Machines with a Memory: Episodic Recall and Graded Compaction for Industrial Waveforms

Version 0.2 — position and design report

Zin Nam

ABSTRACT

Engineers at a machine ask a question that time-series databases do not answer: *have we seen this before, and what was it?* The question is about an episode, an exact stretch of waveform together with the operating context in which it occurred and the outcome it later turned out to have, and it has to be answered years after the episode was recorded. We argue that an archive of machine waveforms should be designed as an episodic memory, and that such a memory has to forget: a single vibration channel sampled at 64 kHz still grows by about 9 GB a day after the best lossless compression we measured. We make three contributions. First, a model of the episode for industrial waveforms and the requirements it places on storage and recall: exact evidence, recall under small time shifts, conditioning on operating context, and the outcome as part of the record. Second, a graded compaction ladder (full, not searchable, represented, summarized, gone) in which every level states which questions it still answers exactly, together with an explicit, versioned keep-value that decides when an episode moves down, and a ledger that makes every step auditable. Third, an evaluation that measures forgetting by what it costs search and by whether known events stay findable. The building blocks are measured in MorletDB, an open PostgreSQL extension: lossless storage at 1.5–1.7 bytes per sample on calibrated vibration data, bitwise exact mergeable chunk summaries at about 1% of that size, and lag-tolerant similarity search whose hits read back the stored samples bit for bit. The ladder, ledger and a policy engine are now implemented in MorletDB, and a first study (F1), frozen before its two confirmation runs, compares four policies on screwdriving cycles and on seismic data. Exempting known events keeps all of them searchable at every reachable budget, where age-based retention keeps roughly the budget's share, and on seismic data the keep-value also answers rare queries better than retention by age at 25% and 10% of the archive. It does not do so on the cycle data, where failed cycles are a quarter of the archive, and novelty alone never beats retention by age; under the rule fixed in advance, retention by age stays the default.

1 INTRODUCTION

A condition-monitoring engineer who hears an unfamiliar knock in a spindle does not want the mean vibration of the last hour. They want to know whether the machine, or one like it, has produced this signal before, under which conditions, and what was found when someone opened it up. Maintenance practice has long reasoned this way: case-based reasoning retrieves past cases and reuses their solutions [1]. Agents built on large language models are now being given the same kind of memory, with retrieval weighted by relevance, recency and importance [20] and memory managed in tiers [19]. Industrial waveforms are a hard place to build it.

Waveforms are large. A vibration channel sampled at 64 kHz produces 5.5 billion samples a day, 44 GB as float64. Our lossless codec stores calibrated vibration and motor-current data at 1.5–1.7 bytes per sample [11, 15], which still leaves about 9 GB a day, about 3.3 TB a year, per channel. A plant has hundreds of channels.

Waveforms are evidence. A spectrum, an envelope or a learned embedding is a view of the signal under assumptions that will change; the samples are what a later analysis, a new feature extractor or a dispute needs. Lossy compression with error bounds [5, 10] trades exactly this away.

Similar is not identical. The same fault produces the same pattern at slightly different times, phases and operating points. Recall must tolerate small shifts and must be conditioned on context, or it finds the wrong neighbours [17].

Forgetting is unavoidable, and silent forgetting is the norm. Time-series databases expire data by age (retention policies, time-to-live) and keep downsampled roll-ups [18, 27, 28]. Age alone is the wrong criterion for a memory: the rare incident from three years ago is worth more than yesterday's thousandth identical cycle.

This report proposes to treat the waveform archive as an episodic memory with deliberate, graded and auditable forgetting. It is a position and design report. Section 2 defines the episode and its requirements; Section 3 the compaction ladder; Section 4 the keep-value and the ledger; Section 5 what is already measured in MorletDB; Section 6 the first evaluation of forgetting policies. Throughout, statements are marked M (measured, with the study), I (implemented and tested for correctness, not measured for benefit) or P (proposed).

2 THE EPISODE

Episodic memory, in the sense introduced by Tulving [29], stores events located in time and context and lets them be re-experienced; it is distinct from semantic memory, the generalized knowledge extracted from many events. Memory research has since argued that forgetting is part of what makes memory useful, not a failure of it [24]. We take both ideas literally.

Definition. An *episode* is a triple of (i) an exact sample interval of one or more recordings, (ii) the operating context valid during it (program, tool, part, load, speed, temperatures), and (iii) zero or more outcomes attached later (a label, a confirmed finding, a work order). The interval is the evidence, the context makes it comparable, and the outcome is what makes it worth recalling.

Requirements. R1 *Exact evidence*: every recalled episode reads back the stored samples bit for bit, so that a conclusion can be checked and re-analysed. R2 *Recall under shift*: similarity tolerates small time shifts between query and stored intervals. R3 *Context conditioning*: recall and comparison are restricted to, or weighted by, comparable operating context. R4 *Outcome linkage*: outcomes attach to intervals, are versioned, and are returned with the hits.

R5 *Consolidation*: frequent, similar episodes are represented by fewer stored examples without losing the ability to say how often and when they occurred. R6 *Accountable forgetting*: whatever is removed is removed by a declared policy, in a recorded step, with what remains stated.

Generalized knowledge, R7, the "semantic" layer of fault libraries and models trained on the episodes, is out of scope here; this report is about keeping the episodes that such knowledge is built from.

3 THE COMPACTION LADDER

Instead of a binary keep-or-delete, we propose that every episode, and every stretch of continuous recording, sits on one of five levels. Each level is defined by the questions it still answers *exactly*; Table 1 summarises them.

L0 to L1. Search representations (envelopes, vectors, index rows) are derived and can be rebuilt from the samples [17]; they also cost about as much again as the samples [17]. Dropping them first removes an episode from similarity recall while keeping every sample, which is the cheapest reversible step.

L2: representatives, not averages. Repetitive operation produces thousands of near-identical episodes. Keeping one *real* episode per cluster, the medoid under the same lag-tolerant distance used for recall [7], and the outliers beyond a threshold, preserves both "what does this usually look like" and "what was unusual" with real evidence. An average waveform is not evidence of anything and would violate R1. Each member keeps its time, context, distance to the medoid and chunk summaries, so frequency, timing and slow drift remain visible. Clusters can be clustered again over longer periods.

L3: exact summaries. Count, minimum, maximum and compensated sums merge exactly over adjacent chunks, so statistics of any chunk-aligned interval computed from summaries equal those computed from the samples, bit for bit [13]. This is the mergeability property of summaries [3] applied to the simplest statistics. Roll-ups to hours, days and months inherit it. Visual summaries such as M4 [6] preserve the extremes per pixel column and are a natural addition at this level.

L4: gone, but accounted for. Removal writes a ledger entry: what, when, under which policy version, with which checksums, and where a cold copy is if one is kept (for example as Parquet files on object storage with the existing exporter).

Moves are one-directional except by restore. An episode moves down by policy; it moves up only by restoring from a cold copy or by recomputing derived data, both of which are recorded too.

4 KEEP-VALUE AND THE LEDGER

Which episodes move down, and when, is the policy. We propose to make it explicit and versioned rather than implicit in a retention interval.

Keep-value. For an episode e we propose

$$v(e) = w_n\,\mathrm{novelty}(e) + w_i\,\mathrm{importance}(e) + w_u\,\mathrm{use}(e) + w_c\,\mathrm{rarity}(\mathrm{ctx}(e)) - w_a\,\mathrm{age}(e),$$

compared against the episode's storage cost. *Novelty* is the distance to the nearest kept episode under the recall distance, which the search already computes. *Importance* comes from outcomes: labels, confirmed findings and membership in an event library. *Use* counts retrievals and citations in investigations. *Context rarity* protects episodes from rare operating conditions, which are underrepresented by construction. *Age* is one term among several, not the policy: the need for a memory falls with time but also follows how often and how recently it was used [4]. The weights are part of the policy version and are returned with every decision. The form follows agent memory, which ranks recall by relevance, recency and importance [20], and prioritised experience replay, which keeps the transitions with the largest learning signal [26]; uniform alternatives such as reservoir sampling [30] are the baseline it has to beat.

Exemptions. Episodes in the event library, the catalogue of confirmed incidents and operations, are not moved by policy at all. An earlier version of this report allowed L1; the first F1 development runs showed that this takes the library out of search exactly when the budget is tight.

Dry run and ledger. Every policy run is first a dry run that reports what would move and how much it would save; the executed run writes the ledger and is resumable. Coverage reports state, for every period, the level it is at, so a search can say which periods it could not see.

5 WHAT IS ALREADY MEASURED

The ladder rests on properties of MorletDB [17], a PostgreSQL extension that stores recordings as losslessly encoded chunks and searches them by lag-tolerant similarity. Table 2 lists the measured building blocks; each comes from a study whose protocol was committed to the public repository before its results [16].

Context conditioning (R3) and outcomes (R4) are implemented as timed, versioned context events, segments cut by context keys, comparison of a segment with earlier segments under the same context, and labels on sample intervals (I); they are tested for correctness but their benefit for recall has not been measured. The episode table, the levels, moves with dry run and ledger, exemptions, and a policy engine with the keep-value are implemented in the main branch (SQL 0.3.5, not yet released; I), and measured in F1 (Section 6) except consolidation into representatives (L2). Roll-ups beyond chunks and cold copies are proposals (P).

6 FIRST EVALUATION: FORGETTING POLICIES (F1)

The protocol of study F1 was frozen in the repository before either confirmation run, after development on separate data [14]; Table 3 summarises the outcome.

Implementation. Episodes are rows with a sample interval, context, importance and a level; a policy run moves them down in transactions through one SQL function that enforces the rules (levels only go down, exempt episodes stay, an L2 move names a representative) and writes the ledger. A similarity window leaves search when episodes out of search cover it and no full episode overlaps it. The policy engine scores episodes with the keep-value,

Table 1: The compaction ladder. Storage is per sample of a 64 kHz channel, from measured sizes (L0: [11, 15]; L3: 66-byte summary per 4,096-sample chunk [13]); L2 depends on cluster sizes; it is implemented but not yet evaluated.

Level	What is kept	What it still answers exactly	Bytes / sample	Status
L0 full	encoded samples, search representations, context, outcomes	everything: samples, similarity, statistics	1.5–1.7	M
L1 not searchable	encoded samples, context, outcomes; search representations dropped	samples and statistics of any interval; not found by similarity	1.5–1.7 (minus index)	M [14]
L2 represented	per cluster of similar episodes: the medoid and outliers raw; for every member its time, context, distance to the medoid and summaries	how often and when; a real typical and the unusual examples; statistics of every member	cluster-dependent	I
L3 summarized	per-chunk summaries (count, min, max, compensated sums) and calendar roll-ups	count, min, max, mean and RMS of any chunk-aligned interval, bitwise identical to decoding	≈ 0.016	M (chunks, [14]), P (roll-ups)
L4 gone	a ledger entry; optionally a checksummed cold copy outside the database	that it existed, under which policy it was removed, and where a copy is	0	M [14] (ledger), P (cold copy)

Table 2: Measured building blocks (M), with their limits.

Result	Study
Lossless storage of calibrated vibration and current at 1.5–1.7 bytes per sample, smallest of the tested codecs on four of six waveform inputs; one dataset per input	G1, B3, B4 [11, 12, 15]
Chunk summaries: statistics bitwise identical to decoding (3,000/3,000), 58× faster for 60 s of 64 kHz at p95, one hour in 36 ms; 4.5% slower ingestion, 1.3% more chunk storage; one machine	CS1 [13]
Lag-tolerant recall beats zero-lag recall on a proxy task (catalogued earthquakes): precision at ten 0.297 vs 0.213, and 0.139 vs 0.060 on held-out stations and year; seismic data only	E5, E5-H [17]
Indexed recall at 97.7% of the exact top ten, p95 57 ms; exact screened search with identical results up to one million windows	E1, E7 [17]
Every hit reads back the stored samples bit for bit; filters and row-level security apply to exact and indexed search	E1 [17]
Ingestion from one connection at 13.7 M samples/s with byte-identical storage; one dataset, one machine	B3.10, B3.12 [11]
Full workflow within 2 CPUs and 512 MB, and on a Raspberry Pi 3B+ with identical search results	EX1, EX2 [17]
Exempt known events stay 100% searchable at every reachable budget; the keep-value beats age-based retention on rare seismic queries at 25% and 10%, not on screwdriving cycles; two datasets	F1 [14]

moves the lowest-valued ones straight to L3 so that the most valuable stay fully searchable, applies the moves in batches of at most 50 and measures the size after each, stopping at the budget.

Policies and budgets. Recency (oldest first, the retention baseline), random (seeded), novelty (distance of an episode's first window to its nearest neighbour in the full archive, a static approximation), and novelty plus importance, in which known events are exempt. Budgets are 50, 25, 10 and 5% of the bytes of chunks, summaries and search representations; a budget the exempt events alone exceed is reported as unreachable, not forced.

Data. Screwdriving cycles of PyScrew scenario s02 [31]: 12,500 cycles, one episode each, 2,988 failed (NOK) cycles as known events; rare queries are the failed cycles, typical queries 300 good ones. Seismic data of four stations over two weeks held out from all earlier work: 7,849 episodes of about ten minutes, 563 containing the predicted arrival of a catalogued event [17]; rare queries are the windows at those arrivals, typical queries 300 windows without events. Development used scenario s01 and one other recording.

Measures and rule. Known events searchable: the share of known-event episodes still at L0. Answer quality: the distance of a query's nearest hit after forgetting over its nearest hit in the full archive (1 means something as similar is still found). The evaluation design in version 0.1 of this report proposed recall at ten against the full archive; the development runs showed that it, and the mean distance of ten hits, penalise removing near-duplicates, which is what novelty does by design, so the nearest hit became primary before the freeze, a change made after seeing development data. Each policy is compared with recency on rare queries by the mean over groups (workpieces, events) of the per-query difference, with a 95% bootstrap interval; "better" requires the interval above zero and at least as many known events searchable. The default is the policy better at 50% and 25% on both datasets, otherwise recency.

Results. Exempting the event library is what keeps it: novelty plus importance keeps every known event searchable at every budget it reaches, recency roughly the budget's share (M [14]). On seismic data, where events are 7% of the episodes, the keep-value also answers rare queries better than recency at 25% and 10%, at no demonstrated cost on typical queries at those budgets. On the cycles, where failed cycles are a quarter of the archive, the event library takes most of the budget: at 50% the rare-query answers are slightly worse on average (the medians are equal), typical answers degrade (median nearest-hit ratio 1.27 against 1.04), and budgets below 30% cannot be met without forgetting events. Novelty alone

Table 3: F1 confirmation. N+I: novelty plus importance. Rare queries: mean difference in nearest-hit ratio against recency with 95% interval; positive favours N+I. n/r: budget not reachable.

Data	Budget	Events searchable Recency	N+I	Rare queries, N+I vs recency
Cycles	50%	46%	100%	−0.05 [−0.09, −0.01]
Cycles	25%	13%	100%	n/r (29.8% kept)
Seismic	50%	45%	100%	+0.01 [−0.02, +0.05]
Seismic	25%	23%	100%	+0.07 [+0.02, +0.12]
Seismic	10%	9%	100%	+0.19 [+0.12, +0.27]

is never better than recency and often worse, on both datasets and for both kinds of query: the near-duplicates it removes are also the nearest neighbours of rare queries. Random and recency do not differ consistently.

Verdict. By the rule fixed in advance, recency stays the default and the keep-value is not built further as a default. What the study does support is narrower: an exempt event library is cheap insurance when events are rare, and pays for itself in rare-query quality there. Not tested: dynamic novelty against the kept rather than the full archive, consolidation into representatives (L2), the use and context-rarity terms, and the visibility of slow drift; each needs its own frozen study.

7 RELATED WORK

Episodic memory and forgetting. The distinction between episodic and semantic memory [29] and the view of forgetting as a feature of useful memory [24] motivate the ladder. Agent memories rank recall by relevance, recency and importance [20] and move content between tiers [19]; they store text, not signals, and do not require exact evidence.

Case-based reasoning in maintenance. Retrieve, reuse, revise and retain [1] is the cycle we support for waveforms: the episode is the case, the outcome its solution.

Compression. Lossless float codecs exploit structure in floating-point series [2, 9, 21]; error-bounded lossy compressors trade exactness for size [5, 10]. Our ladder keeps L0–L2 lossless and makes the move to summaries an explicit, accounted step.

Downsampling and retention. Round-robin archives [18], downsampling in metrics systems [27] and continuous aggregates [28] keep coarse statistics by age. Visualization-oriented aggregation preserves extremes [6]. Our L3 keeps exactly mergeable statistics; what differs is that the move is decided per episode by value, not per table by age.

Representatives and similarity. Medoids [7] keep real examples; reservoir sampling [30] is the uniform baseline; similarity under shift has a long history from dynamic time warping [25] to exact search at scale [22] and motif discovery [32].

Condition monitoring. Vibration-based diagnosis [23] and machine-learning fault diagnosis [8] produce the labels and outcomes that importance draws on; this report is about keeping the evidence those methods are trained and checked on.

8 LIMITATIONS

This is a position and design report with a first evaluation. F1 used two datasets, one budget definition and a static novelty; its verdict is for those, and its primary measure was chosen after development runs. Consolidation, roll-ups and cold copies are proposals; the measured building blocks come from one machine, from seismic data for retrieval quality and from one bearing dataset for storage and speed. Lossless storage depends on a value grid; data without one compresses less. Answer quality measures similarity against the archive's own episodes, a proxy for usefulness to an engineer, which only pilots can establish.

USE OF AI ASSISTANCE AND RESPONSIBILITY

Code, experiment harnesses, analyses and text were developed with Claude (Anthropic), an AI assistant, working under the author's direction. The author reviewed the content and is responsible for it.

LICENCE

© 2026 Zin Nam. The text and figures of this report are licensed under Creative Commons Attribution 4.0 International (CC BY 4.0, https://creativecommons.org/licenses/by/4.0/); the MorletDB source code is licensed under Apache-2.0.

REFERENCES

[1] Agnar Aamodt and Enric Plaza. 1994. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications* 7, 1 (1994), 39–59. https://doi.org/10.3233/AIC-1994-7104

[2] Azim Afroozeh, Leonardo X. Kuffo, and Peter Boncz. 2023. ALP: Adaptive Lossless floating-Point Compression. *Proceedings of the ACM on Management of Data (SIGMOD 2024)* 1, 4 (2023), 1–26. https://doi.org/10.1145/3626717

[3] Pankaj K. Agarwal, Graham Cormode, Zengfeng Huang, Jeff M. Phillips, Zhewei Wei, and Ke Yi. 2013. Mergeable Summaries. *ACM Transactions on Database Systems* 38, 4 (2013), 1–28. https://doi.org/10.1145/2500128

[4] John R. Anderson and Lael J. Schooler. 1991. Reflections of the Environment in Memory. *Psychological Science* 2, 6 (1991), 396–408. https://doi.org/10.1111/j.1467-9280.1991.tb00174.x

[5] Sheng Di and Franck Cappello. 2016. Fast Error-Bounded Lossy HPC Data Compression with SZ. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE, 730–739. https://doi.org/10.1109/IPDPS.2016.11

[6] Uwe Jugel, Zbigniew Jerzak, Gregor Hackenbroich, and Volker Markl. 2014. M4: A Visualization-Oriented Time Series Data Aggregation. *Proceedings of the VLDB Endowment* 7, 10 (2014), 797–808. https://doi.org/10.14778/2732951.2732953

[7] Leonard Kaufman and Peter J. Rousseeuw. 1990. *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley. https://doi.org/10.1002/9780470316801

[8] Yaguo Lei, Bin Yang, Xinwei Jiang, Feng Jia, Naipeng Li, and Asoke K. Nandi. 2020. Applications of Machine Learning to Machine Fault Diagnosis: A Review and Roadmap. *Mechanical Systems and Signal Processing* 138 (2020), 106587. https://doi.org/10.1016/j.ymssp.2019.106587

[9] Panagiotis Liakos, Katia Papakonstantinopoulou, and Yannis Kotidis. 2022. Chimp: Efficient Lossless Floating Point Compression for Time Series Databases. *Proceedings of the VLDB Endowment* 15, 11 (2022), 3058–3070. https://doi.org/10.14778/3551793.3551852

[10] Peter Lindstrom. 2014. Fixed-Rate Compressed Floating-Point Arrays. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (2014), 2674–2683. https://doi.org/10.1109/TVCG.2014.2346458

[11] MorletDB contributors. 2026. B3, B3.1–B3.13: MorletDB and five time-series databases on dense waveforms. https://github.com/merkhet-io/morletdb/blob/main/evaluation/tsdb_b3/RESULTS.md

[12] MorletDB contributors. 2026. B4: the chunk codec against the 2026 float codecs. https://github.com/merkhet-io/morletdb/blob/main/evaluation/storage_b4/RESULTS.md

[13] MorletDB contributors. 2026. CS1: per-chunk summaries for exact interval statistics. https://github.com/merkhet-io/morletdb/blob/main/evaluation/chunk_summaries/RESULTS.md

[14] MorletDB contributors. 2026. F1: forgetting policies — what forgetting costs recall. https://github.com/merkhet-io/morletdb/blob/main/evaluation/forgetting_f1/RESULTS.md

[15] MorletDB contributors. 2026. G1: lossless grid detection without a declared quantum. https://github.com/merkhet-io/morletdb/blob/main/evaluation/grid_detection/RESULTS.md

[16] MorletDB contributors. 2026. MorletDB: source code, frozen study protocols and results. https://github.com/merkhet-io/morletdb

[17] Zin Nam. 2026. MorletDB: Recordings, Not Rows — Lag-Tolerant Waveform Retrieval in PostgreSQL. https://github.com/merkhet-io/morletdb/tree/main/research/paper Technical report, version 0.20.

[18] Tobias Oetiker. 2026. rrdcreate: Set up a new Round Robin Database. RRDtool documentation. Retrieved 2026-11-05 from https://oss.oetiker.ch/rrdtool/doc/rrdcreate.en.html

[19] Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2023. MemGPT: Towards LLMs as Operating Systems. arXiv:2310.08560 [cs.AI] https://arxiv.org/abs/2310.08560

[20] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative Agents: Interactive Simulacra of Human Behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*. ACM, 1–22. https://doi.org/10.1145/3586183.3606763

[21] Tuomas Pelkonen, Scott Franklin, Justin Teller, Paul Cavallaro, Qi Huang, Justin Meza, and Kaushik Veeraraghavan. 2015. Gorilla: A Fast, Scalable, In-Memory Time Series Database. *Proceedings of the VLDB Endowment* 8, 12 (2015), 1816–1827. https://doi.org/10.14778/2824032.2824078

[22] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. 2012. Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*. ACM, 262–270. https://doi.org/10.1145/2339530.2339576

[23] Robert Bond Randall. 2011. *Vibration-based Condition Monitoring: Industrial, Aerospace and Automotive Applications*. Wiley. https://doi.org/10.1002/9780470977668

[24] Blake A. Richards and Paul W. Frankland. 2017. The Persistence and Transience of Memory. *Neuron* 94, 6 (2017), 1071–1084. https://doi.org/10.1016/j.neuron.2017.04.037

[25] Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26, 1 (1978), 43–49. https://doi.org/10.1109/TASSP.1978.1163055

[26] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized Experience Replay. In *International Conference on Learning Representations (ICLR)*. arXiv:1511.05952 https://arxiv.org/abs/1511.05952

[27] The Thanos Authors. 2026. Compactor (Downsampling and Retention). Thanos documentation. Retrieved 2026-11-05 from https://thanos.io/tip/components/compact.md/

[28] Tiger Data. 2026. Understand Continuous Aggregates. TimescaleDB documentation. Retrieved 2026-11-05 from https://www.tigerdata.com/docs/use-timescale/latest/continuous-aggregates/about-continuous-aggregates

[29] Endel Tulving. 1972. Episodic and Semantic Memory. In *Organization of Memory*, Endel Tulving and W. Donaldson (Eds.). Academic Press, New York, 381–403.

[30] Jeffrey S. Vitter. 1985. Random Sampling with a Reservoir. *ACM Trans. Math. Software* 11, 1 (1985), 37–57. https://doi.org/10.1145/3147.3165

[31] Nikolai West and Jochen Deuse. 2025. Industrial screw driving dataset collection: Time series data for process monitoring and anomaly detection. https://doi.org/10.5281/zenodo.15393134 Version 1.2.2, CC BY 4.0.

[32] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. 2016. Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 1317–1322. https://doi.org/10.1109/ICDM.2016.0179