German Unnesting: Final Update

15-799 | 2025 Spring | Project 2

Harivallabha Rangarajan, Roland Liu, Sirui Huang

Agenda

- Project Goals
- Implementation Discussion
- Correctness + Benchmarking
- Next Steps

Project Goals

Project Goals

- 80%: 2025 Algorithm Working for Most Operators
- **100%: 2025 Algorithm Working For One Subquery Type (... ish, more like 90%...)**
- 125%: 2025 Algorithm Fully Working

Project Goals

- **100%: 2025 Algorithm Working For One Subquery Type**
  - There are three main types of subqueries: **`SCALAR`**, **`EXISTS`**, **`ANY`**
  - We only implemented for **`SCALAR`** and without lateral support
    - Temporary Fallback method: if there exists subqueries and they are all of **`SCALAR`** type and there are no lateral joins, run our prototype of 2025 unnesting
    - Otherwise, run what already exists

Project Goals

- **100%: 2025 Algorithm Working For One Subquery Type**
  - There are three main types of subqueries: **`SCALAR`**, **`EXISTS`**, **`ANY`**
  - We only implemented for **`SCALAR`** and without lateral support
    - Temporary Fallback method: if there exists subqueries and they are all of **`SCALAR`** type and there are no lateral joins, run our prototype of 2025 unnesting
    - Otherwise, run what already exists
- What is stopping us from 125%?
  - EXISTS and ANY do not differ substantially, but coming across correctness issues on some of the null semantics
  - Didn’t have time to read over and understand lateral joins

Implementation Discussion

DuckDB Unnesting Overview

- Before:
  - DFS through query plan: operator expressions and operator children
  - Upon discovering subquery: flatten dependent join and hook up duplicate eliminated join
  - Upon discovering nested subquery: ignore, unnest after the current one is unnested
  - Does not take into account equivalence columns
- After:
  - Unnest queries together, 2025 paper unnests child queries with ancestor queries
  - Takes advantage of equivalence columns, no need to use dup-elim-get when an eq column can take its place
  - Updated Unnesting Rules

DuckDB Unnesting Overview

- Before:
  - DFS through query plan: operator expressions and operator children
  - Upon discovering subquery: flatten dependent join and hook up duplicate eliminated join
  - Upon discovering nested subquery: ignore, unnest after the current one is unnested
  - Does not take into account equivalence columns

Important Design Considerations

- Unnesting occurs in the binder, before query optimization
- **Shouldn’t be reinventing the wheel**
  - A lot of 2015 unnesting constructs are still relevant in 2025 version, e.g. adding the correlated columns to the group-by
  - Mark-join insertions, null semantics, group-by null semantics, etc.

Important Design Considerations

- Unnesting occurs in the binder, before query optimization
- **Shouldn’t be reinventing the wheel**
  - A lot of 2015 unnesting constructs are still relevant in 2025 version, e.g. adding the correlated columns to the group-by
  - Mark-join insertions, null semantics, group-by null semantics, etc.
- However, still need to substantially change current structure
  - Currently doesn’t support unnesting multiple subqueries simultaneously

Correctness + Benchmarking

Correctness + Benchmarking

- Correctness: tests provided!
  - Small unit tests when debugging
  - DuckDB has large test suite of millions of tests..

Correctness + Benchmarking

- Correctness: tests provided!
  - Small unit tests when debugging
  - DuckDB has large test suite of millions of tests..
- Unnesting Successful
  - Manually created test cases that have to be manually observed to see if
    - Dup-get isn’t called if an equivalence column can be used
    - No large intermediate cross products

Correctness + Benchmarking

- Correctness: tests provided!
  - Small unit tests when debugging
  - DuckDB has large test suite of millions of tests..
- Unnesting Successful
  - Manually created test cases that have to be manually observed to see if
    - Dup-get isn’t called if an equivalence column can be used
    - No large intermediate cross products
- Performance benchmarking
  - Murky.. Nested queries do not exhibit poor behavior under the original version..

Next Steps

Next steps

- Issues with demonstrating performance improvement with our algorithm
  - DuckDB runs Sam Arch's query in 1.2 seconds
    - Likely due to DuckDB's improvements on speeding up query performance
- Need to polish up the code
- PR within the month
  - Hopefully in two weeks