Data (computing)

6.1 DEFINITION: Data (/ˈdeɪtə/ ***DAY****-tə*, /ˈdætə/ ***DA****-tə*, or /ˈdɑːtə/ ***DAH****-tə*; treated as singular, plural, or as a mass noun) is any sequence of symbols given meaning by specific acts of interpretation. Digital data is the quantities, characters, or symbols on which operations are performed by a computer, stored and recorded on magnetic, optical, or mechanical recording media, and transmitted in the form of electrical signals.[2] A program is a set of data that consists of a series of coded software instructions to control the operation of a computer or other machine.[3] Physical computer memory elements consist of an address and a byte/word of data storage. Digital data are often stored in relational databases, like tables or SQL databases, and can generally be represented as abstract key/value pairs. Data can be organized in many different types of data structures, like arrays, graphs, objects and many more. Data structures can store data of many different types, including numbers, strings and even other data structures. Data pass in and out of computers via peripheral devices.

In an alternate usage, binary files (which are not human-readable) are sometimes called "data" as distinguished from human-readable "text". The total amount of digital data in 2007 was estimated to be 281 billion gigabytes (= 281 exabytes).

6.2 Characteristics

At its heart, a single datum is a value stored at a specific location.

Fundamentally, computers follow a sequence of instructions they are given in the form of data. A set of instructions to perform a given task (or tasks) is called a "program". In the nominal case, the program, as executed by the computer, will consist of binary machine code. The elements of storage manipulated by the program, but not actually executed by the CPU, are also data. The Marvellous twist is that program instructions; and data that the program manipulates, are both stored in exactly the same way. Therefore it is possible for computer programs to operate on other computer programs, by manipulating their programmatic data.

The line between program and data can become blurry. An interpreter, for example, is a program. The input data to an interpreter is itself a program, just not one expressed in native machine language. In many cases, the interpreted program will be a human-readable text file, which is manipulated with a text editor program (more normally associated with plain text data). Metaprogramming similarly involves programs manipulating other programs as data. Programs like compilers,

linkers, debuggers, program updaters, virus scanners etc. use other programs as their data.

To store data bytes in a file, they have to be serialized in a "file format". Typically, programs are stored in special file types, different from those used for other data. Executable files contain programs; all other files are also data files. However, executable files may also contain "in-line" data which is "built-in" to the program. In particular, some executable files have a data segment, which nominally contains constants and initial values (both data).

For example: a user might first instruct the operating system to load a word processor program from one file, and then edit a document stored in another file with the word processor program. In this example, the document would be considered data. If the word processor also features a spell checker, then the dictionary (word list) for the spell checker would also be considered data. The algorithms used by the spell checker to suggest corrections would be either machine code data or text in some interpretable programming language.

Data keys and values, structures and persistence

Keys in data provide the context for values. Regardless of the structure of data, there is always a **key** component present. Data keys in data and data-structures are essential for giving meaning to data values. Without a key that is directly or indirectly associated with a value, or collection of values in a structure, the values become meaningless and cease to be data. That is to say, there has to be at least a key component linked to a value component in order for it to be considered data. Data can be represented in computers in multiple ways, as per the following examples:

RAM

- Computer main memory or RAM is arranged as an array of "sets of electronic on/off switches" or locations beginning at 0. Each location can store a byte (usually 8, 16, 32 or 64 bits depending on the CPU architecture). Therefore any value stored in a byte in RAM has a matching location expressed as an offset from the first memory location in the memory array i.e. 0+n, where n is the offset into the array of memory locations.

Keys

- Data keys need not be a direct hardware address in memory. Indirect, abstract and logical keys codes can be stored in association with values to form a data structure. Data structures have predetermined offsets (or links or paths) from the start of the structure, in which data values are stored. Therefore the data key consists of the key to the structure plus the offset (or links or paths) into the structure. When such a structure is repeated, storing variations of [the data values and the data keys] within the same repeating structure, the result can be considered to resemble a table, in which each element of the repeating structure is considered to be a column and each repetition of the structure is considered as a row of the table. In such an organization of data, the data key is usually a value in one (or a composite of the values in several of) the columns.

Organised recurring data structures

- The tabular view of repeating data structures is only one of many possibilities. Repeating data structures can be organised hierarchically, such that nodes are linked to each other in a cascade of parent-child relationships. Values and potentially more complex data-structures are linked to the nodes. Thus the nodal hierarchy provides the key for addressing the data structures associated with the nodes. This representation can be thought of as an inverted tree. E.g. Modern computer operating system file-systems are a common example; and XML is another.

Sorted or ordered data

- Data has some inherent features when it is sorted on a key. All the values for subsets of the key appear together. When passing sequentially through groups of the data with the same key, or a subset of the key changes, this is referred to in data processing circles as a break, or a control break. It particularly facilitates aggregation of data values on subsets of a key.

Peripheral storage

- Until the advent of non-volatile computer memories like USB sticks, persistent data storage was traditionally achieved by writing the data to external block devices like magnetic tape and disk drives. These devices typically seek to a location on the magnetic media and then read or write blocks of data of a predetermined size. In this case, the seek location on the media is the data key and the blocks are the data values. Early data file-systems, or disc operating systems used to reserve contiguous blocks on the

disc drive for data files. In those systems, the files could be filled up, running out of data space before all the data had been written to them. Thus much unused data space was reserved unproductively to avoid incurring that situation. This was known as raw disk. Later file-systems introduced partitions. They reserved blocks of disc data space for partitions and used the allocated blocks more economically, by dynamically assigning blocks of a partition to a file as needed. To achieve this, the file-system had to keep track of which blocks were used or unused by data files in a catalog or file allocation table. Though this made better use of the disc data space, it resulted in fragmentation of files across the disc, and a concomitant performance overhead due to latency. Modern file systems reorganize fragmented files dynamically to optimize file access times. Further developments in file systems resulted in virtualization of disc drives i.e. where a logical drive can be defined as partitions from a number of physical drives.

Indexed data

- Retrieving a small subset of data from a much larger set implies searching though the data sequentially. This is uneconomical. **Indexes** are a way to copy out keys and location addresses from data structures in files, tables and data sets, then organize them using inverted tree structures to reduce the time taken to retrieve a subset of the original data. In order to do this, the key of the subset of data to be retrieved must be known before retrieval begins. The most popular indexes are the B-tree and the dynamic hash key indexing methods. Indexing is yet another costly overhead for filing and retrieving data. There are other ways of organizing indexes, e.g. sorting the keys (or even the key and the data together), and using a binary search on them.

Abstraction and indirection

- Object orientation uses two basic concepts for understanding data and software: 1) The taxonomic rank-structure of program-code classes, which is an example of a hierarchical data structure; and 2) At run time, the creation of data key references to in-memory data-structures of objects that have been instantiated from a class library. It is only after instantiation that an executing object of a specified class exists. After an object's key reference is nullified, the data referred to by that object ceases to be data because the data key reference is null; and therefore the object also ceases to exist. The

memory locations where the object's data was stored are then referred to as garbage and are reclassified as unused memory available for reuse.

Database data

The advent of databases introduced a further layer of abstraction for persistent data storage. Databases use meta data, and a structured query language protocol between client and server systems, communicating over a network, using a two phase commit logging system to ensure transactional completeness, when persisting data. Formally, "database" refers to the data themselves and supporting data structures. Databases are created to operate large quantities of information by inputting, storing, retrieving and managing that information. Databases are set up so that one set of software programs provides all users with access to all the data.

A "database management system" is a suite of computer software providing the interface between users and a database or databases. Because they are so closely related, the term "database" when used casually often refers to both a DBMS and the data it manipulates.

Outside the world of professional information technology, the term *database* is sometimes used casually to refer to any collection of data (perhaps a spreadsheet, maybe even a card index). This article is concerned only with databases where the size and usage requirements necessitate use of a database management system.

The interactions catered for by most existing DBMSs fall into four main groups:

- **Data definition** – Defining new data structures for a database, removing data structures from the database, modifying the structure of existing data.
- **Update** – Inserting, modifying, and deleting data.
- **Retrieval** – Obtaining information either for end-user queries and reports or for processing by applications.
- **Administration** – Registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control, and recovering information if the system fails.

A DBMS is responsible for maintaining the integrity and security of stored data, and for recovering information if the system fails.

Both a database and its DBMS conform to the principles of a particular database model. "Database system" refers collectively to the database model, database management system, and database.

Physically, database servers are dedicated computers that hold the actual databases and run only the DBMS and related software. Database servers are usually multiprocessor computers, with generous memory and RAID disk arrays used for stable storage. RAID is used for recovery of data if any of the disks fail. Hardware database accelerators, connected to one or more servers via a high-speed channel, are also used in large volume transaction processing environments. DBMSs are found at the heart of most database applications. DBMSs may be built around a custom multitasking kernel with built-in networking support, but modern DBMSs typically rely on a standard operating system to provide these functions. Since DBMSs comprise a significant economical market, computer and storage vendors often take into account DBMS requirements in their own development plans.

Databases and DBMSs can be categorized according to the database model(s) that they support (such as relational or XML), the type(s) of computer they run on (from a server cluster to a mobile phone), the query language(s) used to access the database (such as SQL or XQuery), and their internal engineering, which affects performance, scalability, resilience, and security.

-

Parallel distributed data processing

- Modern scalable / high performance data persistence technologies rely on massively parallel distributed data processing across many commodity computers on a high bandwidth network. An example of one is Apache Hadoop. In such systems, the data is distributed across multiple computers and therefore any particular computer in the system must be represented in the key of the data, either directly, or indirectly. This enables the differentiation between two identical sets of data, each being processed on a different computer at the same time.