One Substrate, Many Standards: An Out-of-Core SPARQL Engine and a Measured Zero-Overhead Evaluation Core Across the W3C/OGC Spec Families

Jesse Wright · the sparq project

DRAFT — in progress. *This paper's headline is a performance and memory claim, and its submission-gating evaluation has* not *run. Every performance and memory figure below is an explicitly marked at-risk slot; no benchmark number is claimed anywhere in this document. The evaluation methodology (§8) — canonical-host baselines against* native *(non-container) competitors, a Sparqloscope run, and a memory-accounting-honest comparison against qEndpoint — is fixed here* before *the measurement so results cannot steer the framing. The architecture, the spec-family breadth, and the correctness evidence (§3–§7) are real and traceable to the codebase today; the competitive numbers are not, and are not asserted.*

Abstract

RDF triple stores have historically forced a three-way trade-off: an engine is fast (QLever, Virtuoso, RDF-3X), or it is broad in reasoning and standards coverage (RDFox, GraphDB, Jena), or it fits large graphs on modest hardware (HDT, qEndpoint) — but a single system rarely claims all three, and no peer-reviewed engine paper combines SPARQL query evaluation, the OWL 2 profiles, RIF-Core, RDF stream processing, GeoSPARQL, and SHACL over one *measured shared* evaluation core. We describe an engine whose organising thesis is that this trade-off is largely an artifact of duplicated evaluation machinery. The system stores triples out-of-core as exhaustively-permuted, block-compressed, memory-mapped indexes (six permutations in the RDF-3X lineage [1], [2]) with inline-tagged term identifiers (the QLever/Virtuoso encoding lineage [3], [4]), so that both the working set and the dictionary spill to disk and only the touched pages are resident. Its query evaluator combines binary joins with worst-case-optimal Leapfrog Triejoin [5], [6] and bind joins over a monomorphic, allocation-lean id-level kernel. The paper's central *systems* contribution is that this kernel — row layout, the numeric tower, the join families, and SPARQL's total term order — is extracted into a single leaf crate consumed unchanged by the query engine *and* by every standards family (the OWL 2 RL/EL/QL reasoners, RIF-Core, an RSP-QL stream processor, GeoSPARQL, and SHACL), so breadth is not a bag of independent subsystems but one core carrying many standards. We frame breadth as a *measured-substrate* claim, not a feature list: the extraction is validated as behaviour-neutral by deterministic byte-exact ratchets and a cross-family differential conformance scoreboard, and the marginal cost of the shared kernel over the engine's hand-tuned original is a measurement (§8), not an assertion. We are explicit about what is *not* yet measured: the competitive performance and memory-frugality numbers that a reviewer will demand are gated on a canonical-host evaluation against native competitors, a Sparqloscope run, and a memory-accounting-honest comparison against qEndpoint; this draft fixes that methodology and states, everywhere a number would go, that the number does not yet exist.

1. Introduction

Three properties of an RDF engine tend to trade against one another. *Speed* — the property that made QLever [3], Virtuoso [4], and the RDF-3X line [1] competitive — usually buys its index and cache structure at the cost of memory and of a narrow query-only feature set. *Breadth* — OWL profile reasoning, rule interchange, stream processing, geospatial querying, shape validation — is the province of RDFox [7], GraphDB [8], and Jena [9], typically as separately engineered subsystems. *Frugality* — serving a graph far larger than RAM on commodity hardware — is the explicit pitch of HDT [10] and qEndpoint [11], usually for a restricted query surface. Systems that claim two of the three are common; a single system that credibly claims all three, over *one* evaluation core, is not something the peer-reviewed record contains: engine systems papers exist for QLever, RDFox, MillenniumDB [12], Tentris

[13], Virtuoso, Jena, and OWLIM/GraphDB, but none combines query, the OWL profiles, RIF, RSP, geo, and SHACL over a demonstrably shared substrate, and neither Oxigraph [14] nor Stardog has a peer-reviewed engine paper at all.

This is a systems paper about how one engine reaches for all three at once, and — as important — about the honesty discipline required to claim it. Breadth alone is not a research contribution: GraphDB, Stardog, and Jena already ship comparable standards coverage commercially, so "we also support SHACL and geo" reads as product engineering. Memory frugality alone invites an accounting attack: an mmap-resident store that reports a small *committed* footprint while relying on the OS page cache can be dismissed as measuring the wrong thing. Our claim is therefore deliberately narrow and falsifiable: *the standards breadth costs one evaluation core, not many, and that core costs no measured marginal overhead over the engine's own hand-tuned evaluation.* The word *measured* is load-bearing and its measurement (§8) is gated, so this draft does not yet assert the number.

We contribute:

1. **An out-of-core storage and evaluation architecture** (§3, §4): six block-compressed, memory-mapped permutation indexes in the RDF-3X/Hexastore lineage [1], [2]; a dictionary that spills to disk with inline-tagged identifiers in the QLever/Virtuoso lineage [3], [4]; and a query evaluator that mixes binary joins, worst-case-optimal Leapfrog Triejoin [5], [6], and bind joins over dictionary-encoded ids. The architecture is real and traceable to the codebase; its *competitive* performance is the at-risk part (§8).
2. **A measured zero-overhead shared evaluation substrate** (§5): the id-level row layout, numeric tower, join kernels, and SPARQL total-order comparator extracted into a single leaf crate, and consumed *unchanged* by the query engine and by every standards family. This is the paper's central systems contribution and the one that makes the breadth claim a claim about engineering economy rather than surface area.
3. **Breadth framed as a measured-substrate claim, not a feature list** (§6): the OWL 2 RL/EL/QL reasoners, RIF-Core, an RSP-QL stream processor, GeoSPARQL, and SHACL, each shown to route its evaluation through the shared kernel, with a deployment surface (§7) that compiles the same core to WebAssembly under a deterministic bundle-size ratchet.
4. **A correctness-evidence layer** (§7): a cross-family differential conformance scoreboard that pins a machine-checked pass floor per standard, so that the substrate extraction is validated as behaviour-neutral rather than merely asserted to be.
5. **A fixed evaluation methodology** (§8) for the four submission-gating experiments, committed in this draft before any of them runs, with the memory-accounting method stated explicitly because it is the most contestable number in the paper. **No benchmark number is claimed in this draft.**

2. Storage: out-of-core, exhaustively permuted, memory-mapped

Six permutations (RDF-3X lineage). A triple (s, p, o) of dictionary-encoded identifiers is stored in all six column orderings — SPO, SOP, PSO, POS, OSP, OPS — so that any triple pattern with any subset of bound positions is answered by a contiguous range scan over exactly one permutation, and any two patterns can be merge-joined on a shared variable without an intermediate sort. Exhaustive permutation indexing is the design of RDF-3X [1] and Hexastore [2]; we adopt it and do not claim it. Each permutation is a *block-compressed, random-accessible* sorted run: the column-0 ids of each block form a directory over which a pattern's bound prefix is binary-searched, and only the addressed blocks are decoded. The permutations are memory-mapped files, so the resident set is the set of *touched* blocks, not the graph.

Inline-tagged identifiers (QLever/Virtuoso lineage). The dictionary maps RDF terms to integer ids, and — following the encoding lineage of QLever [3] and Virtuoso [4] — a class of values (small

integers and other compactly-representable terms) is *tagged inline* in the identifier itself rather than stored in the dictionary, so common numeric and typed-literal operations read the value straight from the id without a dictionary indirection. Terms that are not inlined spill to an on-disk record store; the dictionary keeps a memory-mapped index and does not hold the term bytes in RAM. The consequence is the frugality property: both the triple store *and* the dictionary are out-of-core, and a graph substantially larger than physical memory is queryable with a resident set bounded by the working set of the query, not the size of the data.

Deterministic layout ratchets. The layout has two machine-independent, deterministic size metrics — the store's bytes-per-triple and the dictionary's bytes-per-term — pinned as CI ratchets, so a regression in the on-disk footprint fails the build. These are *deterministic* (a fixed function of the input, identical on any machine), which is exactly why they may become canonical headline evidence when wired (§8); the draft names them without a value.

AT-RISK — gated on the canonical evaluation (bead `sq-vw3ax.12`); no number appears here until it does. The competitive storage-footprint and query-latency numbers — bytes-per-triple against HDT and qEndpoint, cold- and warm-cache scan latency against native QLever and Virtuoso — are gated on the canonical-host evaluation (§8). The *deterministic* layout ratchets above are machine- independent and will be cited via the paper-factory canonical accessor once wired (`engine.store_bytes_per_triple`, `engine.dict_bytes_per_term`); the *competitive* comparison numbers are canonical-EC2 measurements that do not yet exist.

3. Query evaluation: mixed binary, worst-case-optimal, and bind joins

The evaluator plans a basic graph pattern over the six permutations and executes it with three join strategies chosen by the planner. *Binary joins* (merge and hash) handle the common two-relation case; the sorted permutations make a merge join on a shared variable an id-level linear scan with no re-sort. *Worst-case-optimal* multiway joins use Leapfrog Triejoin [5] — the algorithm of Veldhuizen (ICDT 2014), whose optimality is characterised by the AGM bound and the broader worst-case-optimal-join theory [6] — for the cyclic and many-way star/path patterns where a binary plan would materialise an intermediate result asymptotically larger than the output. *Bind joins* push bindings from an outer pattern into an inner one, the strategy that also carries the engine's federation and reasoner integration. All three operate over the same id-level row representation (a small-vector of integer ids) and the same total-order comparator, so no join strategy pays a representation-conversion tax to hand results to another. This uniformity is not incidental — it is the property §5 turns into the shared substrate.

The evaluator's counting and streaming behaviour is lazy: aggregate and cardinality queries that the planner recognises as not needing full materialisation are answered by streaming counts over the addressed permutation blocks, so a `COUNT`-shaped query need not resident-load the graph.

AT-RISK — gated on the canonical evaluation (bead `sq-vw3ax.12`); no number appears here until it does. Per-operator throughput, the crossover point at which the worst-case-optimal path beats a binary plan on cyclic queries, and end-to-end query latency against native competitors are gated on the canonical-host evaluation (§8). No such number appears in this draft.

4. The shared evaluation substrate

The central systems contribution is that the parts of evaluation common to *every* standards family — the id-tuple row layout, the numeric value tower over dictionary-encoded literals, the join kernels, and SPARQL's total order over RDF terms — live in exactly one place and are consumed unchanged by all of them.

Why one core is not the default. The honest starting point (recorded in the project's design audit, `research/shared-eval-substrate.md`) is that this was *not* the state of the code: the query engine and the OWL RL materialiser each had their own join implementation — the engine's merge/hash/bind/Leapfrog families, and the reasoner's own hash-map adjacency indexes and union-find. Two independent join implementations is the norm across the field, and it is why breadth usually means duplicated machinery. The substrate work is an *extraction and unification*, not a relabelling: the common kernels are lifted into a single leaf crate that depends only on the core term/dictionary layer, placed *below* the query engine in the dependency graph so that a reasoner can reach the kernel without taking a dependency on the whole engine (which would pull the planner, the SPARQL protocol client, and the serializers into a lean reasoner or a browser bundle).

Monomorphic and allocation-lean. The kernel is monomorphic over the concrete id and row types with no dynamic dispatch on the hot path, so the shared abstraction is a source-level unification, not a runtime indirection. The extraction is therefore expected to be *behaviour-* and *performance-neutral* by construction — a code-move-and-generalise, not a rewrite — and that expectation is exactly what the measurement in §8 is designed to confirm or falsify.

How zero-overhead is validated, not asserted. Two mechanisms make "zero measured marginal overhead" a testable claim rather than a slogan. First, the deterministic byte-exact ratchets (the WebAssembly bundle size, the store and dictionary layout metrics) are unchanged by the extraction — a behaviour-altering move would move a byte and fail the ratchet. Second, the cross-family differential conformance floors (§7) are bit-stable across the extraction — a semantics-altering move would change a conformance result. The remaining question — whether the generalised kernel costs any *wall-clock* over the engine's hand-specialised original — is a micro-benchmark on the canonical host, and it is the substrate half of the §8 gate.

AT-RISK — gated on the canonical evaluation (bead `sq-vw3ax.12`); no number appears here until it does. The substrate zero-overhead result — the wall-clock delta of the shared kernel versus the engine's pre-extraction hand-tuned join/numeric/compare loops, per kernel, on the canonical host — is the load-bearing measurement for the "zero-overhead" half of the paper's claim and is gated on bead `sq-vw3ax.12`. It is reported nowhere in this draft; §8 fixes how it will be measured (proposed keys `substrate.overhead_<kernel>`, environment="canonical").

5. Breadth as a measured-substrate claim

We deliberately do not present breadth as a coverage checklist. The claim is structural: each standards family below is implemented as a consumer of the shared kernel of §5, so that adding a standard adds a *front end* (its syntax, its entailment or validation rules, its result shape) over a core that already exists, rather than a new evaluation engine.

- **SPARQL 1.1 / 1.2 query** [15], [16] — the reference consumer; the kernel is the engine's own evaluation core.
- **OWL 2 profiles** [17] — RL, EL, and QL reasoners that materialise or rewrite through the shared semi-naive join instead of a hand-rolled adjacency index, plus a Direct-semantics and a D-entailment path.
- **RIF-Core** [18] — rule evaluation over the same join kernel.
- **RDF stream processing (RSP-QL)** [19] — windowed continuous evaluation reusing the id-level join and comparator.
- **GeoSPARQL** [20] — spatial filter functions over the shared numeric/term layer.
- **SHACL** [21] — shape validation, including SHACL-SPARQL, evaluated through the same engine.

The measured-substrate framing is what distinguishes this from the commercial breadth of GraphDB or Stardog: we do not merely claim the families exist, we claim — and in §7 evidence — that they

share one measured-neutral core, and we expose the extraction's cost as a number rather than hiding it behind a product surface.

6. Correctness and deployment evidence

Cross-family differential conformance scoreboard. Correctness is the load-bearing evidence for a *shared* core: if two families shared a kernel but diverged on the standards, the sharing would be a bug, not a feature. The engine carries a cross-family conformance scoreboard (`crates/sparq-conformance`) that pins a machine-checked pass floor per standard — the SPARQL suites, RDFS/OWL-RL entailment, SHACL core and SHACL-SPARQL, the OWL EL and QL profiles, D-entailment, GeoSPARQL, RSP, JSON-LD, and the protocol/service-description lanes — and fails the build if any floor regresses. The scoreboard is a differential gate, not a self-report: a family's results are checked against the standard's own test manifests, and the floors are the appendix evidence that the shared substrate has not been shared at the cost of conformance. We are careful about what a scoreboard is and is not: it is *self-evaluation against fixed suites*, strong for demonstrating that breadth is not achieved by cutting corners, but it is not community-adoption evidence and we do not present it as such (§9).

Deployment surface: one core, compiled to the browser. The same evaluation core compiles to WebAssembly and runs client-side — the full SPARQL 1.1 engine including Leapfrog Triejoin, plus reasoning, RSP, SHACL, and text-search WASM surfaces — under a *deterministic* bundle-size ratchet that fails CI if the compiled artifact grows. Following the honest verdict of the project's own paper-selection audit, we do *not* claim compiling to WebAssembly as a contribution: Oxigraph, Comunica [22], and others already run SPARQL in the browser, and DuckDB-Wasm [23] set the bar for "X-in-WASM as a first-tier result" with async workers, a paged browser filesystem, and JS UDFs — engineering we have not done. The WASM story is therefore one *deployment-surface* section and one *deterministic* figure (the bundle-size ratchet), evidence that the shared core is portable and size-bounded, not a headline.

AT-RISK — gated on the canonical evaluation (bead `sq-vw3ax.12`); no number appears here until it does. The per-family conformance floor counts and the deterministic bundle-size figure are exact, machine-independent facts and will be cited via the canonical accessor once wired (`engine.conformance_families`, `engine.conformance_<family>_floor`, `engine.wasm_bundle_bytes`). They are named without a value here only because the evidence-record wiring is the downstream bead `sq-gum8.9`; they are *not* competitive performance numbers and are not gated on §8.

7. Evaluation methodology (fixed before the measurement)

The competitive evaluation is committed here, before any competitive result exists, so that results cannot steer the method. It is gated on bead `sq-vw3ax.12` (canonical-host competitor baselines) and is *not* run by this drafting bead. Four experiments, each with its honesty constraint stated:

1. **Canonical-host baselines against native competitors.** SP2Bench [24], WatDiv [25], LUBM [26], and BSBM [27] run on a single fixed, quiet canonical host (a pinned EC2 instance type), against *natively built* QLever, Virtuoso, Jena, Oxigraph, and others — not container images, whose scheduling and I/O overhead would confound a systems comparison. Every reported latency or throughput is an `environment="canonical"` record; a work-box or developer-laptop number can never be a headline (the paper-factory `#headline` accessor refuses a non-canonical record by construction). The engine's own results are reported with the same discipline, and losses are reported, not omitted.
2. **A Sparqloscope run.** The Bast-group Sparqloscope benchmark [28] — designed for a fair, comprehensive SPARQL comparison — is run on the canonical host. A reviewer from that group will expect it, and running it on their terms rather than a hand-picked query mix is the honest choice.

3. **A memory-accounting-honest comparison against qEndpoint.** qEndpoint [11] published the closest prior "large graph on commodity hardware" claim, and memory frugality is the paper's most contestable number because an mmap-resident store can appear frugal by relying on the OS page cache. We therefore *fix the accounting method in advance* and report both quantities side by side: peak *committed* resident memory (RSS excluding reclaimable page cache) and the *page-cache working set* under a stated cache budget, for the engine and for qEndpoint, on identical hardware and data. The claim will be phrased in terms of committed bytes under a bounded cache, the accounting a hostile reviewer would demand — never a page-cache-flattered figure.
4. **The substrate zero-overhead measurement.** The micro-benchmark of §5 — the shared kernel versus the engine's pre-extraction hand-tuned loops, per kernel, on the canonical host — quantifies the marginal cost of the shared abstraction. A non-negligible overhead would *weaken the central claim*, and the honest disposition, decided now, is to report it as measured and re-scope the "zero-overhead" language to the measured bound rather than the aspiration.

Reporting rules, fixed now: every performance and memory figure is a canonical-host record cited through the paper-factory evidence accessor; the deterministic ratchets (bundle size, layout bytes, conformance floors) are reported as the machine-independent facts they are; and any experiment that does not favour the engine is reported alongside those that do.

8. Limitations and honest status

This is a draft without the competitive evaluation. The paper's headline is a performance and memory claim, and none of the four gating experiments (§8) has run. This draft therefore claims *no* competitive number; the architecture, the substrate extraction, the standards breadth, and the conformance evidence (§3–§7) are real and traceable to the codebase, but the numbers that would make the extreme-SOTA claim are not, and are not asserted. If the canonical evaluation shows the engine is not competitive on speed, or that the memory frugality does not survive committed-bytes accounting, the honest disposition — decided in advance — is to re-scope the paper to the contribution that survives measurement (the shared-substrate engineering and the conformance breadth) rather than to soften the benchmark.

Breadth is not adoption. The conformance scoreboard (§7) is self-evaluation against fixed standard test suites. It is strong evidence that the shared substrate does not sacrifice correctness, and it is *not* evidence that anyone outside the project uses the engine. We do not present the scoreboard as community-reuse evidence; external adoption, were it to exist, would be a separate and stronger claim.

The substrate claim rests on a measurement not yet taken. "Zero measured marginal overhead" is falsifiable and, in this draft, unfalsified only because §5′s micro-benchmark is part of the §8 gate. The deterministic ratchets bound the *behavioural* neutrality of the extraction today; the *wall-clock* neutrality is the pending measurement, and the claim is scoped to it.

Fragment and scope. The out-of-core claim is bounded by the working-set behaviour of the query, not a promise of RAM-independence for adversarial full-scan workloads; the breadth claim covers the families enumerated in §6 at their pinned conformance floors, not the entire surface of each standard. Blank-node handling, federated-query performance, and update-workload behaviour are out of scope for this draft's evaluation.

9. Related work

Fast SPARQL engines. QLever [3], Virtuoso [4], RDF-3X [1], Hexastore [2], Tentris [13], and MillenniumDB [12] established the storage and join techniques we build on — exhaustive permutation indexing (RDF-3X, Hexastore), inline id encoding (QLever, Virtuoso), and tensor/graph-native execution (Tentris, MillenniumDB). We adopt the six- permutation layout and the inline-id encoding

explicitly and attribute them; our delta is not a new join algorithm but the extraction of the evaluation core into a substrate shared across standards families.

Worst-case-optimal joins. Leapfrog Triejoin is due to Veldhuizen [5]; the worst-case- optimality theory and the AGM bound are due to Ngo, Porat, Ré, and Rudra and collaborators [6]. We use Leapfrog Triejoin as one of three planner-selected strategies and claim no contribution to the algorithm itself.

Broad reasoning/standards engines. RDFox [7], GraphDB/OWLIM [8], and Jena [9] ship broad reasoning and standards coverage, generally as separately engineered subsystems, and without a peer-reviewed claim of a measured shared evaluation core across the families. Our contribution is precisely that shared-core claim and its measurement, not the breadth per se.

Memory-frugal RDF. HDT [10] and qEndpoint [11] are the closest prior art for serving large graphs on commodity hardware; qEndpoint's "Wikidata on commodity hardware" [11] is the nearest published memory-frugality claim, and §8 targets a memory-accounting-honest comparison against it precisely because that number will be attacked.

In-browser engines. Oxigraph [14] and Comunica [22] already run SPARQL in the browser; DuckDB-Wasm [23] is the bar for a first-tier WASM systems result. We therefore fold the WASM story into a deployment-surface section (§7) rather than claiming it, and report only the deterministic bundle-size ratchet.

Delta. Against all of the above, the paper's delta is: (i) a single out-of-core engine that carries SPARQL query, the OWL profiles, RIF, RSP, GeoSPARQL, and SHACL over *one* evaluation core; (ii) that core extracted into a shared substrate whose behavioural neutrality is validated by deterministic ratchets and a differential conformance scoreboard, and whose wall-clock neutrality is a stated pending measurement; and (iii) an honesty discipline — canonical-only performance records, an explicitly-stated memory-accounting method, losses reported — that makes the extreme-SOTA claim falsifiable rather than promotional.

10. Conclusion

The speed/breadth/frugality trade-off that shapes the RDF-engine landscape is, we argue, largely an artifact of duplicated evaluation machinery: engines are narrow because breadth has meant building a second evaluator, and frugal engines are slow because frugality has meant a restricted core. We described an out-of-core engine — six memory-mapped permutation indexes in the RDF-3X lineage, inline-tagged ids in the QLever/Virtuoso lineage, a mixed binary/worst-case-optimal/bind join evaluator — whose evaluation core is extracted into a single substrate shared unchanged across SPARQL query, the OWL profiles, RIF, RSP, GeoSPARQL, and SHACL. We framed breadth as a measured-substrate claim, validated the extraction's behavioural neutrality with deterministic ratchets and a cross-family differential conformance scoreboard, and — crucially — refused to assert the competitive performance and memory numbers that the extreme-SOTA claim ultimately rests on, because the canonical-host evaluation that would earn them has not run. That evaluation, whose methodology (including the contestable memory-accounting method) is fixed in this draft, is the next step; whatever it shows will be reported against the rules committed here.

References

[1] T. Neumann and G. Weikum, "RDF-3X: A RISC-Style Engine for RDF," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, 2008.

[2] C. Weiss, P. Karras, and A. Bernstein, "Hexastore: Sextuple Indexing for Semantic Web Data Management," *Proceedings of the VLDB Endowment*, vol. 1, no. 1, 2008.

[3] H. Bast and B. Buchhold, "QLever: A Query Engine for Efficient SPARQL+Text Search," in *Proceedings of the 26th ACM International Conference on Information and Knowledge Management (CIKM)*, 2017.

[4] O. Erling and I. Mikhailov, "RDF Support in the Virtuoso DBMS," in *Networked Knowledge — Networked Media*, Springer, 2009.

[5] T. L. Veldhuizen, "Triejoin: A Simple, Worst-Case Optimal Join Algorithm," in *Proceedings of the 17th International Conference on Database Theory (ICDT)*, 2014.

[6] H. Q. Ngo, E. Porat, C. Ré, and A. Rudra, "Worst-Case Optimal Join Algorithms," *Journal of the ACM*, vol. 65, no. 3, 2018.

[7] Y. Nenov, R. Piro, B. Motik, I. Horrocks, Z. Wu, and J. Banerjee, "RDFox: A Highly-Scalable RDF Store," in *Proceedings of the 14th International Semantic Web Conference (ISWC)*, 2015.

[8] B. Bishop, A. Kiryakov, D. Ognyanoff, I. Peikov, Z. Tashev, and R. Velkov, "OWLIM: A Family of Scalable Semantic Repositories," *Semantic Web Journal*, vol. 2, no. 1, 2011.

[9] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson, "Jena: Implementing the Semantic Web Recommendations," in *Proceedings of the 13th International World Wide Web Conference (WWW), Alternate Track*, 2004.

[10] J. D. Fernández, M. A. Martínez-Prieto, C. Gutiérrez, A. Polleres, and M. Arias, "Binary RDF Representation for Publication and Exchange (HDT)," *Journal of Web Semantics*, vol. 19, 2013.

[11] A. Willerval, D. Diefenbach, and A. Bonifati, "The qEndpoint: Wikidata on Commodity Hardware," *Semantic Web Journal*, 2024.

[12] D. Vrgoč, C. Rojas, R. Angles, M. Arenas, and others, "MillenniumDB: An Open-Source Graph Database System," *Data Intelligence*, vol. 5, no. 3, 2023.

[13] A. Bigerl, F. Conrads, C. Behning, M. A. Sherif, M. Saleem, and A.-C. Ngonga Ngomo, "Tentris — A Tensor-Based Triple Store," in *Proceedings of the 19th International Semantic Web Conference (ISWC)*, 2020.

[14] T. Pellissier Tanon, "Oxigraph: A SPARQL Graph Database Written in Rust." [Online]. Available: https://github.com/oxigraph/oxigraph

[15] S. Harris and A. Seaborne, "SPARQL 1.1 Query Language." W3C, Mar. 21, 2013. [Online]. Available: https://www.w3.org/TR/sparql11-query/

[16] "SPARQL 1.2 Query Language." W3C, 2025. [Online]. Available: https://www.w3.org/TR/sparql12-query/

[17] B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz, "OWL 2 Web Ontology Language Profiles (Second Edition)." W3C, Dec. 11, 2012. [Online]. Available: https://www.w3.org/TR/owl2-profiles/

[18] H. Boley, S. Hawke, and A. Polleres, "RIF Core Dialect (Second Edition)." W3C, Feb. 05, 2013. [Online]. Available: https://www.w3.org/TR/rif-core/

[19] D. Dell'Aglio, E. Della Valle, J.-P. Calbimonte, and O. Corcho, "RSP-QL Semantics: A Unifying Query Model to Explain Heterogeneity of RDF Stream Processing Systems," *International Journal on Semantic Web and Information Systems*, vol. 10, no. 4, 2014.

[20] "OGC GeoSPARQL — A Geographic Query Language for RDF Data (Version 1.1)." Open Geospatial Consortium, 2024. [Online]. Available: https://www.ogc.org/standard/geosparql/

[21] H. Knublauch and D. Kontokostas, "Shapes Constraint Language (SHACL)." W3C, Jul. 20, 2017. [Online]. Available: https://www.w3.org/TR/shacl/

[22] R. Taelman, J. Van Herwegen, M. Vander Sande, and R. Verborgh, "Comunica: A Modular SPARQL Query Engine for the Web," in *Proceedings of the 17th International Semantic Web Conference (ISWC)*, 2018.

[23] A. Kohn, G. Moerkotte, and T. Neumann, "DuckDB-Wasm: Fast Analytical Processing for the Web," *Proceedings of the VLDB Endowment*, vol. 15, no. 12, 2022.

[24] M. Schmidt, T. Hornung, G. Lausen, and C. Pinkel, "SP2Bench: A SPARQL Performance Benchmark," in *Proceedings of the 25th IEEE International Conference on Data Engineering (ICDE)*, 2009.

[25] G. Aluç, O. Hartig, M. T. Özsu, and K. Daudjee, "Diversified Stress Testing of RDF Data Management Systems (WatDiv)," in *Proceedings of the 13th International Semantic Web Conference (ISWC)*, 2014.

[26] Y. Guo, Z. Pan, and J. Heflin, "LUBM: A Benchmark for OWL Knowledge Base Systems," *Journal of Web Semantics*, vol. 3, no. 2–3, 2005.

[27] C. Bizer and A. Schultz, "The Berlin SPARQL Benchmark," *International Journal on Semantic Web and Information Systems*, vol. 5, no. 2, 2009.

[28] H. Bast and J. Kalmbach, "Sparqloscope: A Comprehensive and Fair SPARQL Benchmark," in *Proceedings of the 24th International Semantic Web Conference (ISWC)*, 2025.

---

sparq project · DRAFT, in progress — the submission-gating canonical evaluation (bead `sq-vw3ax.12`) has not run; no competitive performance or memory number is claimed. Architecture evidence traces to the codebase: the six memory-mapped permutation indexes in `crates/sparq-core` (`compress.rs`), the inline-tagged / dict-spill id encoding (`dict.rs`, `dictspill.rs`), the mixed join families in `crates/sparq-engine`, the shared kernel in `crates/sparq-substrate` (`rows.rs`, `numeric.rs`, `join.rs`, `compare.rs`), the standards families in `crates/sparq-reason{,-el,-ql,-dl}`, `-rsp`, `-geo`, `-shacl`, and the cross-family scoreboard in `crates/sparq-conformance`. Design record: `research/shared-eval-substrate.md`. Positioning per `research/paper-selection.md` §3.4/§3.5/§3.6

1. §5-P3. Performance and memory numbers will flow through the paper-factory canonical evidence

accessor (bead `sq-gum8.9`) once the §8 experiments run; none appears here.