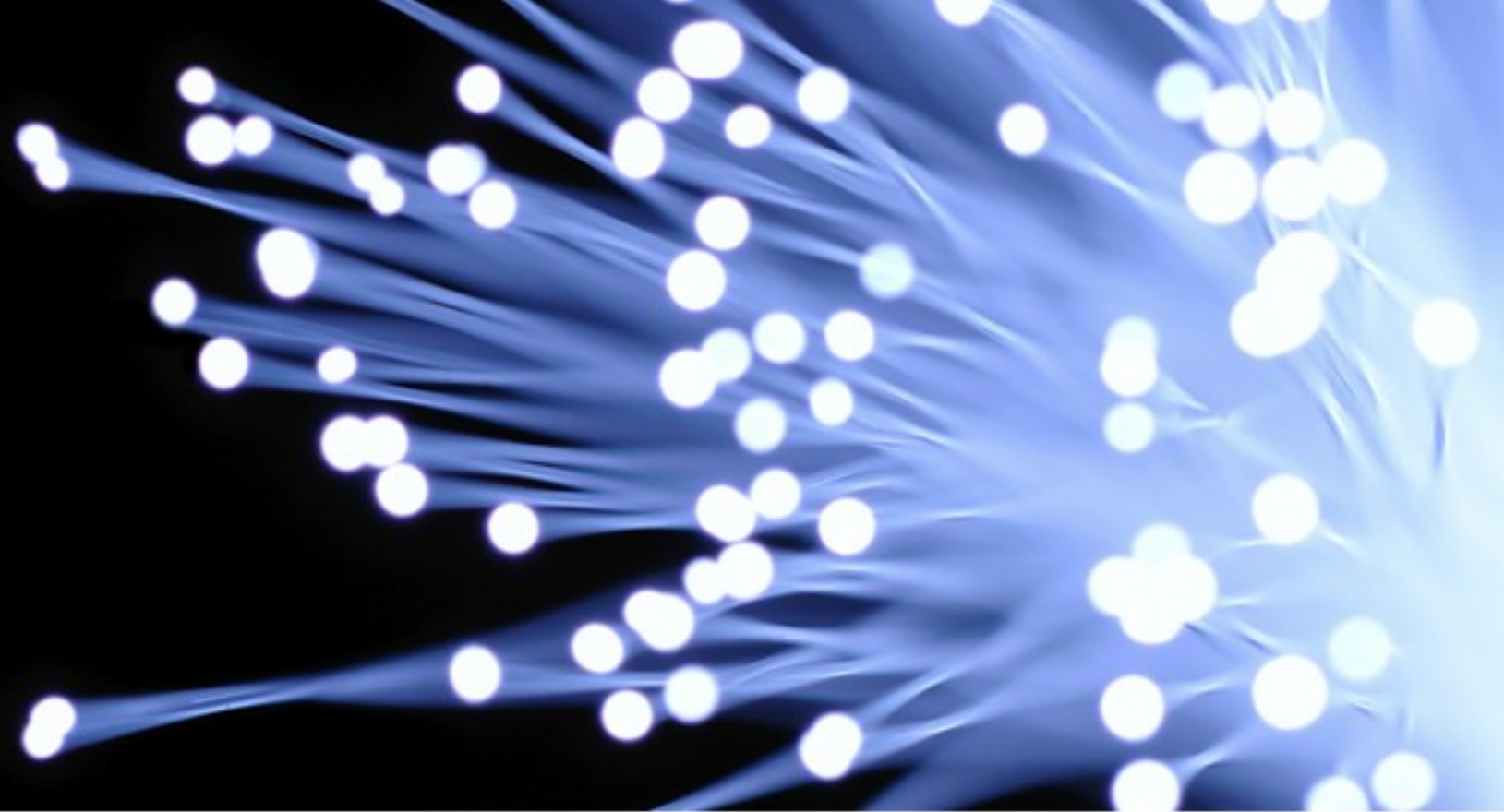


O'REILLY

MySQL

Conference & Expo

INFORMATION UNLEASHED



The MariaDB/MySQL Query Executor In-depth

Presented by: Timour Katchaounov

Optimizer team: Igor Babaev, Sergey Petrunia, Timour
Katchaounov



Monty Program



What's **IN**

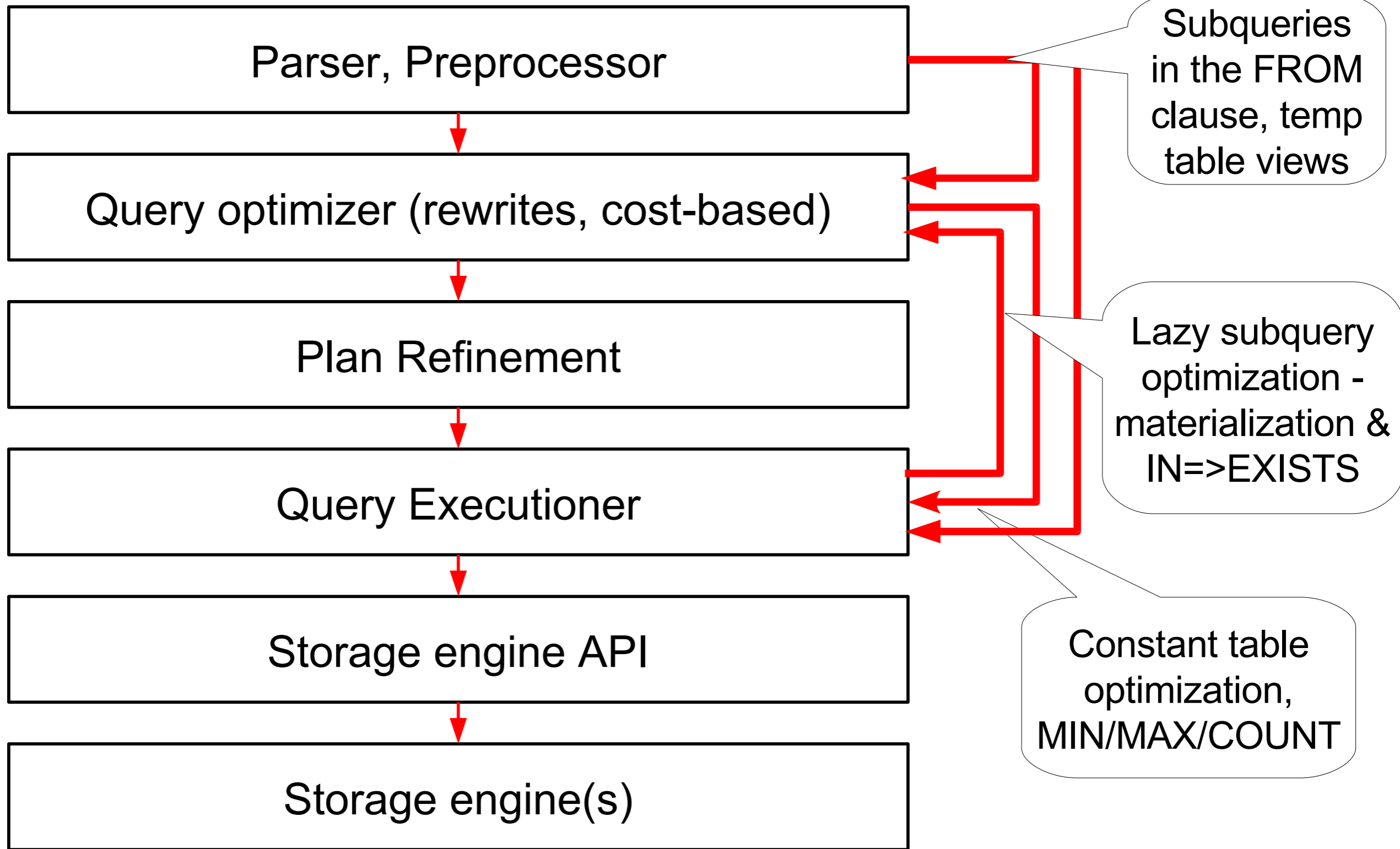
- Query engine architecture
- Execution model
- Representation of query execution plans (QEPs)
- Single-table access methods
- Join methods
- Questions

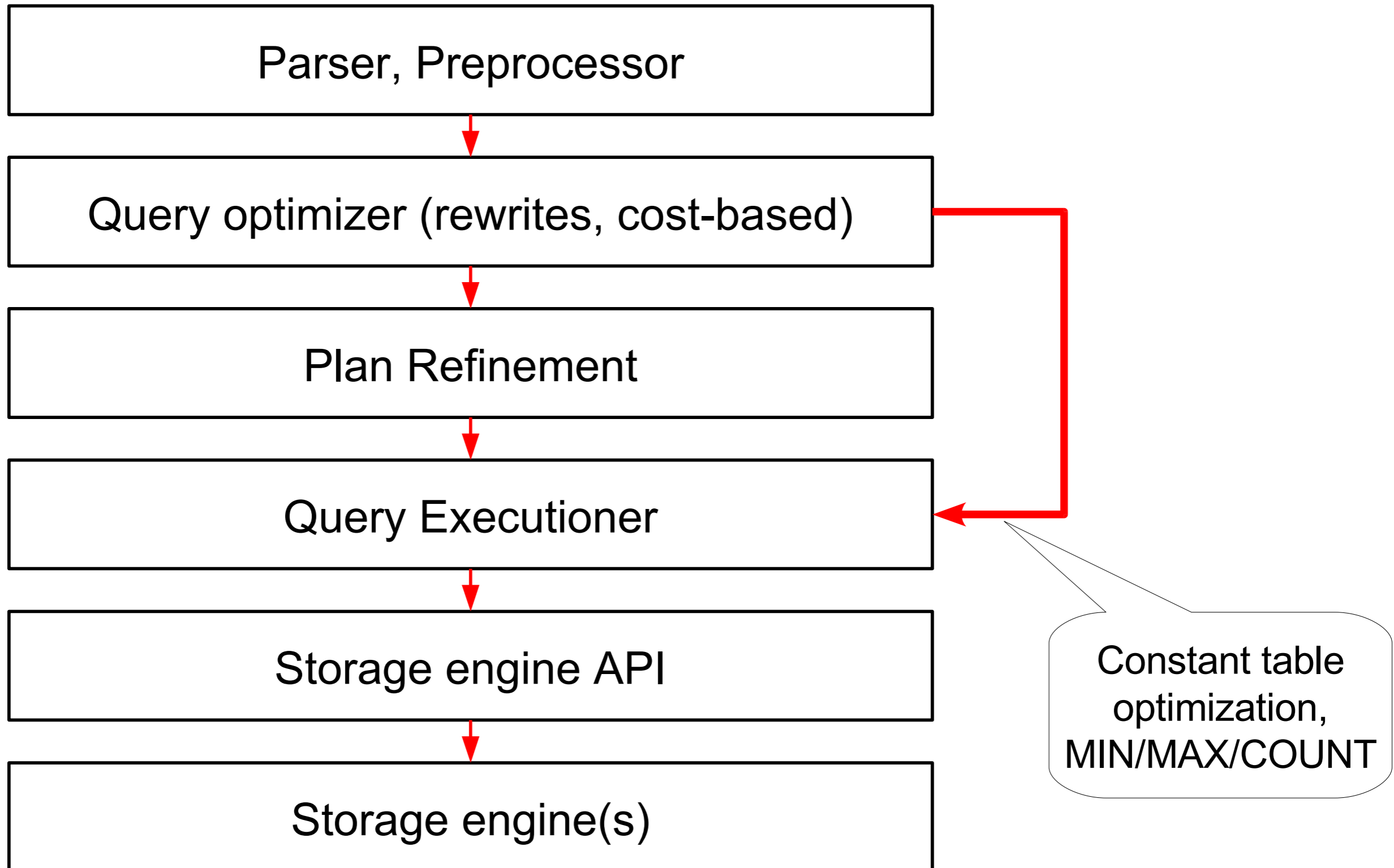
What's **NOT IN**

- Query optimization
- Subquery execution
- Sorting/Grouping/Distinct
- Prepared statements
- Stored procedures
- **INSERT/UPDATE/DELETE**

Comparing latest development versions: MariaDB 5.3 vs. MySQL 5.5

Query engine architecture [MySQL 5.5]

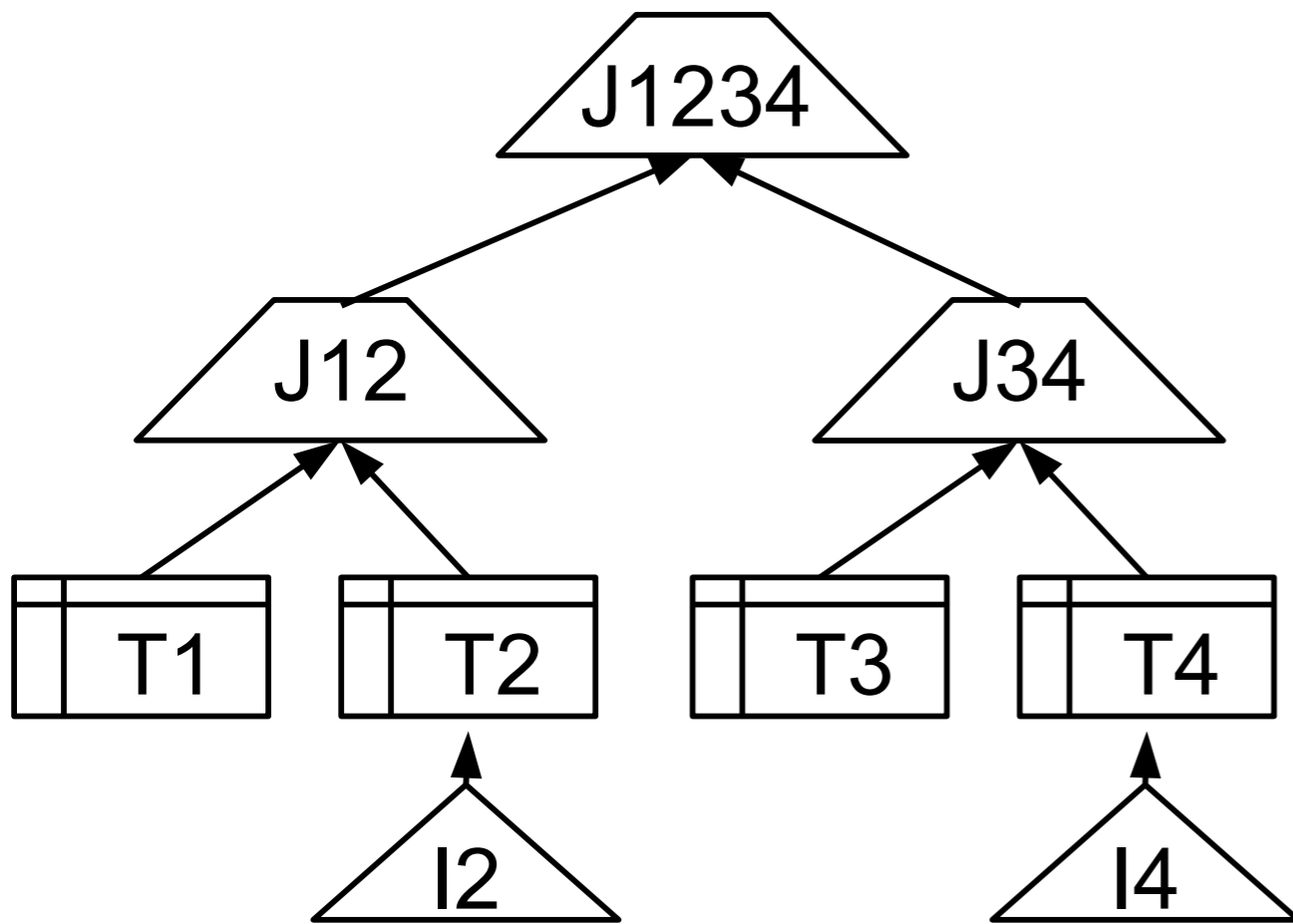




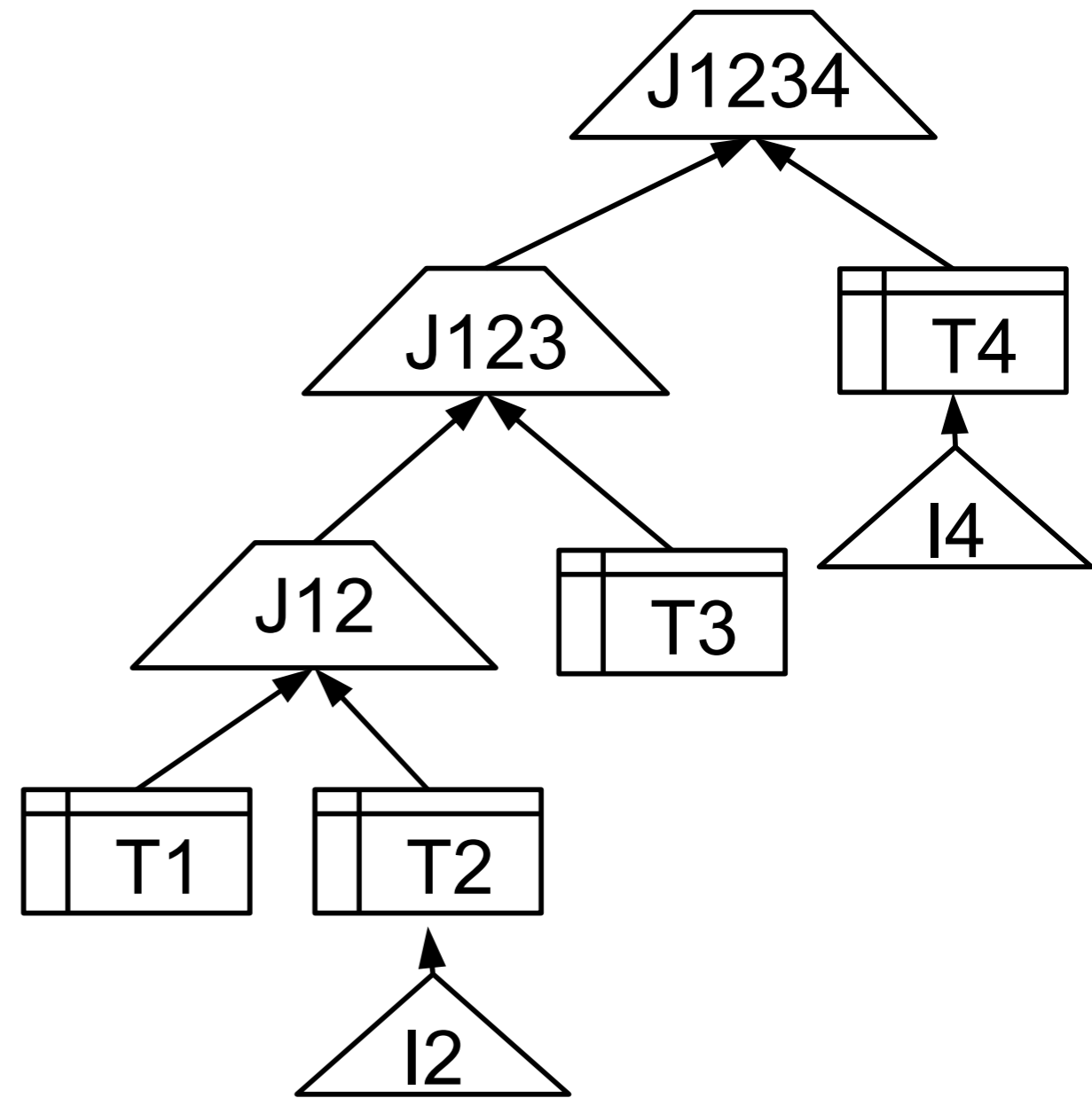
Query plan 'shape' – bushy vs linear



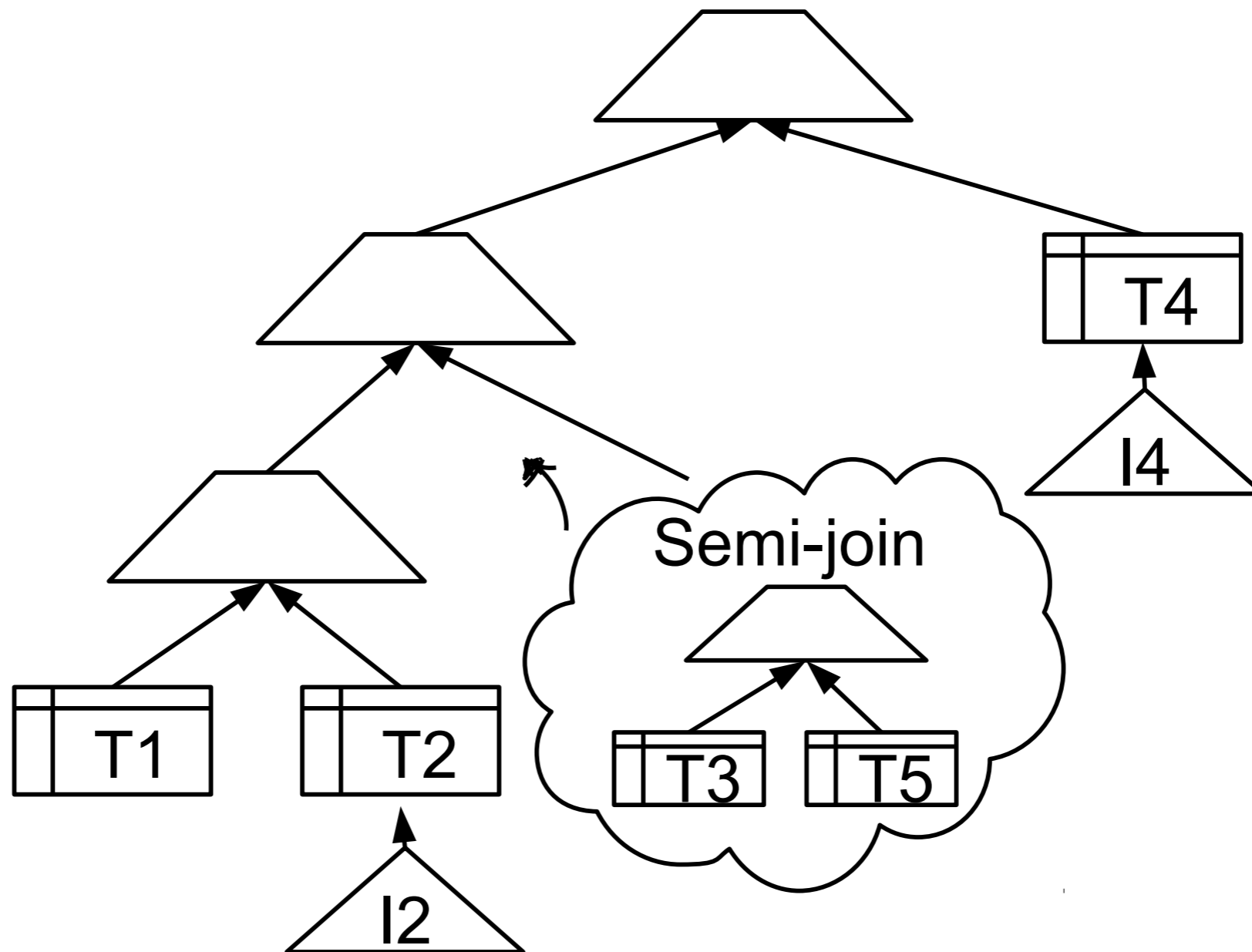
General bushy query plans

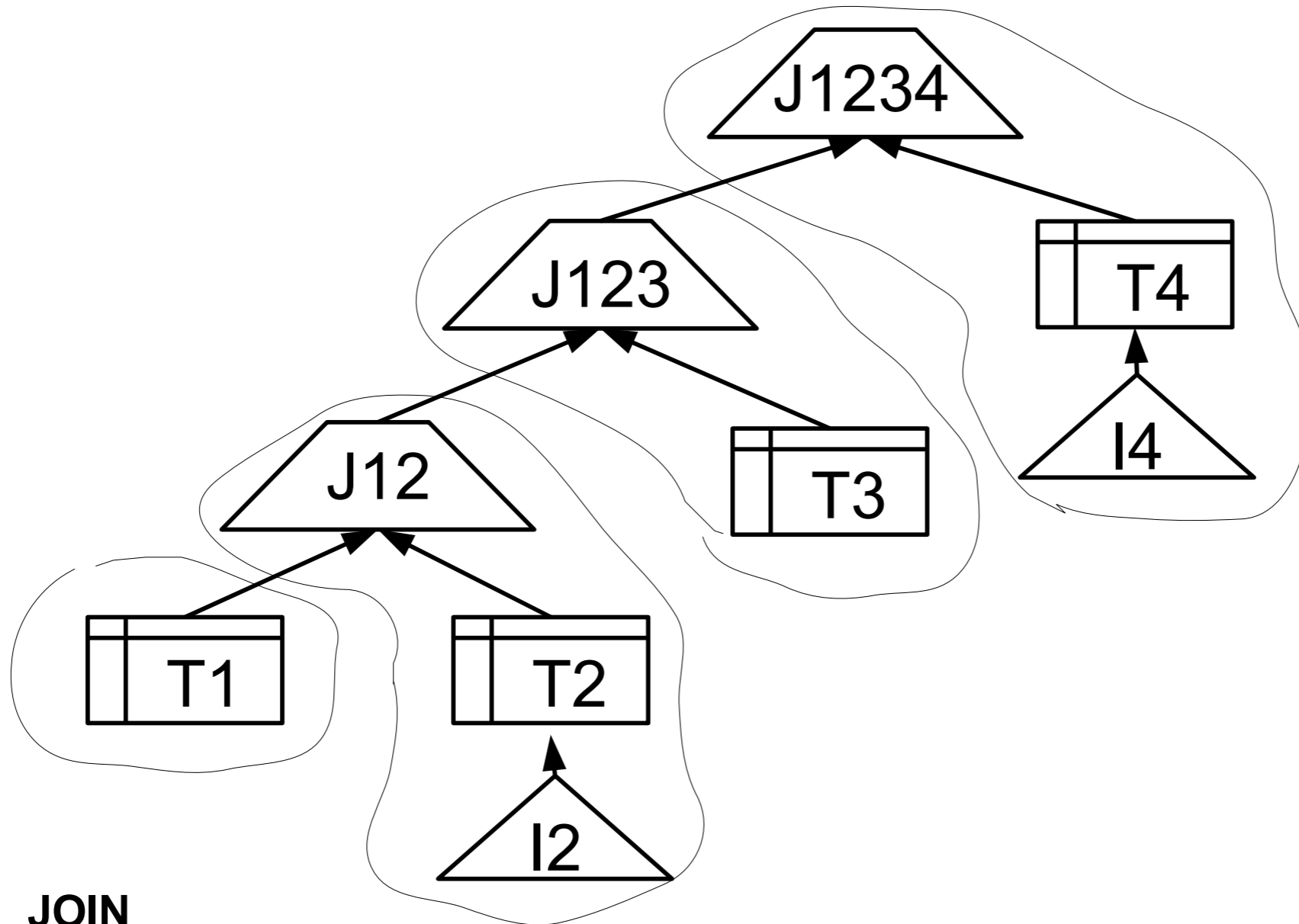


Left-deep query plans

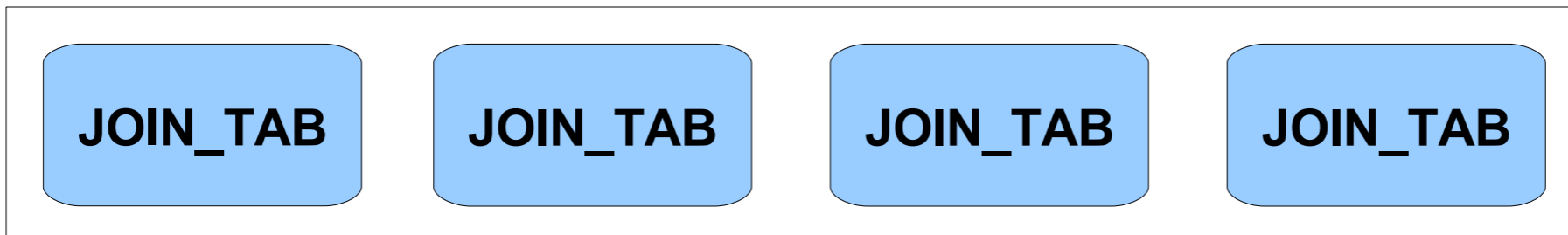


MariaDB 5.3: bushy query plans with semi-join(IN subqueries), and derived tables





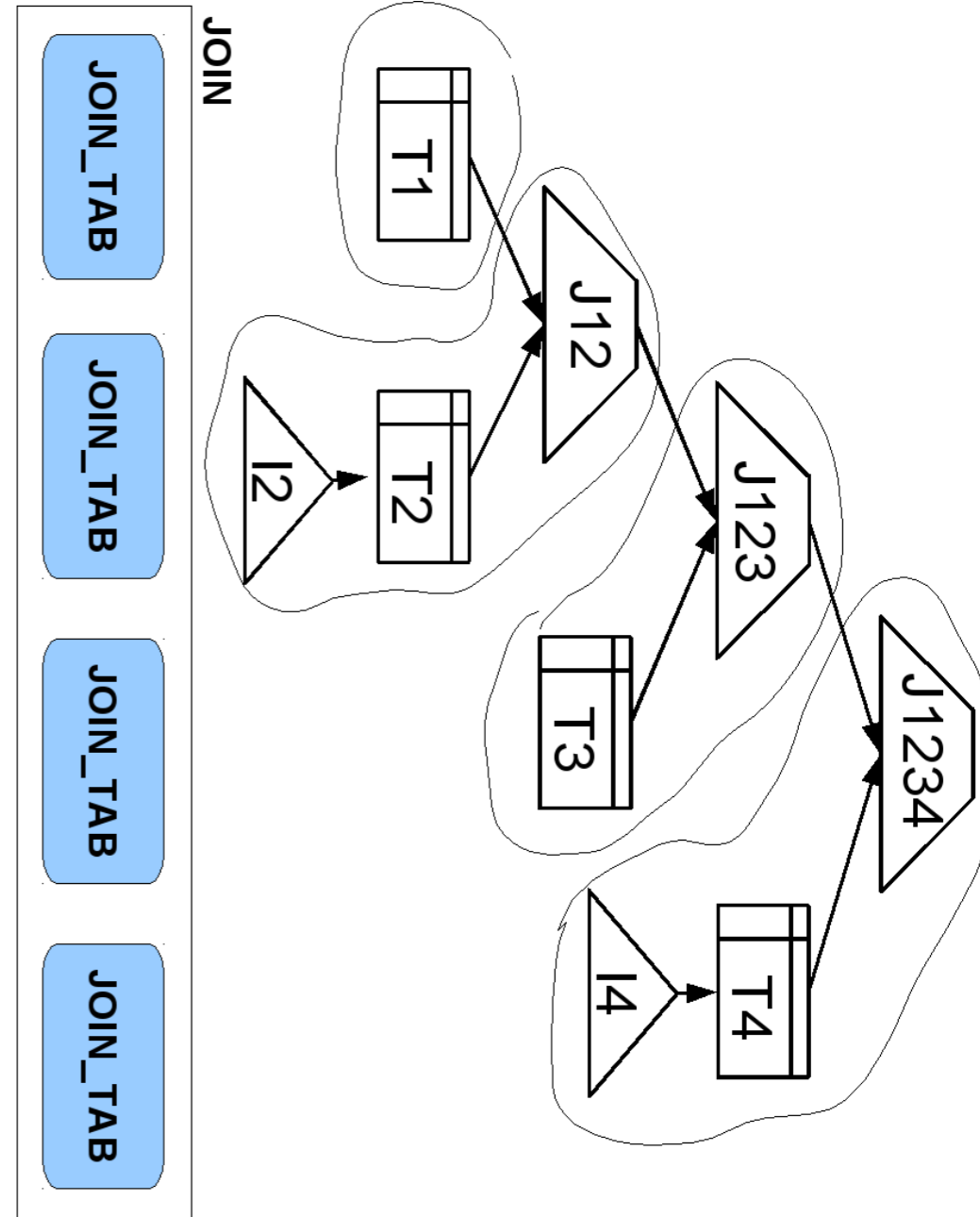
JOIN



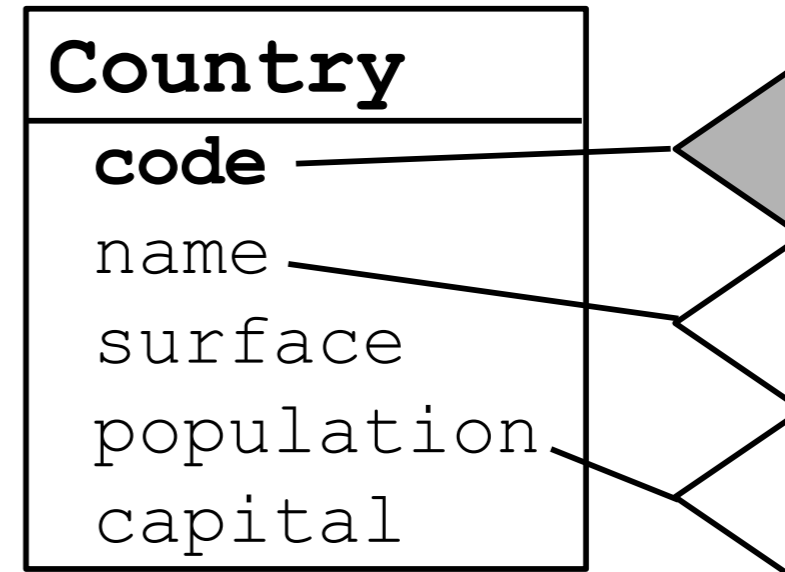
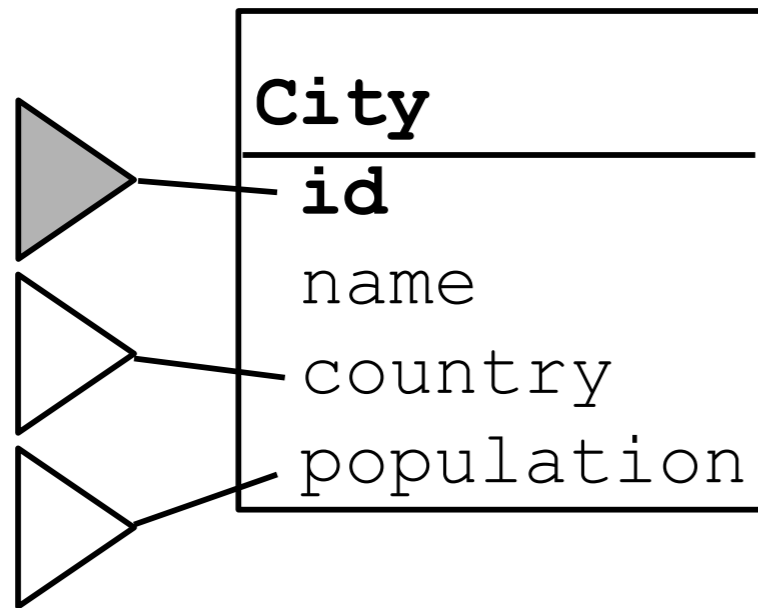
Query plans and EXPLAIN



table	type	possible_keys	key	key_len	ref	rows
T1	ALL	PRIMARY	NULL	NULL	NULL	984
T2	range	K1, K2	K2	3	NULL	30
T3	eq_ref	PRIMARY	PRIMARY	4	C3	1
T4	eq_ref	PRIMARY	PRIMARY	4	C4	1



Example database and query

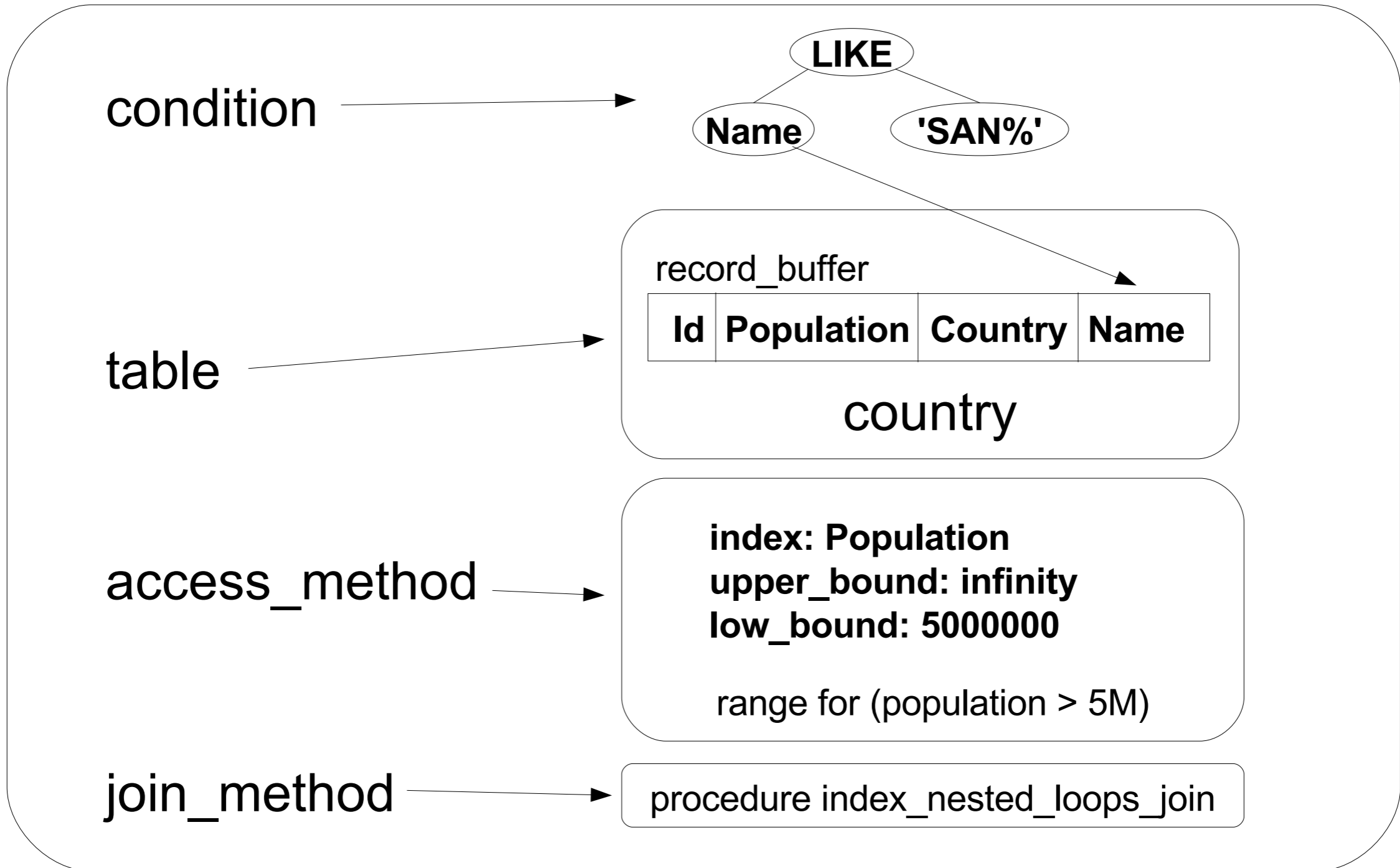


```
SELECT Country.name, City.name, City.population
FROM Country, City
WHERE City.country = Country.code and
      City.id = Country.capital and
      City.population > 5000000 and
      City.name LIKE 'SAN%';
```

Explain for the example



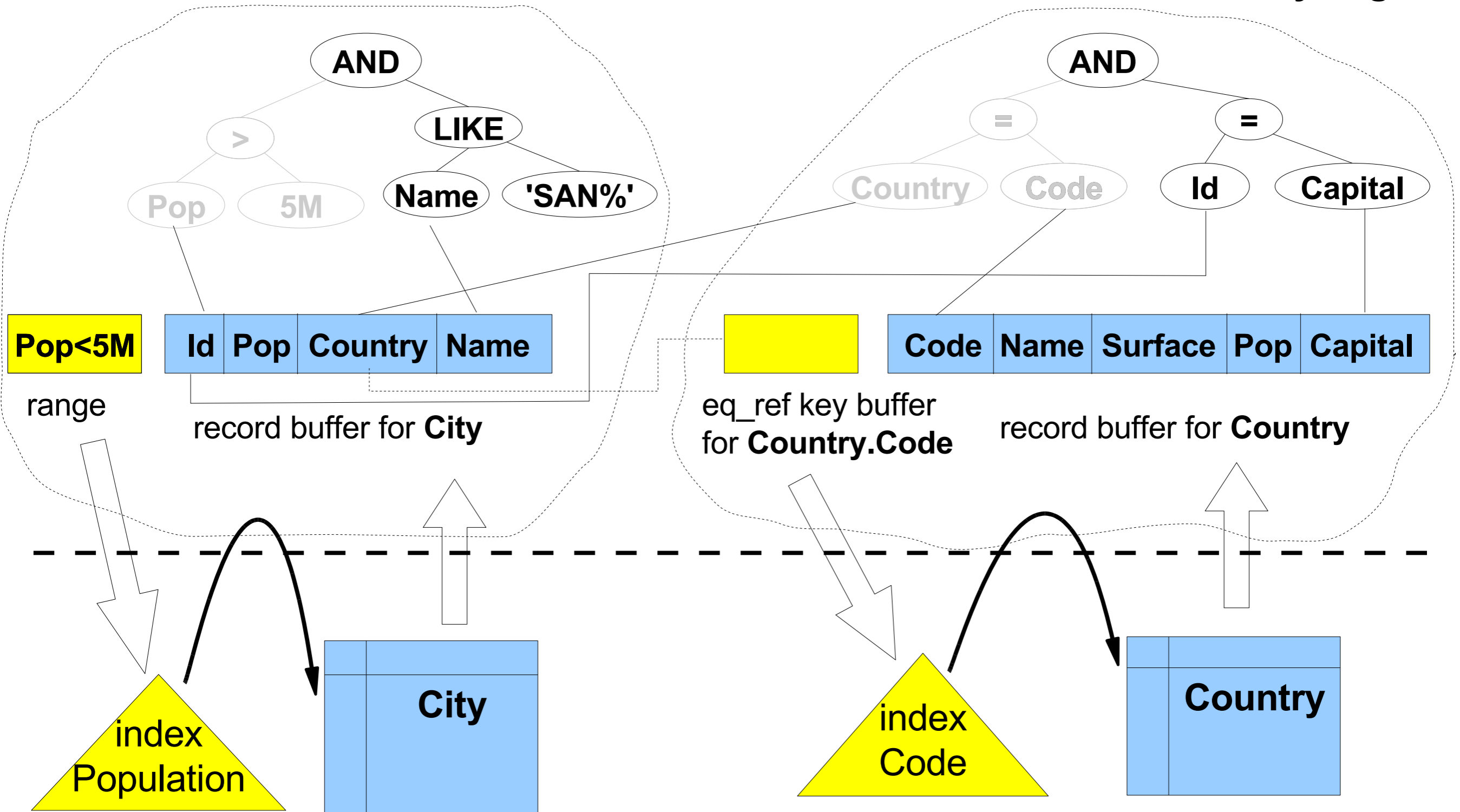
```
***** 1. row *****
      id: 1
    select_type: SIMPLE
      table: City
      type: range
possible_keys: PRIMARY, country, population
      key: population
    key_len: 4
      ref: NULL
     rows: 25
   Extra: Using index condition; Using where; Using MRR
***** 2. row *****
      id: 1
    select_type: SIMPLE
      table: Country
      type: eq_ref
possible_keys: PRIMARY
      key: PRIMARY
    key_len: 3
      ref: world.City.CountryCode
     rows: 1
   Extra: Using where
```



file: sql/sql_select.h, class JOIN_TAB

QEPs and nested loop join execution

Query engine

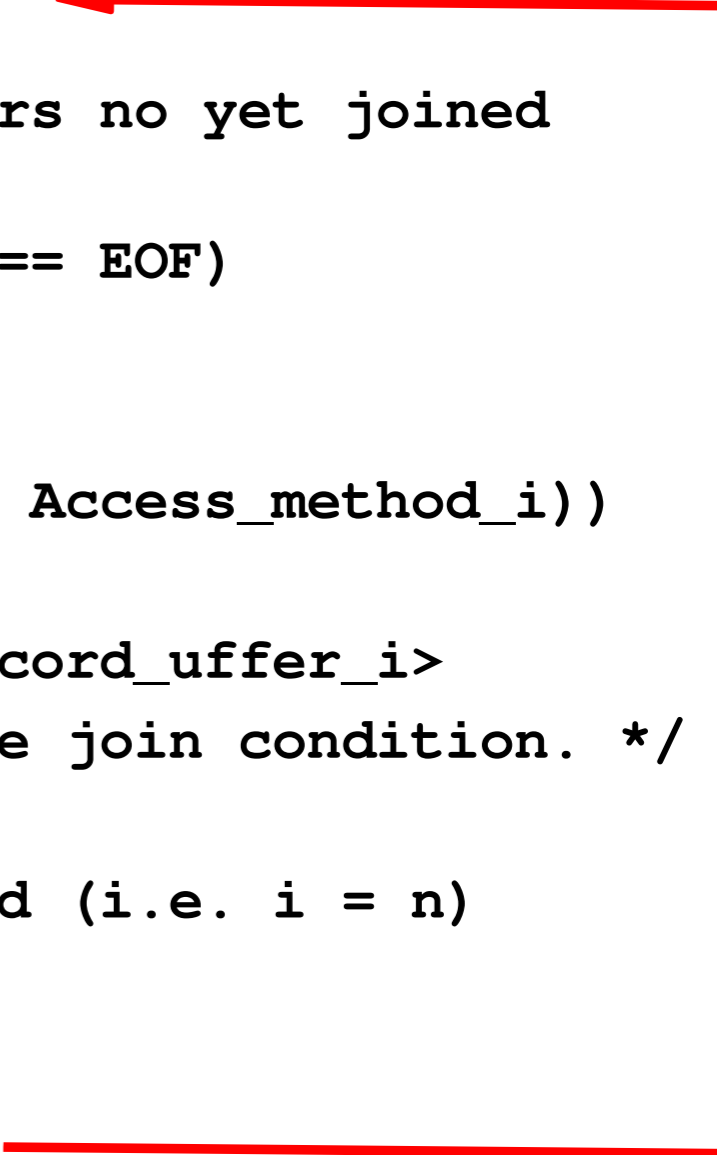



Storage engine

Nested loops join pseudocode

```
procedure nested_loops_join
input: <OP_i, ..., OP_n> // remaining QEP operators no yet joined
{
  if (init_record_scan(Table_i, Access_method_i) == EOF)
    return

  while (curr_record_i = get_next_record(Table_i, Access_method_i))
  {
    joined_record = <record_buffer_1 || ... || record_uffer_i>
    if join_condition_i(joined_record) /* Test the join condition. */
    {
      if joined_record is a complete result record (i.e. i = n)
        output joined_record
      else
        nested_loops_join(OP_[i+1], ..., OP_n)
    }
  }
}
```



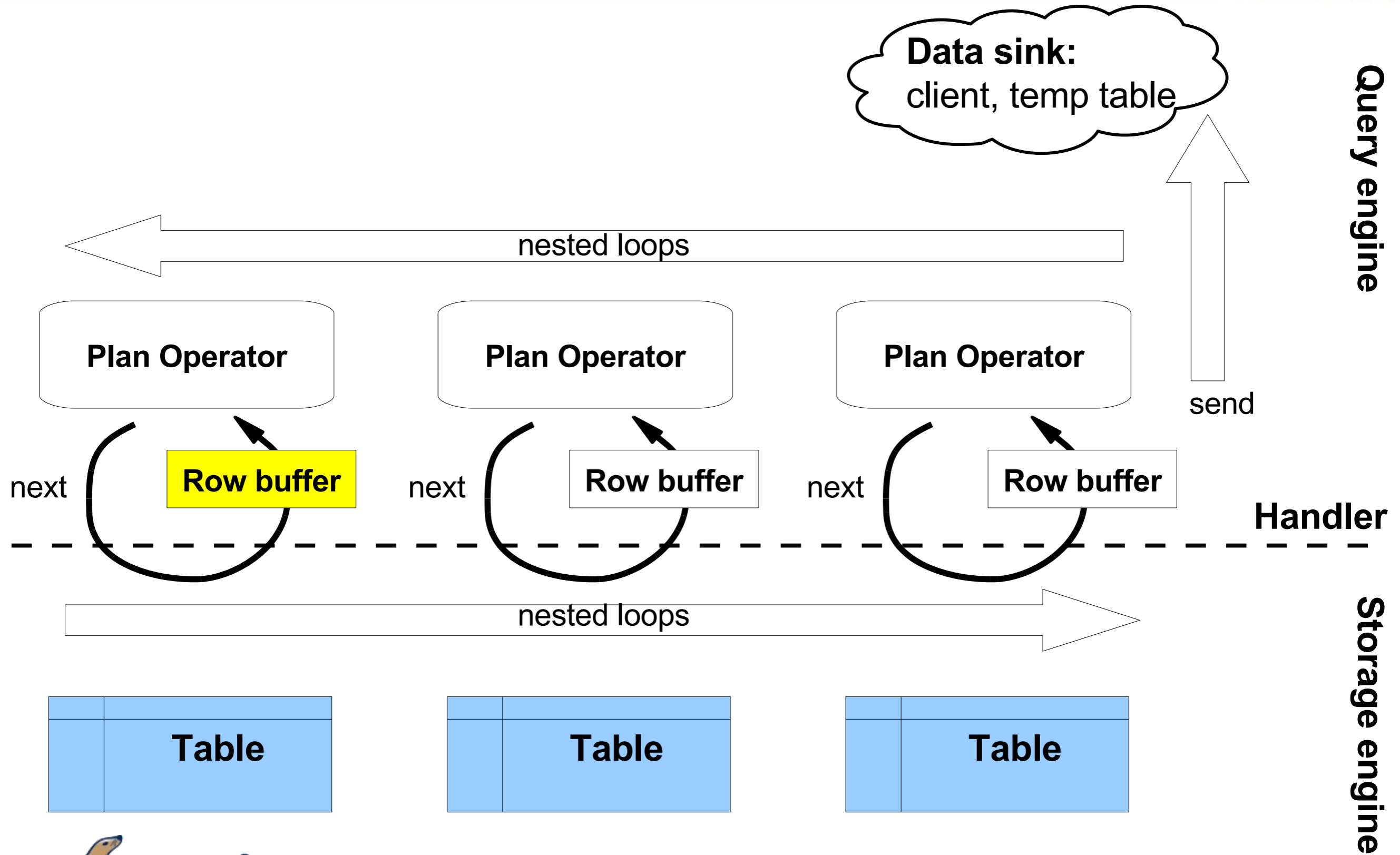


```
enum_nested_loop_state sub_select(JOIN_TAB *remainder)
{
    error= (*join_tab->read_first_record)(join_tab);
    rc= evaluate_join_record(join, join_tab, error);
    while (rc == NESTED_LOOP_OK)
    {
        error= info->read_record(info);
        rc= evaluate_join_record(join, join_tab, error);
    }
}

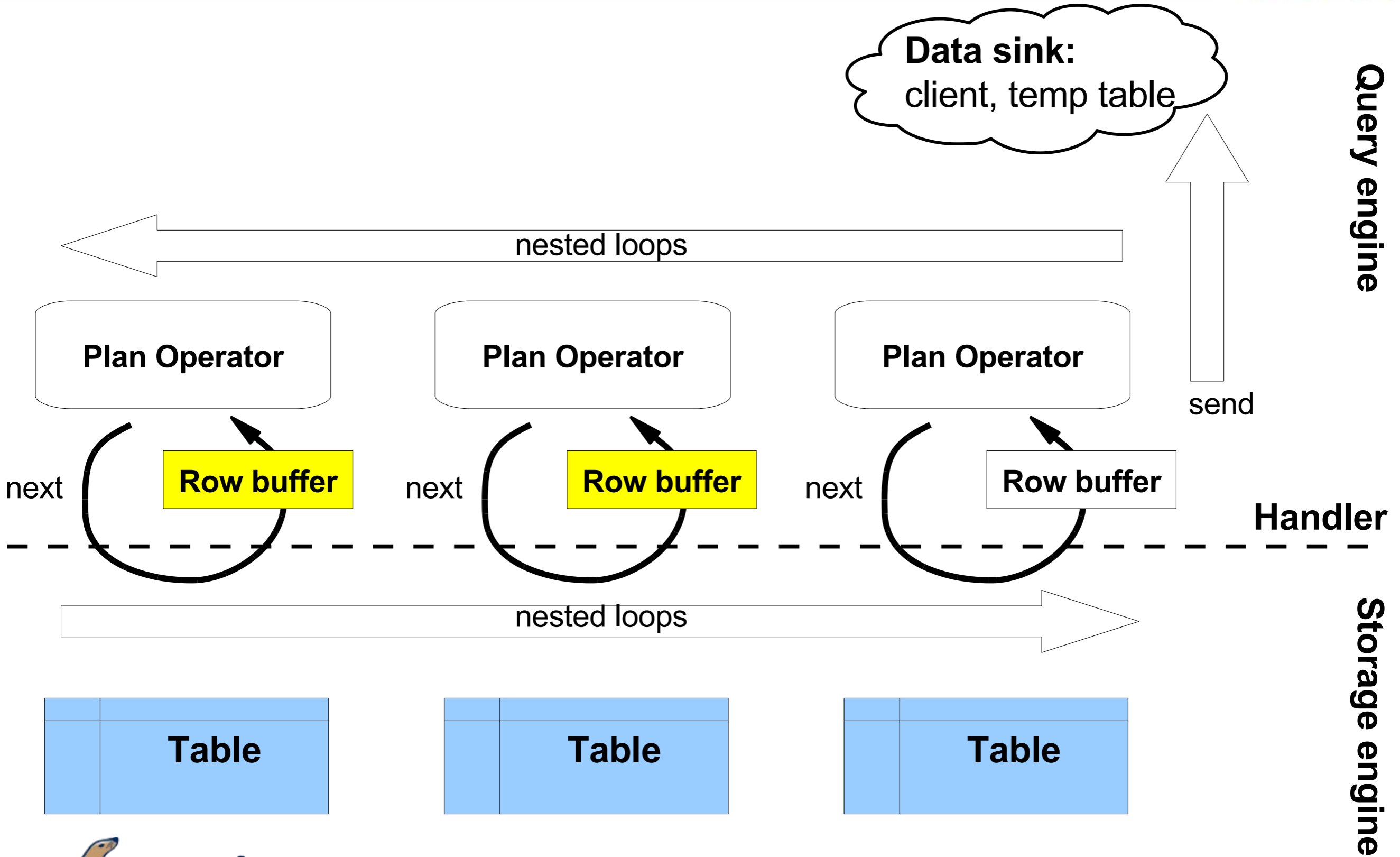
enum_nested_loop_state evaluate_join_record(JOIN_TAB *remainder)
{
    found= test(select_cond->val_int());
    If (found)
        rc= (*join_tab->next_select)(join, join_tab+1, 0);
}
```

file: sql/sql_select.cc

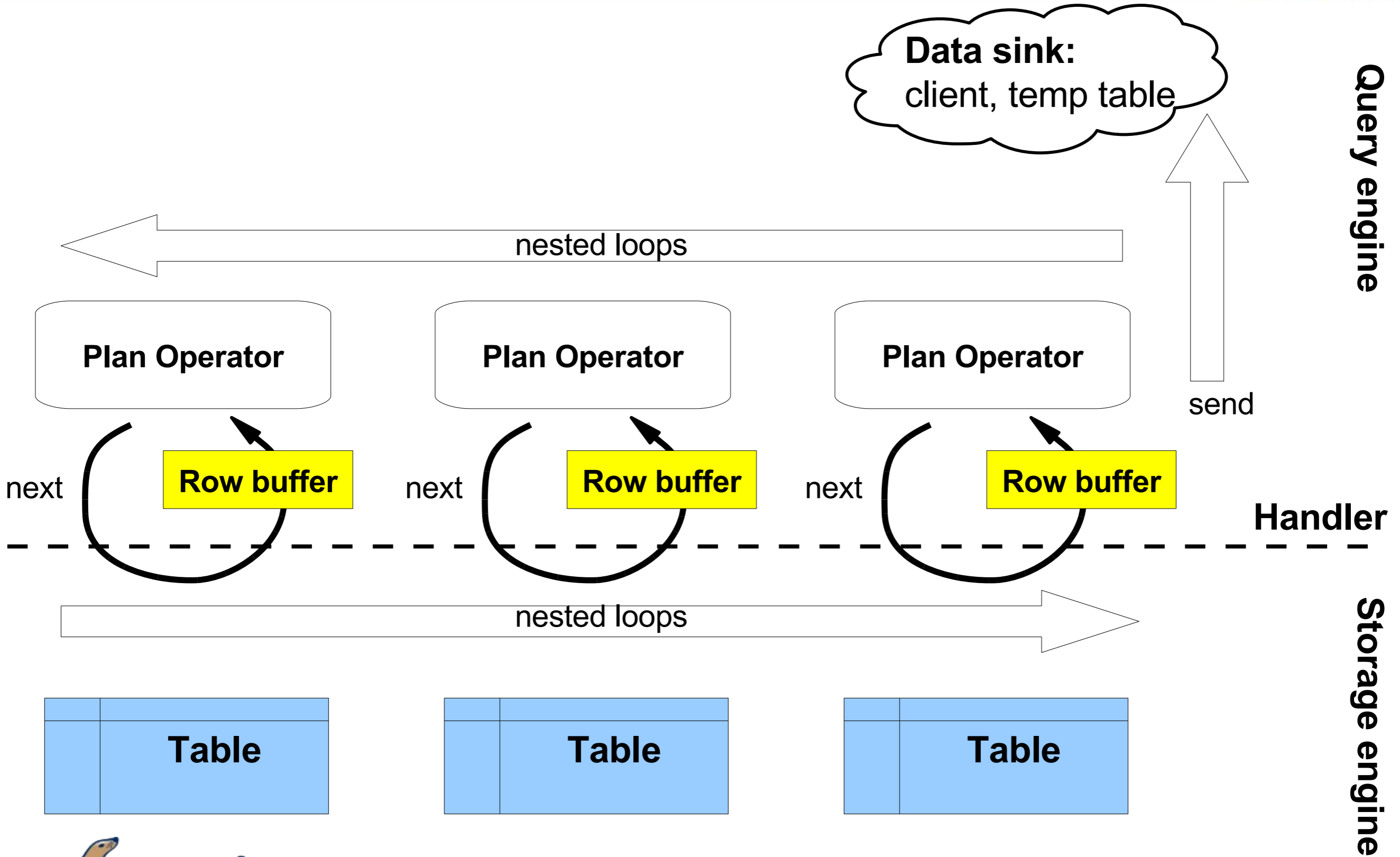
The pull-push execution model



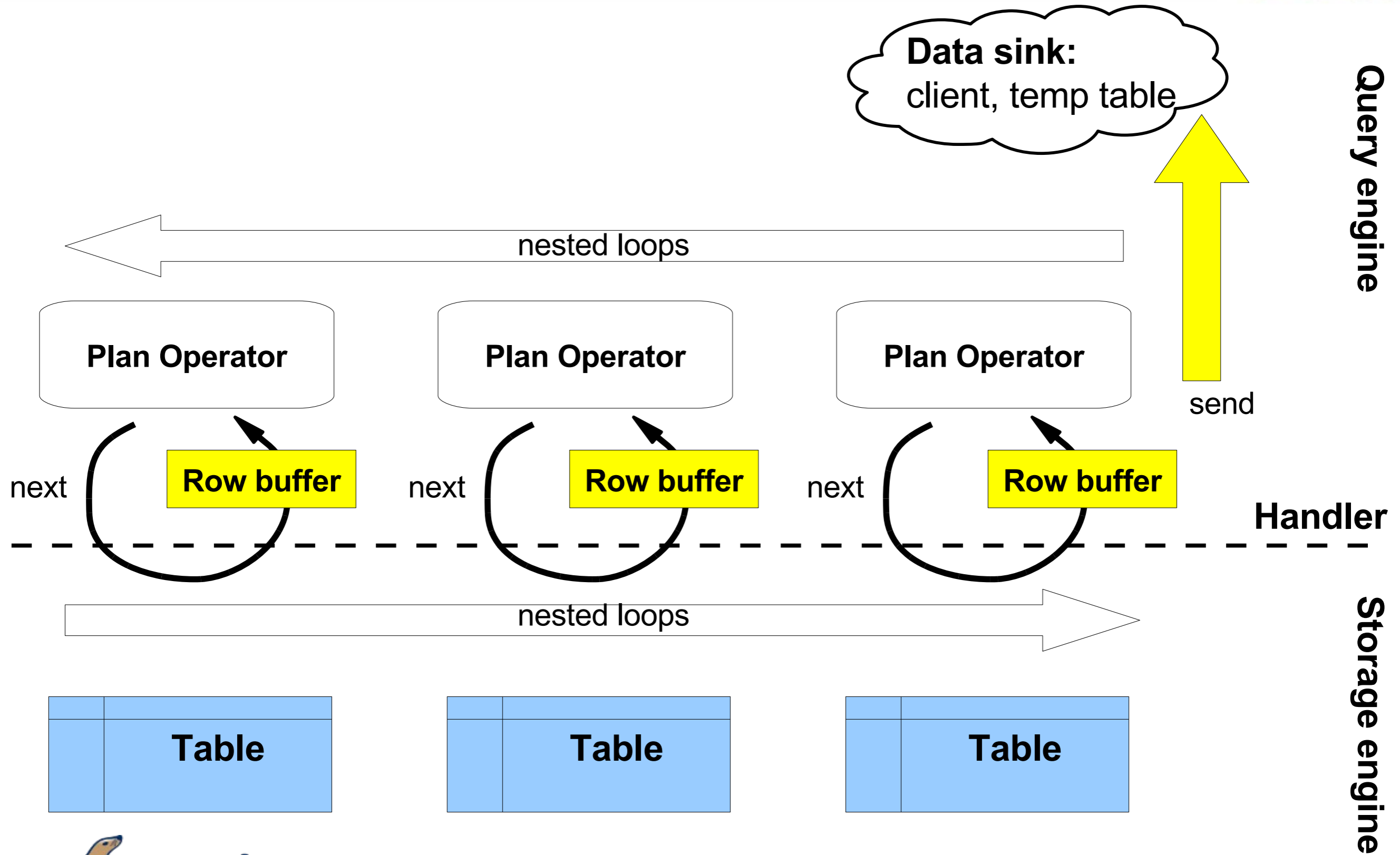
The pull-push execution model



The pull-push execution model



The pull-push execution model



Thank you



Questions?