Client-Driven Offline-First RDF 1.2 using OR-Sets

Jitse De Smet [0009-0002-6513-5013], Ruben Taelman [0000-0001-5118-256X]

IDLab, Department of Electronics and Information Systems, Ghent University – imec.

Abstract. As autonomous agents increasingly write to shared RDF knowledge graphs concurrently, especially in the context of decentralized data ecosystems, principled conflict resolution becomes critical. Existing RDF CRDT approaches either require dedicated infrastructure or introduce consistency boundary problems when storing bookkeeping across multiple resources. We instantiate the state-based OR-Set over RDF 1.2 triple terms — unlike prior work, retaining all bookkeeping within a single dataset without additional infrastructure — and demonstrate a proof-of-concept datastore wrapper to use in query engines, enabling agents to query and update a conflict-free knowledge graph via standard SPARQL, unaware of the underlying merge machinery. Future work will investigate constraint-aware merge semantics for RDF CRDTs.
npm package: https://www.npmjs.com/package/orset-rdf-store
poster: https://orset-rdf.poster.jitsedesmet.be/

1. Introduction

Knowledge graphs expressed in RDF are a central pillar of neuro-symbolic AI: techniques such as GraphRAG [1], GRASP [2], and the Model Context Protocol (MCP) [3] make it increasingly common for multiple agents to read from and write to the same RDF knowledge graph simultaneously — one agent enriches the graph while another reasons over it, and a third corrects errors it discovers.

In decentralized data ecosystems — such as Solid [4], IDSA [5], and Gaia-X [6] — each participant governs their own data without relying on a central authority. A critical challenge is concurrent offline editing: when two agents independently modify the same resource while offline, their updates collide once connection is restored. Asking an agent to resolve such conflicts manually is not always desirable — an agent that has been offline may struggle to grasp how much has changed. Conflict-free Replicated Data Types (CRDTs) [7] offer a principled solution, guaranteeing that any two replicas converge to the same state through well-defined merge algorithms, with no coordination required.

Solutions have been proposed for RDF-based CRDTs [8] and even CRDTs in Solid [9], but the introduction of RDF 1.2 [10] opens new possibilities. RDF 1.2's triple terms — where the object of a triple can itself be a triple — enable efficient, first-class modelling of state-based CRDTs directly within RDF datasets, without any external bookkeeping.

In this paper, we present a **state-based add-wins set CRDT** modelled entirely in RDF 1.2. All bookkeeping stays within the same dataset, enabling atomic updates on the resource without server intervention, and allowing both **CRDT-aware and**

CRDT-unaware clients to coexist without friction. By plugging a CRDT-aware datastore into a query engine, conflict resolution becomes fully transparent to both agents and query engines — as demonstrated by our PoC implementation. Agents can then query and update a conflict-free knowledge graph through standard SPARQL interfaces, without any awareness of the underlying CRDT machinery.

2. Related Work

Decentralized Data Ecosystems and Interface Heterogeneity Decentralized data ecosystems consist of self-governed data stores, each individually positioned in the continuous *Consistency Availability Partition-tolerance* (CAP) space and exposed through heterogeneous interfaces [11] — their only shared infrastructure being the Web itself. Our CRDT is designed to work across such ecosystems by depending only on what every participant already has: the ability to store and exchange RDF data over the Web, plus **ETag support** to avoid mid-air collisions.

CRDTs Existing RDF CRDT approaches require more infrastructure than the Web alone. The m-ld project [12] provides a JavaScript CRDT engine for RDF with a similar eventual-consistency goal, but requires a dedicated m-ld domain server. Gruss et al. [9] store the latest RDF state alongside a binary CRDT representation from a user-chosen library (e.g. Yjs or Automerge), plus a hypermedia description of supported operations. While this lets non-CRDT-aware clients consume the data, it relies on a separate binary blob outside the RDF dataset, introducing a consistency boundary problem where a standard HTTP server cannot atomically update multiple resources. Braid-HTTP [13] proposes adding merge-type headers to HTTP, which is complementary to our approach and could replace our custom patch encoding in a future iteration.
Since we abstract at the RDF dataset level — where only set-based operations exist — we limit ourselves to set CRDTs. The SU-Set [8] extends the OR-Set [7] for RDF using an operation-based approach, whereas we adopt the state-based OR-Set formalization of Bieniusa et al. [14], which aligns naturally with HTTP's Representational State Transfer (REST) semantics — exchanging full dataset state rather than individual operations. Where Bieniusa et al. optimize tombstone removal using causal delivery and a known replica set, we instead exploit NTP time drift bounds — an approach better suited to open decentralized ecosystems where replica sets are unknown.

3. State-based Add-Wins CRDT-RDF

The OR-Set [14] is a replicated set where each element carries a set of unique add-tags; removing an element moves its tags to a tombstone set rather than deleting them outright, ensuring that a concurrent add always wins over a concurrent remove. We instantiate this over RDF 1.2, setting elements as RDF 1.2 triples and unique tags as UUDv4 identifiers. RDF 1.2 triple terms allow a triple to appear in the object position

of another triple, enabling statements to be made about triples directly. We use this to link each tracked triple, its add-tags, and its tombstones through an identifying node, as shown in Fig. 1. A triple is considered present when it has at least one add-tag not in its tombstone set. On merge, multiple identifying nodes for the same triple are consolidated into one.

```
<> a crdt:container .
[] crdt:tags <<( <> a crdt:container )>> ;
   crdt:add "be2f95dd-8ca9-416c-b1d0-81dc45ba54c8"^^crdt:uuid .
:me a :human .
[] crdt:tags <<( :me a :human )>> ;
   crdt:add "96d482e5-3ce7-4f24-a21b-a9d0506ff5b0"^^crdt:uuid ;
   crdt:remove "77c01067-1594-475e-8c64-76f9c4ec4402"^^crdt:uuid .
[] crdt:tags <<( :me a :man )>> ;
   crdt:remove
"216ac011-c2ba-4ff0-825c-7f9cf1efa4ff--2025-03-13T14:00:00Z"^^crdt:stamp-uuid .
```

Fig. 1: RDF 1.2 representation of a state-based OR-Set with two triples present: `<> a crdt:container` and `:me a :human`. The latter has one add-tag not in its tombstone set, so it remains present. Since `:me a :man` has no add-tags, it is not present.

3.1. NTP Tombstone Optimization

Without garbage collection, tombstones accumulate indefinitely — a fundamental challenge for all state-based CRDTs. Bieniusa et al.'s solution requires causal delivery and a known replica set, assumptions that do not hold in open decentralized ecosystems. We instead exploit NTP synchronization: since NTP enters panic mode after a drift exceeding 1000s, any two synchronized clients differ by at most 2000s. By stamping each tag with its creation or tombstone time using `crdt:stamp-uuid` — formatted as `{uuid}--{xsd:dateTime}` — a CRDT dataset can declare a synchronization interval d, after which stale metadata is pruned: 1. an add-tag may be dropped if a newer one exists and all replicas have seen it; 2. a tombstone may be dropped once all replicas have seen it; and 3. the identifying blank node may be dropped if no other triple references it. A tag is universally seen once its timestamp is older than $d + 4000\text{s}$ — a bound derived from the worst-case NTP drift between any two synchronized replicas.

4. Conclusion

We presented a state-based add-wins set CRDT modelled entirely in RDF 1.2, requiring no infrastructure beyond RDF-over-HTTP with ETag support — something most modern Web servers already provide. All bookkeeping stays within the dataset itself, eliminating the consistency boundary problem. An NTP-based tombstone pruning mechanism keeps the CRDT manageable without requiring a known replica

set. By abstracting at the quad-store level, the CRDT can be plugged into existing query engines, making conflict resolution fully transparent to both agents and SPARQL clients alike. A current limitation is the lack of constraint-aware merge: two individually valid states can merge into a state that violates domain constraints, a problem we plan to address in future work.

Acknowledgements. Jitse De Smet is a predoctoral fellow of the Research Foundation – Flanders (FWO) (1SB8525N). The described research activities were supported by SolidLab Vlaanderen (Flemish Government, EWI and RRF project VV023/10). Ruben Taelman is a postdoctoral fellow of the Research Foundation – Flanders (FWO) (1202124N).

References

1. Edge, D., Trinh, H., Cheng, N.: From local to global: A graph rag approach to query-focused summarization. arXiv preprint arXiv:2404.16130. (2024).
2. Walter, S., Bast, H.: GRASP: Generic reasoning and SPARQL generation across knowledge graphs. In: International Semantic Web Conference. pp. 271–289. Springer (2025).
3. Model Context Protocol. https://modelcontextprotocol.io/ (2026).
4. Verborgh, R.: Re-decentralizing the Web, For Good This Time. In: Seneviratne, O. and Hendler, J.A. (eds.) Linking the World's Information - Essays on Tim Berners-Lee's Invention of the World Wide Web. pp. 215–230. ACM (2023). doi:10.1145/3591366.3591385
5. International Data Spaces Association. https://internationaldataspaces.org/
6. Gaia-X. https://gaia-x.eu/
7. Shapiro, M., Preguiça, N., Baquero, C., Zawirski, M.: A comprehensive study of Convergent and Commutative Replicated Data Types. Inria – Centre Paris-Rocquencourt ; INRIA, https://inria.hal.science/inria-00555588 (2011).
8. Ibáñez, L.-D., Skaf-Molli, H., Molli, P., Corby, O.: Synchronizing semantic stores with commutative replicated data types. In: Proceedings of the 21st World Wide Web Conference, WWW 2012. ACM (2012). doi:10.1145/2187980.2188246
9. Gruss, J., Ciortea, A., Salvaneschi, G., Mayer, S.: Real-time Collaboration in Linked Data Systems. In: Proceedings of the ISWC 2023 Posters, Demos and Industry Tracks. CEUR-WS.org (2023).
10. Hartig, O., Champin, P.-A., Kellogg, G., Seaborne, A.: SPARQL 1.2 Query Language. https://www.w3.org/TR/rdf12-concepts/ (2025).
11. Smet, J.D.: Optimizing Write Performance in Decentralized Data Ecosystems. In: The Semantic Web: ESWC 2025 Satellite Events - Portoroz, Slovenia, June 1-5, 2025, Proceedings. Springer (2025). doi:10.1007/978-3-031-99554-5_32
12. m-ld. https://www.m-ld.org/
13. Toomim, M., Little, G., Walker, R., Bellomy, B., Gentle, S.: Braid-HTTP: Synchronization for HTTP. Internet Engineering Task Force, https://datatracker.ietf.org/doc/draft-toomim-httpbis-braid-http/04/ (2023).

14. Bieniusa, A., Zawirski, M., Preguiça, N.M., Shapiro, M., Baquero, C., Balegas, V., Duarte, S.: An optimized conflict-free replicated set. CoRR. (2012).