse_date_time":"11/12/14 11:39",
         "pulse_rate":78
      },
      {
         "pulse_date_time":"11/12/14 11:50",
         "pulse_rate":85
      },
      {
         "pulse_date_time":"11/12/14 10:35",
         "pulse_rate":90
      }
   ]
}
```

Since there is no predefined format of these documents, they can be modified quite easily. If the need arose, adding data about blood pressure or temperature for a single patient would pose minimal challenge. To do the same thing in an RDBMS would require the creation of additional columns or even tables.

Another benefit of the OO family of databases is that they are designed to easily scale horizontally. In other words, the data is often distributed among many commodity computers which leads to nearly limitless storage and very high availability. RDBMS tend to scale vertically, which means that there is a single data store and scaling requires a larger hardware investment. The trade off is that RDBMS usually offer greater data consistency.

There are four characteristics used to describe the reliability of a database: (**1**) **Atomicity** is used to describe database transactions that must be done completely or fail completely. For example, suppose you had a table keeping track of money in different bank accounts, and you wanted to transfer money from one account to another. You would have to do two separate database edits: one to debit the money from the first account and one to credit the money to the second account. It would be completely unacceptable if one of the edits went through and the other one didn't. Since the two edits comprise a transaction that is indivisible, it is said to be atomic. (2) **Consistency** means that any transaction will bring the database from one valid state to another. Data written to the database must satisfy all existing constraints, and if an edit would have resulted in an inconsistent state, the edit should be

3 It is important to remember that each of these documents could contain any number of other objects, including other documents. If this sounds complicated, it is. Too complicated for the boards, anyway.

rejected. For example, suppose a database column is constrained to hold only a date, such as date of birth for a patient. A user accidentally enters an invalid date, such as 2017-02-30. There should be no time where the database contains this inappropriate data. (3) **Isolation** provides that if the database allows transactions to run concurrently, the database must ensure that the same state would be achieved if they had run sequentially. In other words, transactions must have adequate isolation from one another. (4) **Durability** means that once a transaction is complete, the change has to be permanent, even if there is a power loss or the computer crashes.

These properties are usually abbreviated as **ACID**, and most RDBMS provide ACID guarantees. OO databases are usually distributed over several computers and can not provide ACID guarantees (however some do). Instead, there are complex algorithms which are used to determine the most recent or most accurate data to be provided to the user. Some systems provide **eventual consistency,** which means that although the query result may not be accurate at this instant, it will get there eventually, usually within a few seconds. For legal and regulatory reasons, databases that can not provide ACID guarantees are not routinely used to store patient information.

In summary, **relational** databases usually maintain a single, **centralized** data store and usually provide ACID guarantees. NoSQL (object oriented) databases are usually **distributed** over many commodity machines in order to provide high availability while sacrificing consistency. *But not always.*

A **data warehouse** is an intentional abstraction of a database designed for analytics. For example, an Electronic Health Record (EHR) is an example of a database. It is designed for efficiently adding and querying information about a particular patient at a particular point in time. It is very rare that a clinician using an EHR would need to review multiple patients at once.

On the other hand, a hospital administrator is much more interested in the care provided to many patients over longer timeframes (e.g. how many CT scans are being performed on a month-by-month basis?). He would be interested in **aggregate** data instead of **individual** data (Table 6-4).

TABLE 6-4

DIFFERENCES BETWEEN A DATA WAREHOUSE AND A DATABASE

	DATABASE	DATA WAREHOUSE
Definition	An electronic information store, usually organized by data type	A subset of a database, organized for ease of aggregate queries
Optimization	Optimized for online transaction processing (OLTP). Tables tend to be highly normalized and the system is optimized for fast and reliable read-write of individual data points. This usually results in many tables linked in complex ways. However, good process modelling ensures that the data is only written once, resulting in fast response times	Optimized for online analytical processing (OLAP). Tables are usually read-only and are intentionally denormalized to make queries faster and easier. The data are organized into multidimensional arrays (also called cubes or hypercubes). Data are summarized in **pivot tables** and used to garner **business intelligence**
Uptime	Mission critical. Systems are expected to have very little unscheduled downtime. Data errors can be literally life-threatening	Usually, warehouse data is a replica of the original data, running on a separate system so as not to affect performance of the database. Many systems create an overnight "snapshot" of data for this purpose. Since the data are re-created periodically, reliability of OLAP servers is less of a concern

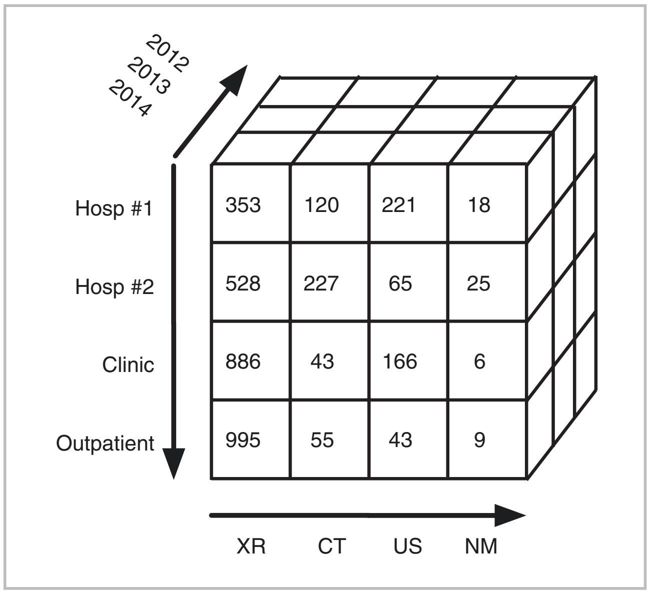


FIGURE 6-6

An example of an online analytic processing (OLAP) cube. This cube shows three dimensions. Along the X-axis is the study type. XR, X-ray; CT, computed tomography; US, ultrasound; NM, nuclear medicine. The Y-axis shows the location of the study. The Z-axis shows the year of the study.

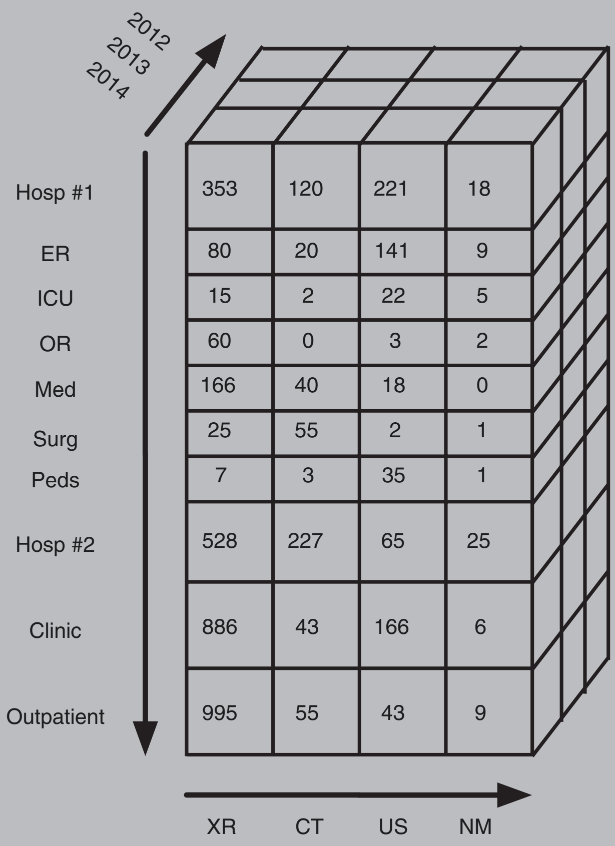
A **data mart** is a smaller slice of the original data, usually restricted to a particular line of business. For example, the radiology department may be given access to a data mart of radiology data, while housekeeping would use a different set of data pertaining to its needs.

Figure 6-6 is a representation of a **OLAP cube**, a multidimensional array used for gathering business intelligence. In this example, the data show the number of different kinds of radiology studies (X-Ray, Computed Tomography, Ultrasound and Nuclear Medicine) done in various clinical environments (Hospital #1, Hospital #2, Clinic and Outpatient) in the timeframe from 2012 to 2014. This type of analysis can help an administrator determine how to provision new equipment or advertise for existing services. In this representation, each axis is called a **dimension.** There is no limit to the number of dimensions in an OLAP cube, although it can be difficult to visualize more than three. Each box in the cube contains a number. For example, in diagram above, the bottommost left cube indicates that there were 995 X-Rays done in the outpatient department in 2014. The data in these cells are referred to as **facts** or **measures.**

There are several OLAP operations which are used to investigate data. **Drill-down** is the process of breaking categories into smaller segments. For example, if the user wants to look at monthly data instead of yearly data, we could break 2014 down into 12 months. The reverse of drill-down is **roll-up** where several categories are combined. For example, the user could combine data from the two hospitals in order to compare it to the outpatient settings. **Slicing** is looking at a subsection of the data when one of the dimensions is held constant. For example, if the user wanted to look at only X-Rays. **Dicing** is looking at a subset of the data based on constraints in more than one dimension. For example, if the user wants to look at only XR and CT studies which were done in the clinic and outpatient departments. A **Pivot** is when the axes of the cube are changed in order to provide a different view. (see Box 6-2)

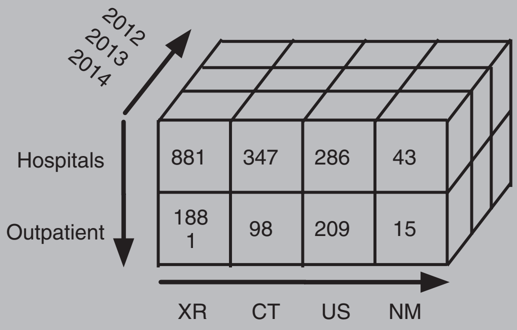
Box 6-2: Types of OLAP operations

Drill Down—Breaking one category into smaller chunks. Here we see that Hospital #1 has an Emergency Room, Intensive Care Unit, Operating Room, Medical service, Surgical service and Pediatrics service.



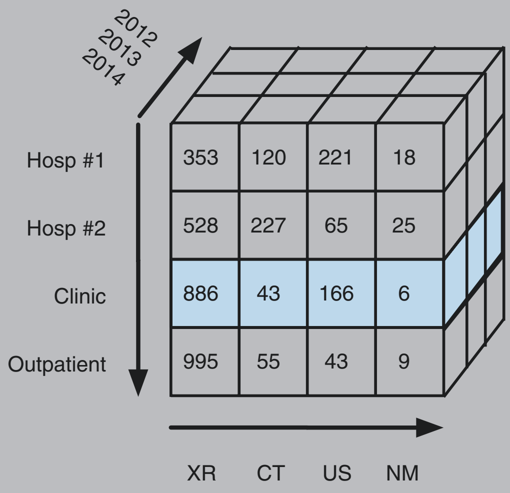
	XR	CT	US	NM
Hosp #1	353	120	221	18
ER	80	20	141	9
ICU	15	2	22	5
OR	60	0	3	2
Med	166	40	18	0
Surg	25	55	2	1
Peds	7	3	35	1
Hosp #2	528	227	65	25
Clinic	886	43	166	6
Outpatient	995	55	43	9

Roll-up—combining categories into larger chunks.

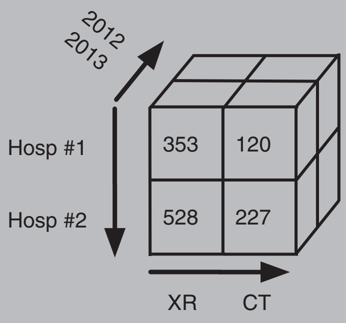


	XR	CT	US	NM
Hospitals	881	347	286	43
Outpatient	188 1	98	209	15

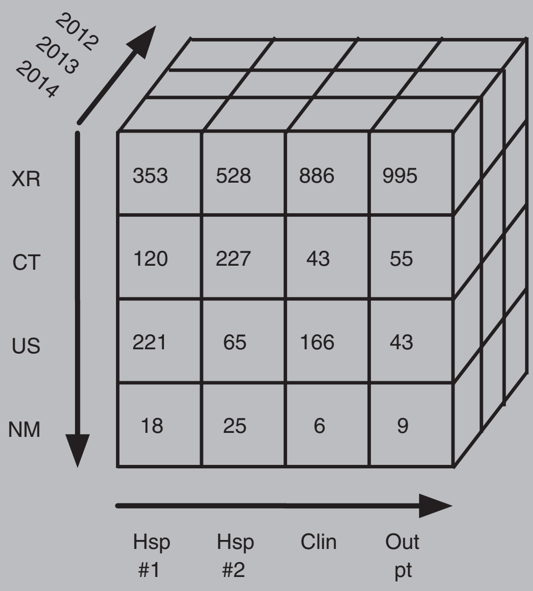
Slice—holding one dimension constant.



Dice—limiting data based on more than one dimension.



Pivot—rotating the data by changing axes.



3.1.2.2 Networks

Networking computers together efficiently is key for establishing interoperability. For further discussion, see Sect. 3.1.3.

3.1.2.3 Data/Database

As we have mentioned earlier, databases are computer programs which allow multiple users to have simultaneous controlled access to different types of data. These have been discussed in other sections.

3.1.3 NETWORKS

Computer networks are a set of protocols and transmission media which allow two or more computers to interact with one another. Networks can be as simple as two computers sharing files over a USB cable or can be as vast as the internet. Any device that can send or receive data is called a **node**. The path along which the data travels is called a **link**.

Some nodes are user nodes, such as computers or display terminals. Other nodes can be devices specifically designed for networking. A **hub** is a node that allows many other nodes to connect to it. Whenever it receives a signal from one node, it rebroadcasts it to all other attached nodes. A **switch** is similar to a hub, however it is more selective. When it receives a signal intended for a particular node, it rebroadcasts the signal only to that node. A **router** is a network device that functions similar to a switch, but it normally sits at the junction of two networks and decides which signals should be allowed to pass through. Some routers provide **Network Address Translation (NAT)** which allows computers to hide their actual identity in order to share a single public-facing address. This is very common in home and corporate networks.

The different aspects of the network are often referred to as **layers**, where the lowest levels deal with physical aspects of the network and the higher levels relate to interactions between programs. There are several models used to describe networks. The two most common are the *Open System Interconnect (OSI)* model and the *Transmission Control Protocol/ Internet Protocol* (TCP/IP) model. (See Fig. 6-7).

In the OSI model, the lowest level is the transmission media, or the **physical layer**. Common physical layers include radio frequency, copper wire and fiber optic. The **data link layer** rests just above this and accounts for correcting errors in the physical layer and

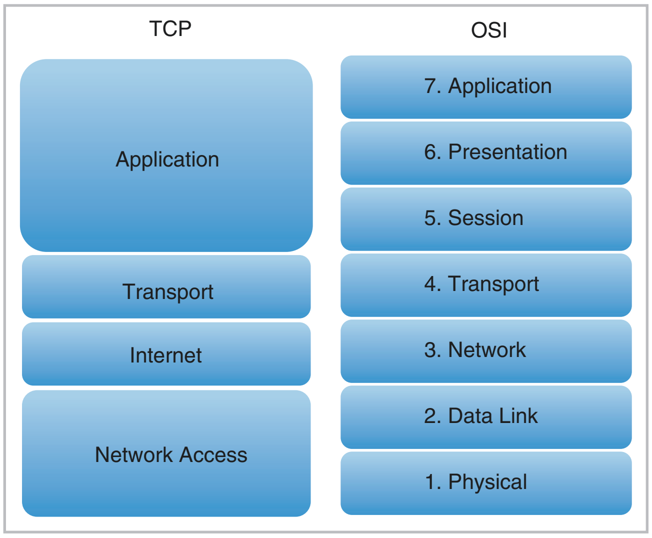


FIGURE 6-7

The Open System Interconnect (OSI) and Transmission control Protocol/Internet Protocol (TCP/IP) network models. OSI, right, contains seven layers. TCP/IP contains four

TABLE 6-5

COMPARISON OF DIFFERENT PHYSICAL NETWORK LAYERS

MEDIA	TYPICAL BANDWIDTH	PROS	CONS
Radio frequency	Cellular phones with 4G LTE can get 5–12 Mb/s; WiFi networks using the latest 802.11 ac are rated to 1.5 Gb/s. In practice, 70–100 Mb/s is common.	Nodes are easily moved and adding another node to the network is as simple as bringing it within range of the transmitter	Radio Frequency is highly subject to electromagnetic interference, and often can not penetrate into basements or through thick walls. As distance from the transmitter increases, bandwidth drops precipitously
Copper wire	Gigabit Ethernet can reliably provide 1Gb/s when using category 6 (CAT-6) cables	In commercial networking, Ethernet is ubiquitous and hardware is sold at commodity prices	Installing a new node requires running physical cables and/or purchasing hardware
Fiber optic	100 Gb/s on OTU-4 lines. Researchers have shown speeds of 1 Tb/s or more	Exceedingly fast and durable. Able to send long distances (>500 km) without repeaters.	Because of high cost, fiber is primarily used in network backbones and WANs

providing synchronization of data transport. Together, these two layers are called the **network access layer** in the TCP/IP model (Table 6-5).

The next layer up is the **network layer** which provides routing and switching capabilities. In the TCP/IP model, this is called the **internet layer.**

The next layer is the **transport layer** (it has the same name in both models). This layer employs error-checking algorithms to ensure that data is transported completely and correctly.

The **session layer** is responsible for opening and closing communications between software applications. The **presentation layer** transforms data into usable formats. For example, converting a binary data stream into a picture. In addition, encryption usually occurs at this layer. The top layer is the **application layer**, where all the higher-order interaction is done, such as authentication, user interaction and quality of service. Everything at this layer is application-specific. Web browsers, File Transfer Protocol (FTP), E-mail and all other applications function at this level.[4] In the TCP/IP model, these top three layers are all referred to as the application layer.

3.1.3.1 Topologies

A **network topology** refers to the map of connections between different nodes. The **physical topology** represents the physical location of nodes and the interconnects that run between them. In contrast, the **logical topology** refers to the way that signals pass from one node to another, regardless of the physical layout. For example, suppose that the finance department and the dietary department share office space. The network engineer connects all the computers with Ethernet cables, but installs specific network devices and protocols to ensure that the two departments' data remains separate.

There are several different network topologies.

Point-to-point is the simplest network topology, and involves two nodes and one link between them. An example is two computers connected with a USB cable. In telephony, a circuit-switching system establishes logical point-to-point network whenever a phone call is made. When the call is completed, the resources are returned to the network (Fig. 6-8).

4 Health Level 7 (HL7), a group responsible for many messaging standards in healthcare got its name from the top layer of the OSI model.

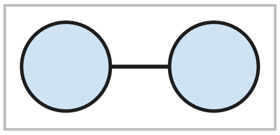


FIGURE 6-8

A simple point-to-point network

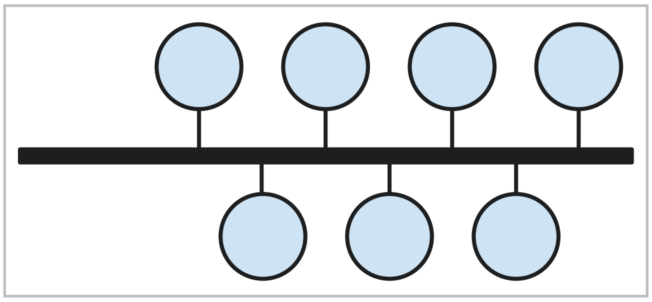


FIGURE 6-9

Network with Bus topology

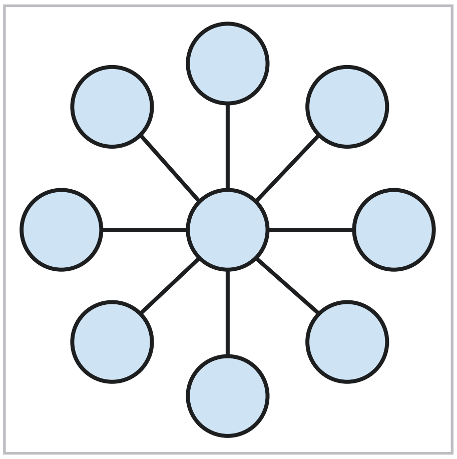


FIGURE 6-10

Star topology

In bus topology, all nodes are connected to a single wire, called the bus, or backbone. As the signal passes from one end to the other, each node examines the data to determine the appropriate recipient. The benefits to this topology are that it is relatively inexpensive, and adding nodes is relatively easy. The downside is that if the wire is cut, all nodes distal to the break are lost (Fig. 6-9).

In **star topology**, every node is connected to a single central node, called a hub. As with bus topology, nodes can be easily added or removed. However, in star topology, if the hub fails, the entire network fails (Fig. 6-10).

In **ring topology**, every node functions as a repeater to keep the signal moving between nodes. When the message reaches the node that originally sent the message, it is terminated. In unidirectional rings, messages travel in only one direction (e.g. clockwise or counter-clockwise). In bidirectional rings, messages are transmitted simultaneously in both directions. Although unidirectional rings use less bandwidth, they are prone to failure. If a single node fails, the whole network is affected. In a bidirectional ring, when one node fails, only that node is affected. When a ring is left open, it is sometimes called a **daisy-chain** (Fig. 6-11).

In a **mesh network**, each node is connected to multiple other nodes. Although this kind of network is very efficient and very secure, it becomes prohibitively expensive as the number of nodes increases. In a fully connected mesh (shown below) every node is connected to every other node. In a partially connected mesh, each node is connected to two or more other nodes, but not to every other node. The main benefit of a mesh network is that if any single node fails, the network remains intact. The internet is an example of an enormous mesh network (Fig. 6-12).

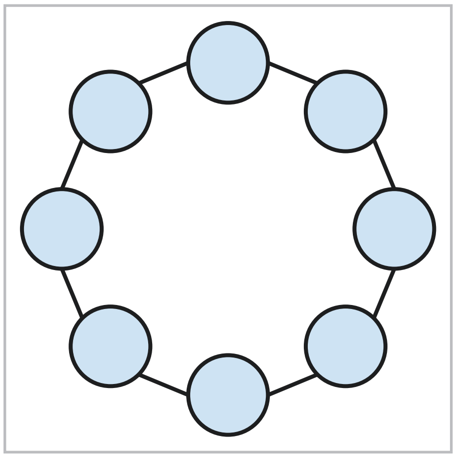


FIGURE 6-11

Network with ring topology

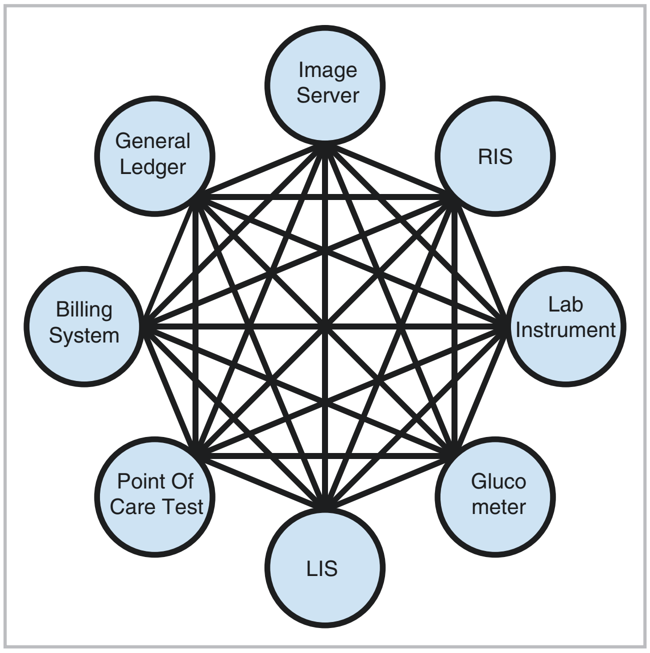


FIGURE 6-12

Network with mesh topology

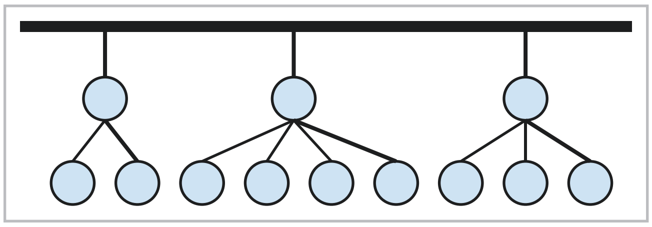


FIGURE 6-13

Hybrid network with tree topology (star networks linked with a bus/ backbone)

A **hybrid network** is a combination of two or more network topologies. For example, A **tree network** is a combination of star networks linked by a bus. This is a very common arrangement in large organizations, where each department has its own hub and all departmental computers connect to it. If the backbone fails, the departments become isolated from one another, but departmental nodes can still communicate with other nodes in the same department (Fig. 6-13).

Other common hybrid networks include the star-ring network, which is a series of star networks, connected by a ring, and the star-star network which is a star of stars (sometimes called a snowflake) (Fig. 6-14).

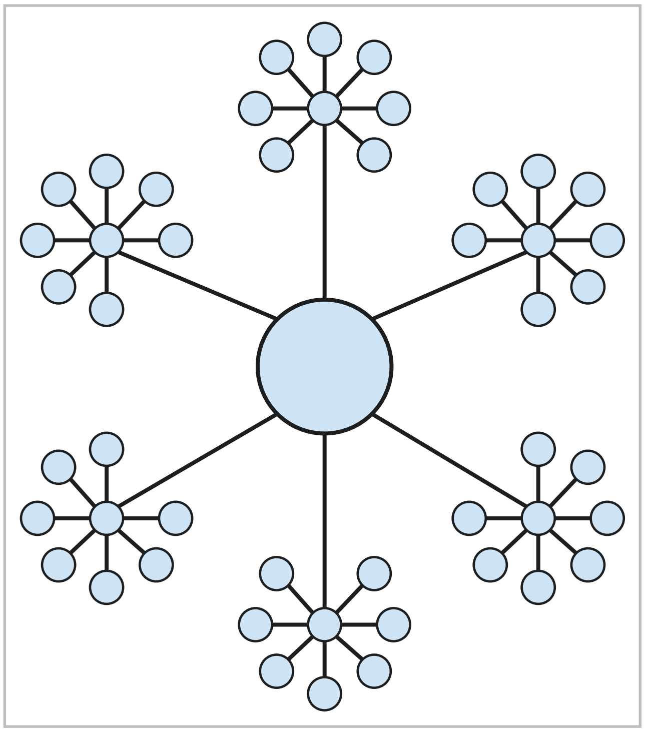


FIGURE 6-14

A star-star network, sometimes called a snowflake

3.1.3.2 Telecommunications

Telecommunication networks enable people and machines to communicate over long distances. In general, these are very large networks shared by many people. The **Public Switched Telephone Network (PSTN)** is the global network of telephones and switching systems. It provides an easy and generally inexpensive method for real-time voice communication. It is sometimes referred to as **Plain Old Telephone Service (POTS)**.

In the early days of telecommunication, when a person wanted to place a call, he would call an operator who would physically plug a jack into a receptacle to create a circuit. If the call required multiple connections, multiple operators were needed. If all the outgoing wires were in use, the call could not go through and the caller would be informed that all circuits are busy. More modern systems function very similarly, although the manual switching process has been replaced with digital switches. This process of sharing a finite number of circuits is called **circuit switching.** Since the circuit is used for only one conversation at a time, there is generally little lag between transmitting and receiving. The downside is that the entire circuit is reserved for one conversation, even if the line is completely silent.

Cellular phones and Voice-over-IP (VOIP) networks do not rely on physical circuits, but instead use **packet switching.** In this technology, each bit of the conversation is encapsulated into a packet of data, which is sent from station to station until it gets to the recipient. The recipient then reassembles the packets and converts them back into voice. Since the packets may arrive in any order, and can sometimes get lost, there is a variable delay between sending and receiving. In high quality local networks, VOIP can be indistinguishable from POTS. However, at the time of this writing, cellular phone communication has significantly more degradation that traditional land-lines.

3.1.4 SECURITY

Protecting patient information while still making it available to decisionmakers is a difficult, if not impossible task. Nearly every few weeks, there are reports of a large company being hacked and personal information being released. Healthcare organizations are required to safeguard patient information and prevent identity theft.

3.1.4.1 The HIPAA Security Rule and Other Government Regulations

In 1996, Congress passed the Health Insurance Portability and Accountability Act (HIPAA). Among the many provisions of the act, the **Privacy Rule** defined characteristics of Protected Health Information (PHI). PHI includes any *individually identifiable information* about the patient, including demographic information (e.g. name, birth date, social security number, address).

Another title in the act, the **Security Rule**, required that the department of Health and Human Services (HHS) develop policies to secure PHI. The Office for Civil Rights (OCR) was given responsibility to enforce the Security Rule and the Privacy Rule.

The Security Rule applies to organizations that transmit health information in electronic form, such as health plans, healthcare clearinghouses, and health care providers. Collectively, these groups are called **covered entities**.

The Security Rule specifically protects electronic protected health information (e-PHI) which includes all health information that a covered entity creates, receives, maintains or transmits in electronic form. Interestingly, the Security Rule does not apply to PHI transmitted verbally or in writing. Faxes are something of a gray area.

In the course of business, Covered Entities often need to enter into agreements with other organizations or **Business Associates** (BA). For example, a physician group (a covered entity) may hire an outside billing company to process claims. In these cases, the covered entity is only allowed to share the **minimum necessary** PHI for business purposes. In most cases, the BA is required to sign a contract called a **Business Associates Agreement** (BAA) with the covered entity to ensure that PHI will remain protected.[5] At a minimum, the BAA should

1. Establish the permitted uses of PHI
2. Prohibit the BA from disclosing PHI inappropriately
3. Establish safeguards to protect PHI
4. Report any breaches of security
5. Allow individuals access to their own PHI
6. Ensure that the BA be held to the same privacy standard as the covered entity itself
7. Allow HHS to review the BA's privacy practices
8. Ensure that the BA will destroy all data when the relationship with the covered entity ends
9. Require any of the BA's subcontractors to follow the same rules as the BA itself
10. Allow the covered entity to terminate the contract if violations occur

The Security Rule divides safeguards into three categories: Administrative, Physical and Technical. Some of the safeguards are mandatory while others are considered addressable, which means that the entity is allowed to make a cost/benefit decision as to whether they want to implement the safeguard or employ some reasonable alternative. In either case, the risk analysis must be thoroughly documented. For example, according to § 164.312(a)(2)(iii), computer systems must have an automatic logoff after a period of inactivity. If the EMR does not directly support automatic logoff, it could be very expensive to add that functionality. Instead, the covered entity decides to use the operating system's own password-based screen saver program. Although the user isn't logged off *per se*, the PHI remains protected, so the safeguard is addressed.

Administrative Safeguards establish policies and procedures within an organization to protect PHI. This process begins with regular risk assessment to identify and analyze vulnerabilities in the information systems. Staff training and compliance monitoring are key to enforcing privacy policies. Information access rules have to be defined so that each person only has access to the minimum necessary to do their job. Finally, there must be an emergency plan to respond to data leaks or lost data.

5 An example Business Associate Agreement (BAA) can be found at http://www.hhs.gov/ocr/privacy/hipaa/understanding/coveredentities/contractprov.html

Physical Safeguards refer to efforts put in place to limit physical access to data or workstations. Some examples include: security personnel to patrol data centers; locks and alarms on workstations; or computer monitor privacy filters which prevent onlookers from reading sensitive data. The covered entity also must have a plan regarding the transfer, removal, disposal, and re-use of electronic media, to ensure appropriate protection of electronic protected health information.

Technical Safeguards include hardware, software, and other technology that limits access to PHI. These controls fall into four basic categories: (1) **Audit Controls** keep track of which users are accessing what kinds of PHI. In the event that PHI becomes public, it is important to review the access logs to see who had access to the leaked information. (2) **Access Controls** restrict access to various systems to users who are authorized. (3) **Integrity Controls** prevent improper alteration or deletion of PHI. (4) **Transmission Security Controls** protect PHI when it is transferred over an electronic network. Specifically, this requirement demands that data is protected in motion and at rest.

Box 6-3: Is texting prohibited under HIPAA?

It is fairly common for physicians to communicate with their patients and with one another by sending messages through their smartphones or other devices. However, since HIPAA requires that PHI be protected in motion and at rest, this could potentially represent a violation. To address this, let's examine how texting works.

A text message passes through five phases during its lifecycle:

1. on the sender's device
2. enroute to the cell tower
3. transiently stored on the carrier's server while waiting for receipt confirmation
4. enroute to the recipient
5. on the recipient's device

Protecting the message while stored on the user's device (stage 1 and 5, above) is problematic. Most devices store copies of sent and received messages for an indefinite period of time. Both sender and receiver have to be careful to remove messages that are no longer needed. Many devices have lock screens which protect unauthorized users from reading their contents. In some cases, such as Apple's iOS 10, messages are automatically encrypted on the device so that even if the device were lost or stolen, it would be impossible for a hacker to read the messages. Android 5.0 uses full disk encryption (FDE) by default.

Protecting data in motion (stages 2 and 4) is easier. Short Message Service (SMS) messages which are sent over GSM networks are encrypted between the device and the cell tower, so that even if a hacker were situated right next to the sender or recipient with a radio receiver, he would only be able to intercept the encrypted message. Apple's iMessage protocol, a competitor to SMS, uses encrypted internet connections in place of cellular radio.

Messages stored by the carrier (stage 3) may not represent a violation, even if unencrypted: these messages are temporary; they are not routinely screened by the carrier; and they are not available on public networks.

So, is physician texting OK? Maybe.

The weakest points in this system are usually the physicians themselves. Forcing doctors to lock their phones has proven difficult, especially for the technically uninclined. Many organizations have opted for a third-party application that ensures end-to-end encryption at the cost of easy usability.

3.1.4.2 Firewalls

In construction, a firewall is a structural entity meant to keep a fire from spreading from one part of a building to another. In computing, a firewall is a device that sits at the junction between two networks and decides what kinds of information is allowed to pass through, thus protecting one network from another (Fig. 6-15).

When information is sent from one computer to another over a network, it is broken down into smaller packets. These packets are reassembled on the target computer to create the original message. (see packet switching, Sect. 3.1.3.2).

The simplest type of firewall is the **packet filter.** The firewall inspects each packet to ensure that it is coming from an acceptable source and is travelling to a permitted endpoint. This is a fairly swift operation because the list of acceptable computers is well known in advance. The weakness of this method is that the contents of the packet are never examined.

The next generation of firewalls were able to overcome this limitation by **stateful inspection** of packets. The packets are temporarily reassembled on the firewall itself and examined for prohibited information. In healthcare organizations, this kind of firewall can be used to prevent users from visiting unauthorized web sites, downloading viruses and playing online games. Since the firewall operates in both directions, it can also be used to prevent users from inadvertently disseminating protected health information. Most modern firewalls employ a combination of stateful inspection and simple packet filtering.

Network Address Translation (NAT) is a process by which computers in a private network are able to share a public-facing address without divulging their local address. Although this function is normally performed by a **router**, many firewalls incorporate this functionality as well. In this scenario, the firewall has a public IP address. Whenever any of the computers on the private network want to connect to a public site, they send packets to the router, which replaces the actual origin address with its own. When the public site returns information, it is replies to the router, which then transmits the packet back to the local machine. In a standard configuration, uninvited packets are simply rejected by the router and can not get into the private network.

3.1.4.3 Virtual Private Networks

Many healthcare networks are closed networks, which means that they can not be easily accessed from outside the walls of the institution. A **virtual private network** (VPN) allows a remote computer to access an otherwise closed network by making it appear as though the

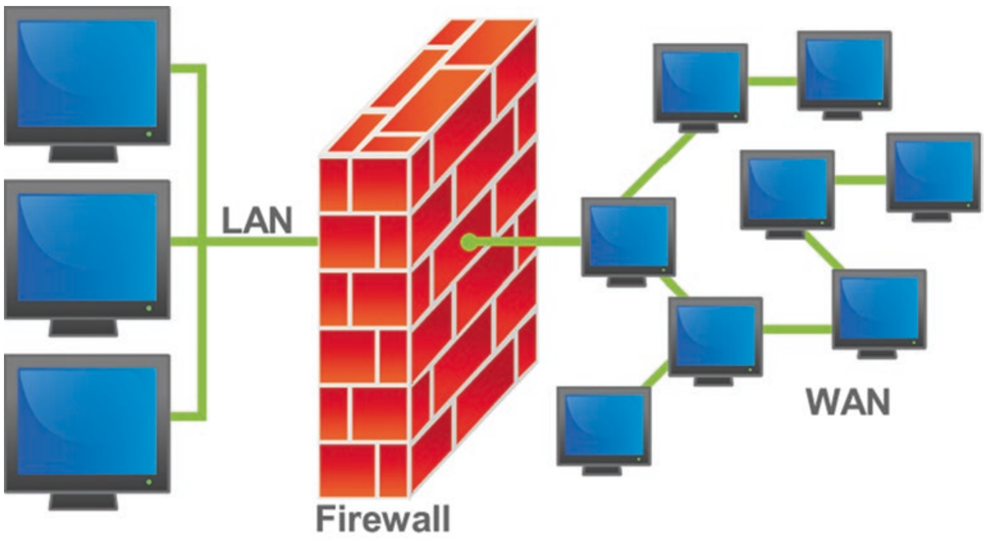


FIGURE 6-15

A firewall separates the LAN from the WAN. Image: "Firewall" by Bruno Pedrozo–Feito por mim. Licensed under CC BY-SA 3.0 via Wikimedia Commons–http://commons.wikimedia.org/wiki/File:Firewall.png#/media/File:Firewall.png remote computer has an address in the local network. The communication between the remote computer and the network is encrypted and can travel in the public internet without fear of interception.

3.1.4.4 Encryption

When private networks are unavailable, health information must be transmitted over public networks such as the internet. In order to prevent interception of sensitive information, data are encrypted by the sending computer and decrypted by the receiving machine. In all cases of cryptography, the original message (**plaintext**) is only known to the sender and recipient, while the encoded message (**ciphertext**) is widely available. In addition, there may be one or more secret keys which are known only to sender or recipient (Fig. 6-16).

One of the simplest forms of cryptography is simple substitution. Each letter of the alphabet is substituted for another letter using a predetermined plan.[6] According to legend, Julius Caesar used this method, and a Caesar rotate is named for him.

For example, a coded message may read as follows.

```
Lqirpdwlfv
```

Before the message was sent, the recipient was given the secret key of 3, which tells him to shift the letters ahead by three and create the following dictionary.[7]

Cipher:	`DEFGHIJKLMNOPQRSTUVWXYZABC`
Plain:	`ABCDEFGHIJKLMNOPQRSTUVWXYZ`

As you can imagine, this method is fairly easy to break, as there are only 25 different encrypted possibilities. If a hacker tried each one, it would take a short time to have the original message. A slightly more difficult method involves substituting letters in random order (instead

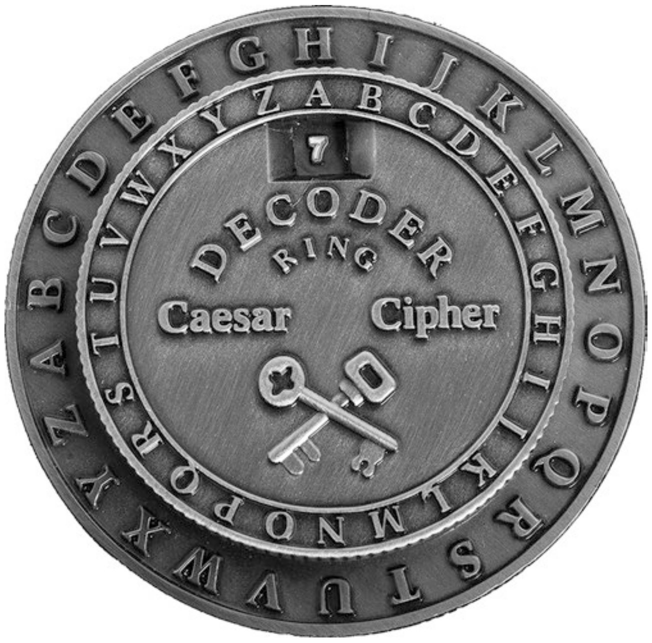


FIGURE 6-16

Decoder ring. Photo courtesy Retro Works, shopretroworks.com

6 Think of the secret decoder ring found in cereal boxes.
7 The plaintext is: Informatics.

of alphabetical order). In this case, the secret key would be much longer than just a number. This complexity results in 4×10^{26} different possibilities, which sounds much harder to break.

Unfortunately, it isn't. Analysis of common texts show that certain letters appear much more frequently than others. Using this well known distribution, a series of educated guesses will allow a hacker to decode a longer message fairly easily.[8]

Modern cryptography employs a combination of substitution, transposition and mixing of plaintext in order to create the ciphertext. In order to make it more difficult to crack, the newly created ciphertext is then passed through the encryption algorithm several times to make it harder to break. After multiple rounds of encryption, the message can be nearly impossible to break.

Advanced Encryption Standard (AES) is a common encryption mechanism. It requires that the sender and recipient possess the same 256-bit key. One way to break this kind of security is to try every possible password (called the **brute force method**). There are approximately 10^{77} possible 256-bit keys. AES was designed so that encryption process is fairly quick, the decryption process takes a significant fraction of a second to complete. As a result, it would take 3×10^{69} years to crack an AES encrypted message.

Another useful type of encryption is called **public key cryptography** or **asymmetric key cryptography.** In this method, the key is composed of two parts. One is the public key, which is well known and the other is the private key, which is known only to the sender. While the two keys are mathematically linked, it is impossible to derive one from another. The public key is used for encryption, while the private key is used for decryption. For example a person could give out his public key and anyone who wants to send him a message could do so securely.

Asymmetric keys can also be used to verify a **digital signature.** For example, a person could use his private key to encrypt a document. If the public key is able to decrypt the document, that proves the authenticity of the original signer.

3.1.5 DATA

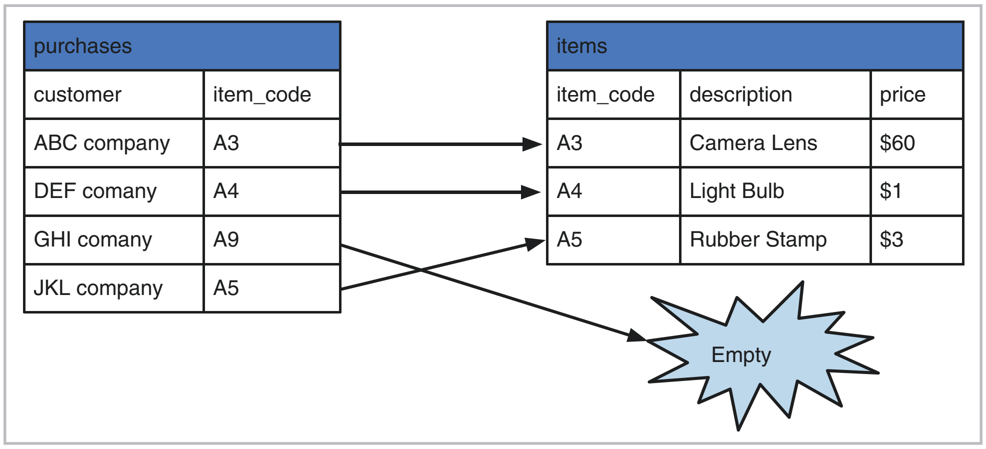
3.1.5.1 Integrity

Data integrity reflects the degree to which we can trust our data. There are two aspects to data integrity. **Physical integrity** refers to the way information is stored on various media. When media is subject to physical insults, the data can become corrupted. One way to protect the data is to employ redundant data stores, such as a Redundant Array of Inexpensive Disks (RAID). **Logical integrity** means that the data makes sense and is appropriate to our needs. For example, if a thermocouple is malfunctioning, an oral thermometer may register a temperature of 300°. Even if the database records the information faithfully, it is logically impossible.

Database integrity includes three special integrity requirements in order to guarantee consistency. **Entity integrity** requires that every table in the database has a unique primary key. This is not always enforced. **Referential integrity** requires that whenever a database column refers to a row in another table, that row exists. **Domain integrity** specifies a specific list of values that are acceptable for a particular column. For example, we can preserve the logical integrity of our temperature field by restricting input values to the range of 20–40 °C.

Consider the following example, shown in figure 6-17. A database contains two tables. The *purchases* table lists the customers and the items that they purchased. The *items* table shows a description of the items. The two tables are related by the common column item_code. The problem with this database is that the GHI company has purchased an item with code A9, but there is no corresponding item in the items table. This database lacks referential integrity because an item in the purchases table refers to a row in the items table which does not exist.

8 For a great demonstration of a computer program to solve cryptoquotes, see http://quipqiup.com/. Note that this works for longer messages, but shorter messages are much harder to solve. For example, all the following are possible plaintexts to the ciphertext Lqirpdwlfv: Objections; Speciously; Spaciously; Operations; Exchangers.

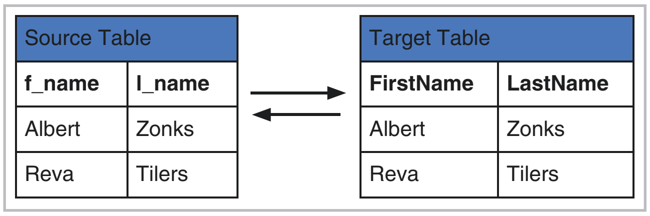


purchases	
customer	item_code
ABC company	A3
DEF comany	A4
GHI comany	A9
JKL company	A5

items		
item_code	description	price
A3	Camera Lens	$60
A4	Light Bulb	$1
A5	Rubber Stamp	$3

FIGURE 6-17

The purchases table has an item with code A9, but this item doesn't exist in the items table, a violation of referential integrity

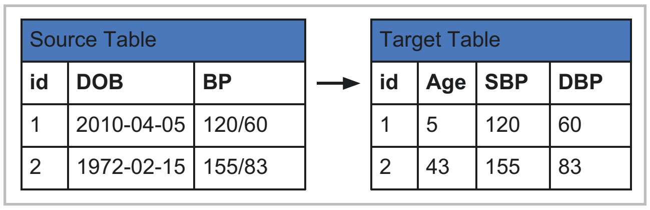


Source Table	
f_name	**l_name**
Albert	Zonks
Reva	Tilers

Target Table	
FirstName	**LastName**
Albert	Zonks
Reva	Tilers

FIGURE 6-18

Direct mapping between source and target tables



Source Table		
id	**DOB**	**BP**
1	2010-04-05	120/60
2	1972-02-15	155/83

Target Table			
id	**Age**	**SBP**	**DBP**
1	5	120	60
2	43	155	83

FIGURE 6-19

Mapping from date of birth to age is possible, but the reverse is not

3.1.5.2 Mapping

Data Mapping is the process by which data is compared or transferred between two different data models. In some cases, the mapping is fairly straightforward, such as when similar elements exist in both models and simply require changing the field name. When this is possible, bidirectional mapping is possible (Fig. 6-18).

In other cases, mapping can be more complicated and require calculation. In the example below, the age (in the target table) can be calculated from the current date and the date of birth (in the source table). The Blood pressure can be separated into systolic and diastolic measures. It can be tricky to provide bidirectional support for this mapping because it depends on knowing the date on which the conversion took place in order to calculate the age. In this case, the age is recorded as an integer, so the exact DOB would be impossible to determine (Fig. 6-19).

Sometimes, the data themselves prohibit bidirectional mapping. Systematized Nomenclature of Medicine-Clinical Terms (SNOMED) is a vast terminology which describes many different clinical entities. The International Classification of Diseases, ninth edition (ICD9) describes diagnoses, but has much fewer terms than SNOMED, so that a single ICD9 term may map to many SNOMED terms and sometimes vice-versa. In these cases, an approximate mapping can be done, but there is no guarantee that the reverse mapping will be relevant.

3.1.5.3 Manipulation (e.g., Querying, SQL, Reporting)

The standardized query language (SQL) is a relatively simple computer language for interacting with databases. There are four basic functions for manipulating data, Create, Read, Update and Delete. These are commonly abbreviated as CRUD. For some more examples of SQL, see Sect. 3.1.1.6

OPERATION	SQL EXAMPLE	EXPLANATION
Create	`INSERT source (id, dob, bp) VALUES (3, ’1937-04-11’, ‘166/75’);`	Insert a new row into the table named "source" with id=3, blood pressure and DOB as shown.
Read	`SELECT id, dob FROM source;`	Show the id and DOB for all patients in the table named "source"
Update	`UPDATE source SET dob = '1955-08-14' WHERE id = 2;`	Update the row of the person who has id=2 and set the birthday to 8/14/1955
Delete	`DELETE FROM source WHERE id = 3;`	Remove the row from the table named "source" for the person who has id=3

Database reporting is one of the most important professional tasks the informaticist will perform. Being able to locate the right piece of data, ensuring that it is logically correct and reflective of the real world is vitally important to making decisions and measuring outcomes. Once the query has been written, it can be re-run periodically to determine if a clinical or operational change has a desired result.

3.1.5.4 Representation and Types

Data exists in many different formats within a computer. The simplest form of data is a whole number (**integer**). In order to store a number, we have to allocate a certain amount of memory for it. If we know in advance that a number falls into a certain range, we can specify how many bytes of data are required to store it. For example, suppose we want to record the number of children a person has and store it in a database. A negative number is impossible; a fraction is impossible, and a number greater than 30 is probably a data error. So what container should we use? In Standard Query Language (SQL), a **tinyint** is defined as a whole number between −127 and +128. Since there are 256 distinct possibilities (2^8), this number can be stored in 8 bits, or one byte[9], and this would be the best answer to our needs.

Now, suppose we also want to record the population of various countries. Again, we can not accept numbers less than zero and we can not accept fractions. The maximum value is going to be very high, as both China and India have over 1 billion residents. Fortunately, there exist larger integer type, such as an **unsigned int** which allows whole numbers between 0 and 4,294,967,295. This represents 2^{32} different possibilities, and thus requires 4 bytes. Different languages and different systems have many options of varying sizes for storing integers[10]. In general, the programmer should choose the data

9 Truthfully, we could express this number using only 5 bits, since $2^5 = 32$, but there is usually little benefit of using only part of an 8-bit byte for storage. It does not improve calculation time or storage requirements.
10 Gangam Style by PSY is the most-watched video on Youtube. When the creators of Youtube initially created their video database they used an int to store the number of times a video is watched. In October, 2014, Gangam surpassed 2.1 billion views and exceeded the available storage capacity, resulting in data

type which requires the smallest amount of space but whose range includes all reasonable values.

What if I wanted to record the mass of different elements or mean temperatures? Real numbers (or **floating point numbers**) are usually represented as a number times a base raised to an exponent. For example, the number 350.15 could be expressed as 3.5015×10^2. In this example, 3.5015 is the **mantissa** or **significand**; 10 is the **base** and 2 is the **exponent**. If we assume a convention where the decimal point is always to the right of the first digit and the base is always 10, this number could be represented as two integers: 35,015 and 2. By varying the number of significant digits available for the significand and the exponent, we can express a wide range of numbers. It is important to remember that this format does not always represent numbers exactly. For example, if you only allow 4 significant digits to the significand, you will find that the number 250,000 is stored the same as 250,001. In practice, a **single precision floating point** number occupies 32 bits and provides 7 significant digits and an exponent up to 127.

Letters and words are also commonly stored as data. The American Standard Code for Information Interchange (**ASCII**, see Table 6-6) encodes the 100 or so most commonly used letters, numbers and punctuation to particular numbers. In this method, one byte corresponds to one letter. **Unicode** is a much more encompassing system and includes 1,114,112 characters, with the more common characters requiring one byte and the more esoteric characters using up to four bytes. While ASCII is limited to Latin-based languages, Unicode is able to represent most known languages, including pictorial languages such as Chinese. For example, consider the string of English letters in the word Informatics. This can be expressed in ASCII as the following series of numbers: 73, 110, 102, 111, 114, 109, 97, 116, 105, 99, 115.

Screen colors are commonly expressed as the value of their Red, Green and Blue (RGB) components, where each component is completely off (0) or completely on (255) or somewhere in between. For example, bright red would be (255,0,0); black, the absence of color is (0,0,0) and white is the combination of all colors (255,255,255).

Using this methodology, it is possible to describe a photograph by dividing the picture into a sufficient number of small boxes (picture elements, or **pixels**). Similarly, a movie can be encoded as a series of still images. Sound can be encoded by digitally sampling the audio in discrete timeframes.

Nearly any kind of information can be expressed as a combination of the above elemental data types. Documents, drawings and even computer programs can be encapsulated and stored this way.

3.1.5.5 Warehousing

A data warehouse is an intentional abstraction of a database which is organized for easy reporting. The data warehouse is often designed with particular queries in mind and organized for maximum efficiency. In addition, it is often created so that the user can interact with the data without needing specific database programming skills. See Sect. 3.1.2.1 for more information on data warehouses and datamarts.

3.1.5.6 Data Mining and Knowledge Discovery

Data mining is a process meant to explore large volumes of data (sometimes called **big data**) looking for patterns or relationships among different variables. In contrast to most biomedical research, there is no hypothesis to be tested by data mining. Instead, the goal is to look for correlations and then work backward to see if causation exists.

errors. Google (the owner of youtube) was forced to rework their database and change the storage capacity from int which is 4 bytes to bigint which is 8 bytes, effectively raising the limit to 1.8×10^{19} (one quintillion).

TABLE 6-6

ASCII TABLE

ASCII	HEX	CHAR	ASCII	HEX	CHAR	ASCII	HEX	CHAR	ASCII	HEX	CHAR
0	0	NUL	32	20	(spc)	64	40	@	96	60	`
1	1	SOH	33	21	!	65	41	A	97	61	a
2	2	STX	34	22	"	66	42	B	98	62	b
3	3	ETX	35	23	#	67	43	C	99	63	c
4	4	EOT	36	24	$	68	44	D	100	64	d
5	5	ENQ	37	25	%	69	45	E	101	65	e
6	6	ACK	38	26	&	70	46	F	102	66	f
7	7	BEL	39	27		71	47	G	103	67	g
8	8	BS	40	28	(	72	48	H	104	68	h
9	9	TAB	41	29	)	73	49	I	105	69	i
10	A	LF	42	2A	*	74	4A	J	106	6A	j
11	B	VT	43	2B	+	75	4B	K	107	6B	k
12	C	FF	44	2C	,	76	4C	L	108	6C	l
13	D	CR	45	2D	-	77	4D	M	109	6D	m
14	E	SO	46	2E	.	78	4E	N	110	6E	n
15	F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Data mining includes three stages: exploration, model building and deployment.

The **exploration** phase involves taking a good look at a sample of the data to identify important variables and to make sure that the data is accurate enough for analysis. Since the source of the data is often vast, the informaticist must make special attention to narrow the scope of data before embarking on further analysis. The process of **model building** includes the competitive evaluation of models which compares various data models on the basis of their predictive ability. When the best model is chosen, it must be validated to ensure that it is able to produce stable results across many different samples. **Deployment** involves using the model selected above and applying it to new data to make predictions.

3.1.6 TECHNICAL APPROACHES THAT ENABLE SHARING DATA

Isolated data is practically useless. When that data is shared among patients, providers, administrators, consultants and public health agencies, it becomes useful and valuable for making important decisions.

3.1.6.1 Integration Versus Interfacing

There are two methods for sharing data between software applications. **Integration** means that the two applications share the same data store. **Interfacing** means that they are able to communicate with one another via some messaging system and can synchronize their data when needed. The communication system is sometimes called a **bridge.**

	INTERFACE	INTEGRATION
What is it?	A messaging system that exists between two software applications enabling them to share data. Interfaces typically use a standard file format, such as XML or HL7	Two applications that share the same data store
Benefits	Consumers can choose "best-of-breed" software tailored to their needs, and synchronization occurs when needed	Both software applications have access to real-time data, so that reports from one system should be equivalent to reports from the other
Pitfalls	Data may be out of sync. The mapping between systems may not be robust enough and the interface may not include all data points needed	Unifying a database is often very costly, and may destroy backwards-compatibility with the original applications

Deciding between an integrated system versus an interfaced system can be difficult. Here are some questions to help decide:

1. Is there particular functionality in the "best-of-breed" system that can not be replicated in the integrated system? If so, interface may be better.
2. Is there a requirement for real-time data? For example, suppose you are choosing a Human Resources and Payroll systems. Since checks are written only once every 2 weeks, the payroll synchronization process is run infrequently, and an interface would be adequate. However, in an EMR system, vital signs should be updated every few minutes, an integrated system may be better.
3. Is isolation required? Are there certain parts of the database that should not be shared because of HIPAA or some other requirement? If so, a selective interface may be better.

3.1.6.2 Dealing with Multiple Identifiers

Whenever patient data is inserted into a database, there are certain fields that are used as unique patient identifiers. For example, a social security number[11] or a medical record number usually identifies a particular person for their lifetime. When comparisons are made within a single health system, these identifiers are usually adequate to uniquely identify a person. These identifiers are stored in a **Master Patient Index (MPI)** or an **Enterprise Master Patient Index (EMPI).** However, when patient data are shared with other systems

11 Social Security Numbers (SSN) are commonly used to identify patients, but they have some drawbacks. Firstly, they are not always unique. Until the 1970s SSNs were not issued to non-workers, and the spouse of a worker would usually use the same number as the worker. In addition, SSN are commonly used in financial transactions such as applying for a credit card, and a data breach could lead to identity theft. For these reasons, many healthcare organizations (such as Kaiser Permanente) do not use SSN to identify patients.

(for example, with a regional health information exchange), the problem of matching and identifying patients can be more tricky.

One solution to this problem is to create a national patient health identifier, such as exists in Denmark. This has failed to gain traction in the US because of privacy and security concerns. In 2014, the Office of the National Coordinator commissioned a report[12] to address methods of patient matching. The two key recommendations included placing emphasis on accurate initial data collection and industry standardization of data identity elements.

There are several ways to make sure data are entered correctly

1. Rigorous training and or certification for registrars
2. Using data entry forms which require the operator to select from a set of validated options instead of allowing free text.
3. Allowing the patient to review the data before it is committed to the database
4. Maintaining a list of abbreviations and diminutive forms to assist in duplicate detection (e.g. William would match Will, Willy, Bill, Billy, etc)

For the second objective, there is no "official" identity data set, but some of the common items are shown below. Note that primary data relates to things that most people have, are unlikely to change, or are unlikely to forget. Secondary items are those that are more variable or less reliable.

PRIMARY	SECONDARY
Legal Name	Date of Death
Date of Birth	Birthplace
Sex	Birth order
Race	Marital status
Ethnicity	Other Phone numbers
Primary Phone Number	Social Security Number
Previous Names	Driver License Number
Mother's Maiden Name	Passport Number
Street Address	E-mail address
	Biometric information

A probabilistic matching algorithm would try to match as much as data as possible, placing more weight on primary than secondary data elements, in order to identify potential duplicates. In some cases, these algorithms can be very complex.

3.1.6.3 Anonymization of Data

In order to permit the secondary use of PHI, data must be de-identified. Section 164.514 of the HIPAA Privacy Rule states that "health information that does not identify an individual and with respect to which there is no reasonable basis to believe that the information can be used to identify an individual is not individually identifiable health information."

There are two methods to meet the HIPAA Privacy rule. The first is the **expert determination** method. An expert is defined as a person with appropriate knowledge and experience in statistics and scientific methods for de-identifying data. If this expert deems that the risk of identifying an individual is very small, the standard is met. It is notable that the term "very small" is not clearly defined.

When using the expert determination rule, emphasis will be placed on characteristics of the population as a whole as well as the population being studied. For example, a patient identified with a first name of Muhammad may be unique in Montana[13] but fairly common in Saudi Arabia.

Another point to take into account is the availability of public databases. Since vital records (birth, death, marriage) are widely available in most states, it may be easy to

12 http://www.healthit.gov/sites/default/files/patient_identification_matching_final_report.pdf
13 Although hardly a canonical reference, see http://names.whitepages.com/first/Muhammad

re-identify a patient whose birthdate and zip code are known. Health plan numbers and patient account numbers that exist within proprietary databases and are harder to collect, however data breaches have been known to exist and re-identification is not impossible. Similarly, at one point, it was assumed that there could never exist a database of IP address or hardware MAC address. That is, until Google was caught collecting this data through its network of Street View vehicles.[14]

The second, and more common, solution to de-identifying data is the **Safe Harbor** method, in which the data is considered de-identified when all of the following identifiers are removed.

1. Names
2. All geographic subdivisions smaller than a state, including street address, city, county, precinct, ZIP code, and their equivalent geocodes, except for the initial three digits of the ZIP code if, according to the current publicly available data from the Bureau of the Census:

   (a) The geographic unit formed by combining all ZIP codes with the same three initial digits contains more than 20,000 people; and
   (b) The initial three digits of a ZIP code for all such geographic units containing 20,000 or fewer people is changed to 000

3. All elements of dates (except year) for dates that are directly related to an individual, including birth date, admission date, discharge date, death date, and all ages over 89 and all elements of dates (including year) indicative of such age, except that such ages and elements may be aggregated into a single category of age 90 or older
4. Telephone numbers
5. Fax numbers
6. Email addresses
7. Social security numbers
8. Medical record numbers
9. Health plan beneficiary numbers
10. Account numbers
11. Certificate/license numbers
12. Vehicle identifiers and serial numbers, including license plate numbers
13. Device identifiers and serial numbers
14. Web Universal Resource Locators (URLs)
15. Internet Protocol (IP) addresses
16. Biometric identifiers, including finger and voice prints
17. Full-face photographs and any comparable images
18. Any other unique identifying number, characteristic, or code

De-identified health information created following these methods is no longer protected by the Privacy Rule because it does not fall within the definition of PHI.

14 Kravets D. An Intentional Mistake: The Anatomy of Google's Wi-Fi Sniffing Debacle. WIRED 2012. https://www.wired.com/2012/05/google-wifi-fcc-investigation/