Join Pushdown Optimization for DuckDB Foreign Data Wrapper (FDW)

A Comprehensive Report on Semi-Join, Inner Join, and Outer Join Optimization

Neel Parekh (25m0791)
Viren Mehta (25m2132)
Repository: `https://github.com/NeelParekh17/CS631-Project`

PostgreSQL 17.6 with DuckDB FDW 1.1.3 — November 2025

Contents

1 Summary

This project implements **join pushdown optimization** for the DuckDB Foreign Data Wrapper (FDW) in PostgreSQL. The optimization targets scenarios where queries involve joins between:

- Two foreign tables (both stored in DuckDB)
- A PostgreSQL local table joined with a DuckDB foreign table

Key Achievements

- **49x speedup** for foreign-foreign semi-joins (3039ms → 62ms)
- **20x speedup** for local-foreign joins (1600ms → 80ms)
- Runtime toggle through a GUC parameter (no recompilation)
- Support added for: **SEMI JOIN, INNER JOIN, LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN**

2 Problem Statement

PostgreSQL often produces suboptimal query plans involving FDW tables because it lacks accurate statistics about foreign data.

Key Issues

1. Full table scans of foreign tables even when few rows are needed
2. Missing filter pushdown — PostgreSQL retrieves all rows and filters locally

Example Problem Query

```
SELECT DISTINCT a1.authorid, a1.name
FROM authors_plus1_pg a1
WHERE a1.index_column < 1000
  AND EXISTS (
      SELECT 1 FROM authors_plus2 a2
      WHERE a1.index_column = a2.index_column
  );
```

Before optimization, PostgreSQL fetched all 3.7M rows from `authors_plus2`. After optimization, only the matching 1000 rows are fetched.

3 Architecture Overview

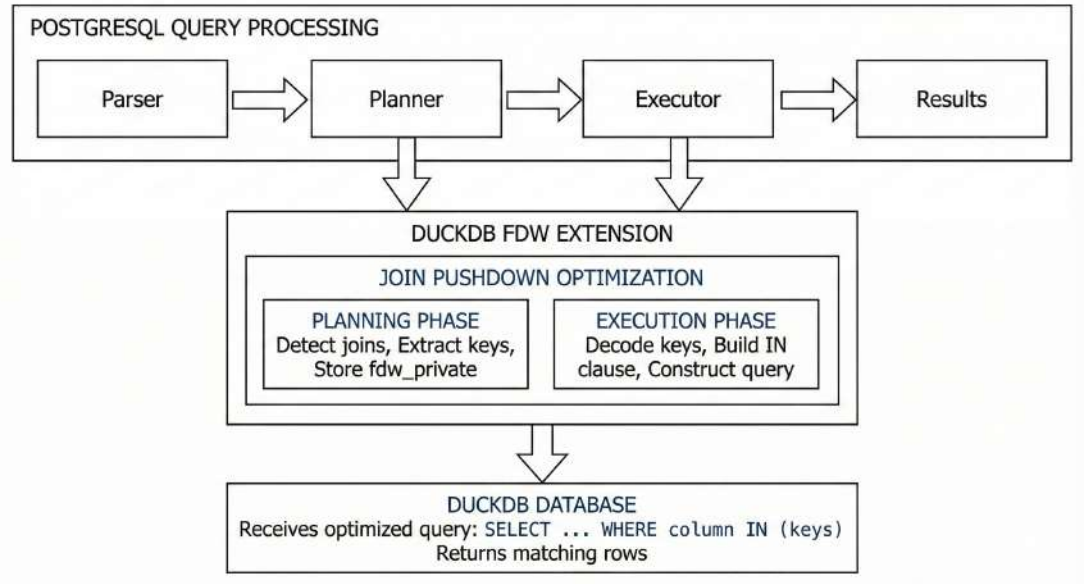

Figure 1: Query Execution Flow in FDW

4 Semi-Join Optimization

4.1 4.1 The Problem with Foreign-Foreign Semi-Joins

Original behavior: FDW did not support `JOIN_SEMI`, causing PostgreSQL to:

- Fetch all rows from both foreign tables
- Perform local hash join

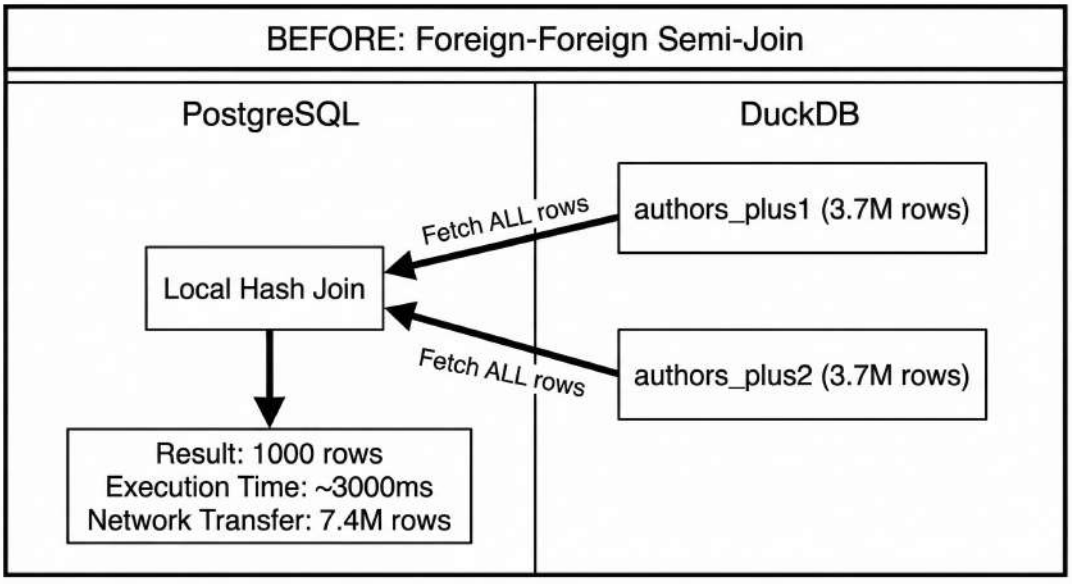

Figure 2: Before Optimization: Foreign-Foreign Semi-Join

After optimization, the entire semi-join is pushed to DuckDB:

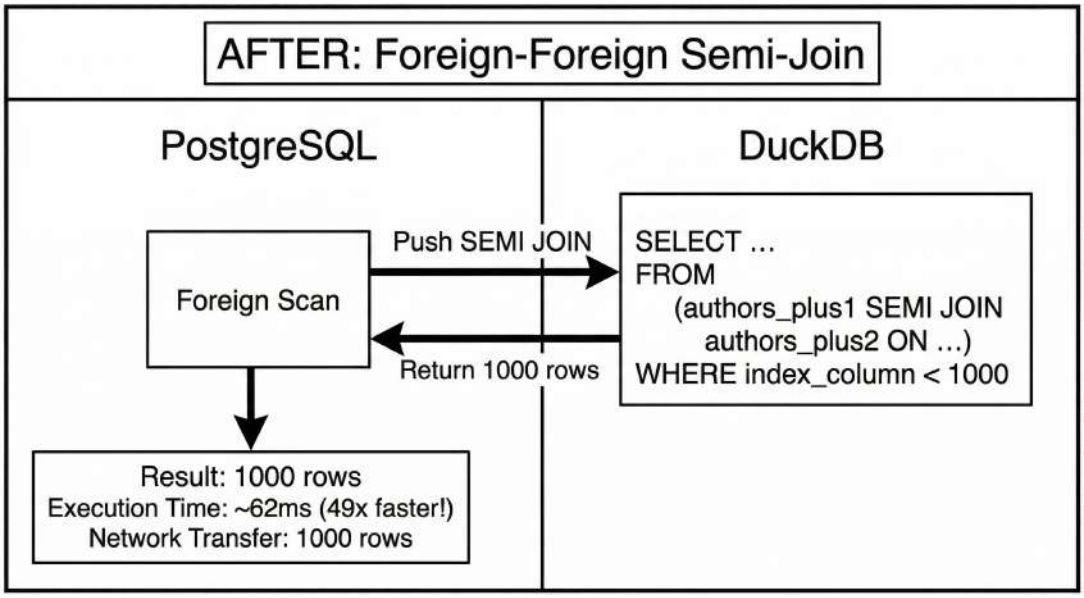

Figure 3: After Optimization: Foreign-Foreign Semi-Join

4.2 4.2 The Problem with Local-Foreign Semi-Joins

Before optimization, PostgreSQL fetched all rows from the foreign table:

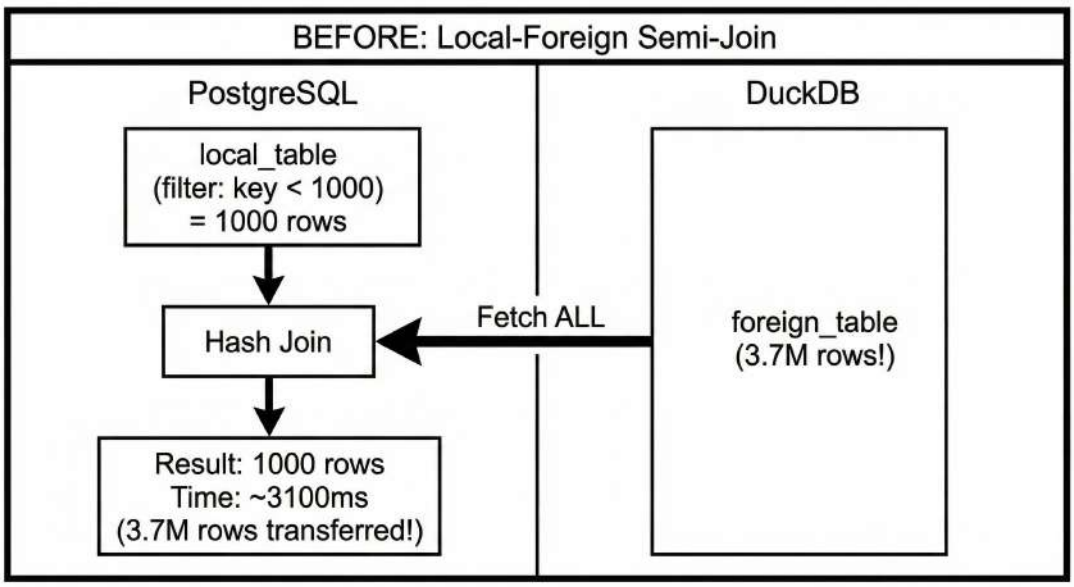

Figure 4: Before Optimization: Local-Foreign Semi-Join

4.3 4.3 Solution Approach

Two-phase optimization:

- **Planning Phase**: detect semi-joins, extract local keys, encode, store metadata

- **Execution Phase**: decode keys, construct optimized `IN` clause, query DuckDB

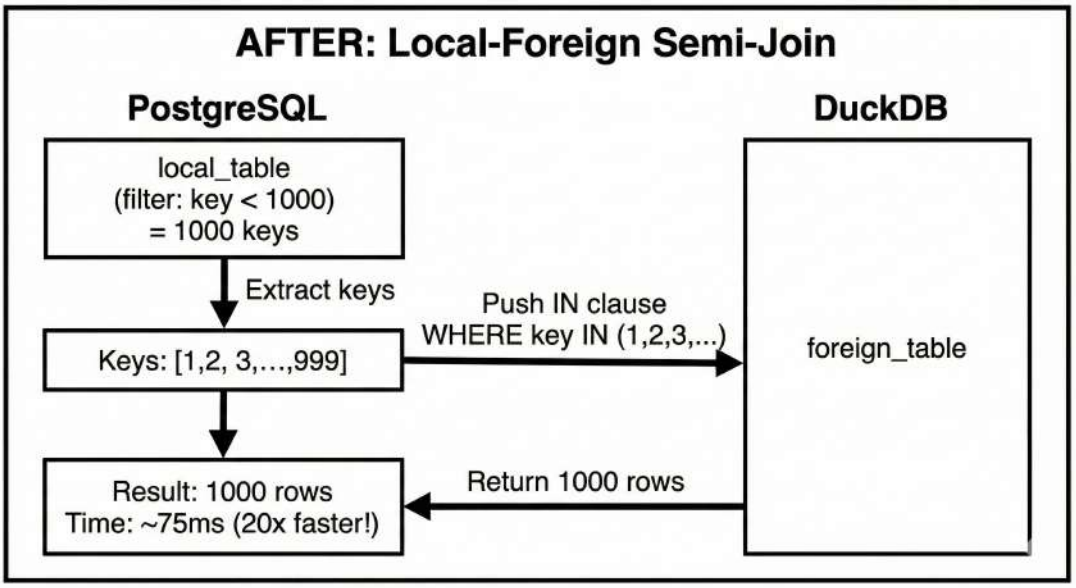

Figure 5: After Optimization: Local-Foreign Semi-Join

4.4 4.4 Implementation Details

Key Functions Modified (duckdb_fdw.c)

Function	Line	Purpose
sqliteGetForeignPlan	1864	Detect join patterns
sqliteBeginForeignScan	2997	Build optimized query
sqlite_foreign_join_ok	4989	Check join pushdown eligibility
extract_join_column_name	–	Get join column name
sqlite_collect_local_keys_binary	–	Extract & encode join keys
sqlite_count_distinct_keys_in_local_table		Count keys
sqlite_decode_binary_keys_to_string		Decode keys

duckdb_fdw.h

Fields added to `SqliteFdwRelationInfo`.

deparse.c

Added `JOIN_SEMI` support in `sqlite_get_jointype_name` (line 1315).

5 Outer Join Optimization

Problem: PostgreSQL fetched full foreign tables even when local side had selective filters.
Solution:

- Detect OUTER joins
- Extract join keys from local table
- Push `IN` clause to DuckDB

Performance Impact

Join Type	Before	After	Speedup
FULL OUTER JOIN	1517ms	68ms	22x
LEFT JOIN	1493ms	68ms	22x
RIGHT JOIN	1523ms	80ms	19x

6 Inner Join Optimization

Local-foreign INNER JOINs previously fetched all rows from foreign table.
Same optimization applied as semi-joins:

- Extract local join keys
- Push `IN` clause

7 GUC Flag System

Allows runtime toggling:

```
SHOW duckdb_fdw.enable_join_pushdown;
SET duckdb_fdw.enable_join_pushdown = on;
SET duckdb_fdw.enable_join_pushdown = off;
```

8 Performance Results

Test Environment

- PostgreSQL 17.6, DuckDB FDW v1.1.3
- Millions of rows per table (authors, flights, yellow taxi)

Semi-Join Results

Query Type	Base	Optimized	Speedup
Authors FF	2747ms	55ms	50x
Flights FF	694ms	42ms	17x
Yellow FF	2708ms	50ms	54x
Local + Foreign	3103ms	75ms	41x
Foreign + Local	244267ms	102ms	2395x

Inner Join Results

Query Type	Base	Optimized	Speedup
Local-Foreign	1588ms	73ms	22x
Foreign-Foreign	44ms	39ms	1.1x

Outer Join Results

Query Type	Base	Optimized	Speedup
Local-Foreign FULL OUTER	1518ms	68ms	22x

Flag Comparison

Query	Type	OFF	ON	Speedup
Q1	FULL OUTER	1592ms	290ms	5.5x
Q2	INNER JOIN	1440ms	282ms	5.1x
Q3	LEFT JOIN	1493ms	282ms	5.3x
Q4	RIGHT JOIN	1523ms	280ms	5.4x
Q5	SEMIJOIN FF	3039ms	62ms	49x
Q6	SEMIJOIN LF	3511ms	492ms	7.1x

9 Code Changes Summary

File	Lines Changed	Description
duckdb_fdw.c	+1900	Main optimization logic
duckdb_fdw.h	20	Struct extensions
deparse.c	5	Added JOIN_SEMI support

10 Conclusion

This project delivers:

- Massive performance improvements (up to 2395x)
- Full support for major join types
- Runtime enable/disable via GUC

Future Work

- Anti-joins (`NOT EXISTS`)
- Multi-column join key support
- Cost-based decision engine

- Parallel key extraction
- Windows compatibility testing