ORACLE

The MySQL Hypergraph Optimizer

What, Why and How

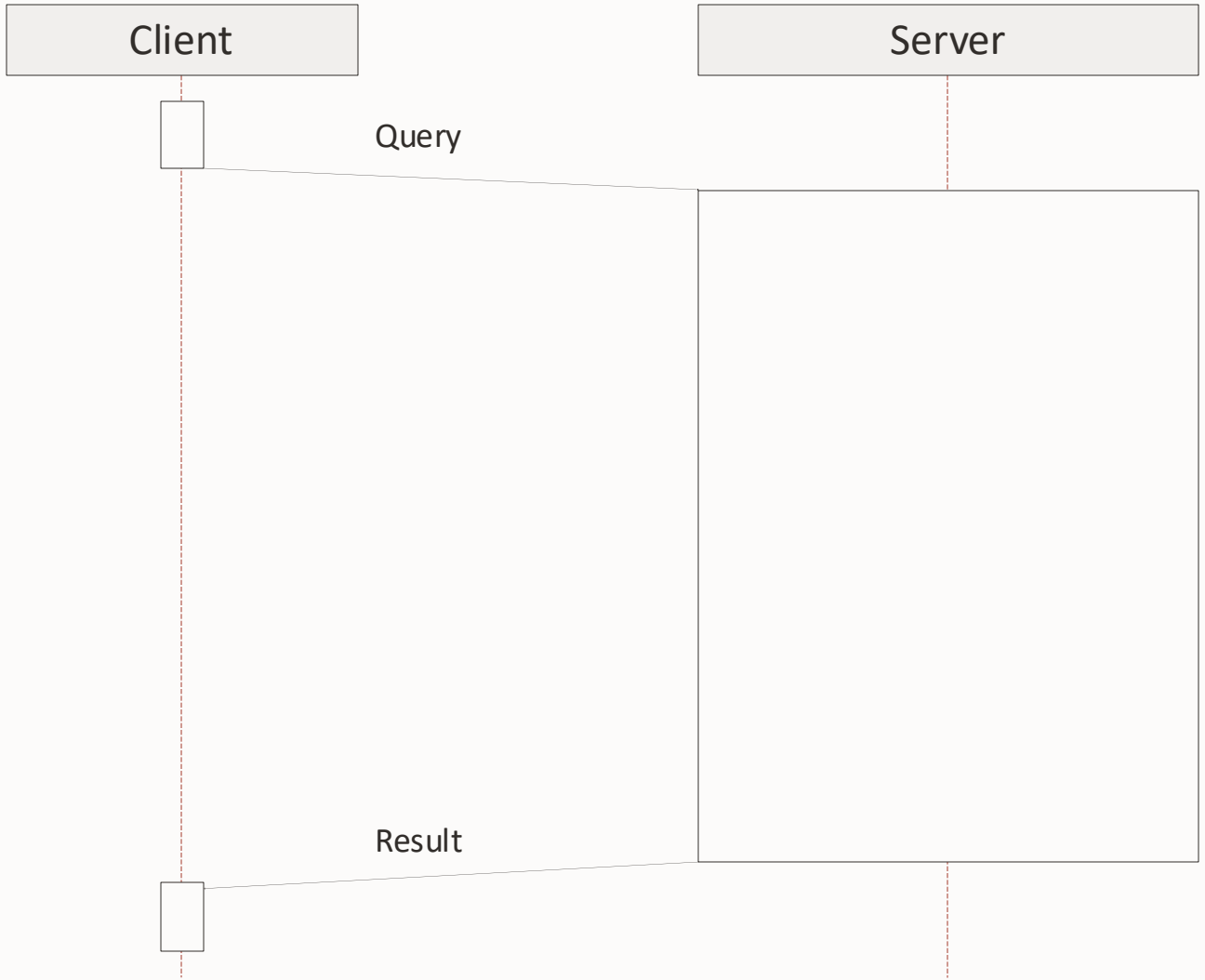
Norvald H. Ryeng
Software Development Director
MySQL Optimizer Team
January 31, 2025

Agenda

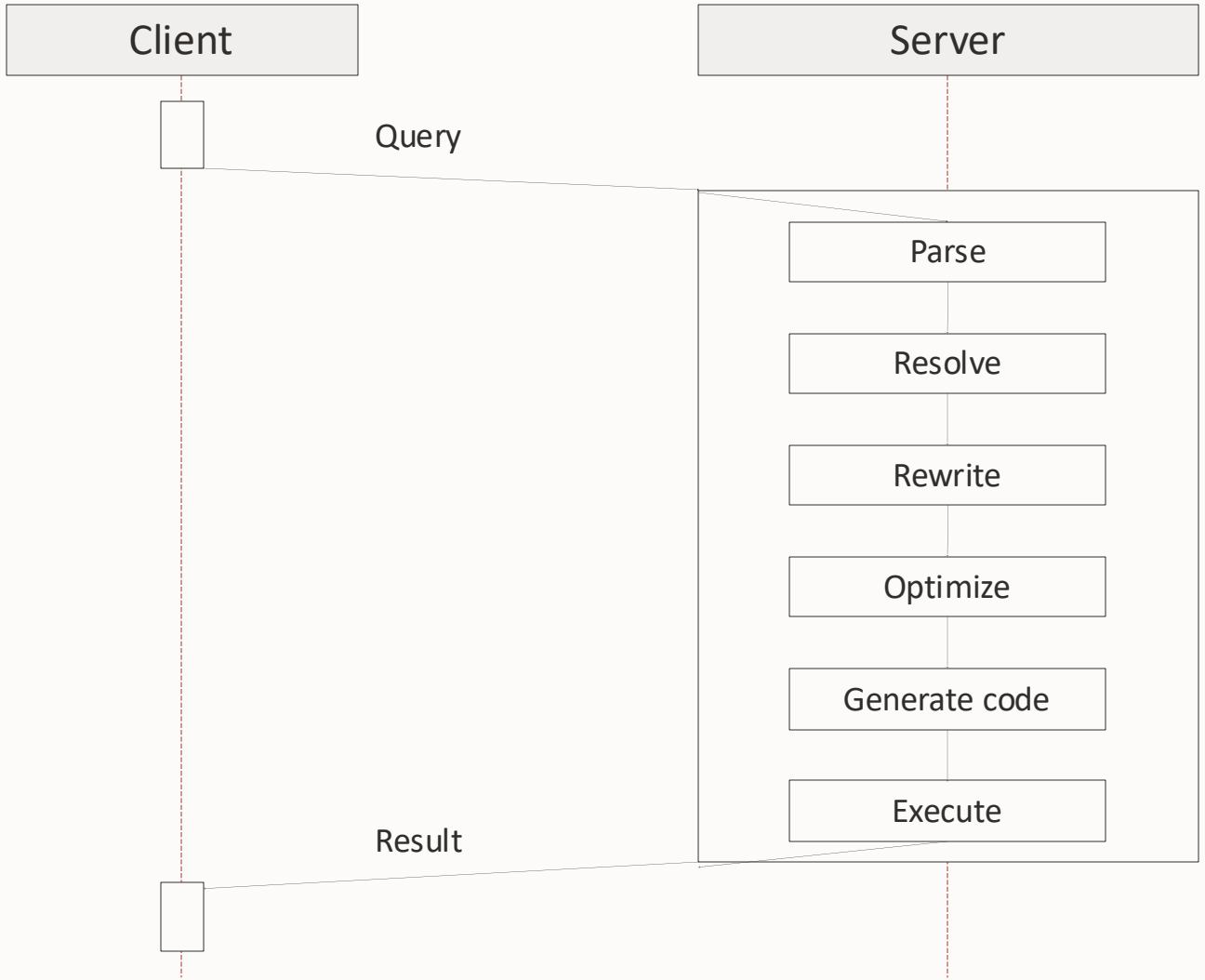
1. Query processing 101
2. Why a new optimizer?
3. What is a hypergraph, and what is it doing in my optimizer?
4. How can I start using the hypergraph optimizer?

Query processing 101

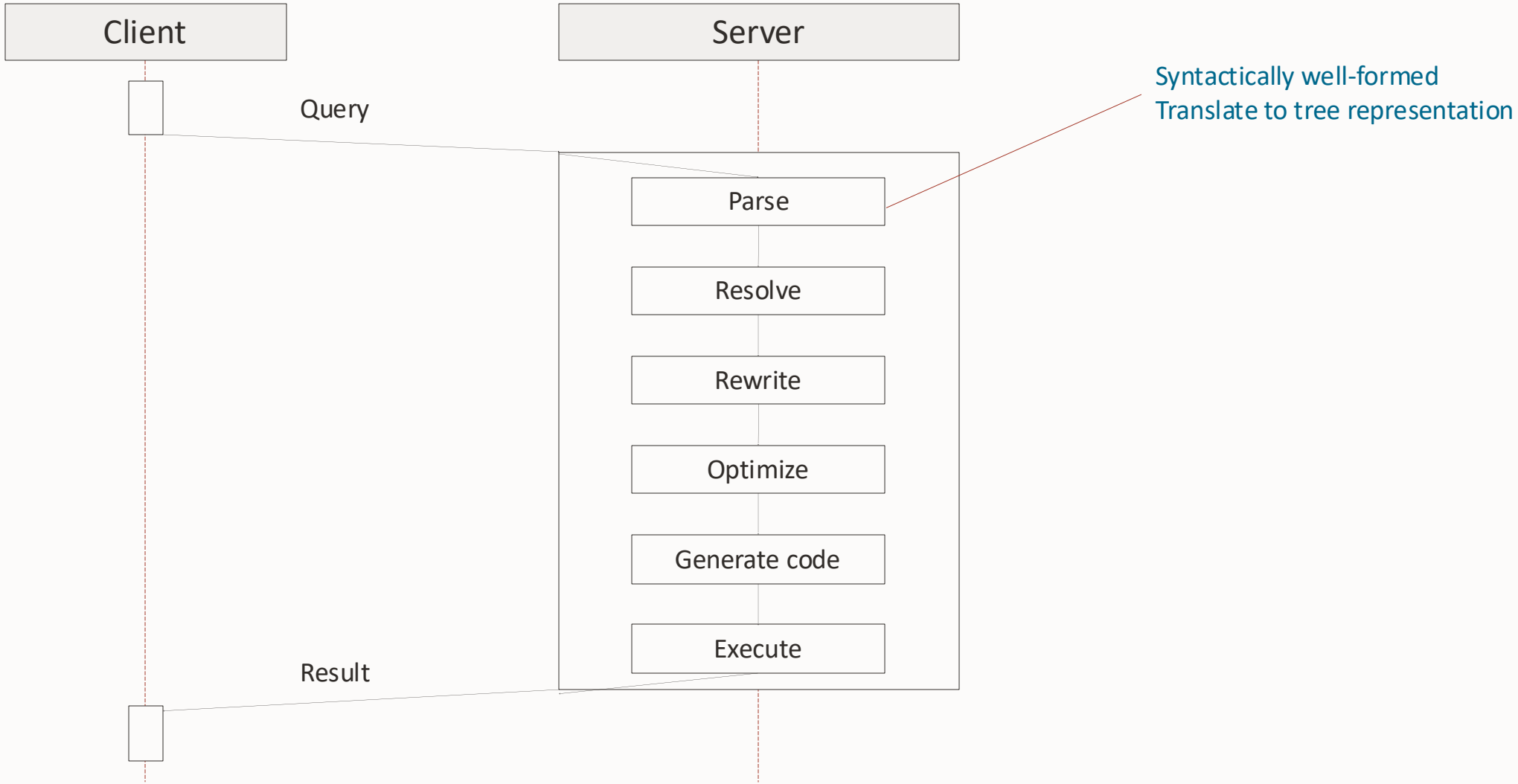
Query processing



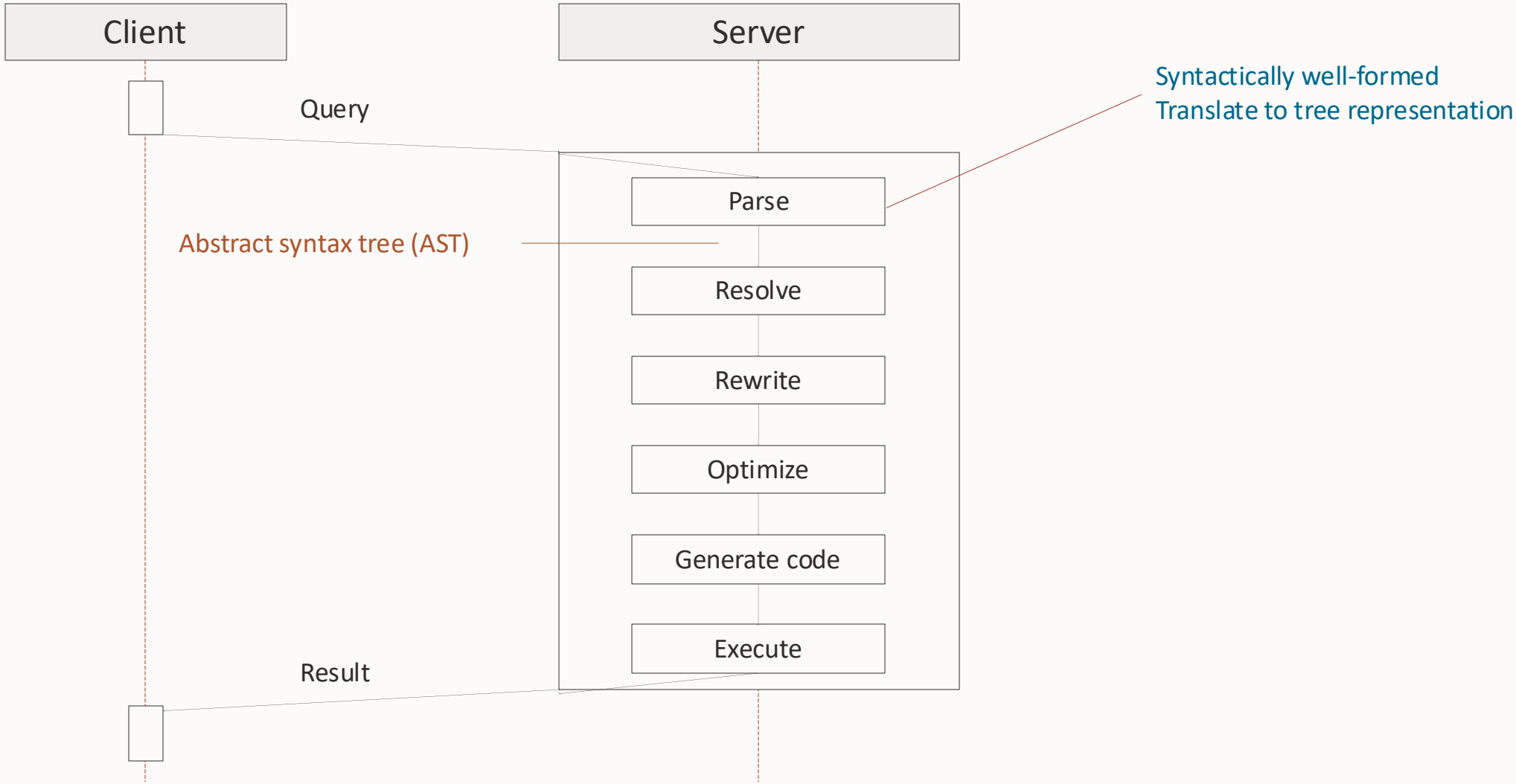
Query processing



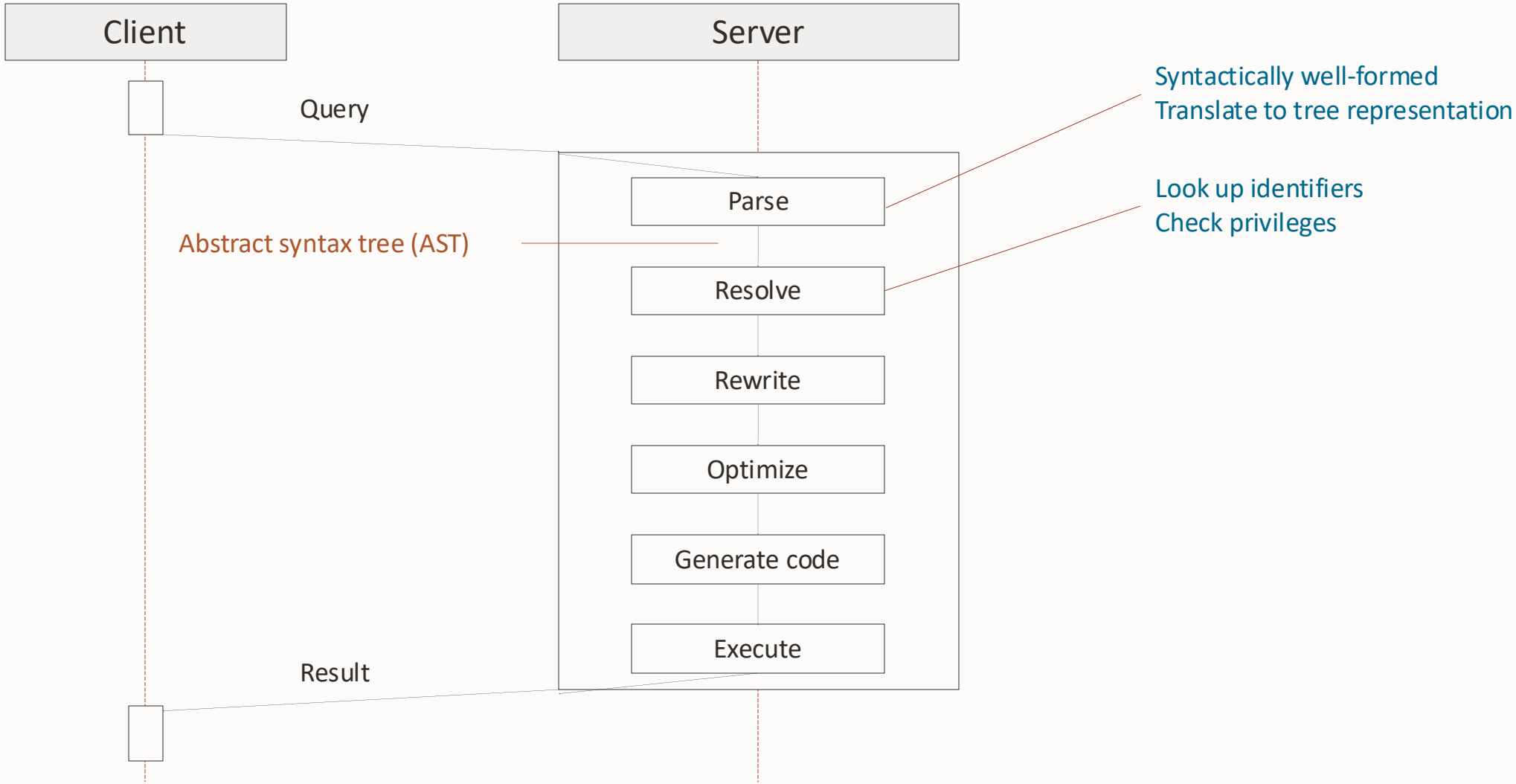
Query processing



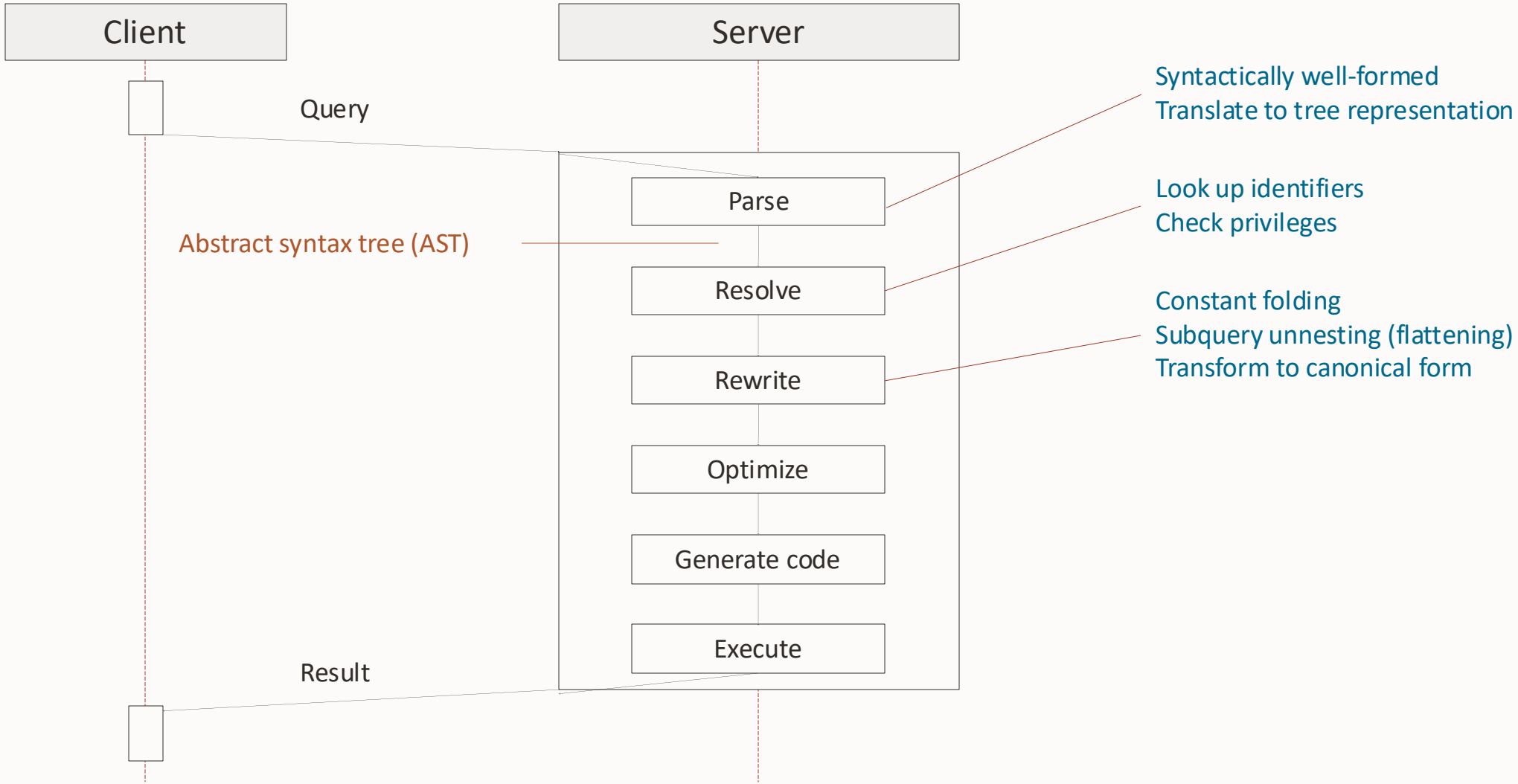
Query processing



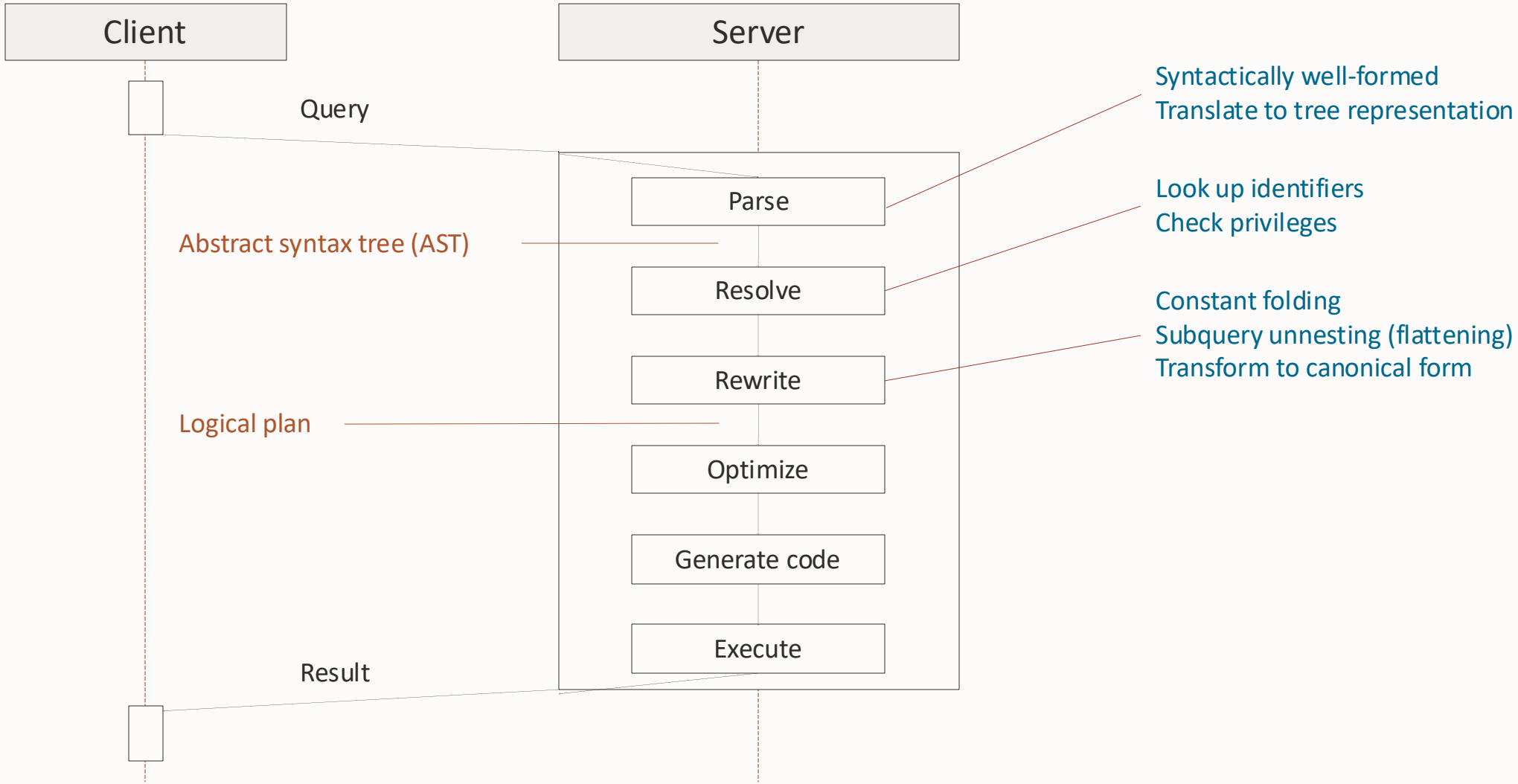
Query processing



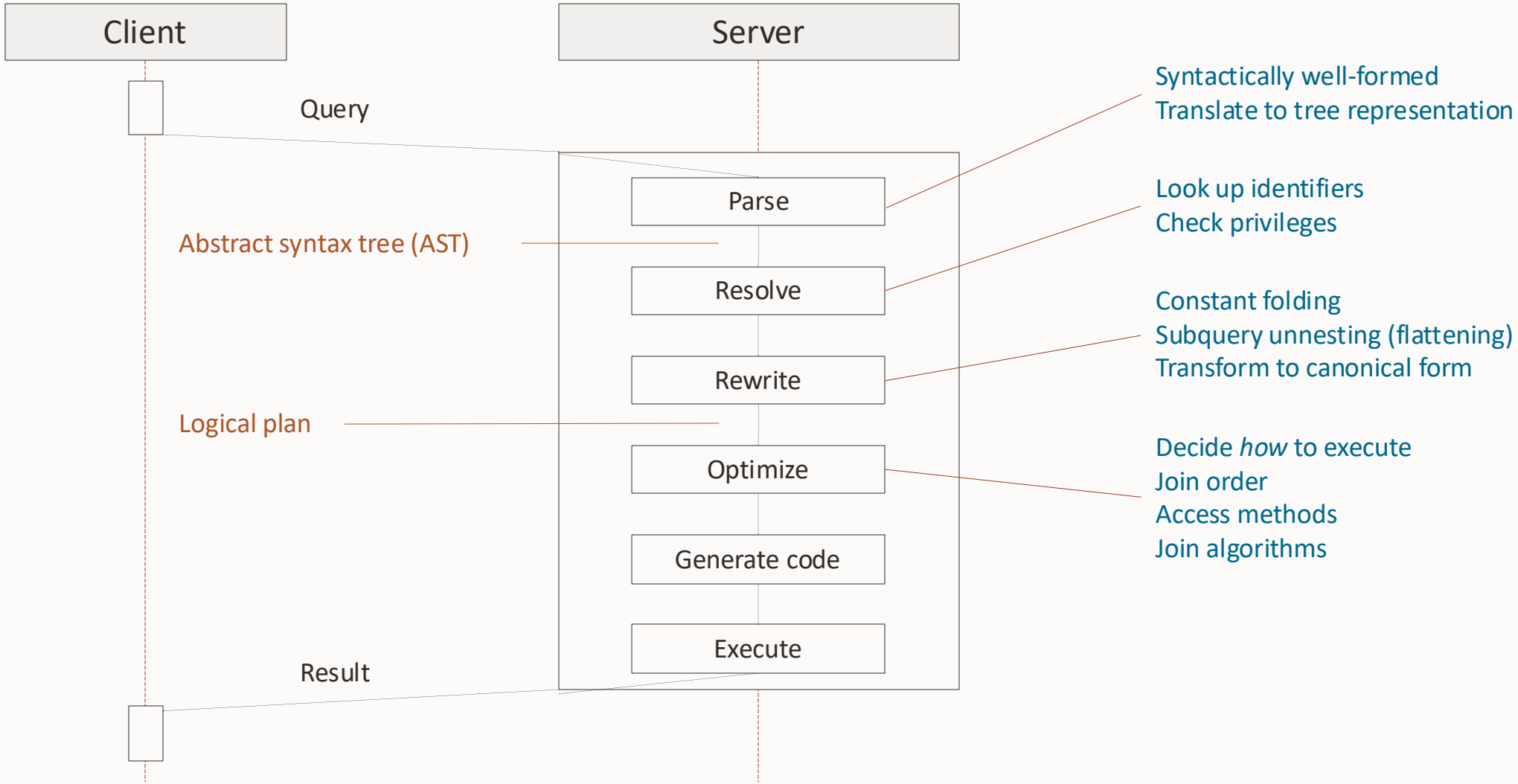
Query processing



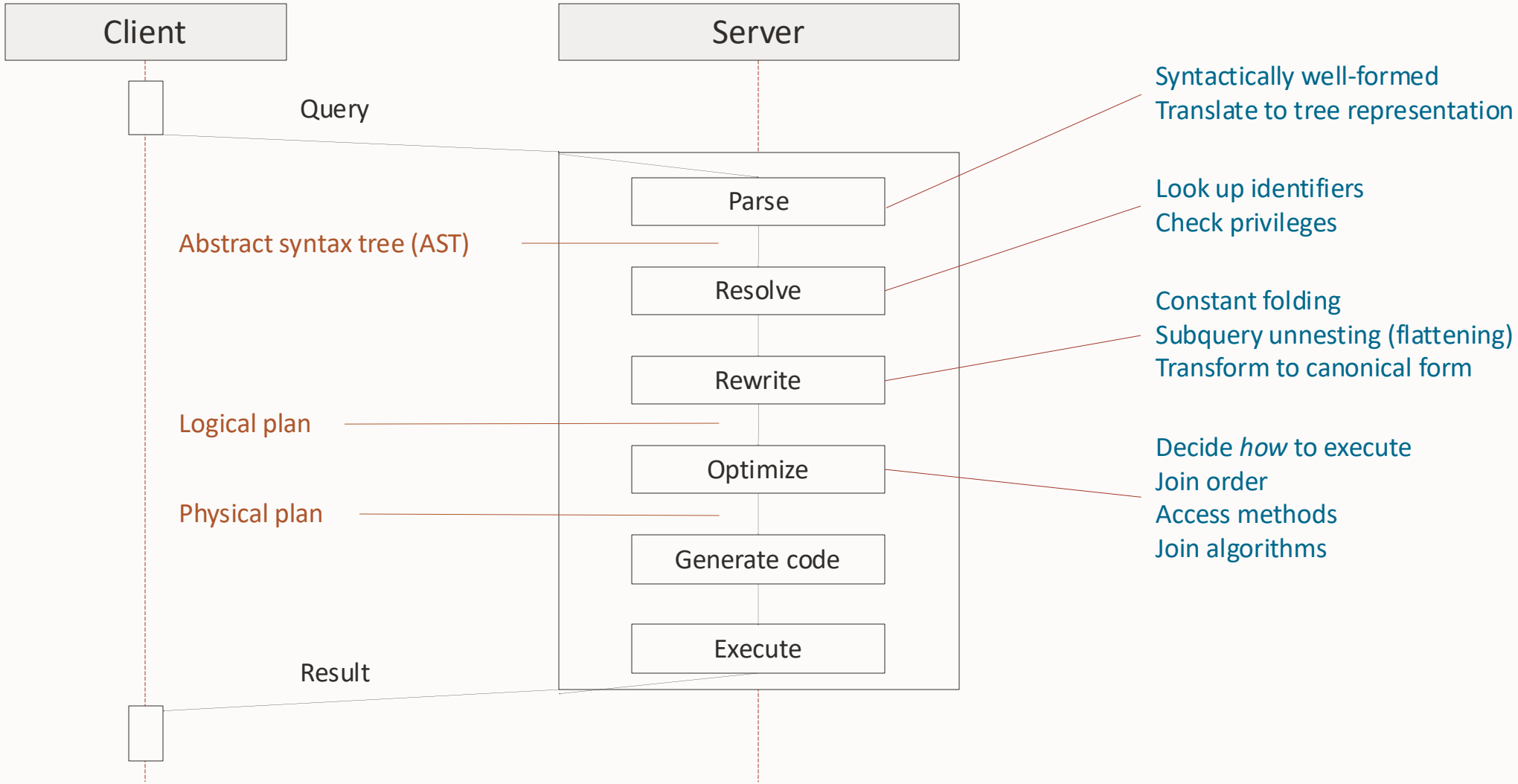
Query processing



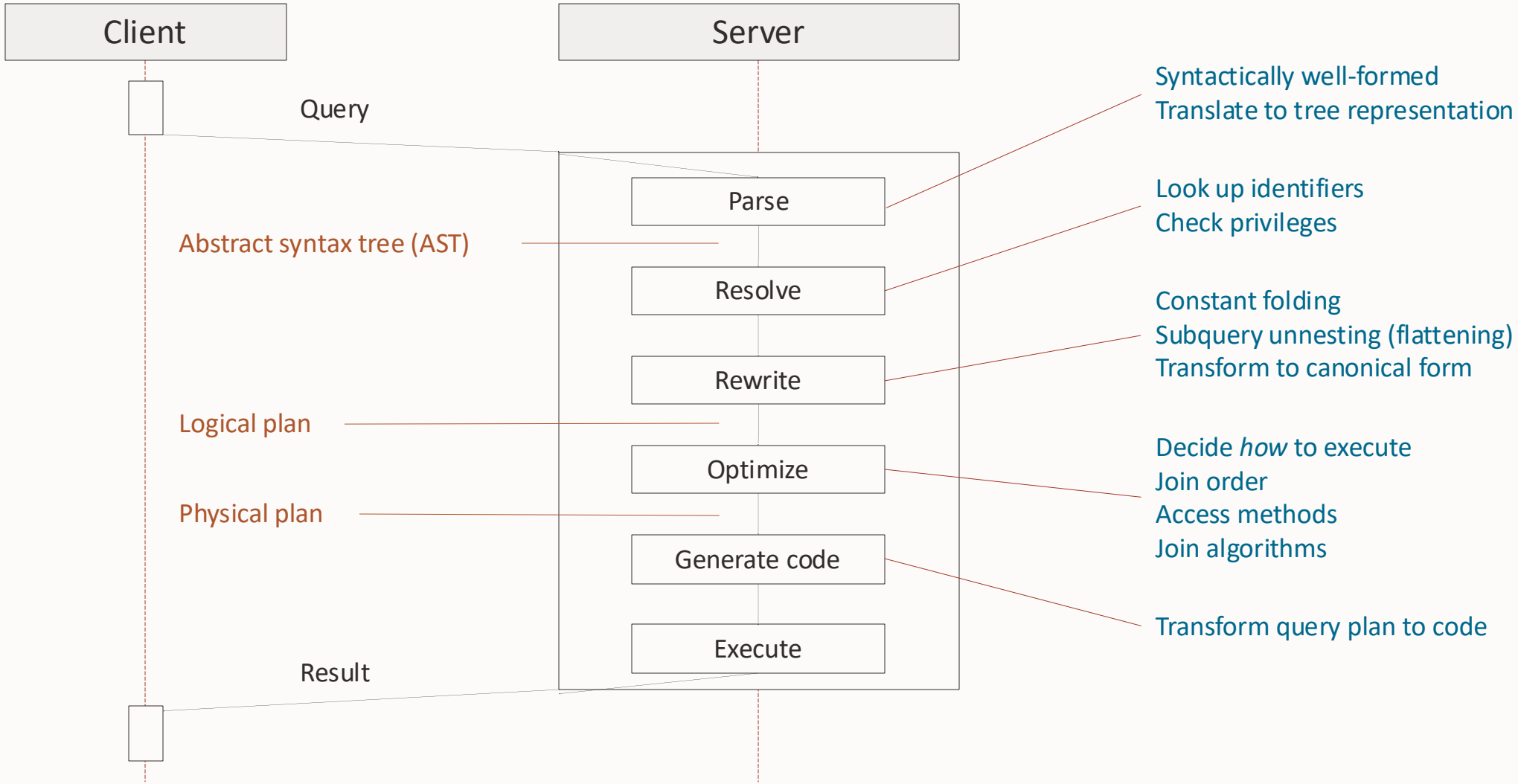
Query processing



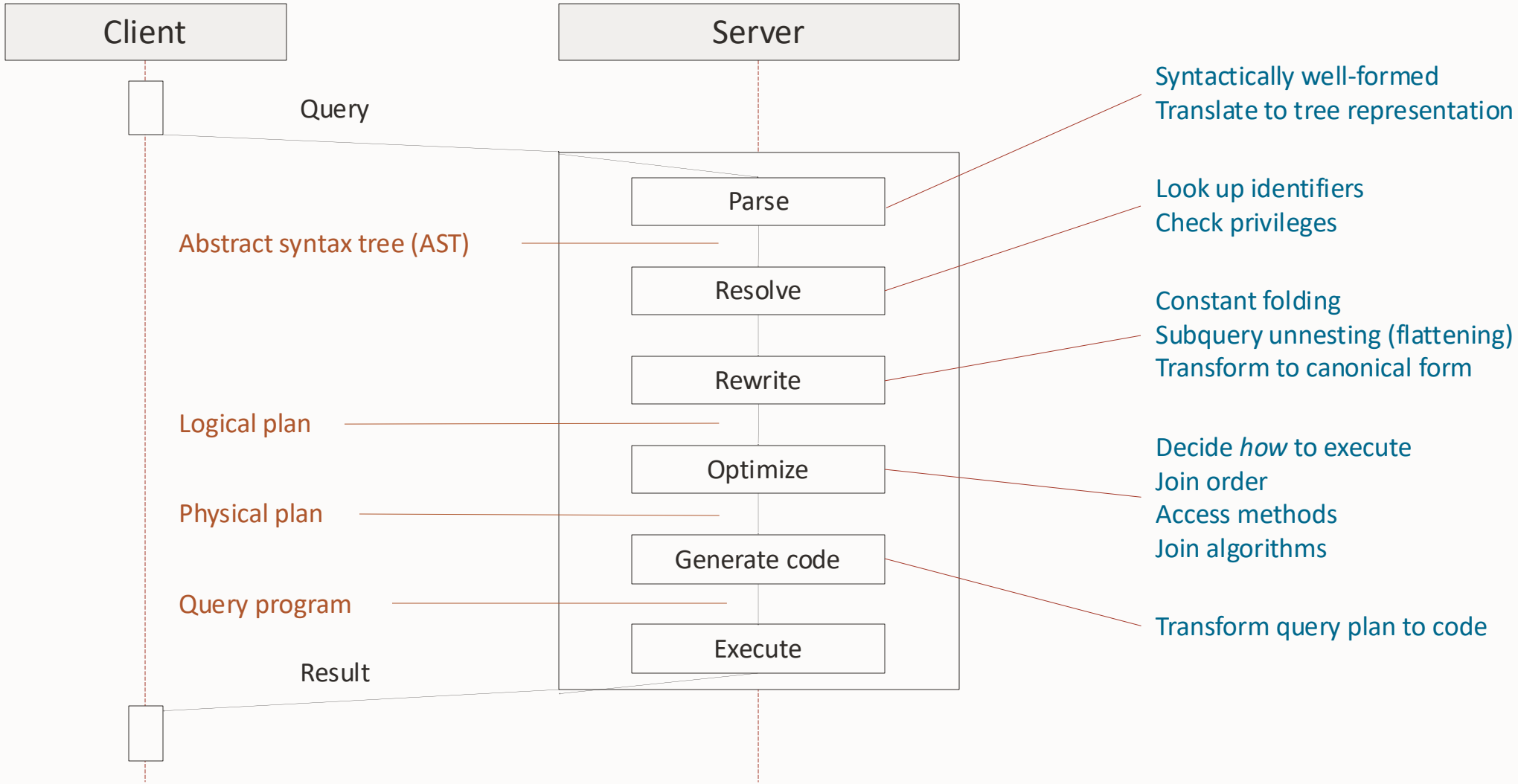
Query processing



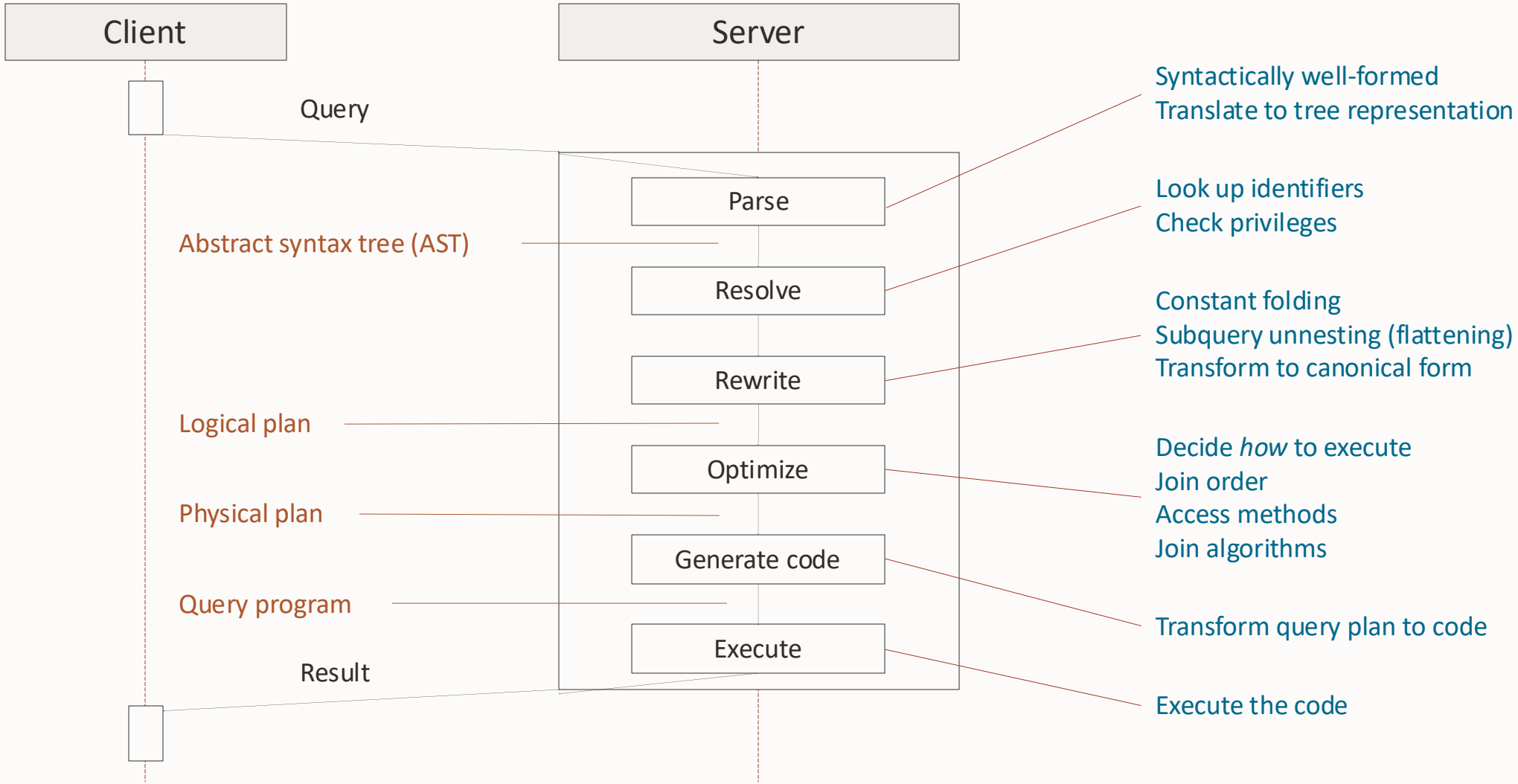
Query processing



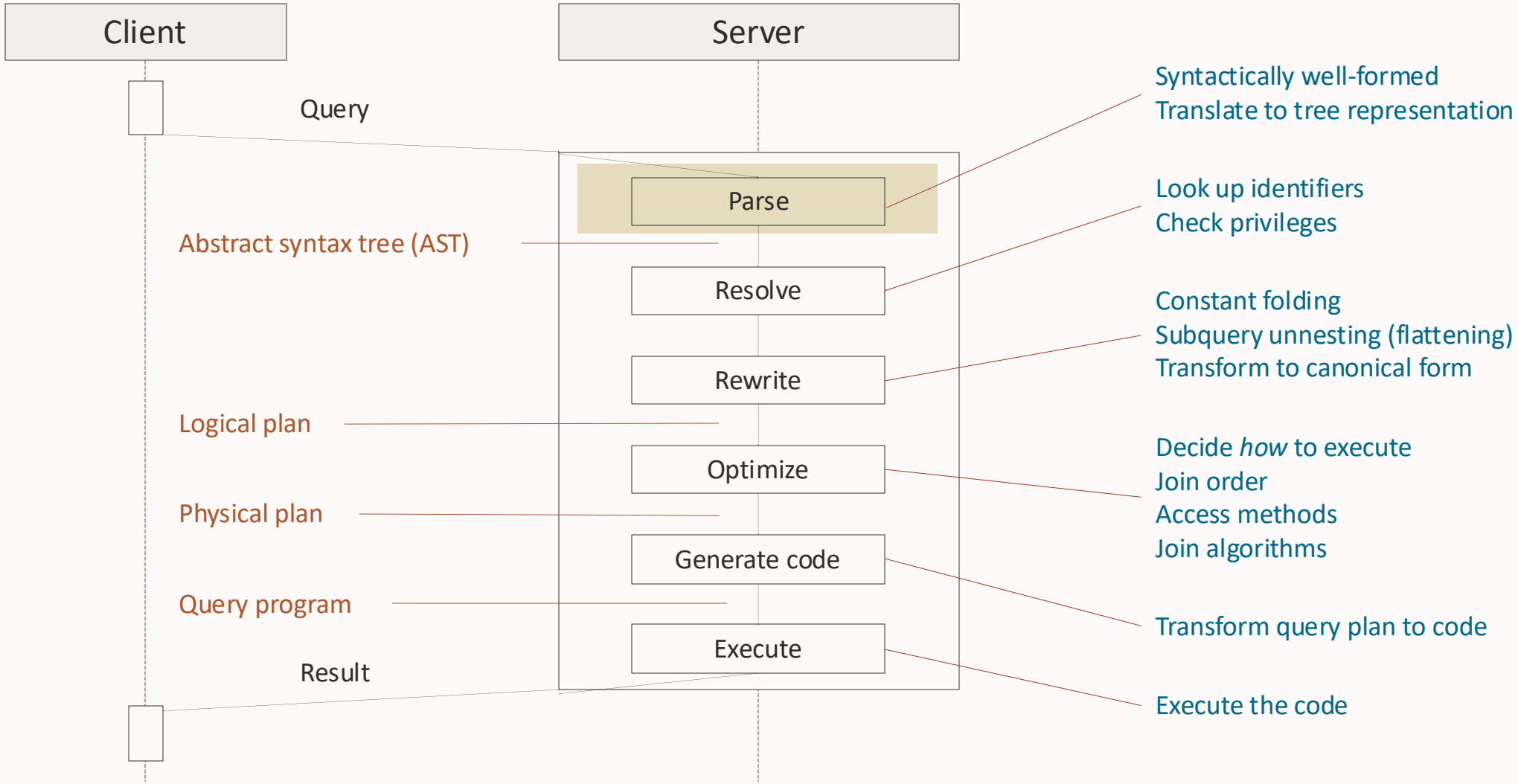
Query processing



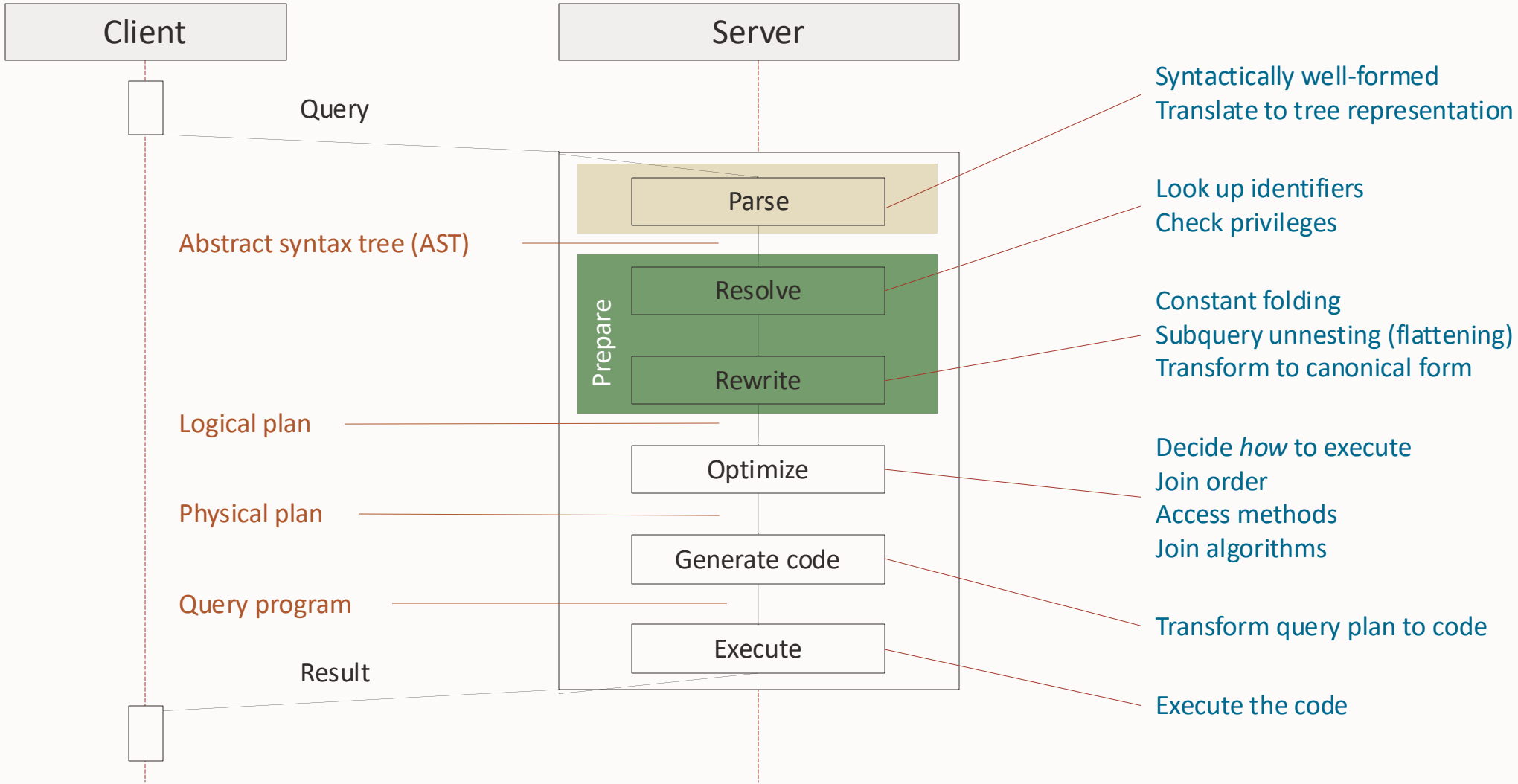
Query processing



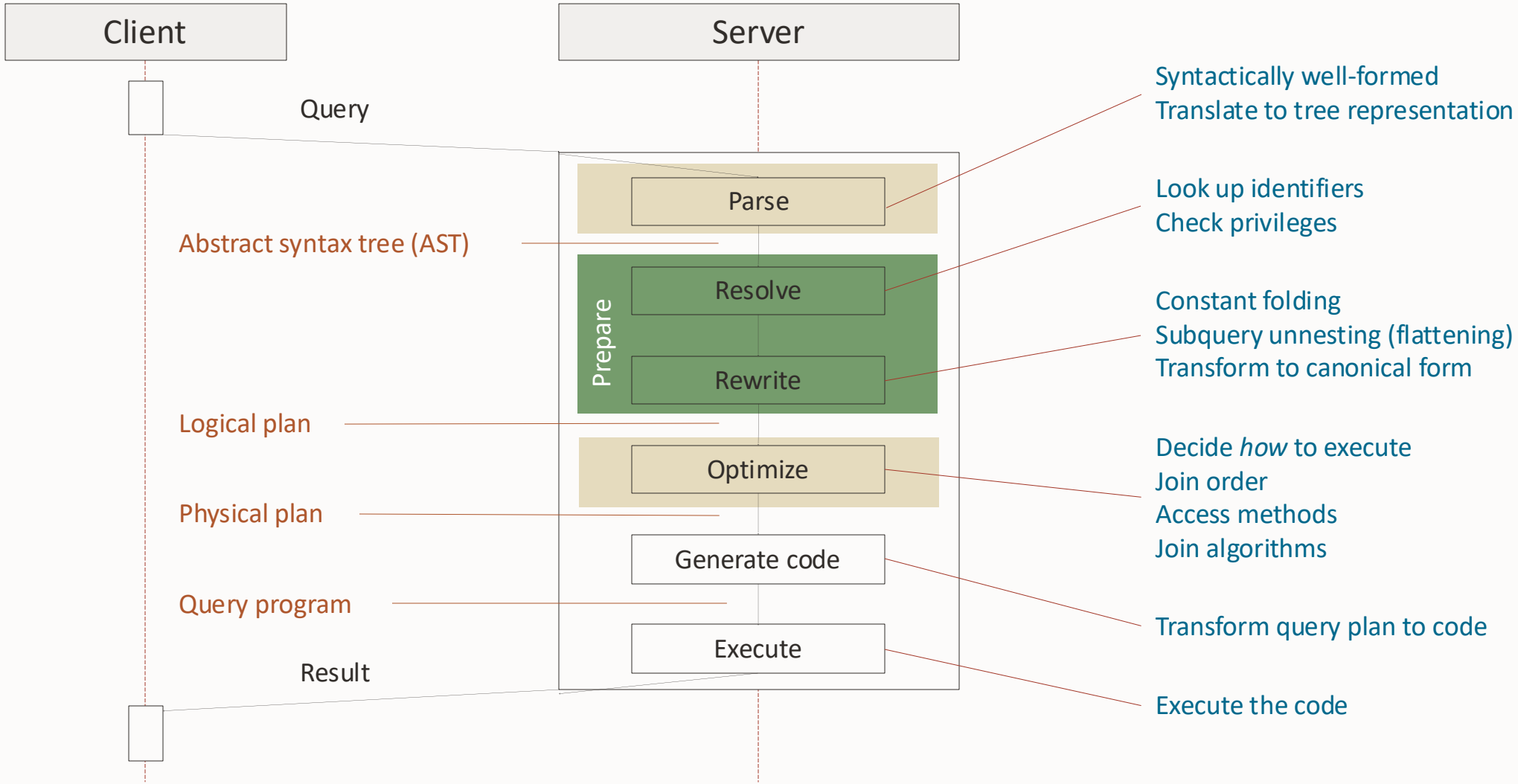
Query processing



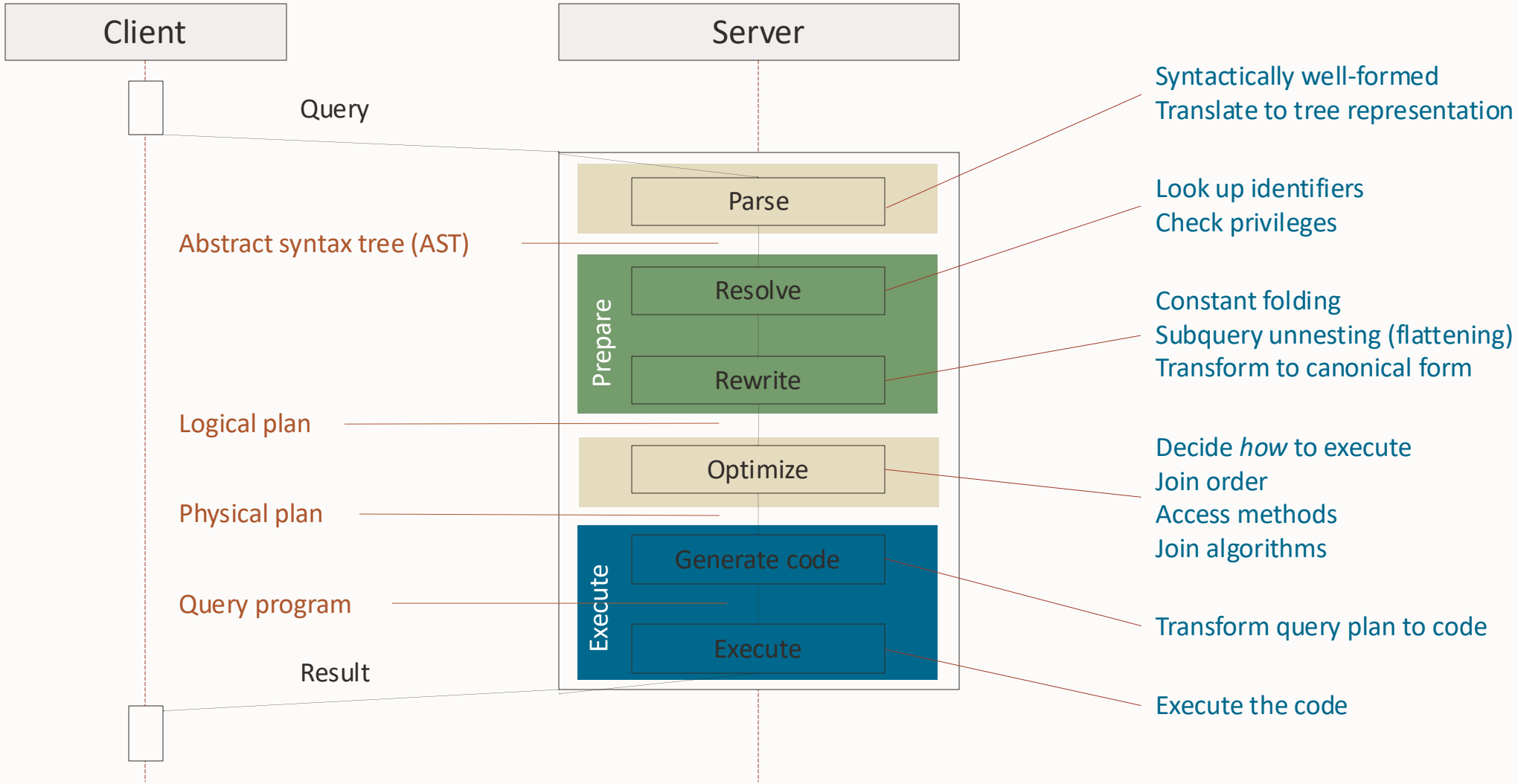
Query processing



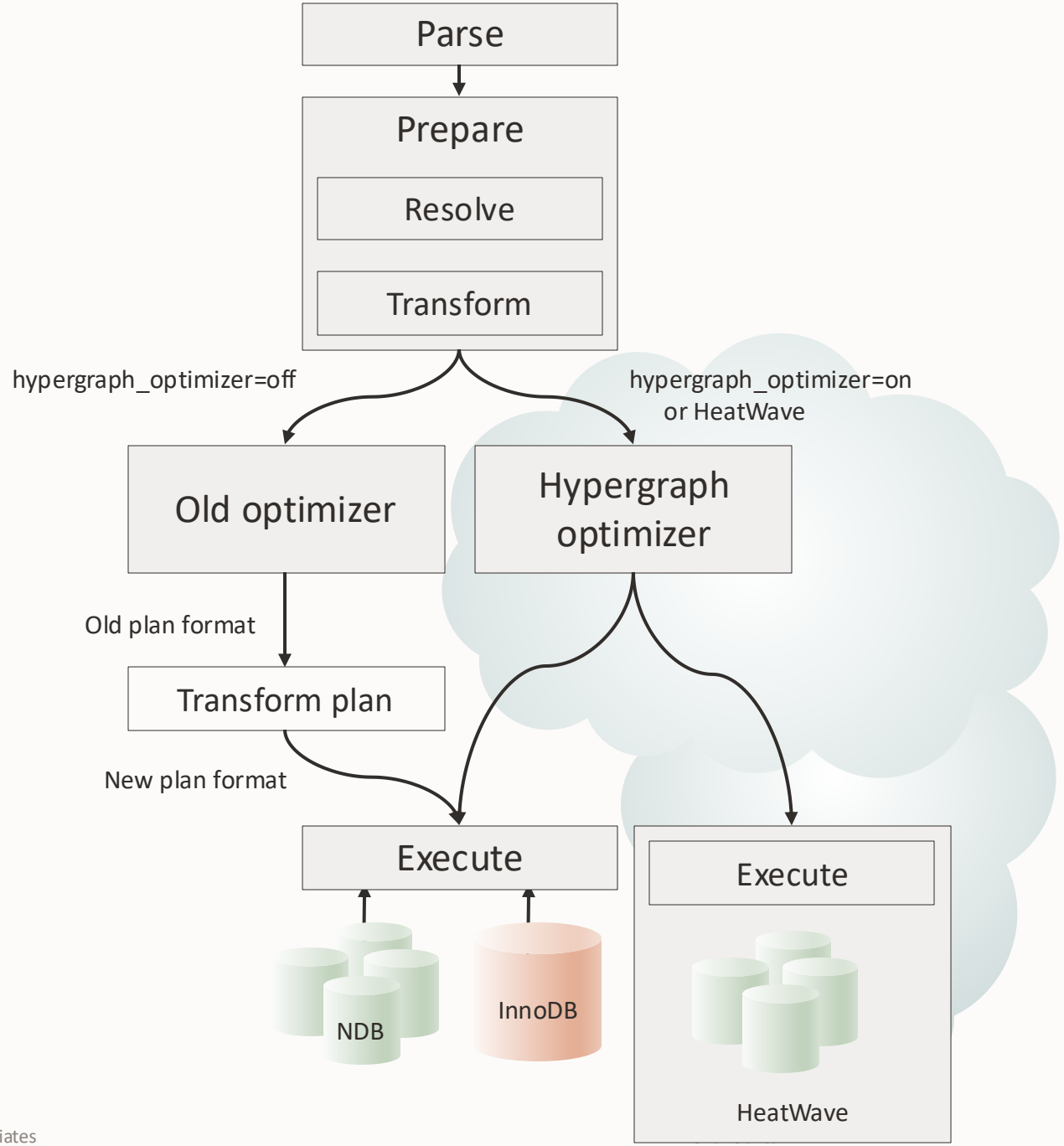
Query processing



Query processing



MySQL 9.0+



Why a new optimizer?

Shortcomings of the old optimizer

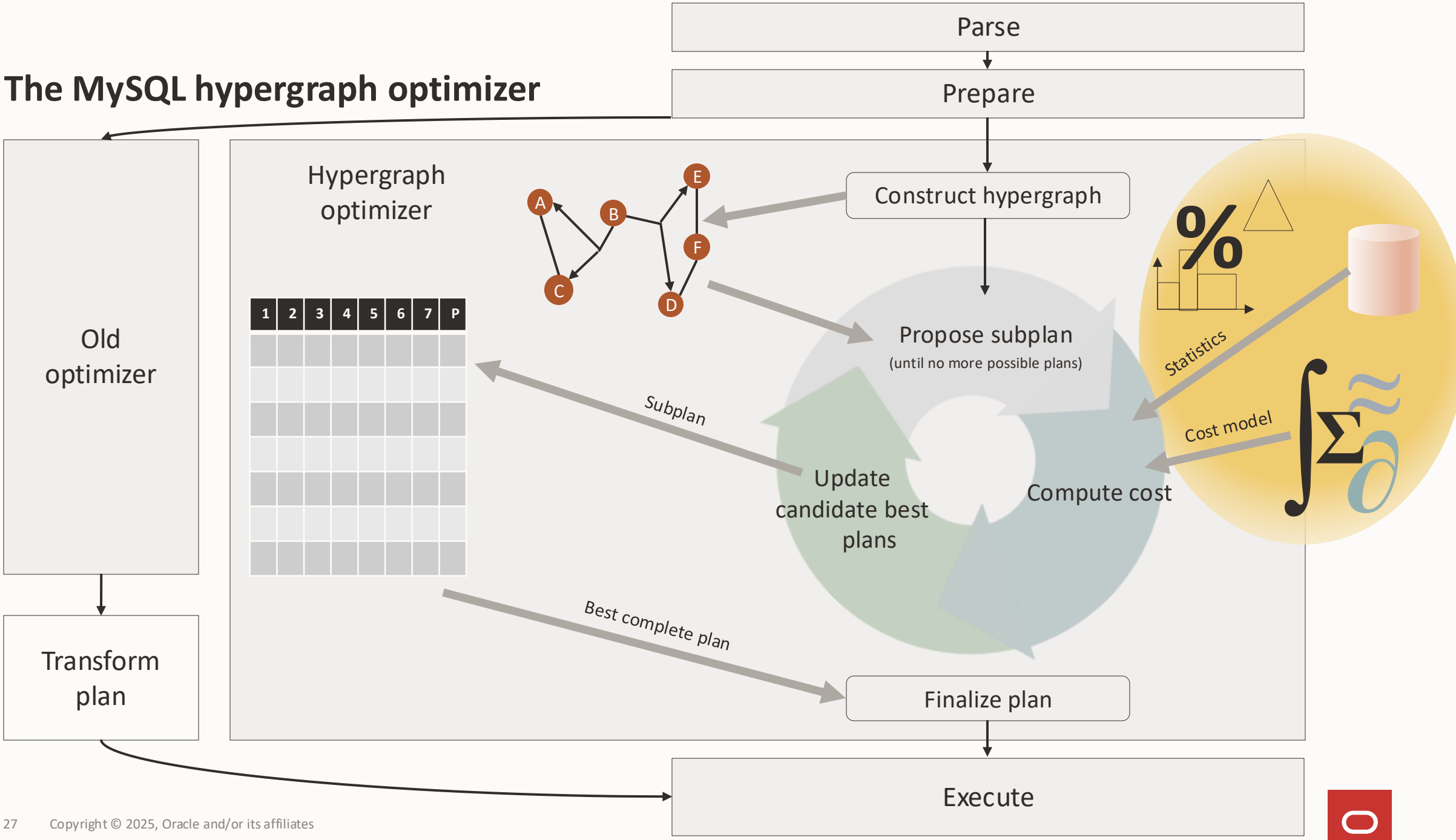
- Too much based on heuristic instead of cost
- Inefficient join ordering algorithm
  - Try all combinations up to a certain number of tables
    - Switches to greedy search when there are too many tables
  - No caching of costs — computes the same numbers several times
- Left deep plans only
- Doesn’t track interesting orders
  - Poor at making the choice between explicit sort and reading in sorted order from an index
- Bad at choosing between join algorithms
  - Inherently nested loop join based
- Presents HeatWave and NDB with only a complete plan
  - No way for HeatWave or NDB to guide the optimizer in plan selection

Design goals

- Modern, based on research
  - But not experimental!
- Cost based
- Interesting order tracking
- Cost based join algorithm selection
- Same optimizer for all storage engines
  - Local row store (InnoDB)
  - Distributed row store (MySQL Cluster/NDB)
  - Distributed column store (HeatWave)
- Bushy plans

What is a hypergraph, and what is it doing in my optimizer?

The MySQL hypergraph optimizer

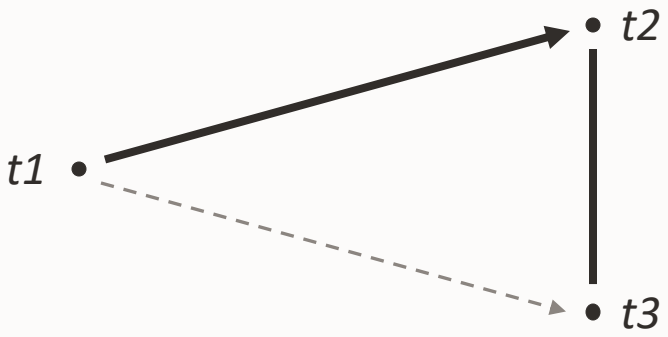


What is a hypergraph?

A hypergraph is a generalization of a graph where **hyperedges** connect two *sets* of vertices

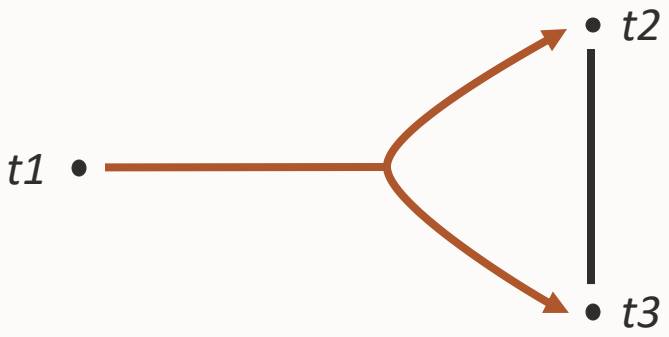
SELECT * FROM *t1* LEFT JOIN (*t2* JOIN *t3* ON *t2.b* = *t3.b*) ON *t1.a* = *t2.a*)

Join graph



There is no edge (*t1*, *t3*) in the graph since there is no join condition connecting *t1* and *t3*.

Join hypergraph



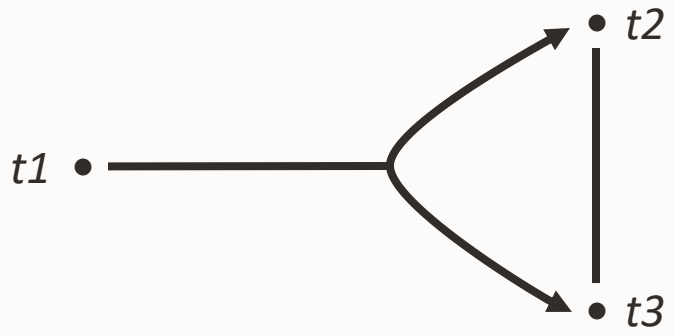
The hypergraph captures the (*t1*, *t3*) connection through the ({*t1*}, {*t2*, *t3*}) hyperedge.

Proposing subplans

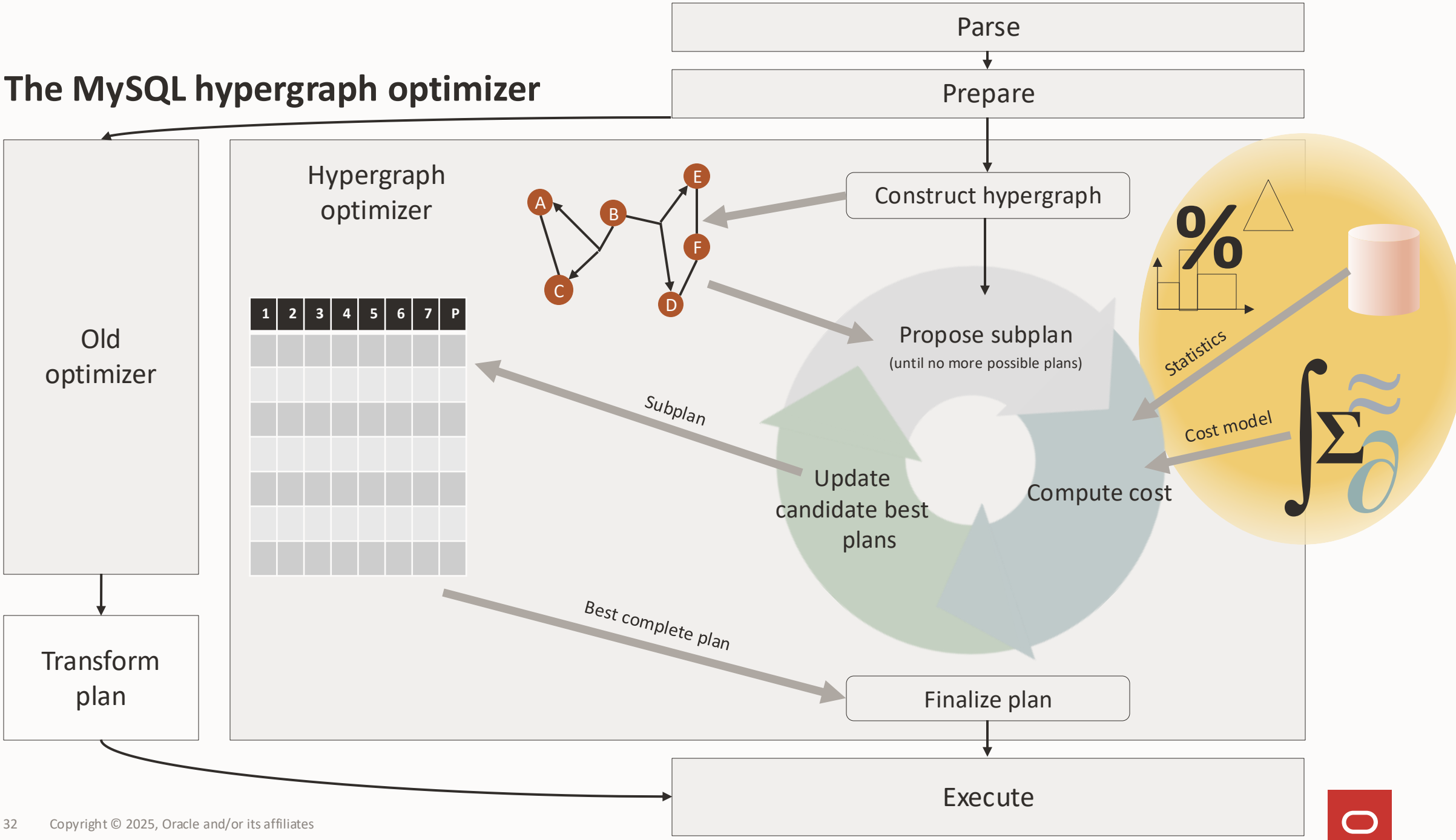
- Start by proposing all the ways we can access a single table
  - Table scan, index scan, index lookup, index range scan, etc.
- Propose all the ways to access another table
- Propose all the ways to join the two tables
  - All join orderings
- Build larger and larger plans
  - Always propose both sides of a join before proposing the join itself

Example:

1. *t2*
2. *t3*
3. *t2* ⋈ *t3*
4. *t1*
5. *t1* ⋈ (*t2* ⋈ *t3*)



The MySQL hypergraph optimizer



Multidimensional cost model

- A more expensive subplan may turn out to be a better choice in the end
  - E.g., it produces a sorted order, and an ordered result is beneficial in the final plan
- Several cost dimensions
  1. Total cost of producing this result once
  2. Initialization cost (e.g., materialize a subquery)
  3. Rescan cost (e.g., read a materialized table the second time)
  4. Dependency on other tables (subplan is a candidate only when those tables are present)
  5. Ordering
  6. Does provide row IDs (sorting may be faster if it does)
  7. Can update/delete while scanning the table (allows UPDATE/DELETE without materialization)
- A plan candidate is kept if it is not dominated by any other plan
  - I.e, there is at least one dimension where this plan is better than the current set of best plans

When will the optimizer finish processing?

- Theoretical ideal: Finish when the time spent to find a better plan is more than the potential savings
  - I.e., minimize the total query processing time (optimization + execution)
  - Impossible to know up front
- Bottom-up planner
  - No complete plan until (almost) all orderings have been tried
- MySQL hypergraph optimizer: Finish when there are no more plans to propose
- Planning time depends on query properties
  - Number of tables
  - Join conditions
  - Number of relevant indexes
  - ...

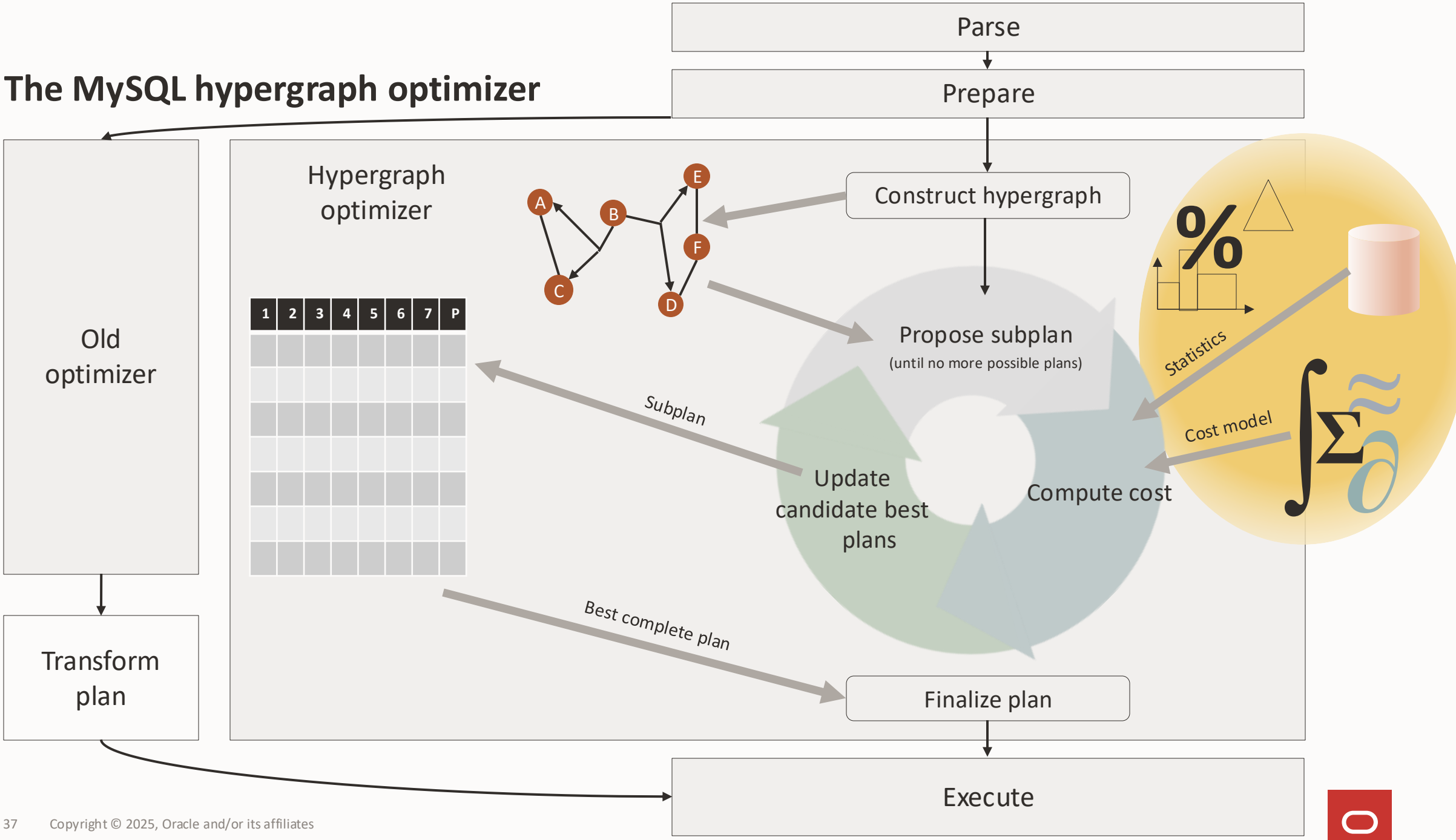
Back to heuristics

- Resort to heuristics if computing all possible join orderings takes too long
  - Currently limited by number of possible join orderings (not time)
- Use heuristics to add new hyperedges to the join hypergraph
  - Encodes a locked join ordering
    - The full set of tables at each end must be computed before computing the hyperedge

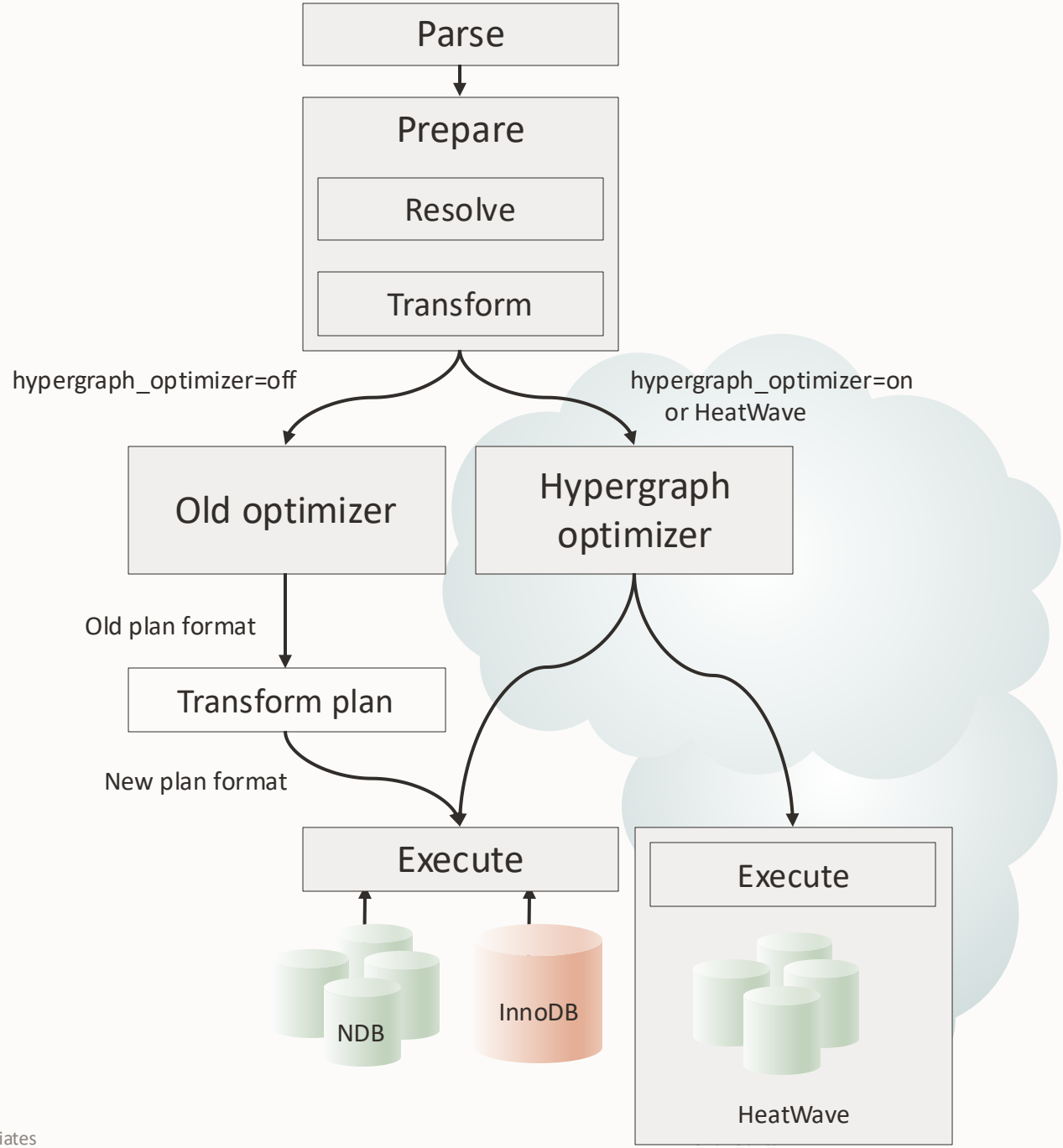
Current algorithm:

1. Set n=1
2. Add n artificial hyperedge(s)
3. Compute number of possible join orderings (skip cost evaluation to make it fast)
4. If there are still too many possible join orderings, set n=2n and go to (2)
5. Bisect between n/2 and n to find minimum number of hyperedges to add without generating too many plans
6. Re-run the optimizer and find best plan

The MySQL hypergraph optimizer



How can I start using the hypergraph optimizer?



Start using the hypergraph optimizer

- Conly available in HeatWave MySQL cloud services (OCI or AWS)
  - Available since 9.0.0
- Switch optimizers by setting an optimizer switch:

SET optimizer_switch='hypergraph_optimizer=on';

Thank you


ORACLE

Our mission is to help people see data in new ways, discover insights, unlock endless possibilities.