PGcuckoo

Injecting Physical Plans into PostgreSQL

Denis Hirn

denis.hirn@uni-tuebingen.de

University of Tübingen

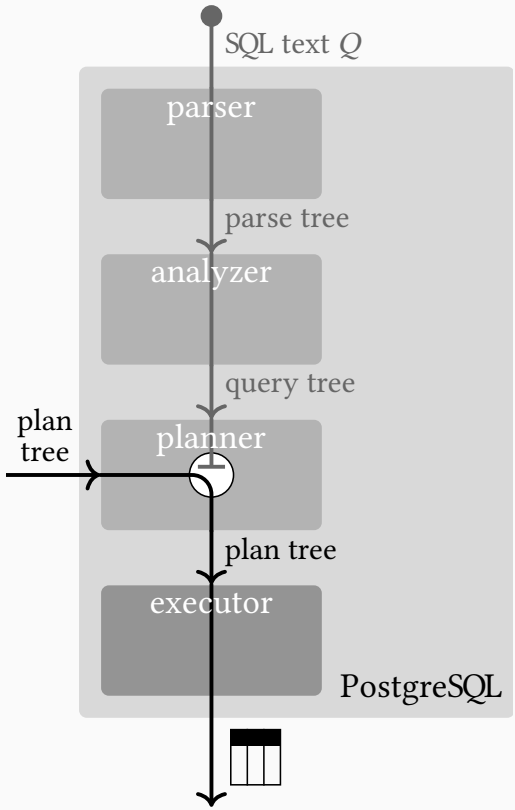
BTW 2019, Rostock
March 5, 2019

THE QUERY PIPELINE OF POSTGRESQL

The query pipeline of PostgreSQL consists of four phases:

1. Parser: create query AST from query string
2. Analyzer: semantic analysis and rewriting
3. Planner: find the best evaluation strategy
4. Executor: evaluate the query

Each phase creates a new data structure.



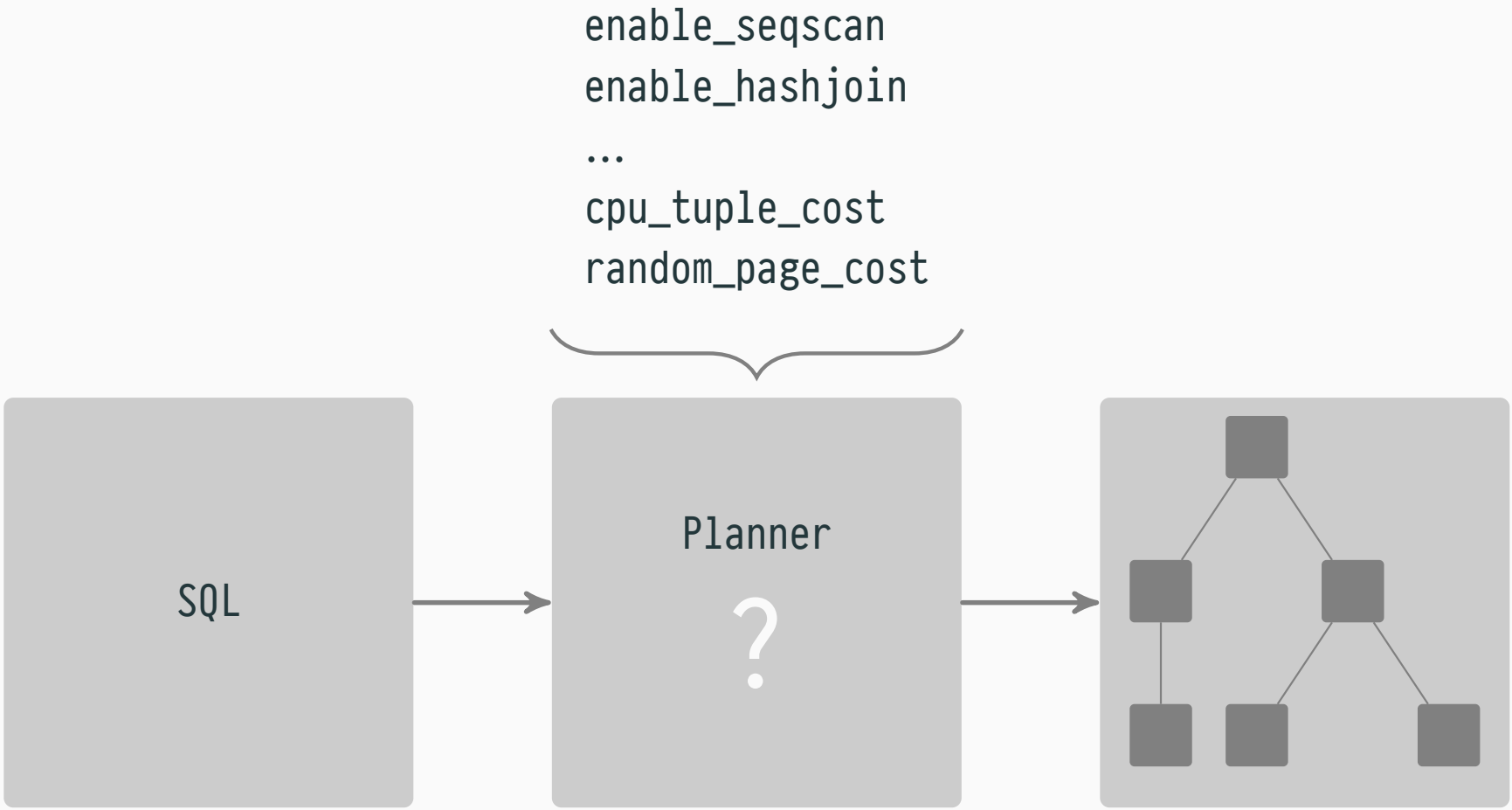
WHAT IS AN EXECUTION PLAN?

Query planning is critical to database performance:

- SQL only specifies *what* to compute but not *how*
- There are many equivalent plans for non trivial queries
- The planner enumerates *all* possible plans and chooses the cheapest one, based on a cost model (System R Algorithm)
- Query runtime depends on the quality of the execution plans

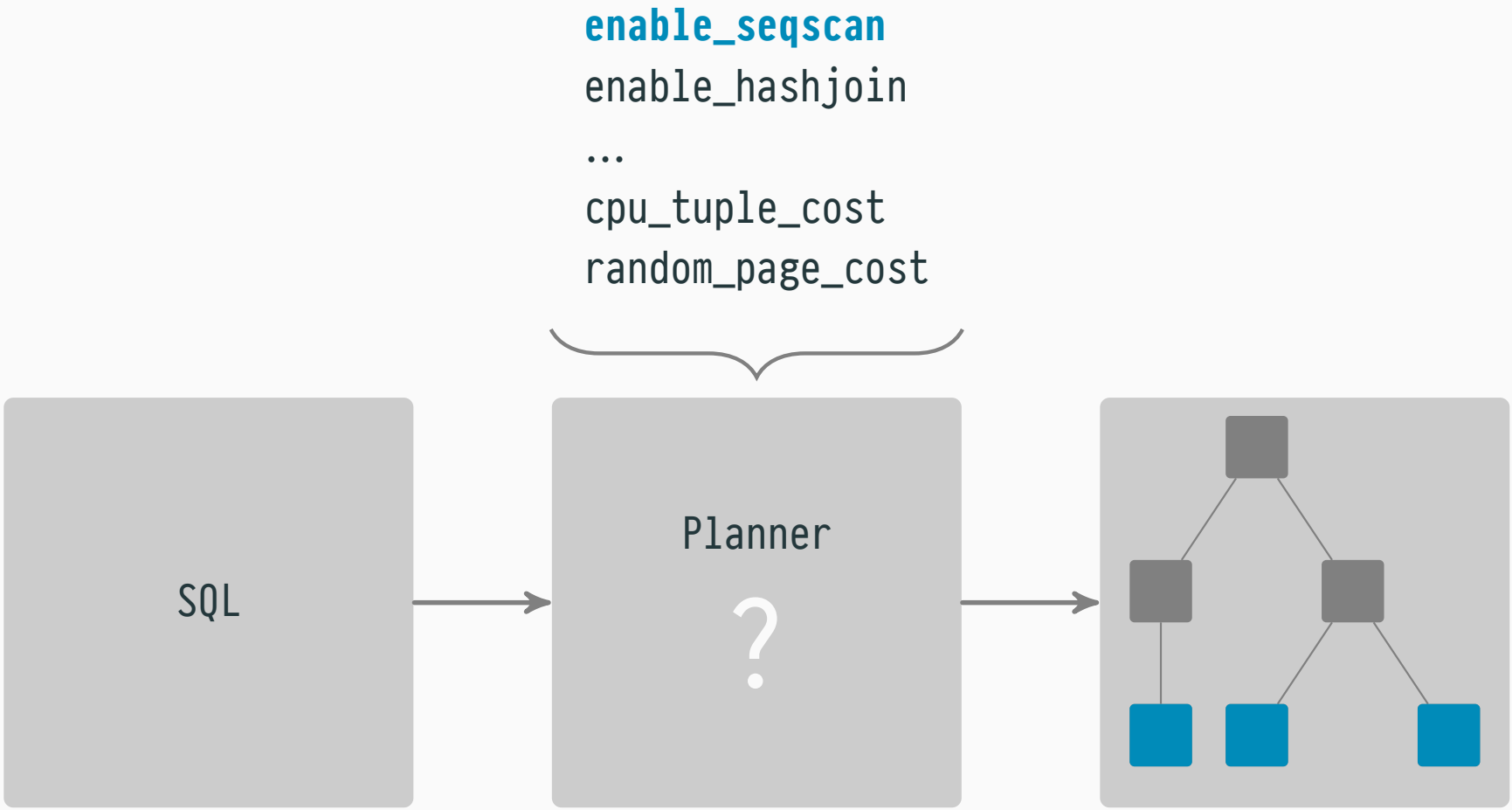
Can we *hint* a specific plan?

PLANNER CONFIGURATION



- It is hard to predict which plan gets selected by the planner

PLANNER CONFIGURATION



- It is hard to predict which plan gets selected by the planner

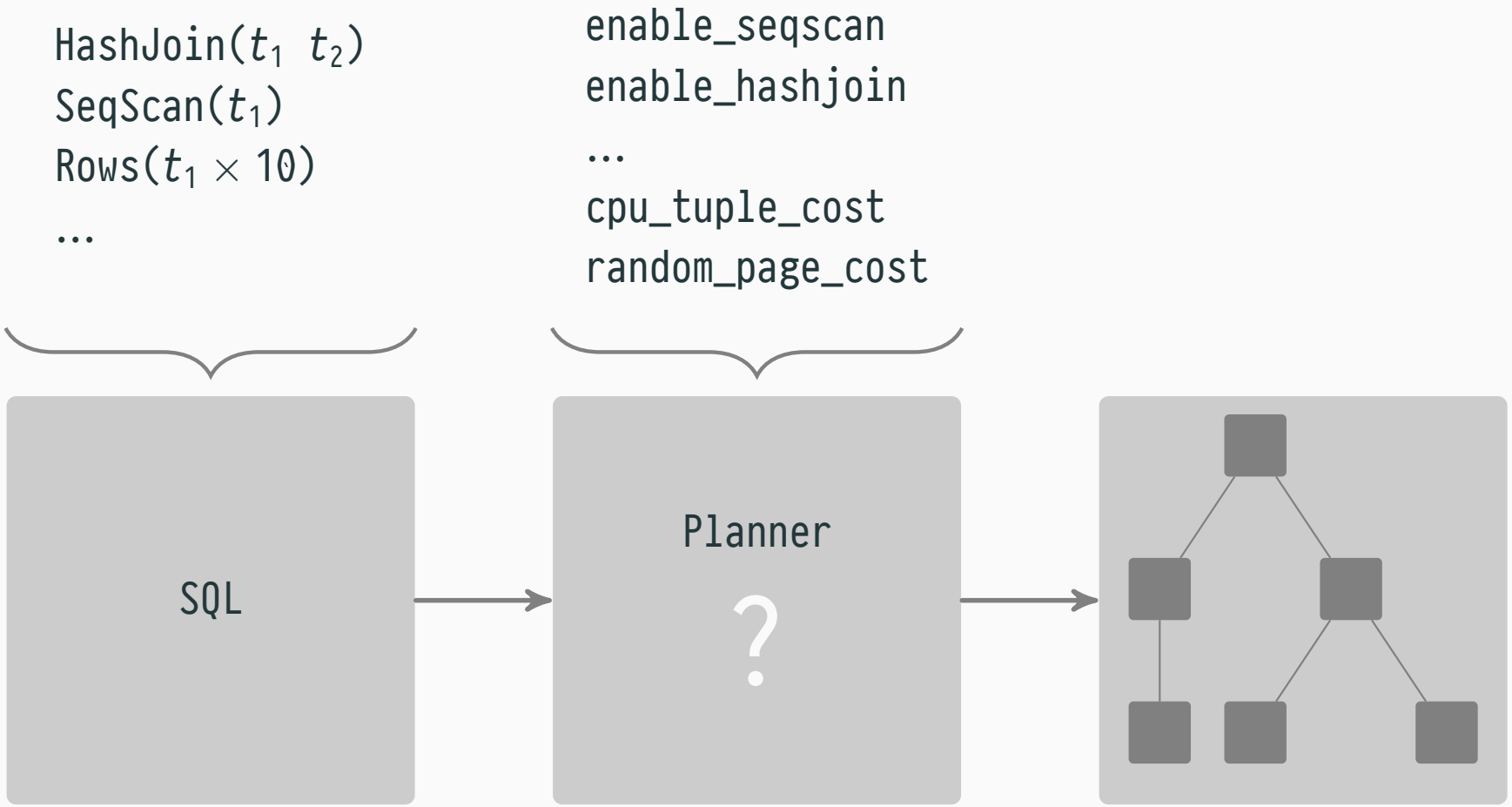
EXTENSION pg_hint_plan

PostgreSQL does not support plan hinting by default.

- pg_hint_plan uses SQL comments to tweak execution plans. This allows to hint scan methods, join orders and algorithms, as well as row count estimates

```
postgres=# /*+
postgres=#     Rows(t1 + 100)
postgres=#     SeqScan(t1)
postgres*#     NestLoop(t1 t2)
postgres*#     MergeJoin(t1 t2 t3)
postgres*#  */
postgres-# SELECT * FROM table1 t1
postgres-#     JOIN table2 t2 ON (t1.key = t2.key)
postgres-#     JOIN table3 t3 ON (t2.key = t3.key);
```

PLANNER CONFIGURATION



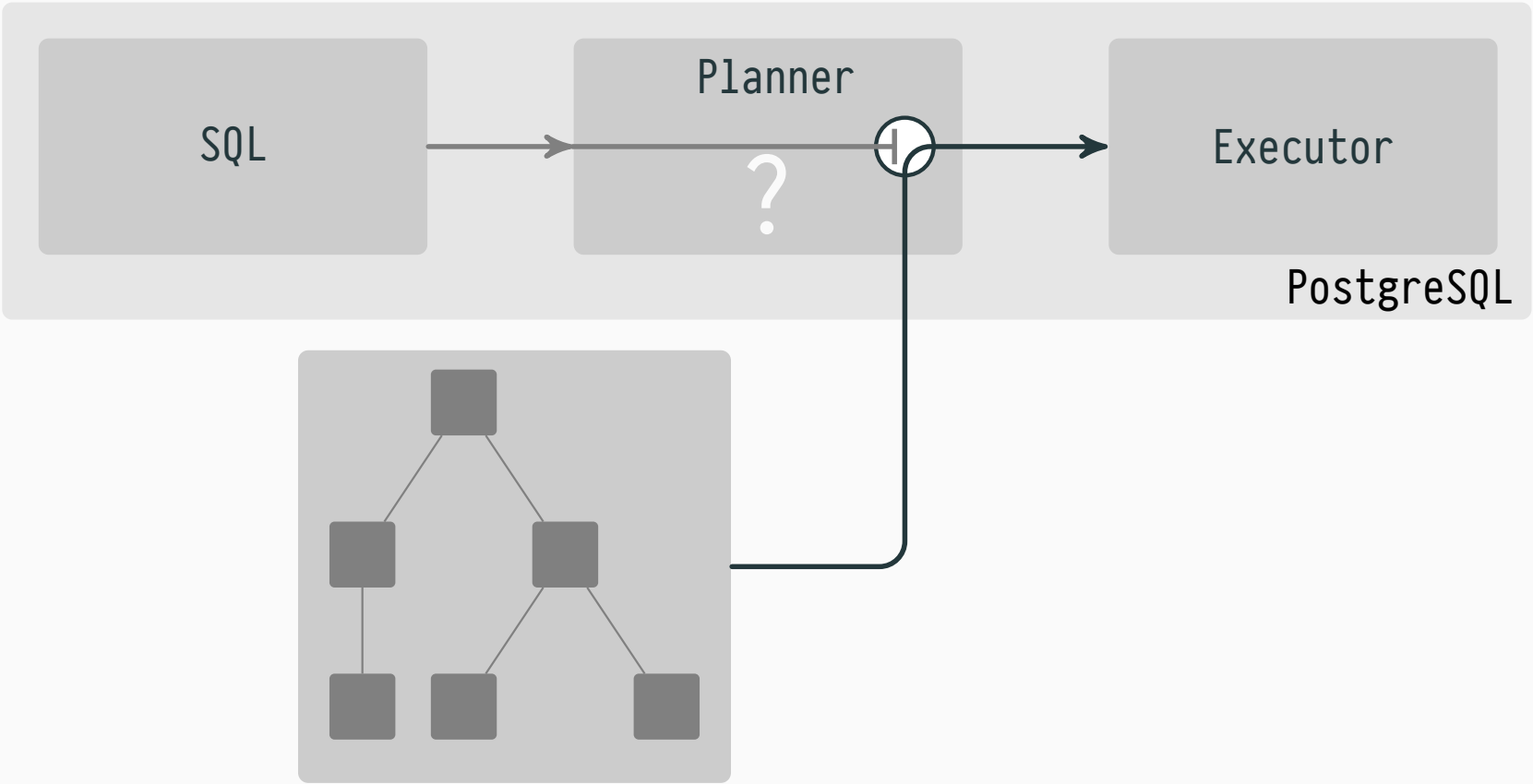
- It is hard to predict which plan gets selected by the planner
- In general it is not possible to select a specific plan

PLAN FORCING

Open problems:

1. If a plan is not part of the search space, the planner can not select it. No hint or planner configuration can change this
2. It is impossible to design execution plans from scratch

Solution: Get rid of the planner and inject a plan directly instead.



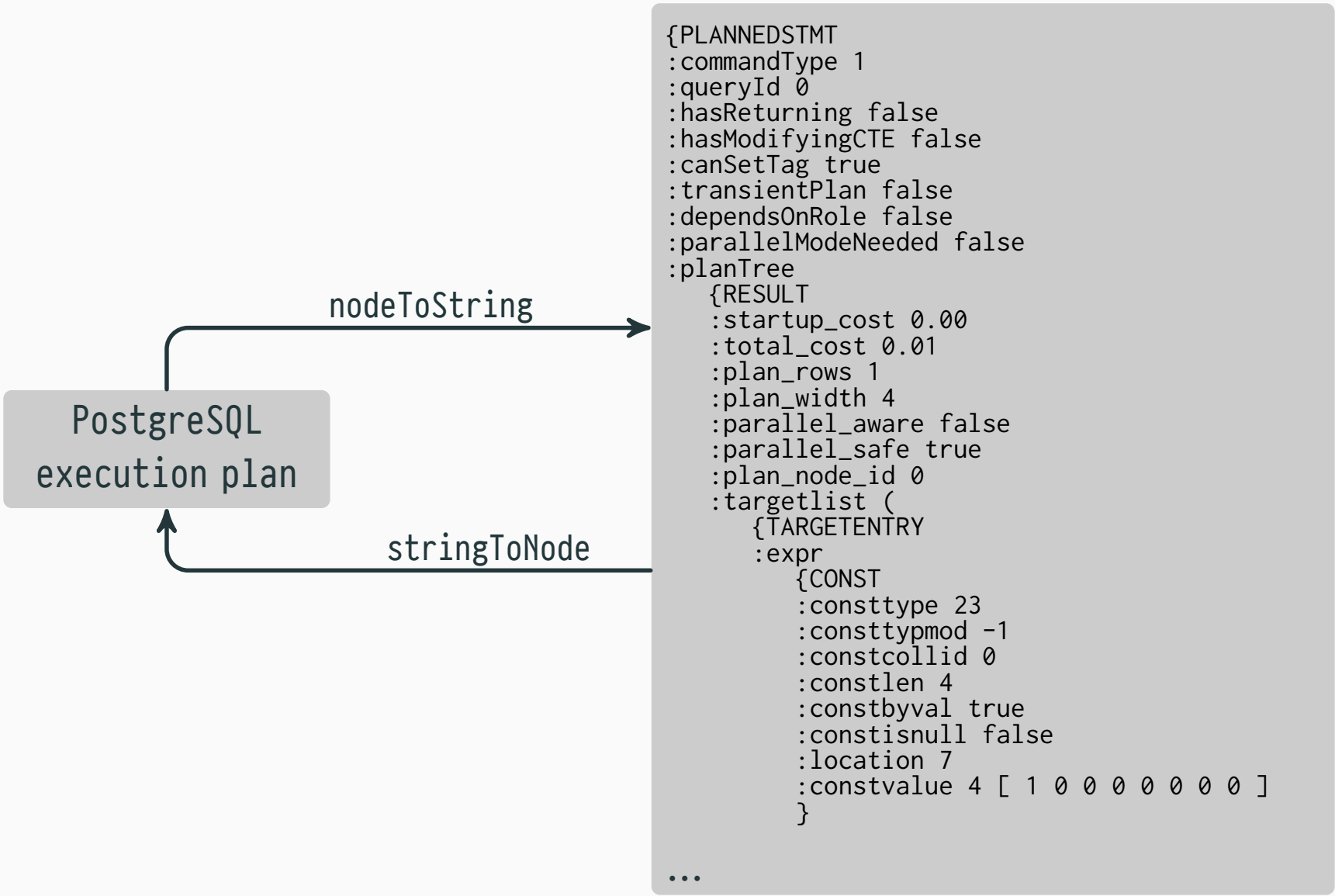
REQUIREMENTS OF PLAN FORCING

Language C extension needs an interface to:

1. Load and store execution plans
2. Execute loaded plan
3. Return result of the execution as table valued function

SERIALIZATION AND DESERIALIZATION OF EXECUTION PLANS

PostgreSQL has internal modules to load and store execution plans

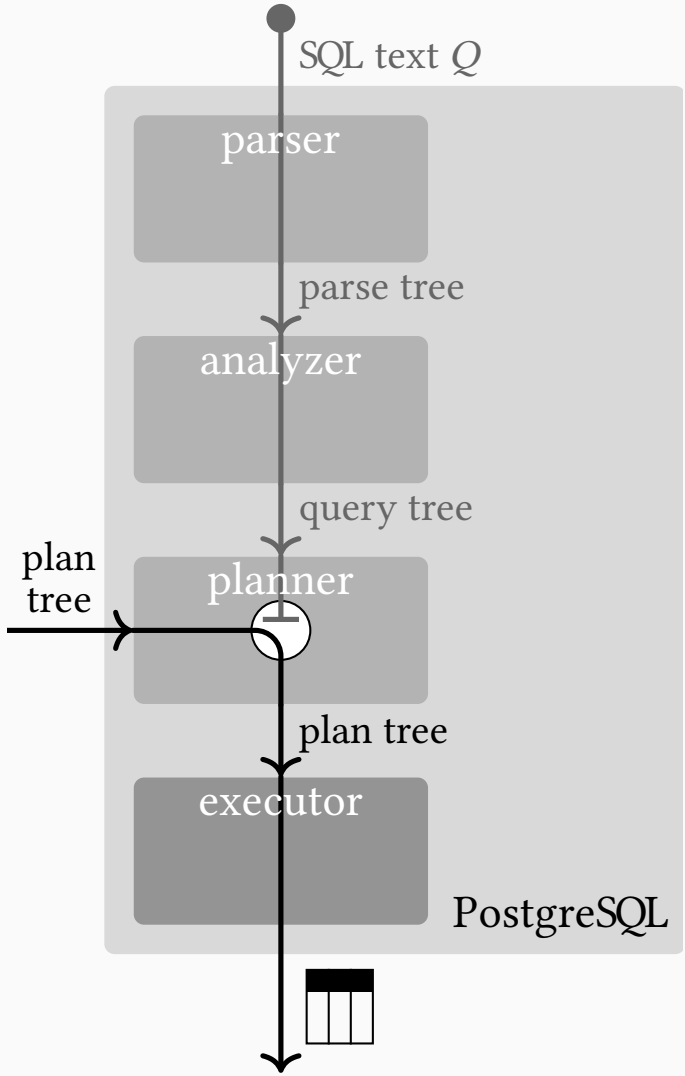


PLANNER HOOK

```
PlannedStmt *
planner(Query *parse, int cursorOptions, ParamListInfo boundParams)
{
  PlannedStmt *result;

  if (planner_hook)
    result = (*planner_hook) (parse, cursorOptions, boundParams);
  else
    result = standard_planner(parse, cursorOptions, boundParams);
  return result;
}
```

INJECTION OF A PHYSICAL PLAN

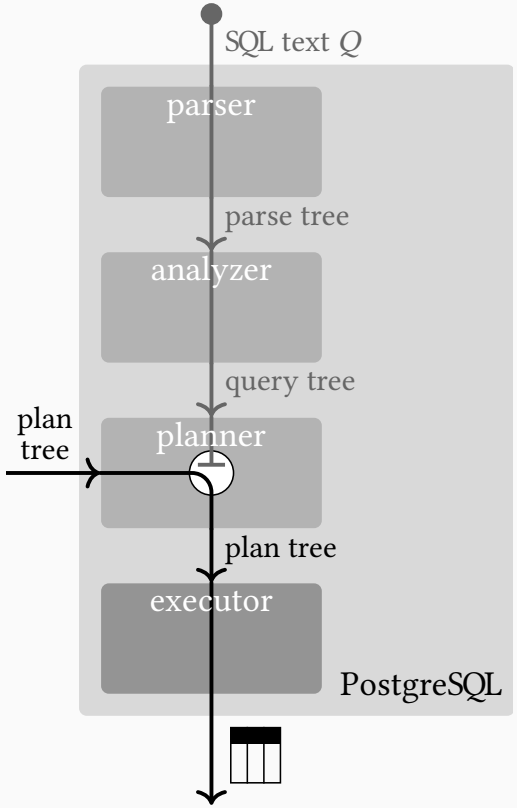
```c
//Define global variable
//for physical execution plan
PlannedStmt *myPlan;

PlannedStmt *
myPlanner(Query *parse,
          int cursorOptions,
          ParamListInfo boundParams)
{
    // Statically return 'myPlan'
    return myPlan;
}

Datum plan_execute(String plan)
{
    // [...]
    myPlan = (Node) stringToNode(plan);
    // Bypass standard_planner
    planner_hook = &myPlanner;

    // issue dummy query
    res = SPI_exec("select 1;", 0);

    planner_hook = NULL;
    return res;
}
```



APPLICATIONS OF PLAN INJECTION

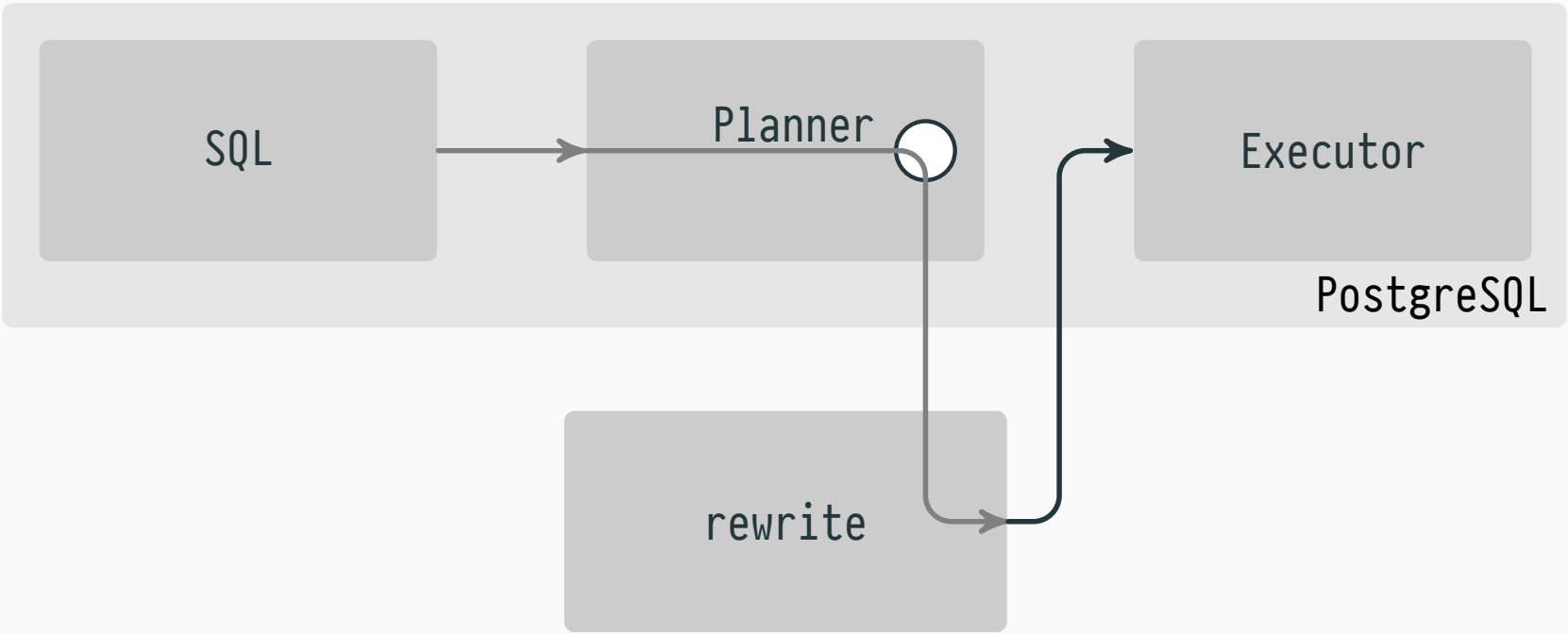
Plan injection enables *full* control over execution plans.

- Simulate planner functionality that is not (yet) available
- Externalize former strictly system-internal query processing steps (Advanced algebraic rewriting of plans, e.g. for unnesting of correlated subqueries)



EXTERNAL REWRITING

1. Use PostgreSQL to create an initial plan
2. Transform plan into relational algebra, if needed
3. Rule based plan optimization
4. Execute improved plan



PGCUCKOO

INJECTING PHYSICAL PLANS INTO POSTGRESQL

Denis Hirn

denis.hirn@uni-tuebingen.de

University of Tübingen

BTW 2019, Rostock
March 5, 2019