

MEMORY STORAGE CALCULATIONS

Professor Jonathan Eckstein (adapted from a document due to M. Sklar and C. Iyigun)

An important issue in the construction and maintenance of information systems is the amount of storage required. This handout presents basic concepts and calculations pertaining to the most common data types.

1.0 BACKGROUND - BASIC CONCEPTS

Grouping Bits - We need to convert all memory requirements into bits (b) or bytes (B). It is therefore important to understand the relationship between the two.

A bit is the smallest unit of memory, and is basically a switch. It can be in one of two states, "0" or "1". These states are sometimes referenced as "off and on", or "no and yes"; but these are simply alternate designations for the same concept. Given that each bit is capable of holding two possible values, the number of possible different combinations of values that can be stored in n bits is 2^n . For example:

1 bit can hold $2 = 2^1$ possible values (0 or 1)
 2 bits can hold $2 \times 2 = 2^2 = 4$ possible values (00, 01, 10, or 11)
 3 bits can hold $2 \times 2 \times 2 = 2^3 = 8$ possible values (000, 001, 010, 011, 100, 101, 110, or 111)
 4 bits can hold $2 \times 2 \times 2 \times 2 = 2^4 = 16$ possible values
 5 bits can hold $2 \times 2 \times 2 \times 2 \times 2 = 2^5 = 32$ possible values
 6 bits can hold $2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^6 = 64$ possible values
 7 bits can hold $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^7 = 128$ possible values
 8 bits can hold $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^8 = 256$ possible values
 ⋮
 n bits can hold 2^n possible values
 ⋮

Bits vs. Bytes - A byte is simply 8 bits of memory or storage. This is the smallest amount of memory that standard computer processors can manipulate in a single operation. If you determine the number of bits of memory that are required, and divide by 8, you will get the number of bytes of memory that are required. Similar, to convert from bytes to bits, you must multiply by 8.

Standard Datatypes - Many standard kinds of data occupy either 1, 2, 4, or 8 bytes, which happen to be the data sizes that today's typical processor chips are designed to manipulate most efficiently.

- 1 byte = 8 bits:
 - A single character of text (for most character sets). Thus, an MS Access field with datatype *Text* and field width n consumes n bytes. Example: *Text(40)* consumes 40 bytes.
 - A whole number from -128 to +127. This is what you get in the MS Access *Number/Byte* datatype
 - A whole number from 0 to 255
 - MS Access *Yes/No* fields also consume 1 byte. In principle, you only need a single bit, but one byte is the minimum size for for a field.
- 2 bytes = 16 bits, or two bytes:
 - A whole number between about -32,000 and +32,000; this is MS Access' *Number/Integer* datatype, often also called a "short" integer
 - A single character from a large Asian character set
- 4 bytes = 32 bits:
 - Can hold a whole number between roughly -2 billion to +2 billion. This is MS Access' *Number/Long Integer* datatype
 - A "single precision" floating-point number. "Floating point" is basically scientific notation, although the computer's internal representation uses powers of 2 instead of powers of 10.

This is MS Access' *Number/Single* datatype, with the equivalent of about 6 decimal digits of accuracy.

- 8 bytes = 64 bits:
 - Can hold a “double precision” floating-point number with the equivalent of about 15 digits of accuracy. This is MS Access *Number/Double* datatype, and is the most common way of storing numbers that can contain fractions.
 - Really massive whole numbers (in the range of + or – 9 quintillion). This is essentially the way MS Access stores the following datatypes
 - *Date/Time*
 - *Currency*.

Multiplier Prefixes - Memory requirements can become huge, and standard metric-system prefixes are utilized to keep the ultimate value manageable, according to the usual metric system:

1 kilobit (kb) or kilobyte (kB) = 1000 bits or 1000 bytes, respectively

1 megabit (Mb) or megabyte (MB) = 1000 kilobits or 1000 kilobytes, respectively

1 gigabit (Gb) or gigabyte (GB) = 1000 megabits or 1000 megabytes, respectively

1 terabit (Tb) or Terabyte (TB) = 1000 gigabits or 1000 gigabytes, respectively

Because computers tend to work in powers of 2, computer engineers have taken liberty with the above by substituting the multiplier $1024 (= 2^{10})$ for 1000. As a result, for many applications:

1 kilobit (kb) or kilobyte (kB) = 1024 bits or 1024 bytes, respectively

1 megabit (Mb) or megabyte (MB) = 1024 kilobits or 1024 kilobytes, respectively

1 gigabit (Gb) or gigabyte (GB) = 1024 megabits or 1024 megabytes, respectively

1 terabit (Tb) or Terabyte (TB) = 1024 gigabits or 1024 gigabytes, respectively

We'll call these two different systems “decimal-style” and “binary-style”, respectively. Which one gets used depends on the convention for marketing or measuring a particular component.

When you buy a 128 MB RAM chip for a computer, you actually get 128 binary megabytes, or about 134.22 million (128 MB x 1024 KB/MB x 1024 B/KB). Your computer BIOS will read the RAM as 128 MB (134.22 / (1.024 x 1.024)). When you buy a 15 GB hard drive, however, you might well get 15 *decimal* gigabytes, so when the drive is formatted, your computer's operating system might state its size as 13.97 *binary* GB (15 / (1.024 x 1.024 x 1.024)). You haven't lost 1 GB; the size was measured using two different systems.

2.0 DETERMINING MEMORY REQUIREMENTS FOR A SINGLE UNIT

Each type of data has its own specialized name for its "unit", as follows:

Numeric data is typically stored in the following standard formats (see above)

- Byte (8 bits/1 byte): whole numbers from 0 to 255, or –128 to +127
- “Short” (16 bits/2 bytes), or what Access calls “integer”: whole numbers from approximately –32,000 to +32,000
- “Long integer” (32 bits/4 bytes): whole numbers from approximately –2 billion to +2 billion
- “Single: or “float” (32 bits/4 bytes): scientific notation numbers with approximately 6 digits of precision
- “Double” or “double precision” (64 bits/8 bytes): scientific notation numbers with approximately 15 digits of precision

For textual data, a unit is a "character"; a file consists of a number of characters.

For picture data, a unit is a "dot" or "pixel"; a file consists of a number of dots or pixels.

For sound data, a unit is a "sample"; a file consists of a number of samples.

For video data, a unit is a "frame"; a file consists of a number of frames (each frame is a picture)

Despite the different names, the manner by which we calculate the memory requirements of a "unit" of data is similar.

2.1 TEXTUAL DATA - UNIT SIZE

The storage requirement for a single character (letter, number, punctuation mark, and symbol) depends upon the size of the character set used. Each character have a unique representation, so the larger the character set, the larger the memory requirement for each character in the set.

- Early character sets (containing 26 capital letters, 10 numbers, a space character, and assorted punctuation) allowed 64 characters. A popular example is "ASCII 64". These sets require 6 bits per character. Because of the need to include punctuation and/or special symbols in the character set, 6-bit character sets cannot differentiate between small and capital letters, and are now virtually unused.
- Current western character sets contain either 128 or 256 characters, requiring either 7 or 8 bits per character. Each is typically stored in one byte (even if only 7 bits are used).
 - There are two standards for representing characters, ASCII (used most places) and EBCDIC (used only on some "mainframe" equipment of older design). In ASCII, for example, the character "3" is represented by 00110011, "A" by 00100001, "b" by 01100010, and "\$" by 00100100.
 - "Markup" information like font family, bold, italic, and so forth must be represented separately – it is not part of the basic 128/256 character set
- Some Asian character sets have space for as many as 64K characters, requiring up to 16 bits per character.

2.2 PICTURE DATA - UNIT SIZE

These days, most picture data is represented in "raster" or "bitmap" format – a rectangular array of dots, each with its own color.

For picture files, the concepts of "dots" and "pixels" are identical. In normal usage, "dots" is associated with dots-per-inch (dpi), a standard measure of resolution for scanners and printers, while "pixels" is associated with the working resolution of a computer monitor. The memory requirement for a single dot or pixel depends upon the level of color or shade resolution desired in the picture. Typical applications include:

For black-and-white pictures:	Line Art (black and white only)	1 bit/pixel
	16 shade grayscale	4 bits/pixel
	64 shade grayscale	6 bits/pixel
	256 shade grayscale	8 bits/pixel
For color pictures	16 color (basic EGA color)	4 bits/pixel
	256 color (Basic VGA or 8 bit color)	8 bits/pixel
	16 bit color (65,536 colors)	16 bit/pixel
	24 bit color (16,777,216 colors, or true color)	24 bits/pixel
	30 bit color (~1 billion colors, a scanner resolution)	30 bits/pixel
	32 bit color (4.29 billion colors, or "true" color)	32 bits/pixel
	40 bit color (a scanner resolution)	40 bits/pixel
48 bit color (a scanner resolution)	48 bits/pixel	

Colors are represented by numbers

- For black and white: a number indicating how bright the dot is
- For color, three number indicating how bright the dot is with respect to each of the primary wavelengths detected by the human eye (red, blue, and green)

Most pictures that look like anything recognizable have large areas of similar colors. This property can be used by mathematical compression algorithms like JPEG (.jpg) to reduce the amount of storage needed. The degree of compression depends on the complexity of the picture.

2.3 SOUND DATA - UNIT SIZE

Sound consists of a oscillating wave of air pressure. The simplest approach is record the air pressure variation using a sequence of numbers.

Sound files need to represent combinations of individual sounds across part or all of the audible spectrum (for humans the audible range consists of vibrations ranging from 20 Hz (cycles per second) to over 20 kHz. Most CD-Quality recordings have standardized on an upper range of 22.05 kHz. This require sampling the air pressure 44,100 times per second. CD's measure air pressure as a 16-bit number.

There tend to be standard patterns in series of air-pressure readings that sound like something intelligible. Therefore, there are numerous ways to compress sound samples, such as MP3.

2.4 VIDEO DATA - UNIT SIZE

Video data consist of a a series of pictures called *frames*. Video requires a rapid sequence of pictures (typically 24 frames per second) to provide realistic animation. Like sound samples, video can make extensive use of compression technologies.

See section 2.2 to calculate memory requirements for a single pixel

See section 3.2 to calculate units that make up a picture

3.0 DETERMINING HOW MANY "UNITS" MAKE UP A FILE

Once the memory requirement for a "unit" is determined, then the number of units in a file must be determined. The methods are specific to the type of data, as outlined below.

3.1 ALPHANUMERIC FILES - NUMBER OF UNITS

The number of characters in a file can usually be determined from existing data. As an example, the number of characters in a book can be determined by multiplying:

$$(\text{characters/line}) \times (\text{lines/page}) \times (\text{pages/book}) = \text{Characters/book}$$

An 80-page book with 50 lines per page and 80 characters per line would have

$$(80 \text{ characters / line}) \times (50 \text{ lines / page}) \times (80 \text{ pages / book}) = 320,000 \text{ characters}$$

Remember that spaces are characters, so half lines, half pages and blank pages need to be included. The number of characters per line might be exact (in set width fonts) or an average (if both sides are justified to provide a "block" appearance). The number of lines per page is normally exact, so long as the font size is consistent. Alternatively, the total number of characters in a file may be specified.

A note about word processor (Word, WordPerfect, etc) files - they are not true alphanumeric files. Word processors allow extensive formatting, font styles, colors and sizes, and inserted objects that increase the overall file size. Their

size can be approximated in characters as long as three addition variables are known. One of these is a "setup overhead" which is specific to the word processor being used. The second is a "variable overhead" (a multiplier). The last is the size of any non-text objects (such as pictures) inserted into the file. The equation for determining the effective size is as follows:

$$\text{(Size)} = \text{(Setup Overhead)} + \text{(Variable Overhead)} \times \text{(Alphanumeric File Size)} + \text{(Inserted Objects)}$$

3.2 PICTURE FILES - NUMBER OF UNITS

In picture files the number of dots or pixels make up the number of "units".

Occasionally the size of the file is given in pixels, and no additional calculations are required to determine the number of "units" in the file. A perfect example is digital cameras, which are usually specified by the maximum resolution of the pictures they take. An Olympus C-3030Zoom camera is specified as a 3.14 mega-pixel camera. This is because its maximum picture resolution is 3,145,728 pixels. Beware, however, that an emerging trend is to specify digital cameras based upon the size of their image pickup unit. This size will always be larger than the maximum "unit" resolution of the files generated by the camera (For the Olympus camera the image pickup unit is rated at 3.34 million pixels).

Picture files are more likely to be specified by their length and width, in pixels. This is also true for standard resolutions of computer monitors. To determine the number of units in this type of file, you simply multiply the length by the width. An example would be the maximum resolution from the Olympus camera described above. The camera can produce picture files with a resolution of 2048 by 1536 pixels.

$$\text{(2048 pixels wide)} \times \text{(1536 pixels long)} = \text{3,145,728 pixels total}$$

Some problems can require you to size pictures to a computer screen (or in the case of video, a portion of a computer screen), or determine the file size generated by a digital camera. In such cases, one of the following pixel resolutions should be used:

160 × 120	Video phone and Quicktime movie resolution
320 × 240	Video phone and Quicktime movie resolution
512 × 400	Common video game "movie" resolution
640 × 480	Standard VGA resolution, typical for 14" and 15" monitors
800 × 600	Basic SVGA resolution, typical on 15" and 17" monitors
1024 × 768	SVGA resolution common on 17" and larger monitors
1280 × 960	Maximum effective resolution for most 17" monitors
1600 × 1200	Maximum "supported" resolution for most 19" and 21" monitors

A note regarding digital cameras - most digital cameras standardize on one or more of the above resolutions. High-end "consumer" cameras also support the 2048 × 1536 resolution.

Another common practice is to provide a dot or pixel density. This is the case for most computer printers and scanners, which will specify one or more possible densities using "dots per inch" (dpi). Typical resolutions are 300 dpi, 600 dpi, or 1440 dpi. Scanners will usually support a much larger variety of densities, up to the scanner's maximal resolution. To determine the total number of units in this type of file, you need to know the file's overall size. As an example, a 4-inch long by 6-inch wide photo, which is scanned at a resolution of 600 dpi, will contain:

$$\begin{aligned} \text{(4 inch long)} \times \text{(600 dpi)} &= \text{2400 dots long} \\ \text{(6 inch wide)} \times \text{(600 dpi)} &= \text{3600 dots wide} \end{aligned}$$

$$\text{(2400 dots long)} \times \text{(3600 dots wide)} = \text{8,640,000 dots total}$$

(Note: most scanners list two different types of resolutions, optical and digital. The optical resolution notes the "true" ability to discriminate details, while the digital resolution describes the ability to "zoom" the optical

output to larger scale with reasonable accuracy. If two different optical resolutions are given - for horizontal and vertical capability - the real optical resolution is normally the lower value.)

3.3 SOUND FILES - NUMBER OF UNITS

The number of units in sound files is based upon time. Samples are normally specified as samples per second, so you must also know the total length of the recording.

The number of samples can then be calculated from the relationship:

$$\text{(samples rate per second)} \times \text{(total time in seconds)} = \text{total number of samples}$$

Any one of the three variables can be determined if the other two are known.

Digital sampling of analog sound sources requires a minimum of two samples per Hz. Since the upper range of human hearing is considered to be somewhere near 20,000 Hz, the sampling rate for CD audio is standardized at 44,100 samples per second, and 16 bits per sample. CD audio is not compressed since the technology was developed before it was economic to embed compression technology in consumer products.

Compression techniques, such as MP3 format, compromise slightly on sound quality to obtain a much smaller file size. Digital cellphone and answering machine audio is highly compressed, and the resulting distortion quite audible.

3.4 VIDEO FILES - NUMBER OF UNITS

The number of units in a video file is based upon time. Samples are normally specified as frames per second (fps), thus the total time of the video file must also be known.

The number of frames can then be calculated from the relationship:

$$\text{(frame rate per second)} \times \text{(total time in seconds)} = \text{total number of frames}$$

Any one of the three variables can be determined if the other two are known.

Sampling of analog video sources signals requires a minimum of about 10 fps to provide low quality video, and at least 24 fps for high quality. 30 fps is quite common.

A related issue for video is the refresh rate (how often a screen is updated) for the video monitor. Television uses a refresh rate of 60 Hz, which coincides with the frequency of AC power supplies. Some "flickering" of the picture may be seen, especially when viewed closely. On CRT computer monitors, refresh rates need to exceed 72 Hz or the eye may perceive flickering. Flickering is distracting, tiring on the eyes, and can cause headaches. Normally a higher refresh rate is better.

4.0 FINAL CALCULATIONS, AND THE EFFECT OF COMPRESSION ON FILE SIZE

As stated initially, file size is the product of "unit" size and number of units that make up the file. In the case of alphanumeric and picture files, this calculation determines the uncompressed file size. For sound and video files, if the sample or frame rate is provided, the calculation determines the compressed file size.

Compression is a process in which file size is reduced while maintaining all critical file components. There are many different compression techniques, most of which are optimized to specific file types. A key concept is that of "compression ratio". Compression ratio is described as:

$$\text{(Original file size)} / \text{(Compressed file size)} = \text{Compression ratio}$$

4.1 ALPHANUMERIC FILES - FINAL SIZE AND COMPRESSION

For alphanumeric files, the memory requirement calculation is:

$$\text{(Memory / character)} \times \text{(characters / file)} = \text{memory requirement}$$

Assuming a 7-bit character set, the memory requirement for the book in section 3.1 would be:

$$\begin{aligned} \text{(7 bits / character)} \times \text{(320,000 characters / file)} &= \text{2,240,000 bits} \\ \text{2,240,000 bits} \times \text{(1 byte / 8 bits)} &= \text{280,000 bytes} \end{aligned}$$

Text compression relies on text not being “uniformly” random. For example, if the last three characters in an English text were space, “b”, and “r”, the next character is most likely a vowel and not a “z”. Text compression is usually “lossless” – by applying the right algorithm, one can completely reverse the compression.

Compression ratios of 1.5 to 5 are commonly achieved on text files files.

4.2 PICTURE FILES - FINAL SIZE AND COMPRESSION

For picture files, the memory requirement calculation is:

$$\text{(Memory / dot or pixel)} \times \text{(dots or pixels / file)} = \text{memory requirement}$$

Assuming a 16-shade grayscale resolution, the memory requirement for the scanned picture in section 3.2 would be:

$$\text{(4 bits / pixel)} \times \text{(8,640,000 pixels / picture)} \times \text{(1 byte / 8 bits)} = \text{4,320,000 bytes} = \text{4.32 MB}$$

The above result is for an uncompressed picture file, commonly called a bitmap file.

There are a variety of standardized picture compression techniques. Many take advantage of the similarity of adjacent pixels over large areas of any picture. Algorithms describing one or more geometric shapes can exactly represent large groups of adjacent identical pixels (identical in terms of color). An algorithm describing a rectangle of 25 by 10 pixels with the same 24-bit color might take up 10 to 20 bytes of space (while the original 250 pixels would take up 750 bytes). The less complex a picture is, the higher the percentage of the picture represented by these algorithms, and the higher the level of compression. This type of compression is lossless, and can achieve a compression ratio of as much as 4 to 8.

Even greater compression is available if a “lossy” compression is allowed. In lossy compression, all major attributes of a picture are retained, but some of the detail might be modified or lost. For picture files, this might mean that a few pixels are changed during compression. Or it may mean a change in color might be described as a smooth transition, eliminating less consistent details. In some compression techniques (notably JPEG) the user can choose the tradeoff between loss of detail and increasing compression ratio. Compression ratios of 4 to 8 can be achieved with very minimal loss of detail. Higher compression ratios (up to 25) can be achieved with a more noticeable loss of detail. In general, only lossless compression should be used on photo-quality pictures that might be blown-up and printed. All other pictures can be evaluated for lossy compression.

4.3 SOUND FILES - FINAL SIZE AND COMPRESSION

For sound files, the memory requirement calculation is:

$$\text{(Memory / sound sample)} \times \text{(sound samples / file)} = \text{memory requirement}$$

or, remembering that sampling rates and times are often supplied:

$$(\text{Memory} / \text{sound sample}) \times (\text{sound samples} / \text{sec}) \times (\text{seconds} / \text{file}) = \text{memory requirement}$$

For either equation, problems can be constructed to solve for any single variable by supplying the other variables. These types of problems imply an analysis of already-compressed sound files.

Many sound files are stereo, meaning that there are two sound channels, one for the left speaker, and one for the right speaker. In the absence of compression, stereo recordings require twice as much storage as one-channel “mono” recordings.

The compression techniques used on sound files are varied. A trait of audio samples is that each is very similar to the sample before it. Certain compression techniques minimize storage requirements by saving only the differences from one sample to the next.

4.4 VIDEO FILES - FINAL SIZE AND COMPRESSION

The memory requirement calculation is very similar to the sound equation described above:

$$(\text{Memory} / \text{video frame}) \times (\text{video frames} / \text{file}) = \text{memory requirement}$$

or, remembering that frame rates and video times are often supplied:

$$(\text{Memory} / \text{video frame}) \times (\text{video frames} / \text{sec}) \times (\text{seconds} / \text{file}) = \text{memory requirement}$$

For either equation, problems can be constructed to solve for any single variable by supplying the other variables. These types of problems imply an analysis of already-compressed video files.

Video files take advantage of compression techniques for individual pictures, and also benefit from the compression techniques applied to audio files. Video files are large sequences of pictures, so techniques that reduce the size of individual frames with some minimal loss of detail (lossy compression) are utilized. The minimal loss of detail is usually acceptable because no single video frame is presented for more than a fraction of a second. Like audio files, the differences between any two consecutive video frames are minimal. Therefore compression techniques that store only the difference from one frame to another are utilized. A standard video compression technique such as MPEG can achieve overall compression ratios of 50 to 100.

5 GENERAL DATA TABLES

Finally, consider the amount of space taken by a table of the sort we are beginning to study in relational databases. Sometime, tabular data is stored as text data in essentially the same form it is displayed, in which case the rules for estimating the sizes of text files apply. Within relational database systems, however, each field in the table is stored in a manner determined by its datatype – in Access, its “Data type” and “field size” properties.

The general rule is to find the amount of storage needed by each row of the table, which means adding the memory required by each field. Then, you multiply by the number of rows in the table.

Example: The USA contains about 3100 counties. Suppose you have a table that stores, for each county, its name (up to 40 characters in 8-bit ASCII), its state (a two-letter code), its population, and its median income (both as 32-bit numbers). How much space would the whole database take in binary-style K?

Space needed per row of the table:

Name	40 Characters	40 bytes
State	2 Characters	+ 2 bytes
Population	32 bit integer	+ 4 bytes (32 bits/8 bits/byte = 4 bytes)
Median Income	32 bit integer	+ <u>4</u> bytes
		50 bytes

If there are 3100 rows, that means $3100 \text{ rows} \times 50 \text{ bytes/row} = 155,000 \text{ bytes}$ for the whole table. Finally, $155,000 \text{ bytes} / 1024 \text{ bytes/binary KB} \approx 151 \text{ KB}$.

Requirements for common kinds of fields in MS Access are as follows (see also section 1.0 above):

- Text: 1 byte per character
- Number
 - Byte: 1 byte (of course): whole numbers from -127 to $+128$
 - Integer: 2 bytes (often called a “short” integer): whole numbers from $-32,768$ to $+32,767$
 - Long integer: 4 bytes: whole number from approximately -2 billion to $+2$ billion
 - Single: 4 bytes: “floating” numbers with about 6 digits precision
 - Double: 8 bytes: “floating” numbers with about 15 digits precision
- Date/Time: 8 bytes (this is actually a specially formatted 8-byte integer)
- Currency: 8 bytes (again, actually a specially formatted 8-byte integer)
- Yes/No: 1 byte (in theory only one bit is really need, but in practice a whole byte is typically used)

Note that there are many factors that can make the actual amount of storage needed considerably larger.

- Sometimes a few extra “filler” bytes can be introduced between fields of the table, so that data are “aligned” properly for efficient manipulation by the processor.
- Relational database often introduce additional information called “indexes” to help them quickly locate particular rows of the table, without having to search through all the rows exhaustively.
- Sometimes, as in Access, the space taken up by deleted records may not be immediately reclaimed. Instead you need to “compress” the database (a different meaning from “compression” above) to reclaim this space.
- Other objects and information may also be stored in the database file – for example, the descriptions of all the Access objects related to the table, including the table design view, forms, queries, and reports.
- The file may be longer than needed, to leave room to add more table rows or other information in the future.

Thus, the calculation of $(\text{space per row}) \times \text{number of rows}$ is only a lower bound on the total amount of space needed.