

A Short History of the Business Sort

From the punched-card sorter to Linux: what each era got right, and the four things that never changed

Bill Ahlbrandt

President and CEO, Ahlbrandt Software

Abstract

Businesses have sorted records by machine for more than 130 years, and by stored program for almost eighty. This paper follows the business sort through five eras: punched cards; stored programs and magnetic tape; sort as a separately sold software product; open systems and clusters; and hardware help on the current mainframe. For each one I ask what it added that later eras kept. The techniques changed completely, several times. The contract between the sort and the business stayed put. Four things survived every change: a defined treatment of records with equal keys, an explicit collating sequence, control through statements that describe the result, and the discipline of the batch window. If you move sort work between platforms, you're moving that contract too. Test it on purpose. Don't assume it comes along with the data.

What I found

- The card sorter's right-to-left method only worked because each pass kept the order left by the one before. **Stability was a mechanical necessity** decades before it was a software option.
- Sort was one of the first programs a computer **generated for you**. Betty Holberton's sort-merge generator for UNIVAC I (1951–52) took a description of the file and produced the program. The sort control statement comes from that idea.
- A stable, documented interface made sort one of the earliest markets for **independent system software**. Syncsort's founders started in 1968, a year before IBM unbundled software from hardware.
- Sort benchmark records went from 980 seconds for one million records (1987) to 134 seconds for 100 terabytes (2016). The benchmark measures throughput on uniform records. It doesn't test collating sequences, packed-decimal keys or the order of equal keys.
- In 2019 IBM put sort into the z15 processor. Seventy-four years after von Neumann measured EDVAC against IBM's card sorters, sorting still earned **dedicated silicon**.

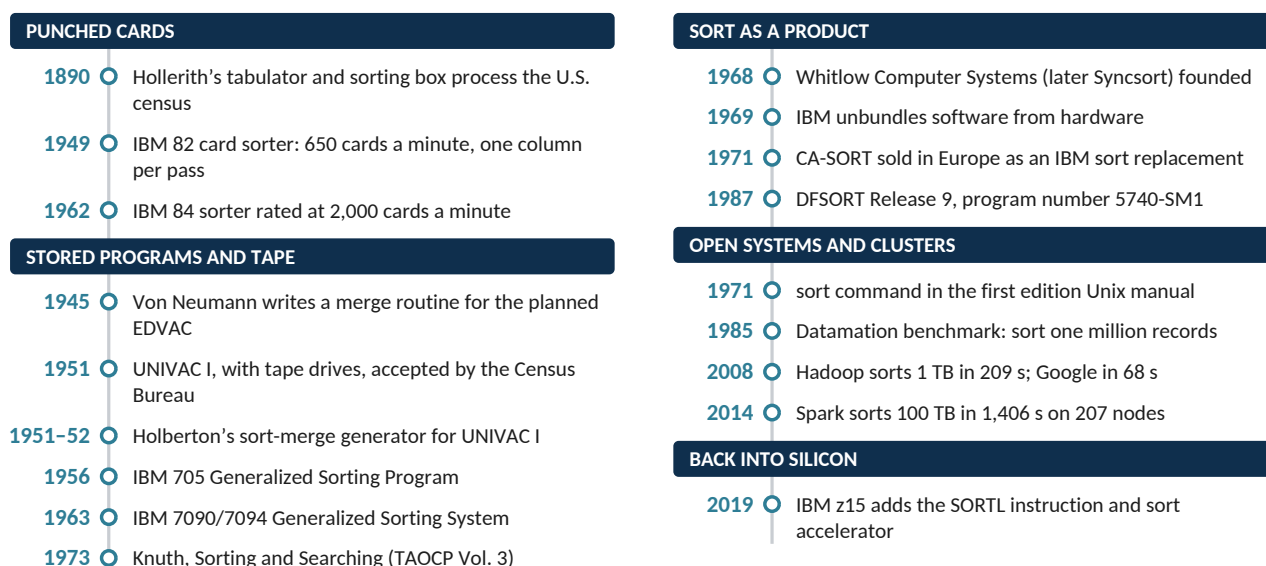
1. Why write a history of sort?

Sorting is the oldest commercial data processing workload that's still running. It's half a century older than the stored-program computer, and it's been rebuilt on every generation of equipment since: card sorters, magnetic tape, disk, the Unix pipeline, clusters of commodity servers and, since 2019, a dedicated instruction on the IBM Z processor. Not many business functions have been reimplemented that often, and fewer still have behaved the same way from the outside the whole time.

That second point is what this paper is about. Most writing on sorting is about algorithms, and the algorithms did change, over and over. What an accounts department expected from a sort stayed the same. The output had to be in an order the business could state ahead of time. Records with equal keys had to behave predictably. And the run had to finish before the next business day started. The history below is split into five eras, summarized in Exhibit 1, and for each one I ask what it added that later eras kept.

There's a practical reason to care. A migration from z/OS to Linux or the cloud crosses several of these transitions in one step, and what survived every earlier crossing is a pretty good guide to what has to survive the next one.

Exhibit 1. Seventy years of the business sort, with its punched-card prehistory



Events grouped by era, and in date order within each era. Sources as cited in the text.

2. Punched cards

The business sort starts with the United States census. Herman Hollerith got a patent for an electromechanical tabulating machine on January 8, 1889, and his equipment won the contract for the 1890 census after a trial in which it sorted test data into categories in 5.5 hours. The two competing methods took 44.5 and 55.5 hours.¹ Sorting was built in from the start. As each card was read, a compartment in the attached sorting box opened according to what was punched in it, and the clerk put the card in by hand.² An experienced clerk could do 80 cards a minute.³ Even though it collected more data, the Census Bureau published the 1890 results 18 months sooner than it had published the 1880 results.¹

The mature form was the free-standing card sorter. IBM's 82 sorter, introduced in 1949, was rated at 650 cards a minute.² By 1962 IBM's reference manual rated the 83 at 1,000 and the 84 at 2,000.⁴ The principle stayed the same. The sorter read one column per pass and dropped each card into one of 13 pockets: one for each of the 12 punch positions, plus a reject pocket for cards with nothing punched in that column. To sort on a field several columns wide, the operator started, in the manual's words, "with the right-hand (units) column of the field", restacked the pockets in order, and repeated column by column from right to left. Alphabetic columns have both a zone punch and a digit punch, so they took two passes each.⁴

Two things about that procedure outlived the machines. It's a least-significant-digit radix sort, and it only works because each pass keeps the cards in each pocket in the order they came in. Shuffle them and every earlier pass is lost. Stability (keeping input order among equal keys) was a mechanical necessity long before it was a software option. And the cost of a sort was visible, and it went up with the length of the key, as Exhibit 2 shows.

Exhibit 2. Rated feed time to sort 10,000 cards on an IBM 82 and an IBM 84 (illustrative)

Sort key	Columns	Passes	IBM 82 at 650 cards/min	IBM 84 at 2,000 cards/min
Six-digit account number	6 numeric	6	92 minutes	30 minutes
Account number and four-digit date	10 numeric	10	154 minutes	50 minutes
Eight-letter surname	8 alphabetic	16	246 minutes	80 minutes

Illustrative calculation from IBM's rated speeds and its rule of one pass per numeric column and two per alphabetic column.⁴ Doesn't include restacking, handling or re-runs, which aren't in the rated speed.

3. Stored programs and tape

The stored-program computer was measured against the card sorter from day one. In the spring of 1945 John von Neumann wrote one of the earliest programs for the planned EDVAC: a routine to merge two separately sorted sequences of records into one, which is the central step of what we now call merge sort. Donald Knuth, working through the surviving manuscript in 1970, gave two reasons for the choice. A sort was a tough test of whether the proposed instructions could control a complex process, and IBM's special-purpose sorting machines gave von Neumann "a standard against which he could measure the proposed computer's speed."⁵

The Census Bureau accepted the first UNIVAC I on March 31, 1951. Its UNISERVO drives were the first tape drives sold with a commercial computer, and they could read and write in either direction while the processor computed, which suited sort and merge work.⁶ Tape brought external sorting. You read the file in pieces small enough to fit in memory, sort each piece into a run and write it out, then merge the runs, pass by pass, across several tape units until there's one sorted file. A lot of ingenuity went into cutting down the merge passes and rewinds. Knuth's third volume catalogs the results, including the polyphase and cascade merges.⁷ The same structure is still there in the work data sets of every mainframe sort.

The idea from this period that lasted longest was a way of asking for a sort. In 1951–52 Betty Holberton, one of the original ENIAC programmers, came up with a sort-merge generator for UNIVAC I. The user described the file and the order they wanted, and the generator wrote the sorting program. Grace Hopper credited it as the source of her first ideas about compilation.⁸ IBM went the same way. Its 705 Generalized Sorting Program dates from 1956, with a companion merge program in 1957,⁹ and its Generalized Sorting System for the 7090 and 7094 was documented in 1963.¹⁰ "Generalized" is the key word. It meant one supplied program, driven by parameters, in place of a sort hand-written for each file.

There was plenty of work to justify the effort. In *The Art of Computer Programming*, whose volume on sorting first came out in 1973, Knuth recorded that computer makers in the 1960s estimated more than a quarter of their machines' running time, across all customers, went to sorting, and that at a lot of installations it was over half.⁷ A workload that big was going to support a market of its own.

4. Sort becomes a product

IBM's System/360 and its operating system gave sort the shape it still has on the mainframe: a system utility called through job control language (JCL), driven by control statements, with defined exit points where installation code can look at or change records.¹¹

You can follow the product line through its program numbers, from OS Sort/Merge (5734-SM1) to OS/VS Sort/Merge (5740-SM1), which was renamed Data Facility Sort, or DFSORT.¹² By April 1987 IBM was publishing the Release 9

programming guide for DFSORT under the same program number, 5740-SM1.¹³ DFSORT is still IBM's sort for z/OS.

A documented, stable interface has a commercial side effect: somebody else can implement it. In 1968 Duane Whitlow and Stan Rintel founded Whitlow Computer Systems, later renamed Syncsort, to build a faster sort for IBM mainframes. It was designed as a drop-in replacement that a site could swap in for IBM's sort without changing JCL or application code.¹⁴ In 1969, with a U.S. antitrust suit filed against it earlier that year, IBM announced it would price software and services separately from hardware. IBM's own history credits unbundling with giving birth to the software and services industries.¹⁵ Syncsort came first, so unbundling didn't create it. What unbundling did was let independent system software compete on its merits, and sort was a prominent early example. An IDC survey done in 1983 reported SyncSort in use at 75 percent of the IBM OS/VS1 and MVS customers surveyed.¹⁶

A second independent line started in Europe. Computer Associates AG, founded in Zurich in 1970, developed a sort originally for the pharmaceutical company Hoffmann-La Roche, and from 1971 sold it in Europe as CA-SORT, a plug-in replacement for IBM's sort on System/360 and System/370. In 1976 the product went to the newly formed Computer Associates, Inc. in the United States.¹⁷ CA Sort is now a Broadcom product, since Broadcom bought CA in 2018.¹⁸

Three products implementing one statement language had a lasting effect. Each one added functions (reformatting, selection, summation, reports, joins), and each had to accept the others' statements to win their customers. So the language became a de facto standard, extended by competition with no standards body involved. Inside, the products competed on technique. DFSORT, for example, picks among several internal methods, and its preferred one is called Blockset.¹¹ The job never saw the technique. Customers bought speed and kept their control statements.

5. Unix and open systems

Unix saw it differently. A sort command, first implemented in Multics, was in the first edition of Unix, written by Ken Thompson at Bell Laboratories,¹⁹ whose manual is dated November 3, 1971.²⁰ Unix *sort* treats a file as lines of text and keys as fields between delimiters, and it's designed to sit in a pipeline between other programs. It went into the X/Open Portability Guide in 1987 and from there into POSIX. The GNU implementation that's standard on Linux uses a merge sort.¹⁹

Its defaults differ from the mainframe's in two ways that matter when you migrate. The first is collating. Unless you tell it otherwise, GNU *sort* compares using "the character collating sequence specified by the LC_COLLATE locale", so the same file can sort differently under two locale settings. The manual says to set LC_ALL to C if you want byte order. The second is equal keys. When all the keys compare equal, GNU *sort* by default compares the whole lines as a last resort. The `--stable` option turns that off so those lines keep their input order.²¹ Neither default is wrong, and neither one matches the mainframe. EBCDIC also puts lowercase letters before uppercase and letters before digits, the reverse of ASCII,²² and mainframe records carry packed-decimal and binary fields in fixed- or variable-length records with no line terminators at all.

Independent vendors followed business data onto the new platforms. Syncsort moved into Unix client/server environments in the 1990s, introduced its DMExpress data-integration product in 2004, and renamed itself Precisely in 2020.¹⁴ This era added portability. It also raised the question every migration since has had to answer: do you convert to the new platform's conventions, or reproduce the mainframe's behavior on it?

6. Benchmarks, and back into silicon

In April 1985 an article in *Datamation*, "A Measure of Transaction Processing Power", published under the byline "Anon. et al." and led by Jim Gray, proposed a small set of standard tests. One of them was the time to sort one million 100-byte records.^{23, 24}

The sort test outlasted the others. The Sort Benchmark, now run by a volunteer committee, lists the first recorded result for the one-million-record test as 980 seconds on a Tandem system in 1987, and the last, before the category was retired, as 0.44 seconds on 32 Linux PCs in 2001.²⁴ Later categories raised the volume to a terabyte and then, as

GraySort, to 100 terabytes. Exhibit 3 lists some of the records.

Two of them stand out. In May 2008 a Yahoo! team sorted a terabyte in 209 seconds with Apache Hadoop on 910 nodes.²⁵ That November Google reported sorting the same amount in 68 seconds on 1,000 computers with MapReduce, and a petabyte in six hours and two minutes on 4,000.²⁶ In 2014 Databricks sorted 100 terabytes with Apache Spark in 1,406 seconds on 207 cloud nodes, sharing the record with the University of California, San Diego's TritonSort.²⁴ The previous Hadoop record, Databricks pointed out, had needed 2,100 machines and 72 minutes.²⁷

Exhibit 3. Selected sort benchmark records, 1987–2016

Year	Test	Result	System
1987	One million 100-byte records (100 MB)	980 s	Tandem
1994	One million records	7 s	AlphaSort, Digital Equipment
2001	One million records	0.44 s	NOW-sort, 32 Linux PCs
1998	1 terabyte	151 min	Nsort, SGI Origin 2000, 32 processors
2008	1 terabyte	209 s	Apache Hadoop, 910 nodes (Yahoo!)
2008	1 terabyte (not a benchmark entry)	68 s	MapReduce, 1,000 computers (Google)
2009	100 terabytes (GraySort)	173 min	Apache Hadoop, 3,452 nodes (Yahoo!)
2014	100 terabytes (GraySort)	1,406 s	Apache Spark, 207 cloud nodes (Databricks); TritonSort 1,378 s on 186 nodes
2016	100 terabytes (GraySort)	134 s	Tencent Sort

General-purpose (“Daytona”) results where the benchmark separates them from benchmark-specific (“Indy”) entries. Sources: Sort Benchmark record tables; Yahoo!, Google and Databricks reports.^{24, 25, 26, 27}

A sort works the disk, memory, network and processor all at once, so a fast sort is evidence of a balanced system, and these records moved the question of scale from the processor to the cluster. They have a limit, though. Benchmark records are uniform, fixed-length and randomly keyed. The results say nothing about packed-decimal keys, collating sequences, the order of records with equal keys, or the reformatting and selection that business sort steps do along with the sort itself. A sort engine can set a throughput record and still be the wrong replacement for a payroll job.

The mainframe's answer was silicon. The IBM z15, announced on September 12, 2019, added a SORT LISTS (SORTL) instruction with an on-chip accelerator, which DFSORT uses for eligible sorts once APAR PH03207 is applied.²⁸ Seventy-four years after von Neumann measured EDVAC against IBM's card sorters, sorting was still important enough to get dedicated hardware in IBM's flagship processor.

7. What never changed

The eras don't share much technology. What they share is what the business needs from a sort, and each era made its lasting contribution to one of those needs (Exhibit 4).

Equal keys

The card sorter kept input order because its method needed it. Mainframe sorts make it an explicit choice. DFSORT's EQUALS option keeps the original order of records with equal control fields, and the installation sets the default.¹¹ GNU *sort* only gives you the same guarantee with `--stable`.²¹ Business processes often depend on this without saying so, for example by expecting transactions to stay in posting order within each account.

The collating sequence

The order of the output is part of the specification. On the mainframe it's EBCDIC unless you name another sequence (DFSORT's ALTSEQ, for example).¹¹ On Linux it's the locale unless you set it otherwise. The same records in a different order make a different file, and downstream reports, matches and merges will show it.

Statements that describe the result

From Holberton's generator on, the user of a business sort has said what order they want and left it to the sort to work out how. A statement like `SORT FIELDS=(1,10,CH,A,15,5,PD,D)` names positions, lengths, formats and directions and nothing else. That separation is why Syncsort and CA-SORT could replace IBM's sort without changing any jobs, and why three products could share one language. It's also why a replacement sort engine on another platform can, in principle, run the same statements.

The batch window

The census had a publication deadline. The overnight run has the start of the business day. Every advance, from faster card feeds to fewer merge passes, clusters and on-chip acceleration, has been a way to fit more records into the same window.

Exhibit 4. What each era added, and where you can still see it

Era	Characteristic technology	What it added	Where you still see it
Punched cards, 1890–1960s	Tabulator; IBM 82, 83, 84 sorters	Multi-column keys sorted by stable passes; a machine run with a deadline	The stability requirement; batch schedules
Stored programs and tape, 1945–1965	EDVAC design; UNIVAC I; IBM 705, 7090	Merge sort; external sorting in runs and merge passes; generated sorts	Sort work data sets; parameter-driven sort
Sort as a product, 1964–1990	OS/360 Sort/Merge, DFSORT, Syncsort, CA-SORT	Control statements, JCL interface and exits; an independent market	One statement language shared by three mainframe products
Open systems, 1971 onward	Unix sort, GNU sort	Sort as a filter you can chain; locale-aware collation	Linux batch pipelines; defaults you have to check
Clusters and benchmarks, 1985–2016	Datamation test; Hadoop; Spark	Scale-out; sort as a measure of system balance	Distributed data platforms
Silicon, 2019 onward	IBM z15 SORTL and accelerator	Hardware sort assistance	Accelerated DFSORT on current IBM Z

Summary of Sections 2–6.

8. Bottom line

- **The contract outlived every algorithm.** Radix passes gave way to tape merges, clusters and hardware instructions. The business still needed a defined order, predictable equal keys, statements that describe the result, and a finishing time.
- **What each era left behind was one of those requirements.** Cards made stability unavoidable, and generators made sort declarative. The product era turned the control statement into a language three vendors shared.
- **Throughput records don't prove fidelity.** The benchmarks measure speed on uniform records. A business sort also has to reproduce collating, key formats and the order of equal keys exactly.
- **A migration crosses several eras at once.** Moving a sort from z/OS to Linux changes the character set, the record format, the defaults and the platform all together. Test each property in Section 7 explicitly.

Afterword: why the problem hasn't gone away

I've worked on sort for more than thirty years, a fair slice of this history, and I'm sometimes asked why sorting is still worth a white paper. My answer is that the algorithm was never the hard part for long. Every generation of hardware has made sorting a record cheaper, and every generation of business has found more records to sort and less time to sort them in. What stays hard is the contract: the unwritten promise that the output will be in exactly the order the business expects, down to the last pair of equal keys, and that it'll be there in the morning.

What strikes me in this history is how often the advance that lasted was a matter of discipline. There was the stacking rule on the card sorter, the generator that asked for a description instead of a program, and the statement language three competitors agreed to share. Those are what let you move a sort to another platform and trust the result, and they deserve as much attention in the next migration as the throughput numbers.

About the author

Bill Ahlbrandt is President and CEO of Ahlbrandt Software, which he founded in 1993. Before that he was a systems programmer, Director of Information Security and Director of R&D at Acxiom. He has been working on mainframe sort and batch for more than thirty years, and his company's sort engine runs mainframe sort jobs on Linux and other platforms for banks, insurers and government agencies.

Full disclosure: my company sells sort software for platforms other than the mainframe. This paper is a history, and it doesn't evaluate any current product.

References

1. U.S. Census Bureau, "History and the Census: Herman Hollerith and Mechanical Tabulation", census.gov, January 2016.
2. F. da Cruz, "Hollerith 1890 Census Tabulator" and "IBM Card Sorters", Columbia University Computing History.
3. U.S. Census Bureau, "The Hollerith Machine", Census Bureau history, census.gov.
4. IBM, *Reference Manual: IBM 82, 83, and 84 Sorters*, form A24-1034-1, July 1962 (minor revision). Computer History Museum archive.
5. D. E. Knuth, "Von Neumann's First Computer Program", *Computing Surveys* 2(4), December 1970, pp. 247–260.
6. "UNIVAC I", Wikipedia, accessed September 2026 (Census Bureau acceptance; UNISERVO tape).
7. D. E. Knuth, *The Art of Computer Programming, Vol. 3: Sorting and Searching*, Addison-Wesley, 1st ed. 1973; 2nd ed. 1998, p. 3 and §5.4.
8. J. A. N. Lee, "Frances Elizabeth (Betty) Snyder Holberton", *Computer Pioneers*, IEEE Computer Society, history.computer.org/pioneers.
9. IBM, *705 Generalized Sorting Program*, form 32-6831, 1956; *705 Generalized Merge Program*, form 32-7626, 1957.
10. IBM, *IBM 7090/7094 Generalized Sorting System: 7090/7094 Sort*, form C28-6307-0, 1963.
11. IBM, *z/OS DFSORT Application Programming Guide*, SC23-6878 (EQUALS, ALTSEQ, exits, Blockset).
12. "Mainframe sort merge", Wikipedia, accessed September 2026 (IBM program numbers).
13. IBM, *DFSORT Application Programming Guide, Release 9*, SC33-4035-12, 5740-SM1, April 1987.
14. L. Johnson, "Oral History of Duane Whitlow", CHM, 8 May 1998; "Precisely (company)", Wikipedia.
15. E. W. Pugh, "Origins of Software Bundling", *IEEE Annals of the History of Computing* 24(1), 2002, pp. 57–58; IBM Archives, "Chronological History of IBM: 1960s".
16. IDC, *Seventh Annual Survey of Sort Programs Used in IBM OS, OS/VS, and MVS Environments*, January 1984. Cited from secondary sources.
17. Computerworld, 19 January 1976, p. 17, and 18 October 1976, p. 44; S. M. Lewis and A. Woodward, "Computer Associates International, Inc.", *International Directory of Company Histories*, vol. 49, St. James Press, 2003, pp. 94–97.
18. Broadcom Inc., announcement of completion of its acquisition of CA, Inc., 5 November 2018.
19. "sort (Unix)", Wikipedia, accessed September 2026 (origins, standardization, GNU implementation).
20. K. Thompson and D. M. Ritchie, *UNIX Programmer's Manual*, 1st ed., Bell Telephone Laboratories, 3 November 1971.
21. Free Software Foundation, *GNU Coreutils manual*, "sort invocation", gnu.org.
22. IBM, EBCDIC code page 037 (CCSID 37) character assignments; ANSI X3.4 (ASCII).
23. Anon. et al. [J. Gray and others], "A Measure of Transaction Processing Power", *Datamation* 31(7), 1 April 1985.
24. Sort Benchmark, sortbenchmark.org (C. Nyberg, M. Shah), record tables, accessed September 2026.
25. O. O'Malley, "TeraByte Sort on Apache Hadoop", Yahoo!, May 2008.
26. G. Czajkowski, "Sorting 1PB with MapReduce", The Official Google Blog, 21 November 2008.
27. R. Xin, "Spark officially sets a new record in large-scale sorting", Databricks blog, 5 November 2014.
28. IBM, z15 announcement, 12 September 2019; *z/Architecture Principles of Operation*, SA22-7832-12 (SORTL); APAR PH03207.