Running Loops Around Recursive SQL

Daan de Graaf [1], Altan Birler [2], Thomas Neumann [2], and Nikolay Yakovets [1]

[1] Eindhoven University of Technology, Netherlands
{d.j.a.d.graaf,hush}@tue.nl
[2] Technische Universität München, Germany
{altan.birler,neumann}@tum.de

Recursive common table expressions (CTEs) make SQL Turing-complete, but their semantics are poorly suited to iterative workloads such as graph analytics. For example, a shortest path algorithm decreases tentative distances as shorter paths are discovered, but the SQL standard only allows adding tuples, not updating them. Furthermore, many graph algorithms are naturally expressed using multiple pieces of state, but this requires mutual recursion, which most systems do not implement.

We propose `WITH LOOP`, a declarative looping construct for SQL that permits:
(1) carrying multiple state tables between iterations,
(2) removing or updating tuples, and
(3) specifying an explicit termination condition.
Figure 1 shows an example using these features. `INIT` queries compute the initial state tables, `NEXT` queries compute the state for the next iteration, and an `UNTIL` query decides when to stop. Optionally, *in-place aggregation* aggregates tuples into a table by key (e.g., minimum distance per node), and *convergence checking* terminates the loop once a table no longer changes.

```
WITH LOOP reach
INIT reach AS SELECT 0 AS id,
     front AS SELECT trg AS id
              FROM edge
              WHERE src = 0
NEXT reach AS
    SELECT id FROM reach UNION
    SELECT id FROM front,
front AS SELECT trg AS id
    FROM edge, front
    WHERE front.id = edge.src
    EXCEPT SELECT id FROM reach
UNTIL SELECT COUNT(*) = 0 FROM front
```

Figure 1: Reachability (BFS) using `WITH LOOP`.

`WITH LOOP` subsumes recursive CTEs and prior extensions such as `USING KEY` [2]. It is also a natural compilation target for higher-level languages: loops in the graph algorithm language GraphAlg [1] map directly onto `WITH LOOP`, and Datalog programs compile to `WITH LOOP` with semi-naive evaluation.

We integrated `WITH LOOP` into the Umbra database system. On LDBC Graphalytics, `WITH LOOP` is up to 15× faster than an external driver issuing one query per iteration, making Umbra competitive with relational and graph-native systems. On Datalog workloads, we outperform specialized engines on three out of four workloads. Our work shows that, with the right looping construct, a relational database can be an efficient platform for iterative algorithms.

References

[1] Daan de Graaf, Robert Brijder, Soham Chakraborty, George Fletcher, Bram van de Wall, and Nikolay Yakovets. GraphAlg Playground: An Online Platform for Learning and Experimenting with the GraphAlg Language. In *PVLDB*, volume 19, August 2026.

[2] Denis Hirn and Torsten Grust. A Fix for the Fixation on Fixpoints. In *CIDR*, 2023.