WWW.PICODATA.IO

PICODATA

Picodata SQL internals

What it is?

PICODATA

Plan

PICODATA

1. Overview
2. Read query 1
3. Read query 2
4. DML
5. DDL

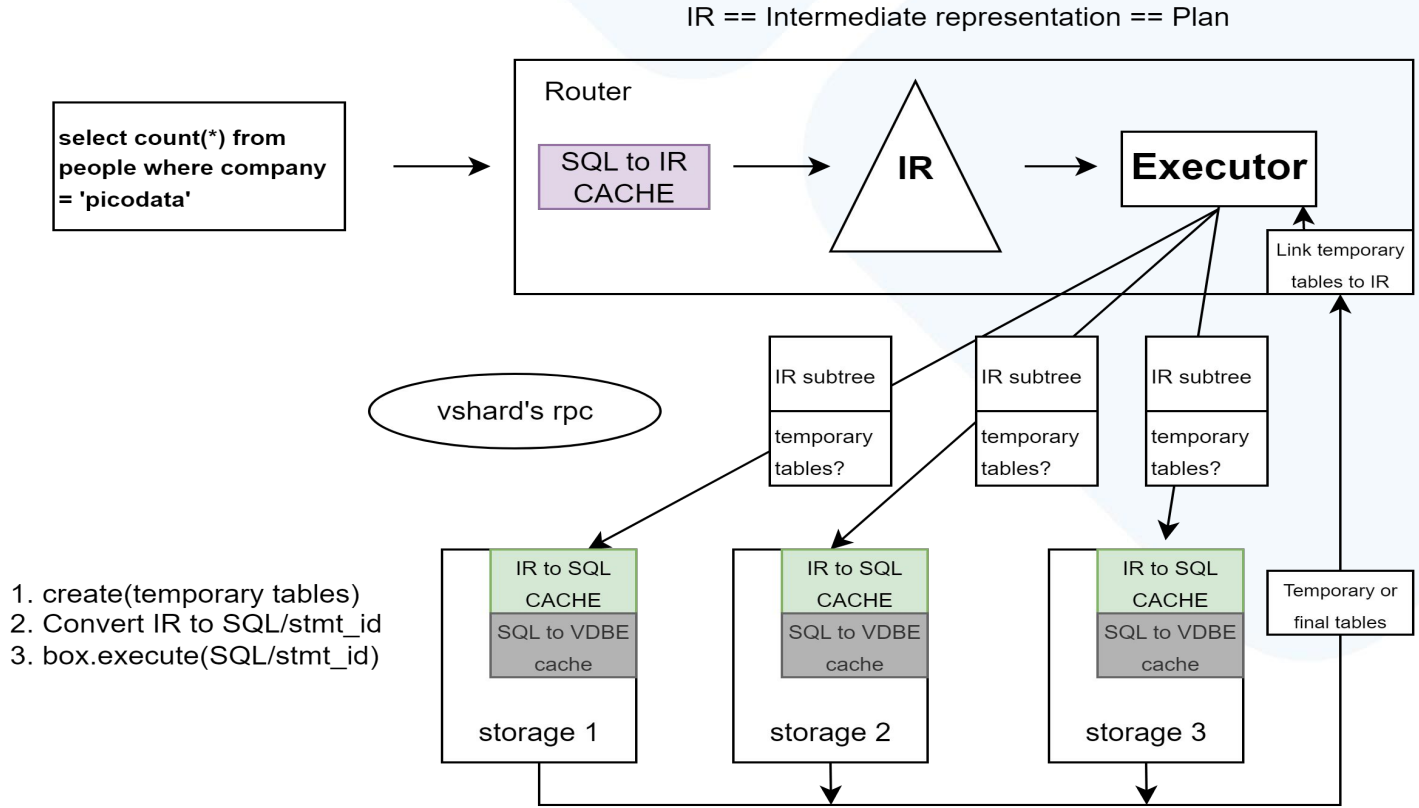
Overview

Что такое Picodata SQL

PICODATA

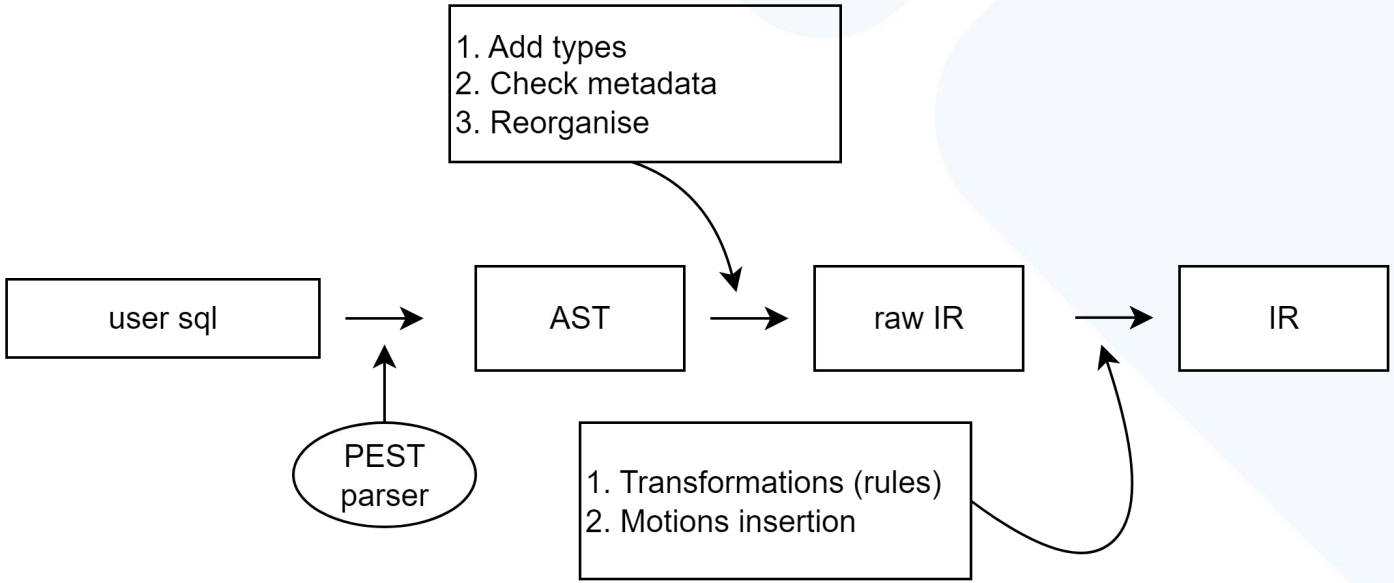
1. Library for distributed SQL
2. Uses vshard for queries dispatch
3. Needs several traits to be implemented before usage (e.g DDL)
4. For executing on single node uses SQL Tarantool (read queries), tarantool-module (update/insert/delete)

Overview: big picture

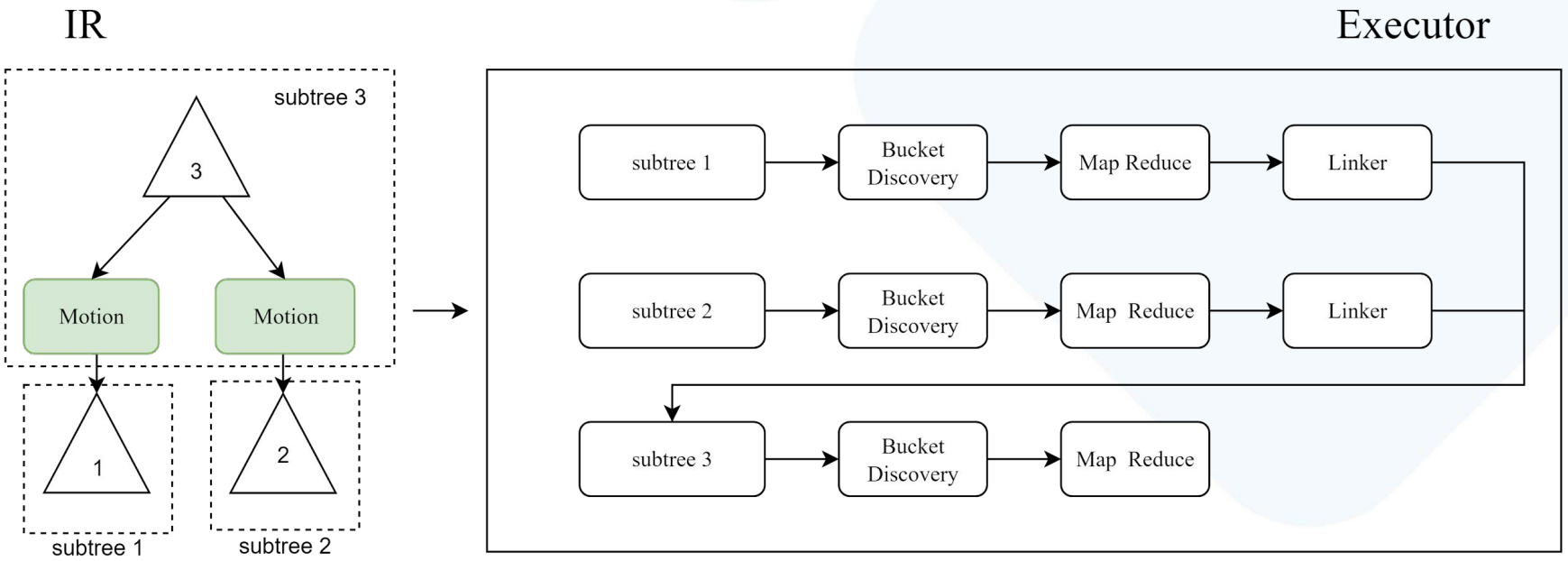


Overview: IR creation

PICODATA



Overview: Executor



Example 1

Query

PICODATA

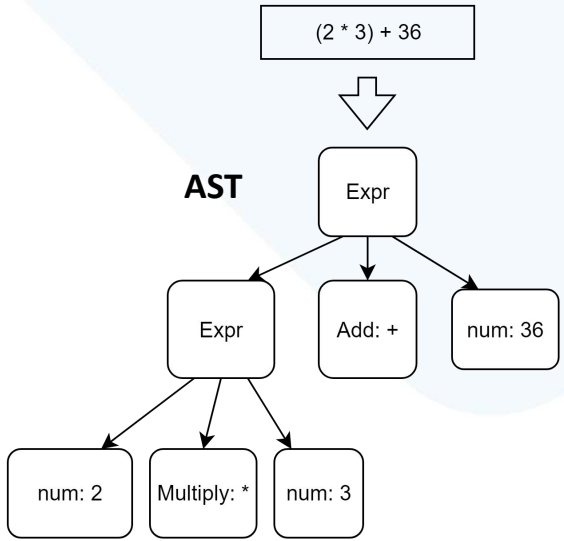
```
SELECT b FROM t WHERE a = 1
```

Grammar

PICODATA

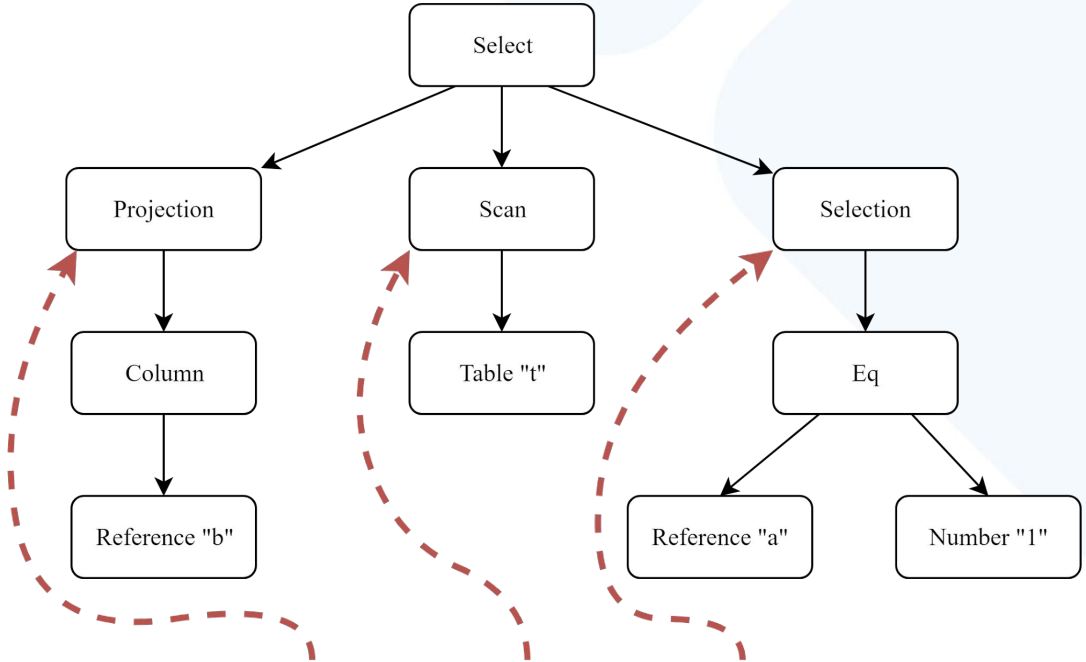
- Grammar - a way to describe formal language
- AST (abstract syntax tree) - representation of sentence (that satisfies grammar) in form of a tree.
- PEST - external library for building AST using PEG (Parsing Expression Grammar)

```
num = @{ ("+" | "-")? ~ ASCII_DIGIT+ }
operation = _{ add | subtract | multiply | divide}
    add      = { "+" }
    subtract = { "-" }
    multiply = { "*" }
    divide   = { "/" }
expr = { term ~ (operation ~ term)* }
term = _{ num | "(" ~ expr ~ ")" }
calculation = _{ SOI ~ expr ~ EOI }
```



AST

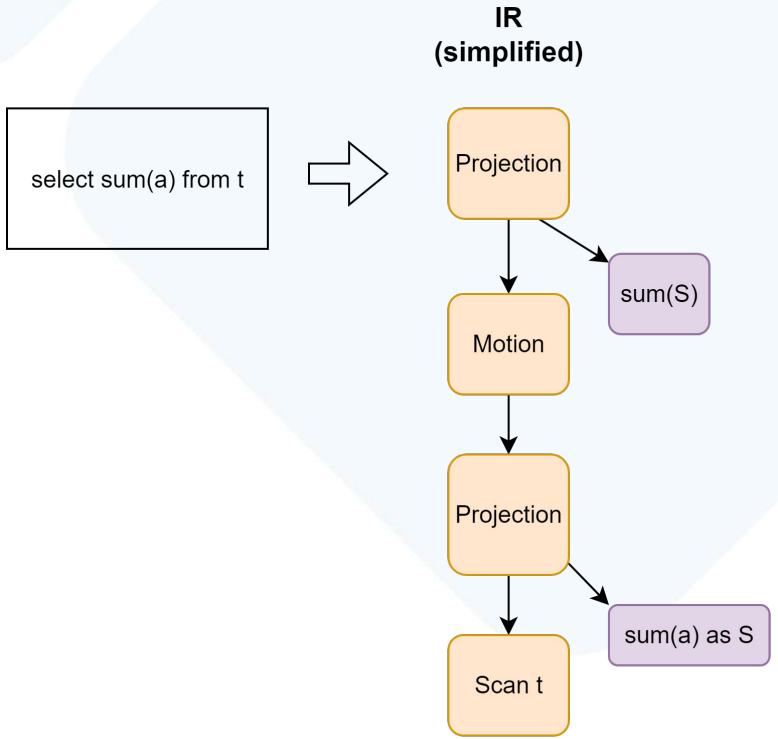
PICODATA



IR

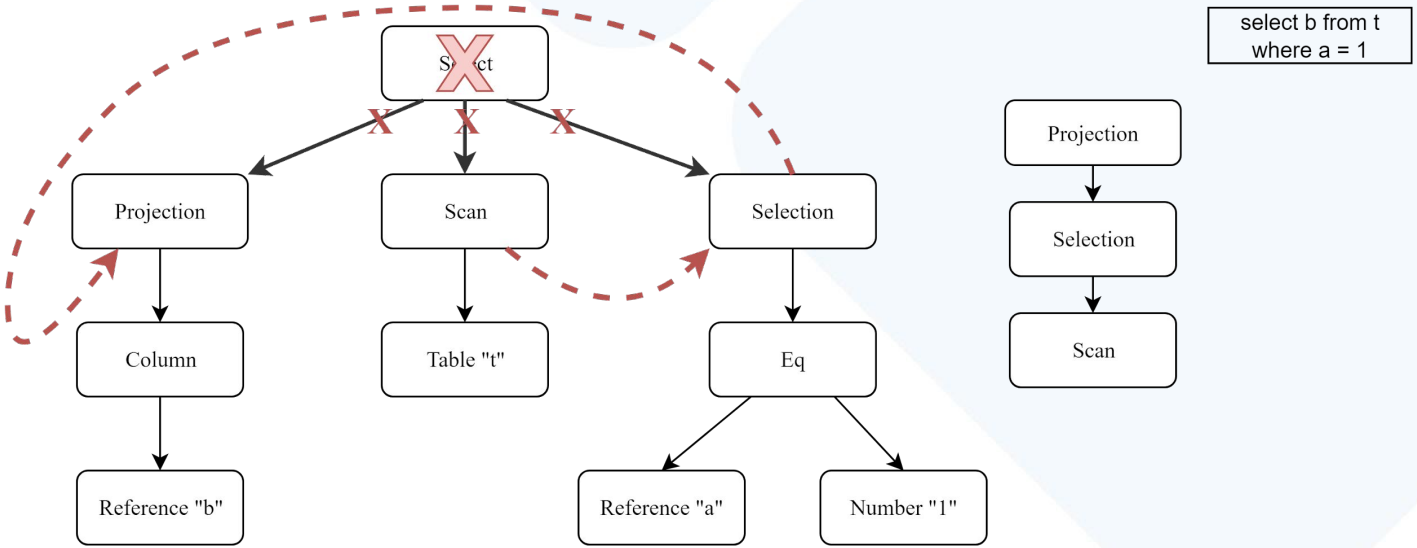
PICODATA

1. IR (intermediate representation) - represents a plan for distributed query execution
2. It is a tree consisting of Relational and Expression nodes
3. Motion - special Relational node, that materializes subtree below it
4. Each subtree below Motion can be converted to valid tarantool's SQL



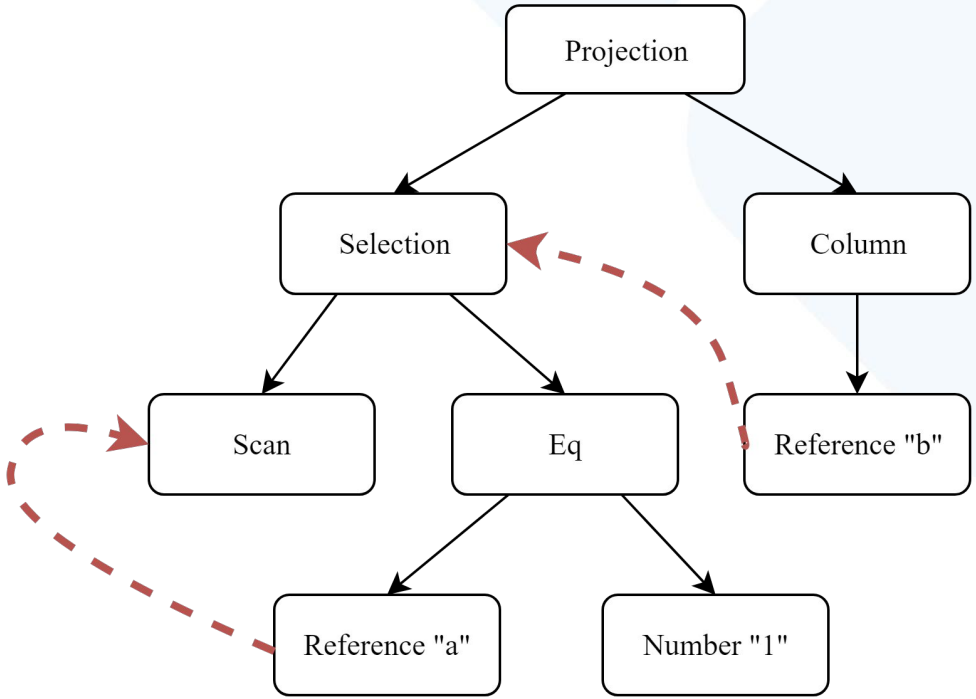
AST: transform select

PICODATA



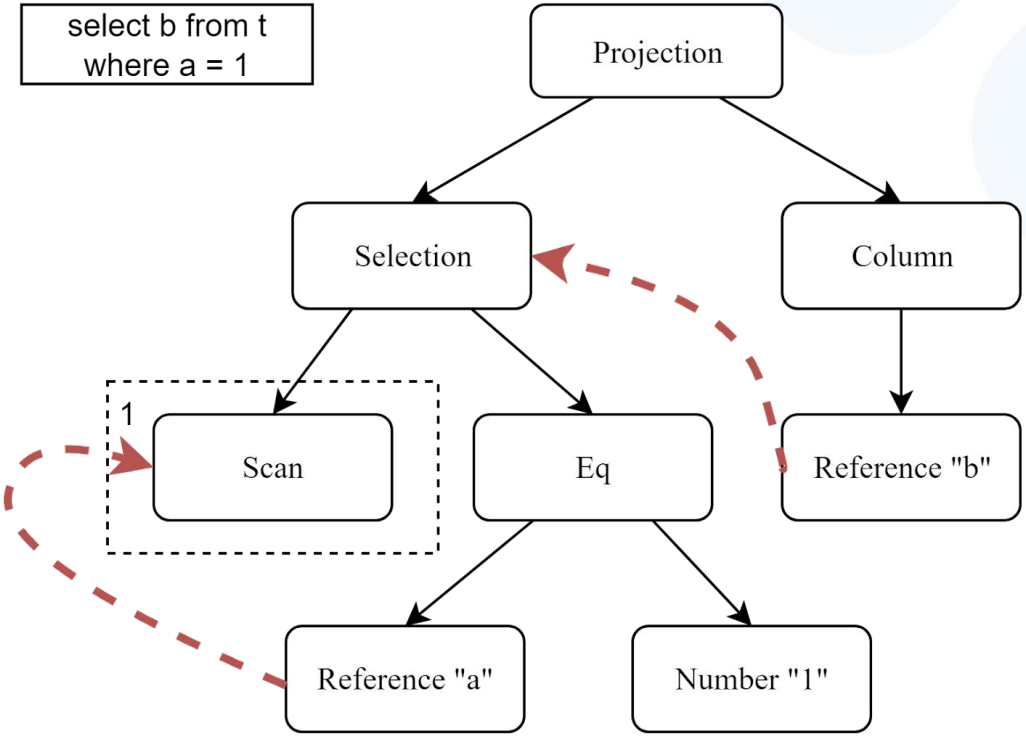
AST: bind references

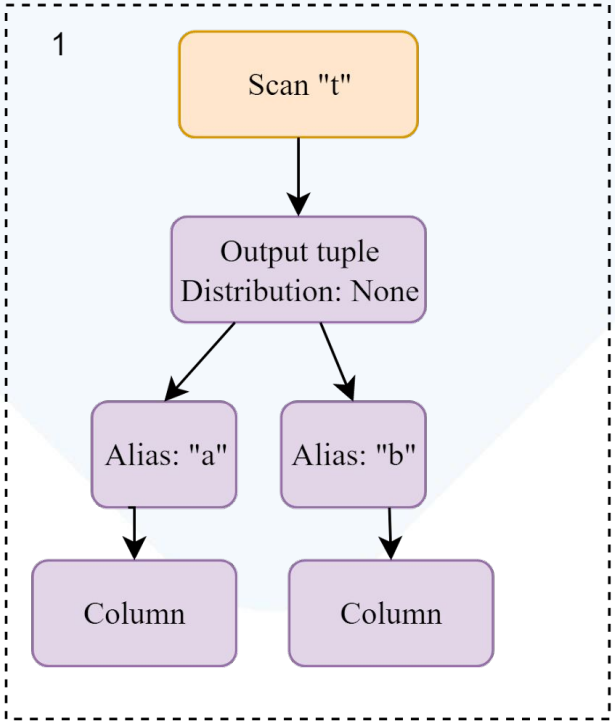
PICODATA



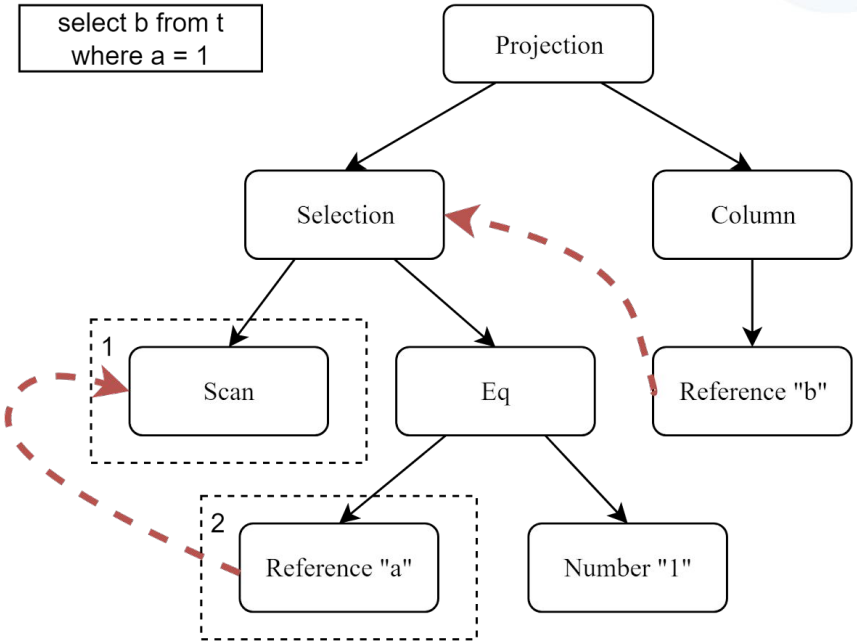

```
select b from t
where a = 1
```

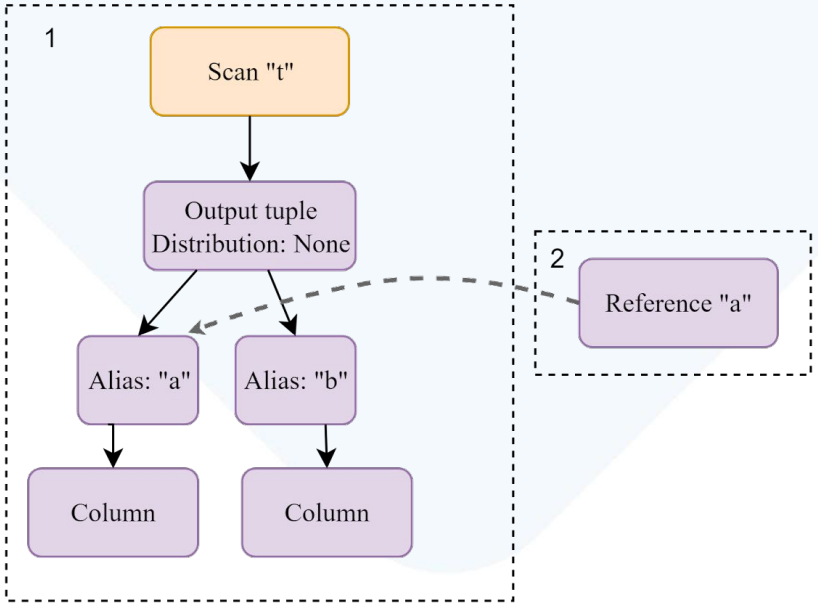
AST to raw IR


select b from t
where a = 1
Projection
Selection
Column
1
Scan
Eq
Reference "b"
Reference "a"
Number "1"

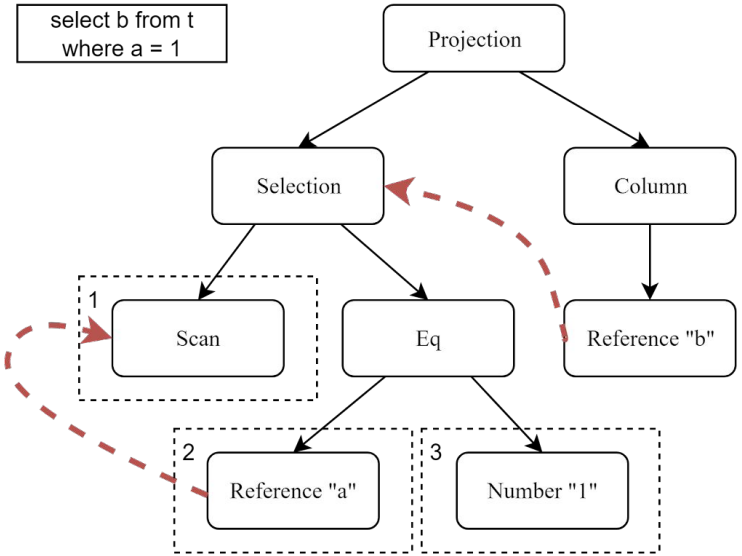

1
Scan "t"
Output tuple
Distribution: None
Alias: "a"
Alias: "b"
Column
Column

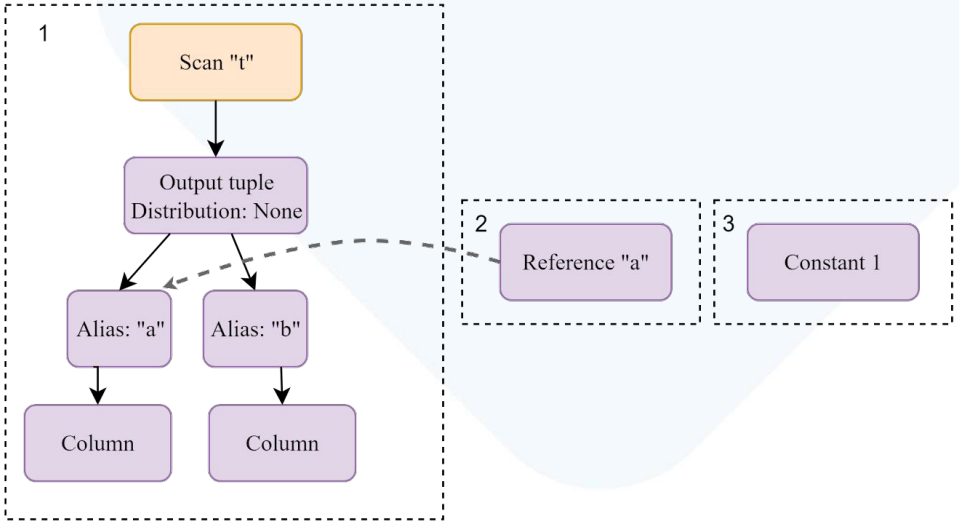
AST to raw IR


select b from t
where a = 1
Projection
Selection
Column
1
Scan
Eq
Reference "b"
2
Reference "a"
Number "1"

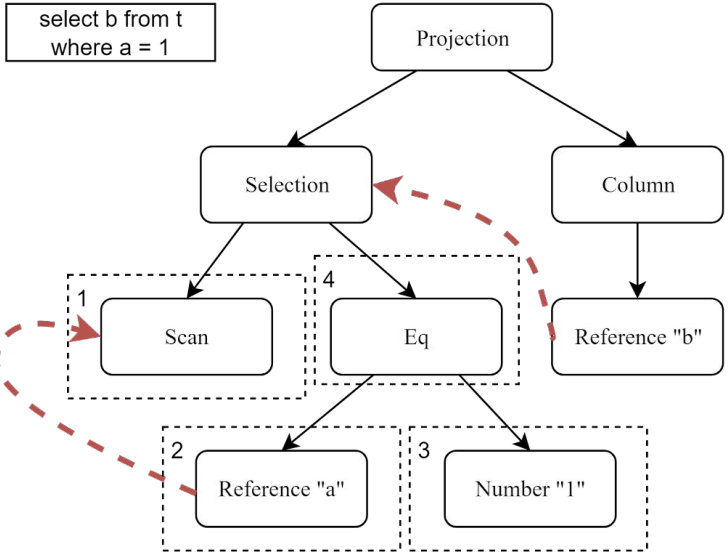

1
Scan "t"
Output tuple
Distribution: None
2
Reference "a"
Alias: "a"
Alias: "b"
Column
Column

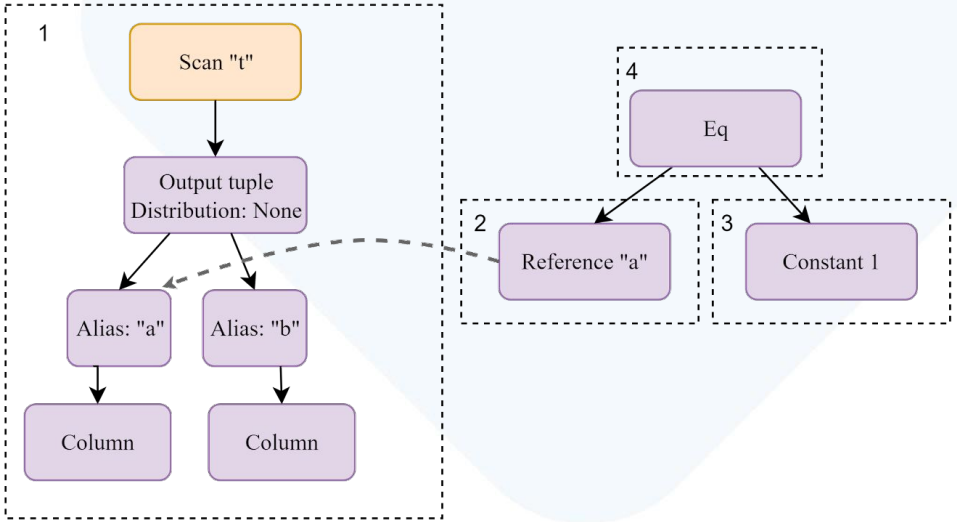
AST to raw IR


select b from t
where a = 1
Projection
Selection
Column
1
Scan
Eq
Reference "b"
2
Reference "a"
3
Number "1"


1
Scan "t"
Output tuple
Distribution: None
Alias: "a"
Alias: "b"
Column
Column
2
Reference "a"
3
Constant 1

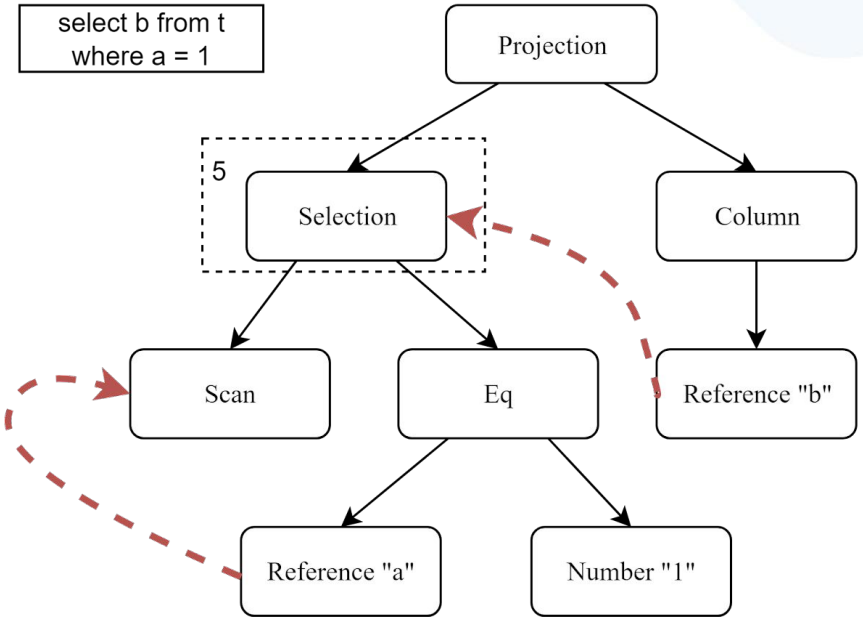
AST to raw IR

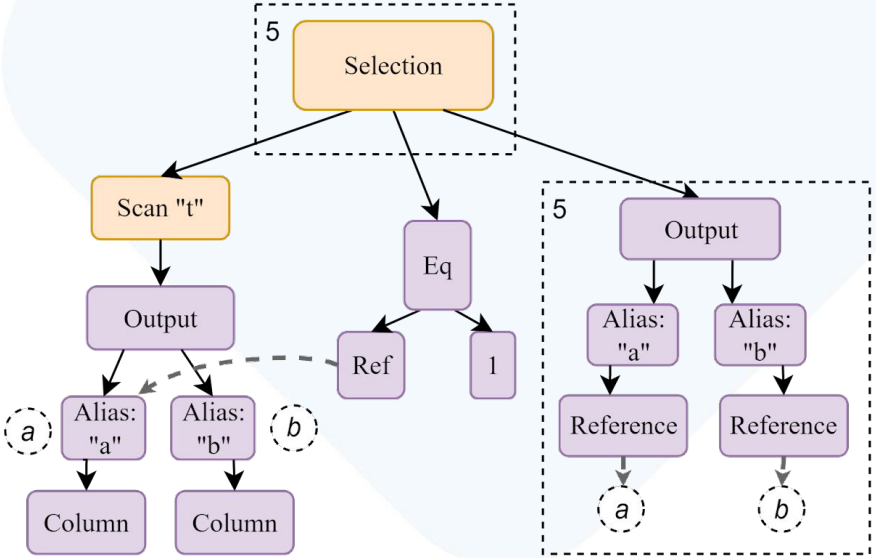

select b from t
where a = 1
Projection
Selection
Column
1
Scan
4
Eq
Reference "b"
2
Reference "a"
3
Number "1"


1
Scan "t"
Output tuple
Distribution: None
Alias: "a"
Alias: "b"
Column
Column
4
Eq
2
Reference "a"
3
Constant 1

AST to raw IR

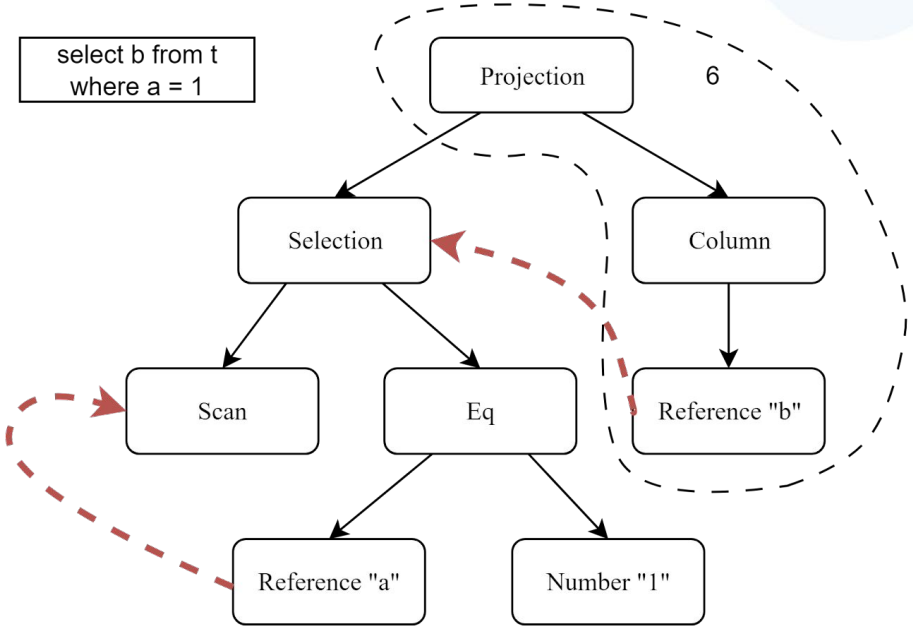
PICODATA

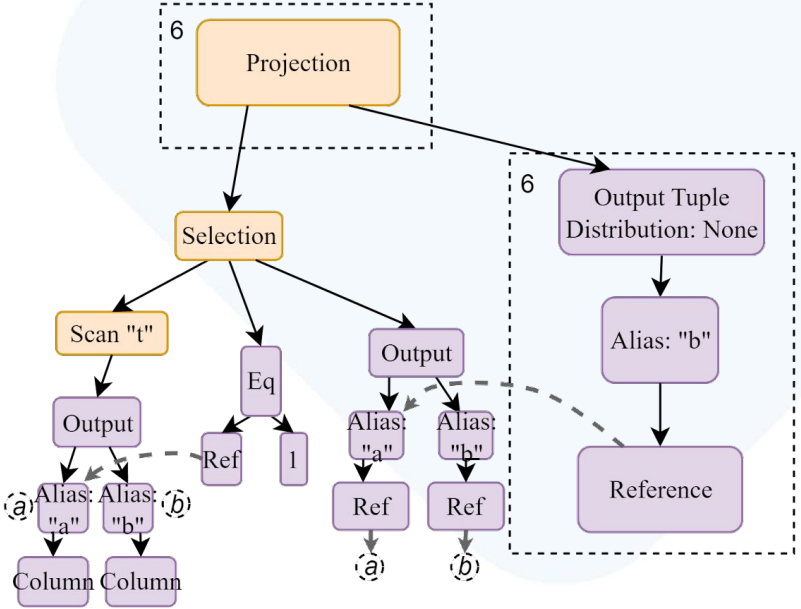




AST to raw IR

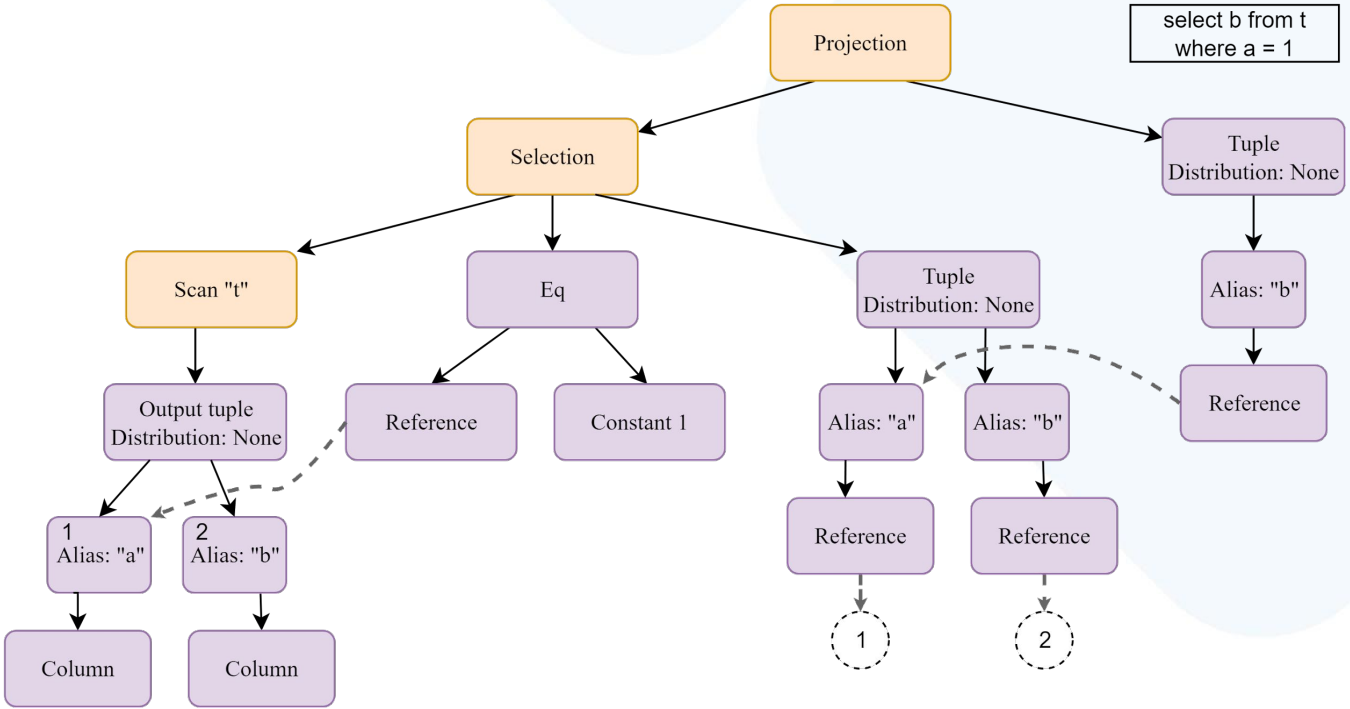
PICODATA





raw IR

PICODATA

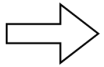


Sharded tables

PICODATA

t sharded by a

a	b	bucket_id
1	1	100
1	2	100
150	1	2400



Storage 1
buckets: 1 - 1000

a	b	bucket_id
1	1	100
1	2	100

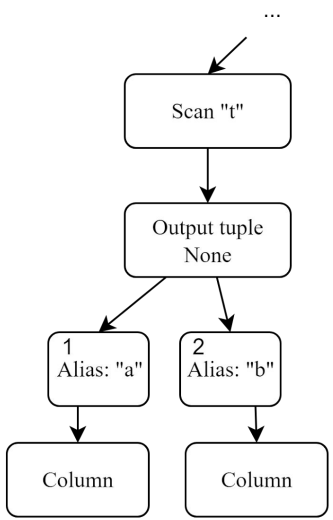
Storage 2
buckets: 1001 - 3000

a	b	bucket_id
150	1	2400

Distributions

PICODATA

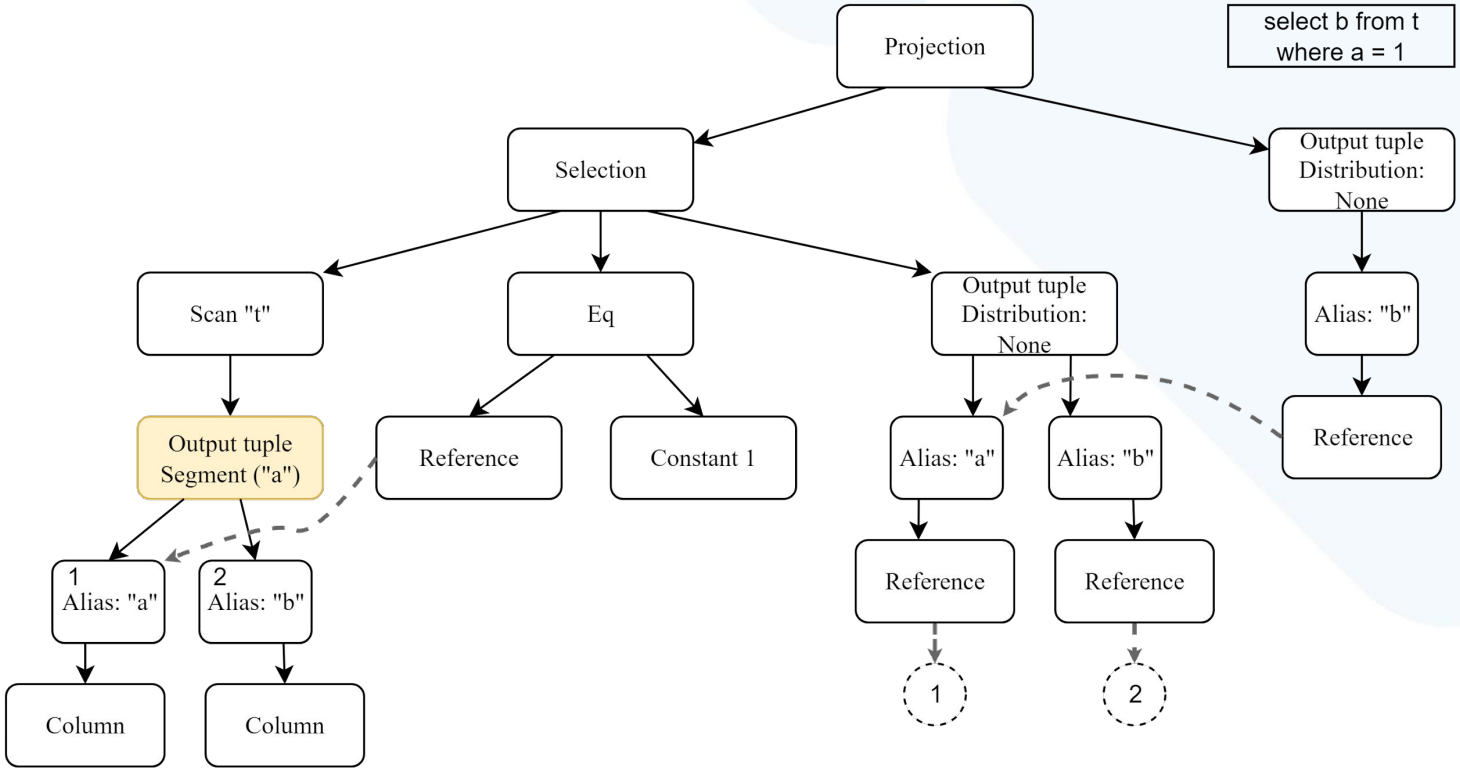
- How during IR build, we understand that Motion must be inserted?
- Distribution - properties of output table, that must be true for Relational node output's table after subtree with its node was dispatched and materialized on storages




```
select b from t
where a = 1
```

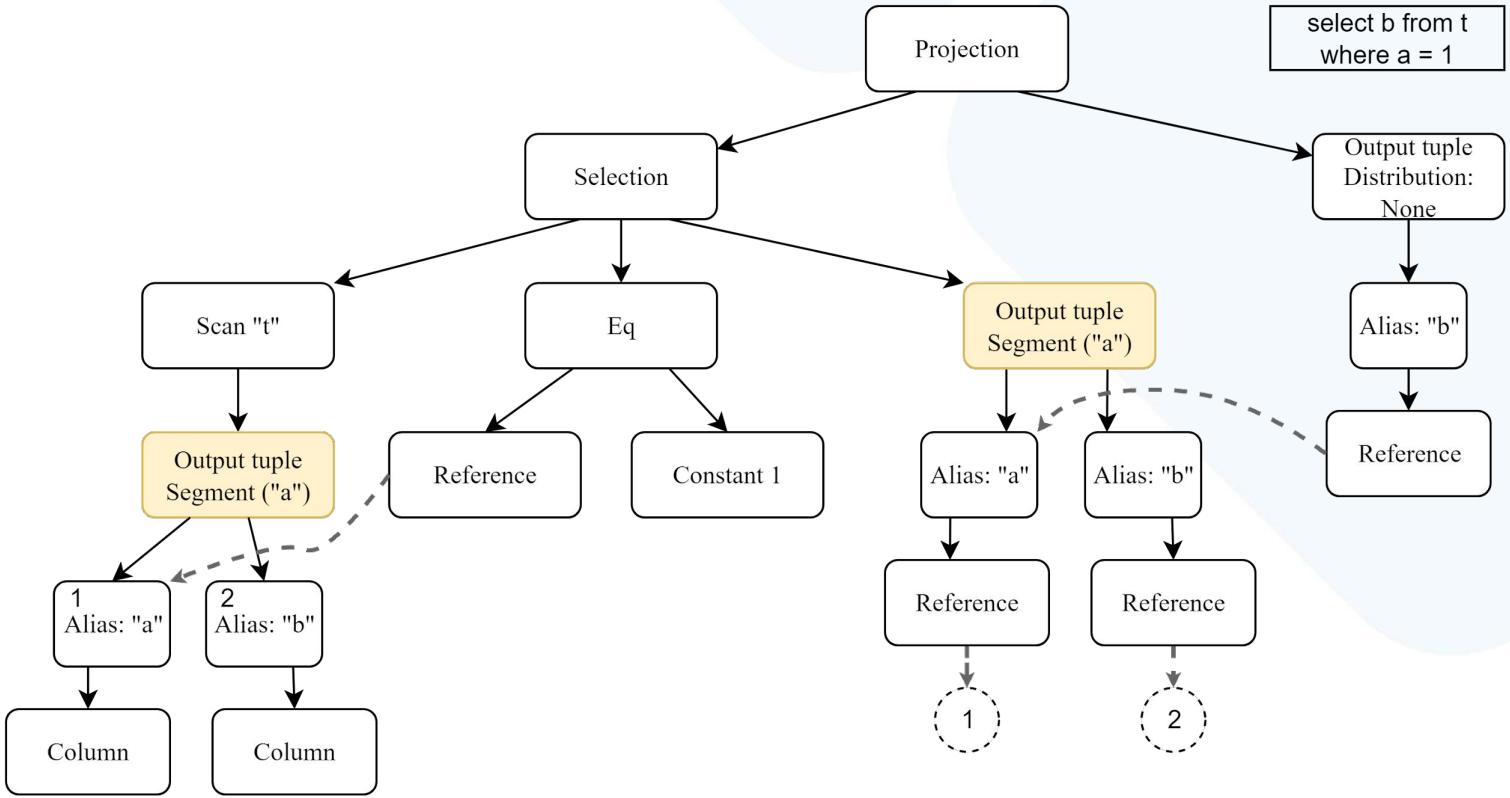
Raw IR to IR

PICODATA



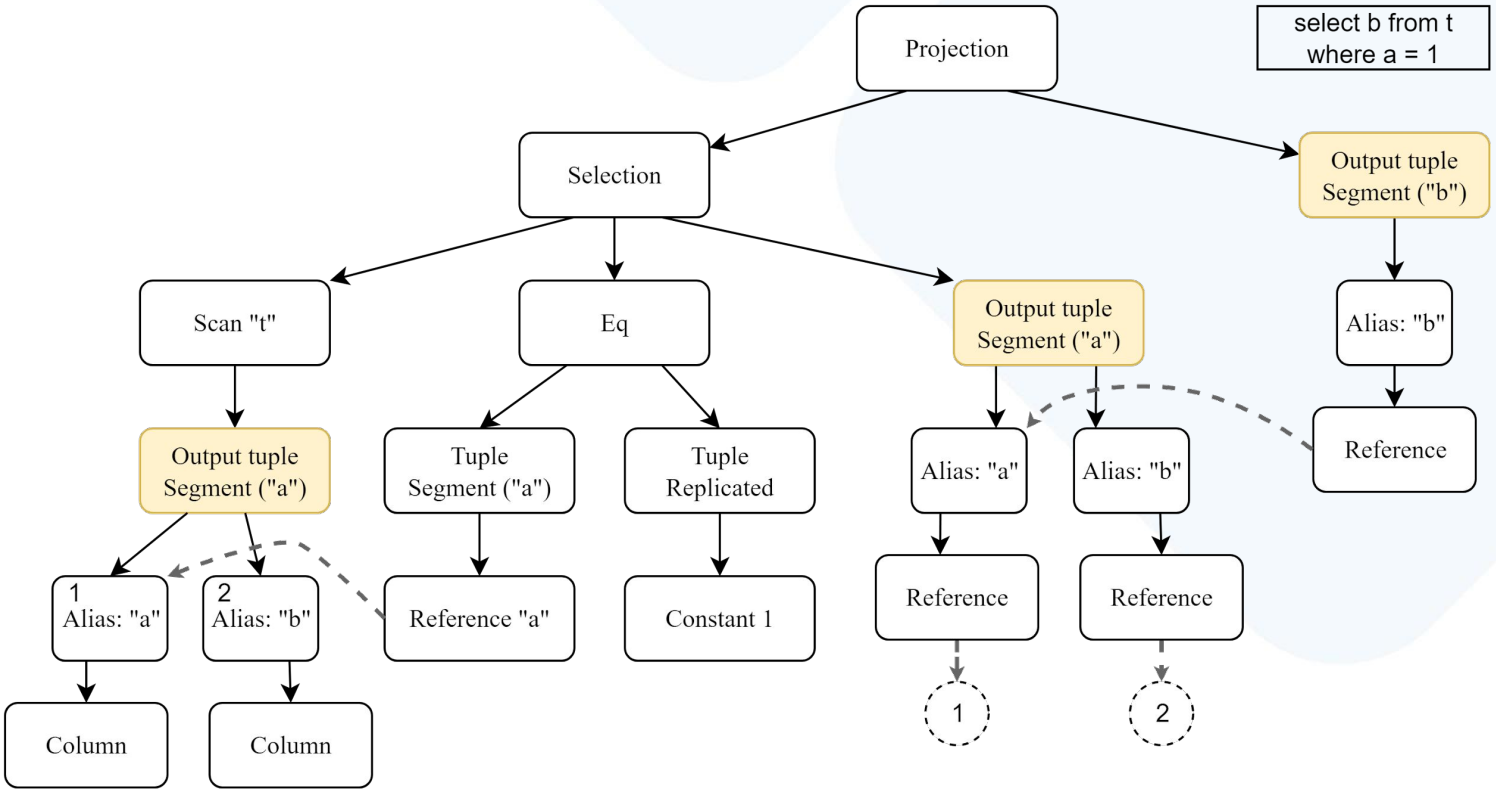
Raw IR to IR

PICODATA



Raw IR to IR

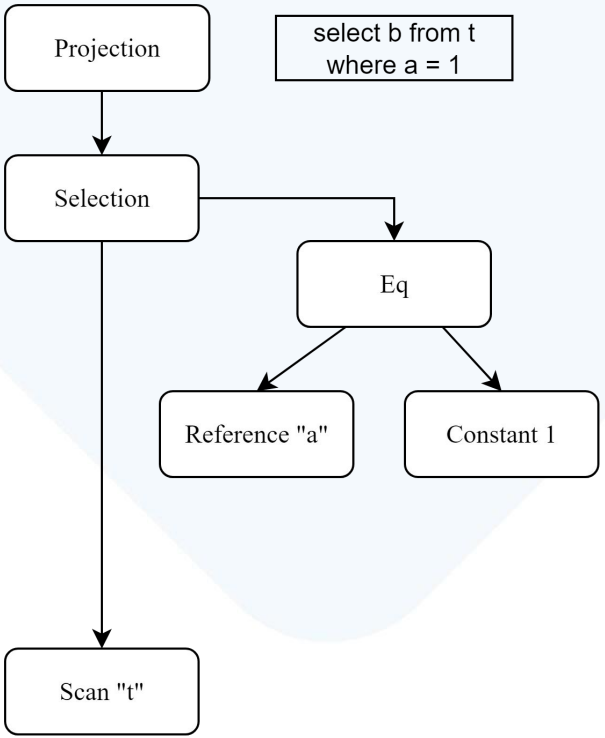
PICODATA



Bucket discovery

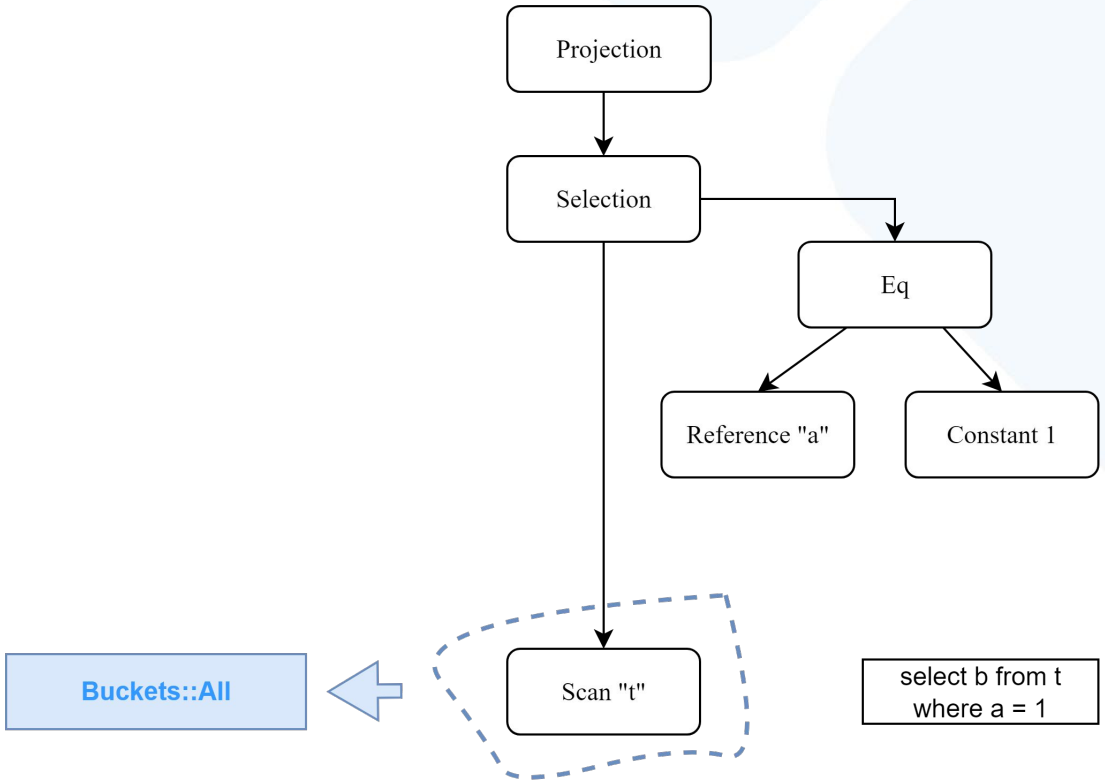
PICODATA

- Bucket discovery - calculation of buckets, where IR subtree must be executed.



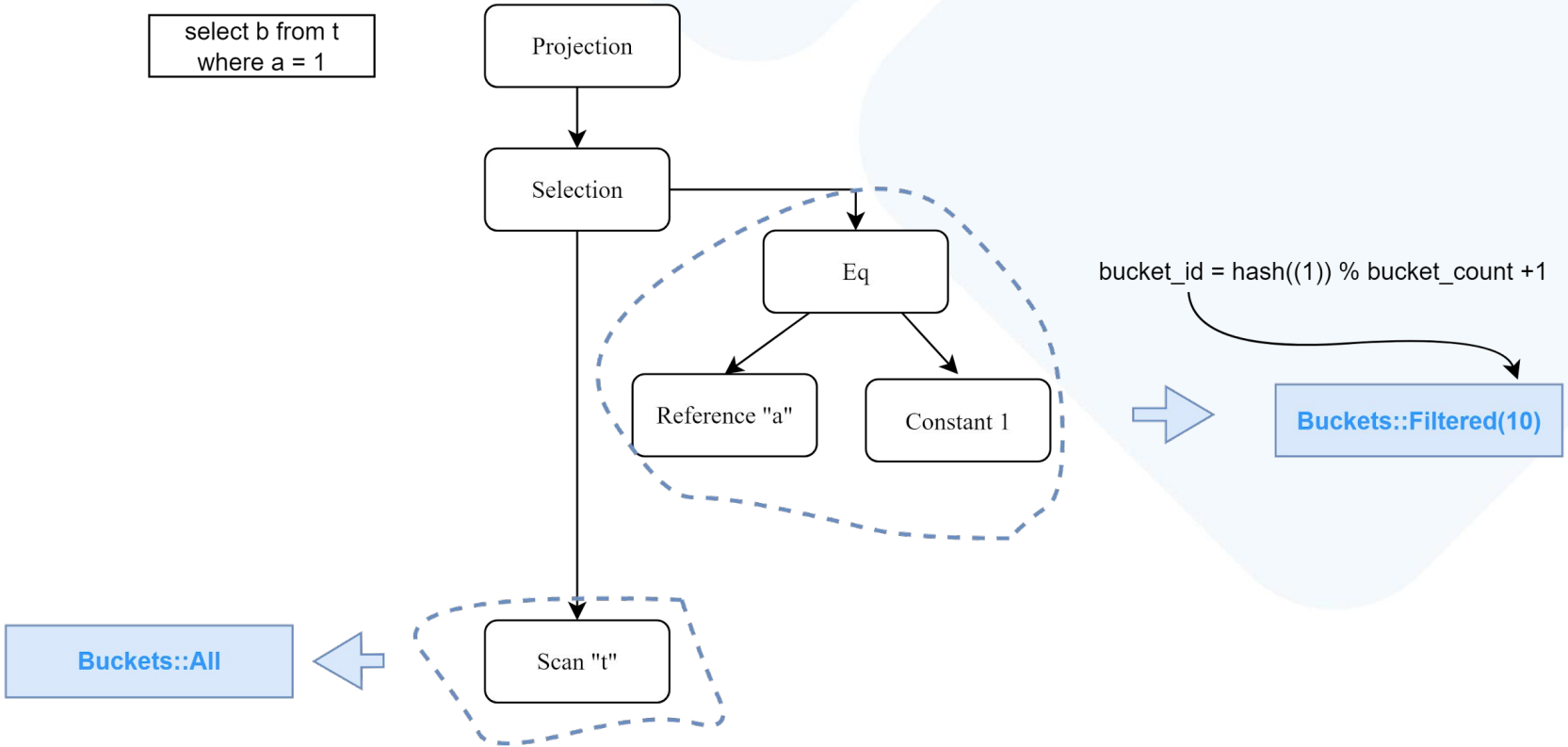
Execution: bucket discovery

PICODATA

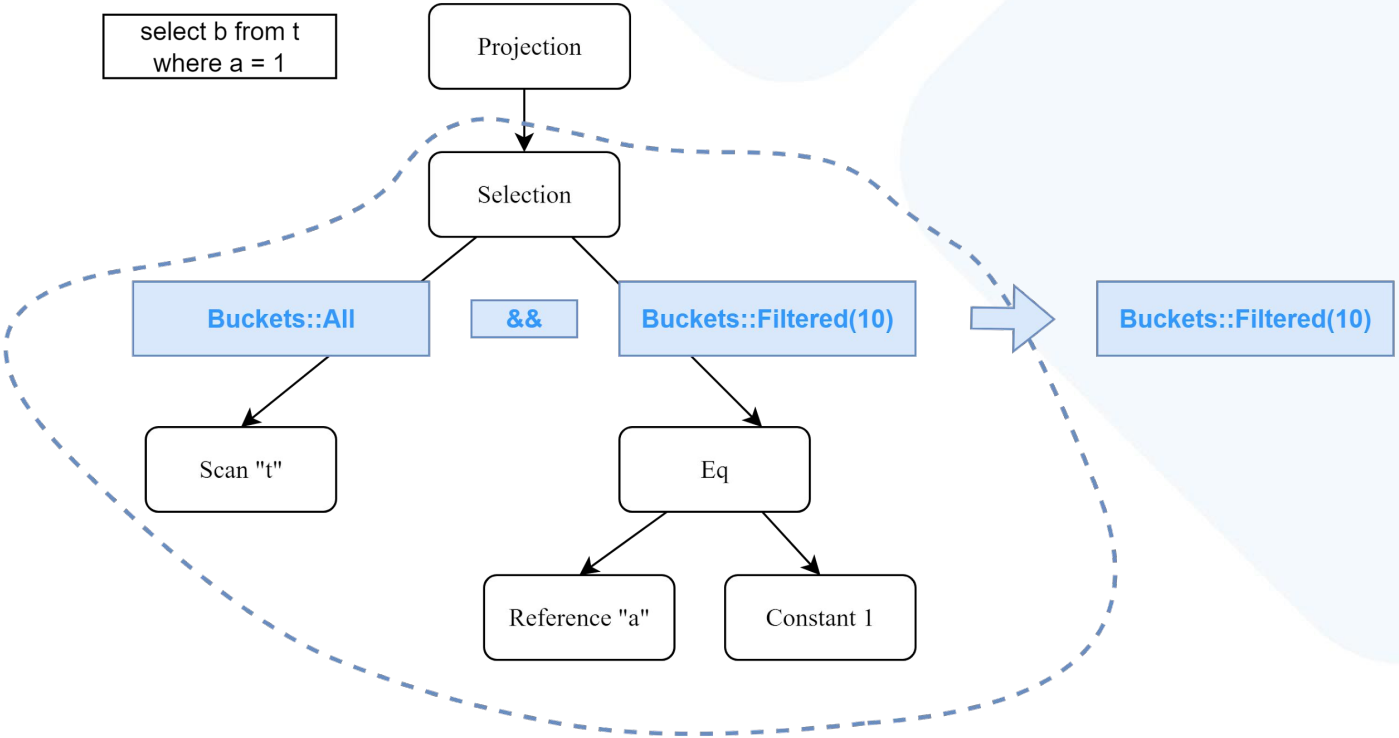


Execution: bucket discovery

PICODATA

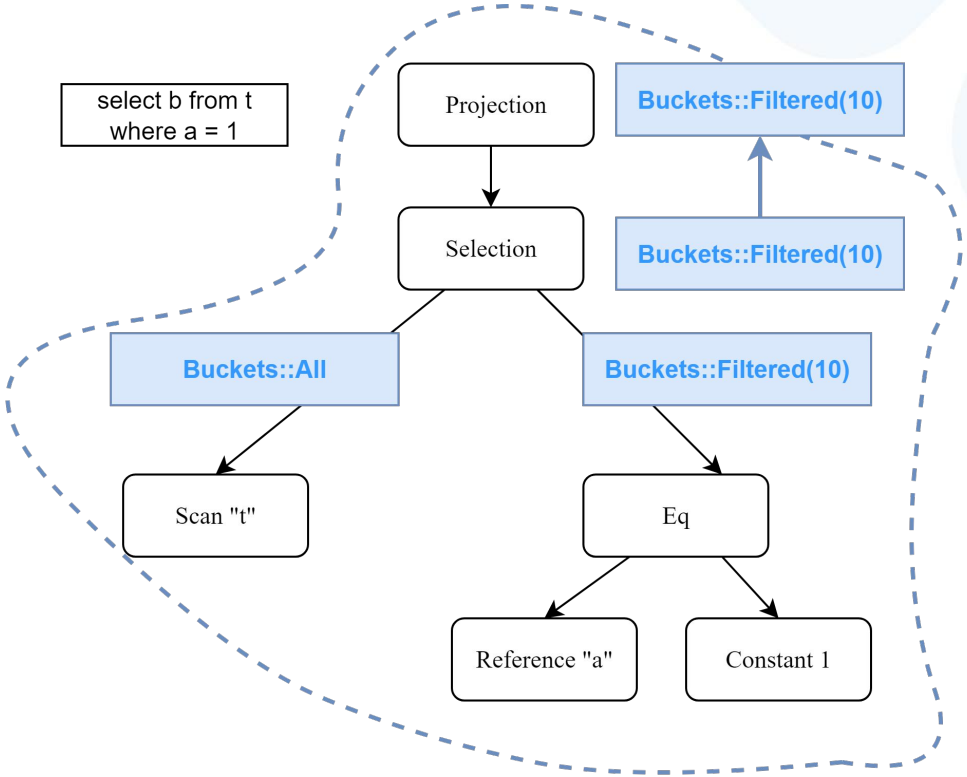


Execution: bucket discovery

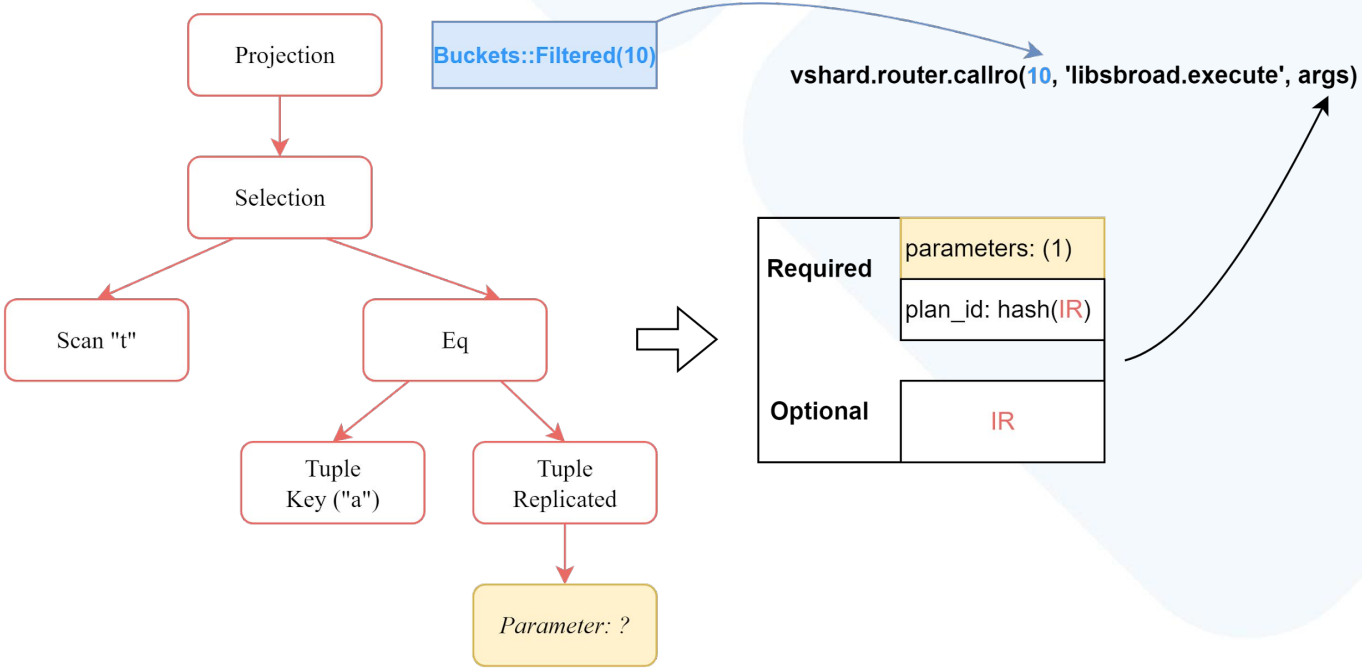


Execution: bucket discovery

PICODATA

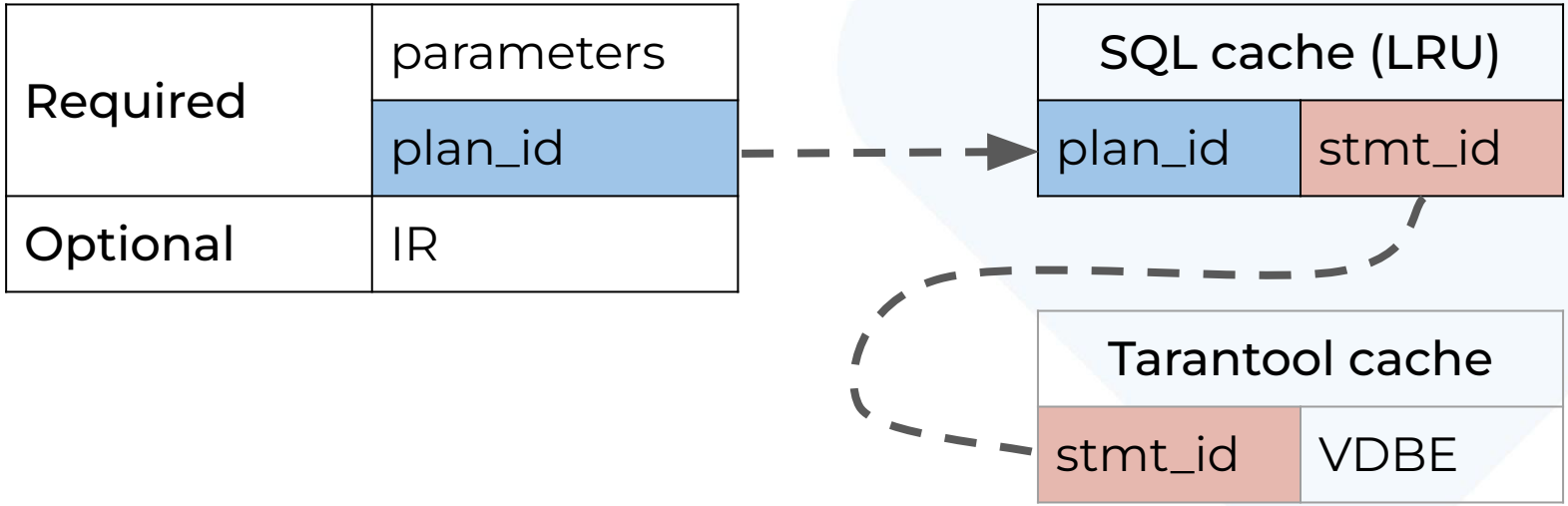


Execution: vshard rpc



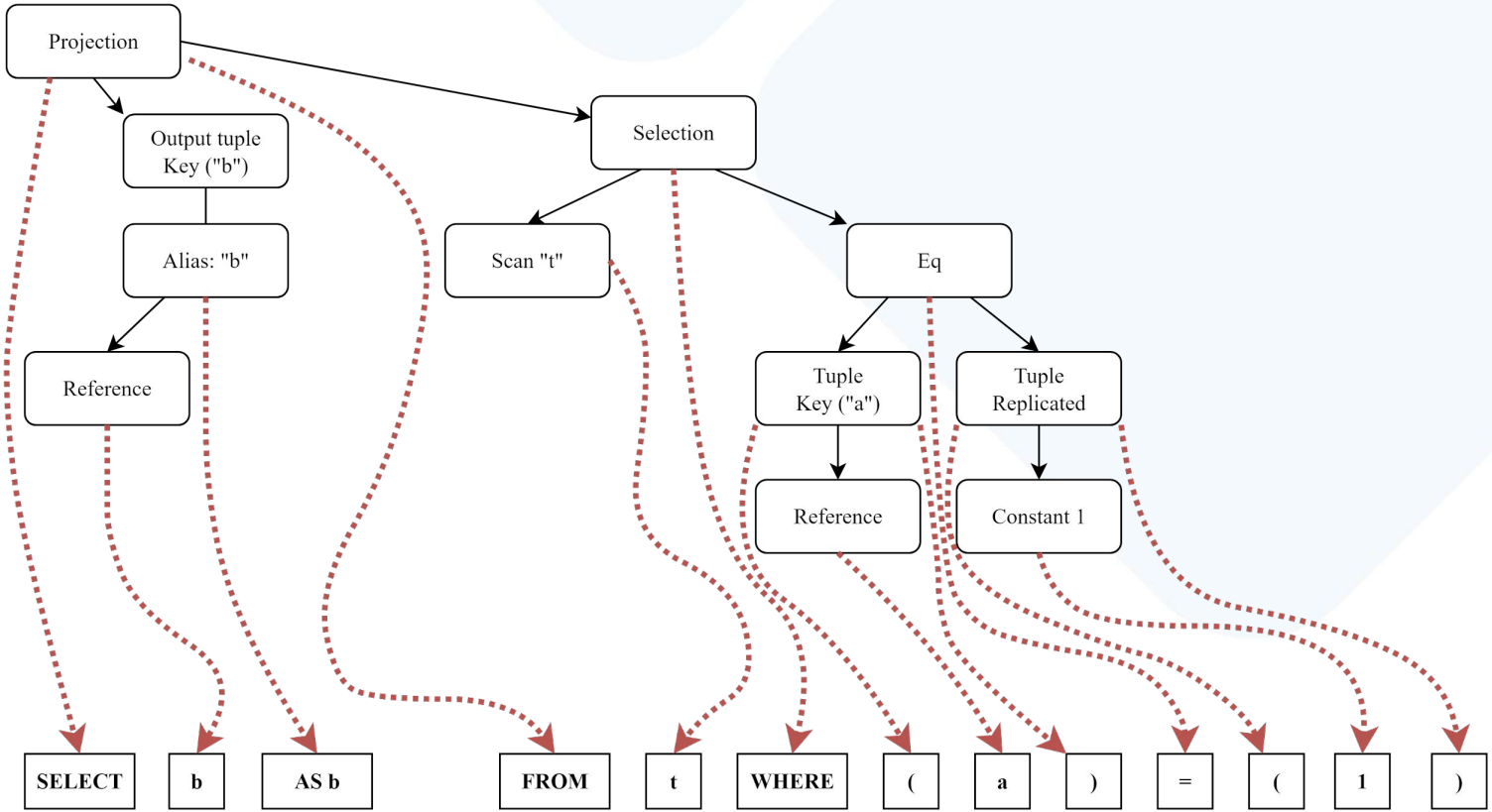
Storage cache

PICODATA



IR to SQL

PICODATA



Tarantool SQL: VDBE

PICODATA

```
box.execute([[
        explain select "b" from "t"
        where "a" = 1
]])
```

idx	opcode	p1	p2	p3	p4	p5
0	Init	0	1	0		00
1	IteratorOpen	1	0	0	space<name=t>	02
2	Integer	1	1	0		00
3	IsNull	1	5	0		00
4	MustBeInt	1	10	0		00
5	SeekGE	1	10	1	1	00
6	IdxGT	1	10	1	1	00
7	Column	1	1	2		00
8	ResultRow	2	1	0		00
9	Next	1	6	0		00
10	Halt	0	0	0		00

Example 2

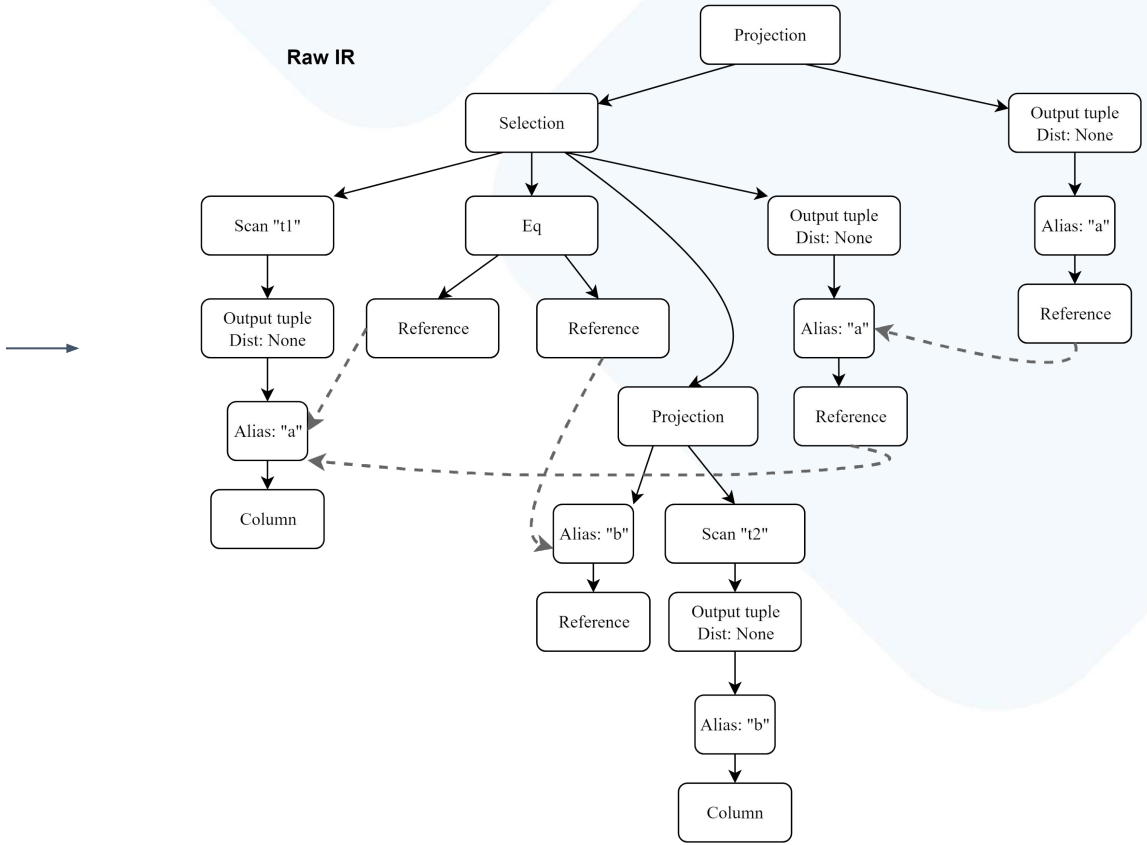
Query

PICODATA

```
select a from t1 where a in (select b from t2)
```

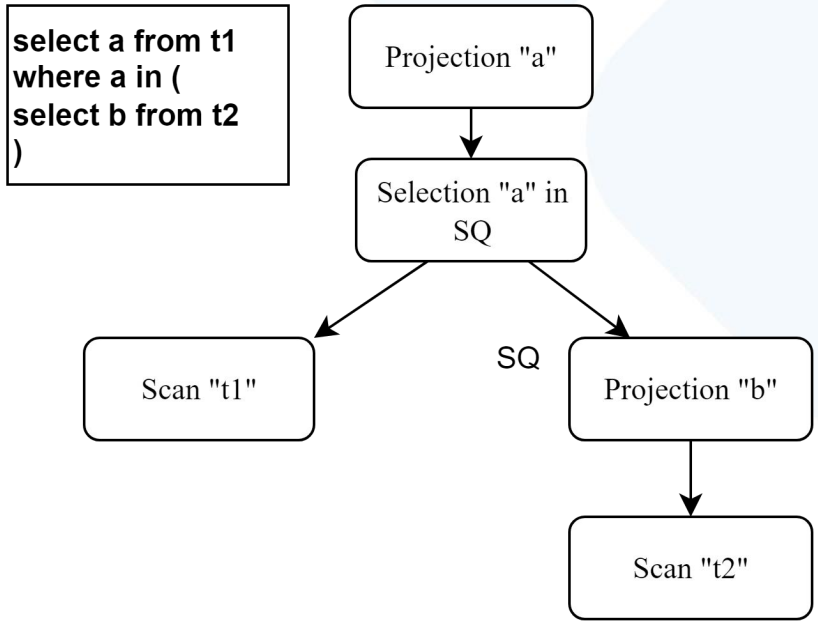
Raw IR

select a from t1
where a in
(select b from t2)



Simplified raw IR

PICODATA



Case 1: t1(a), t2(b)

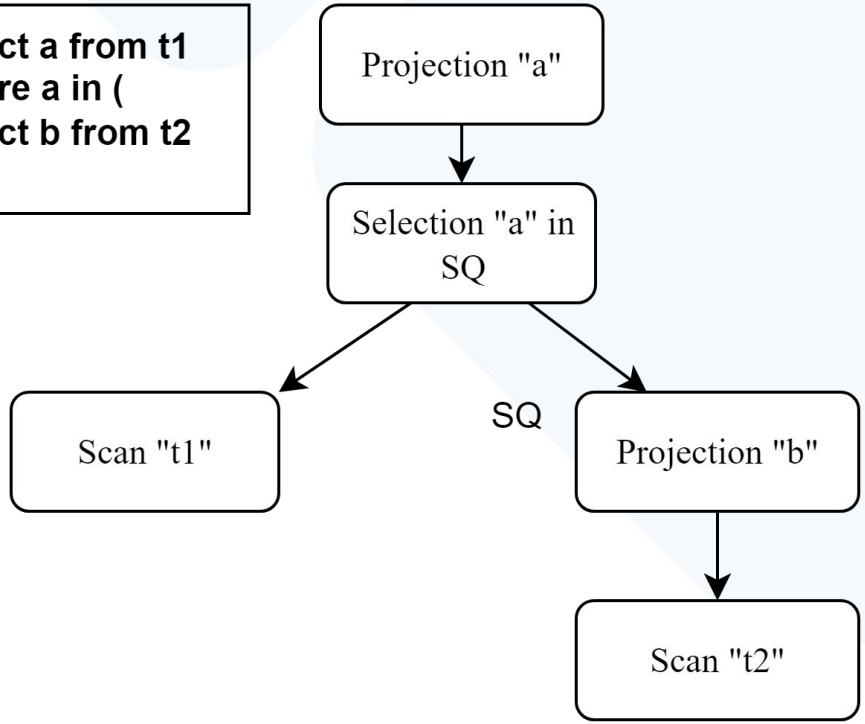
Initial conditions

PICODATA

t1 sharded by "a"

t2 sharded by "b"

```
select a from t1
where a in (
select b from t2
)
```



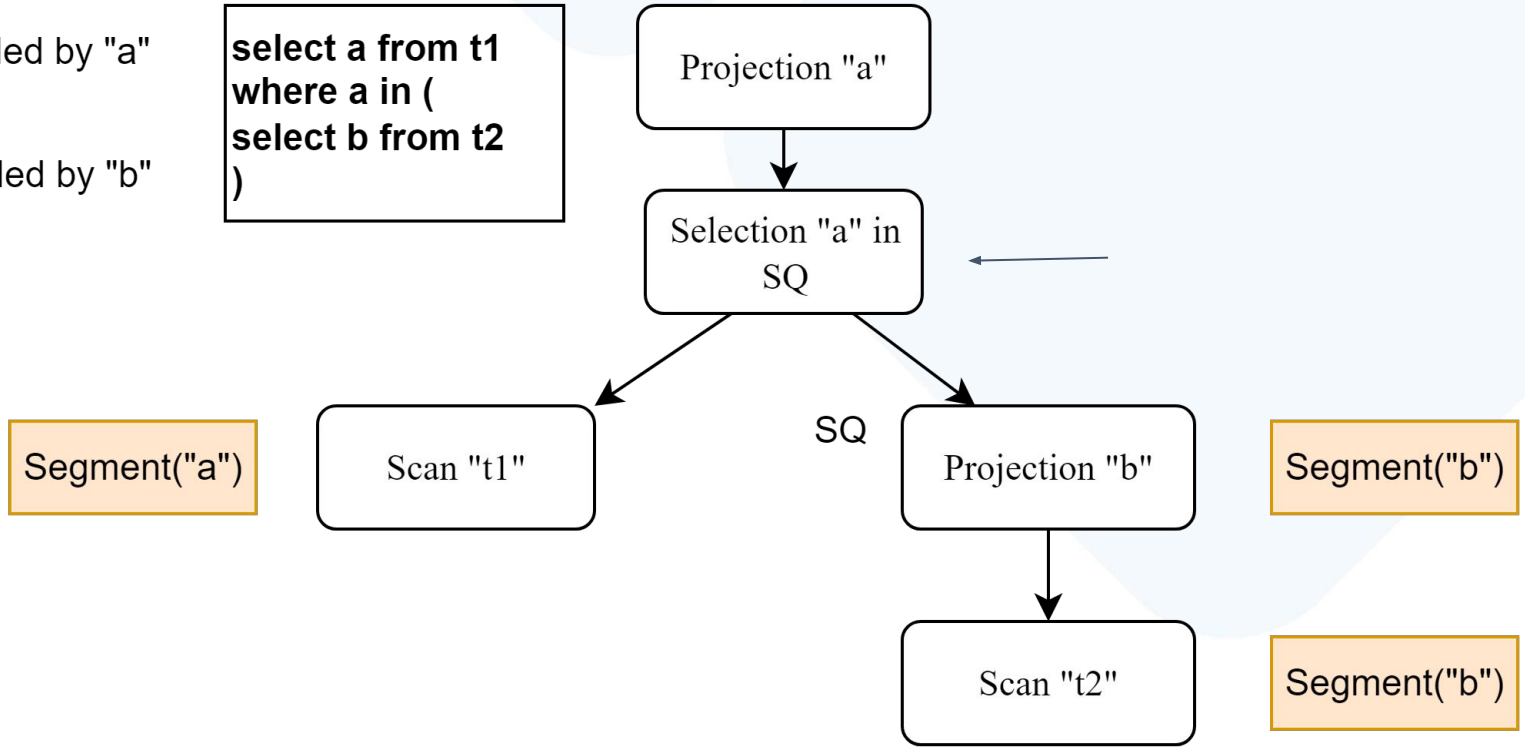
Distribution conflict?

PICODATA

t1 sharded by "a"

t2 sharded by "b"

```
select a from t1
where a in (
select b from t2
)
```



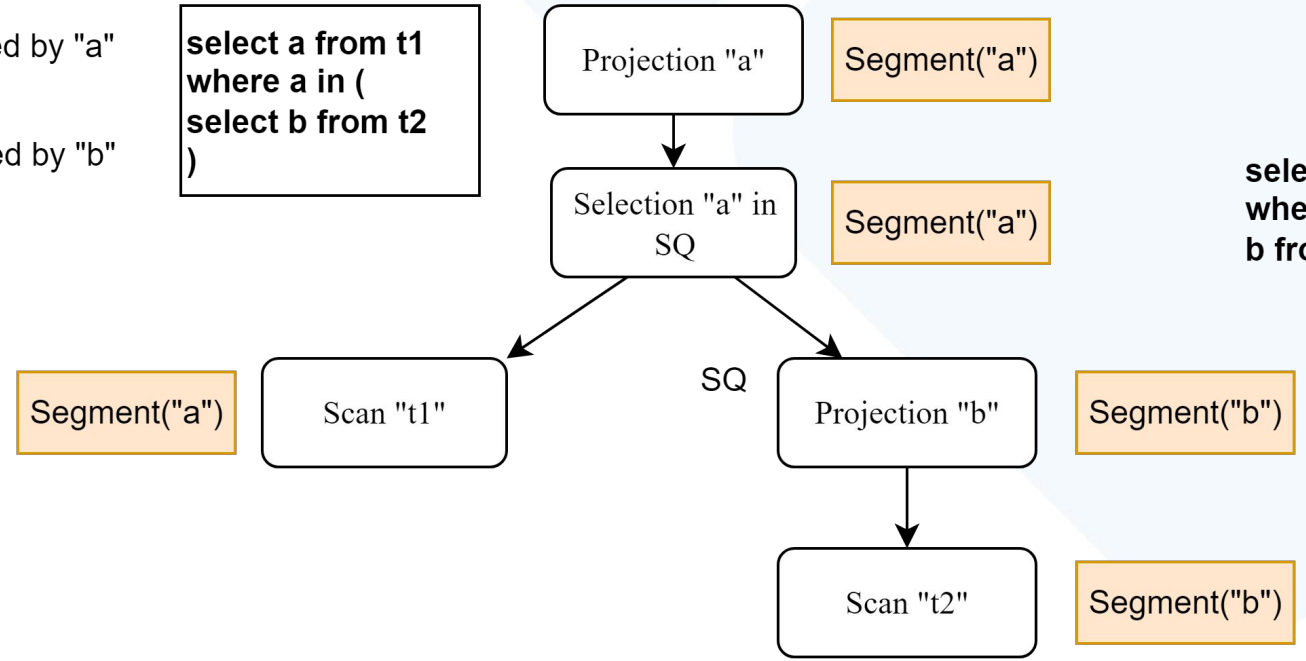
Answer

PICODATA

t1 sharded by "a"

t2 sharded by "b"

select a from t1 where a in (select b from t2)



select a from t1 where a in (select b from t2)

Case 2: different filter

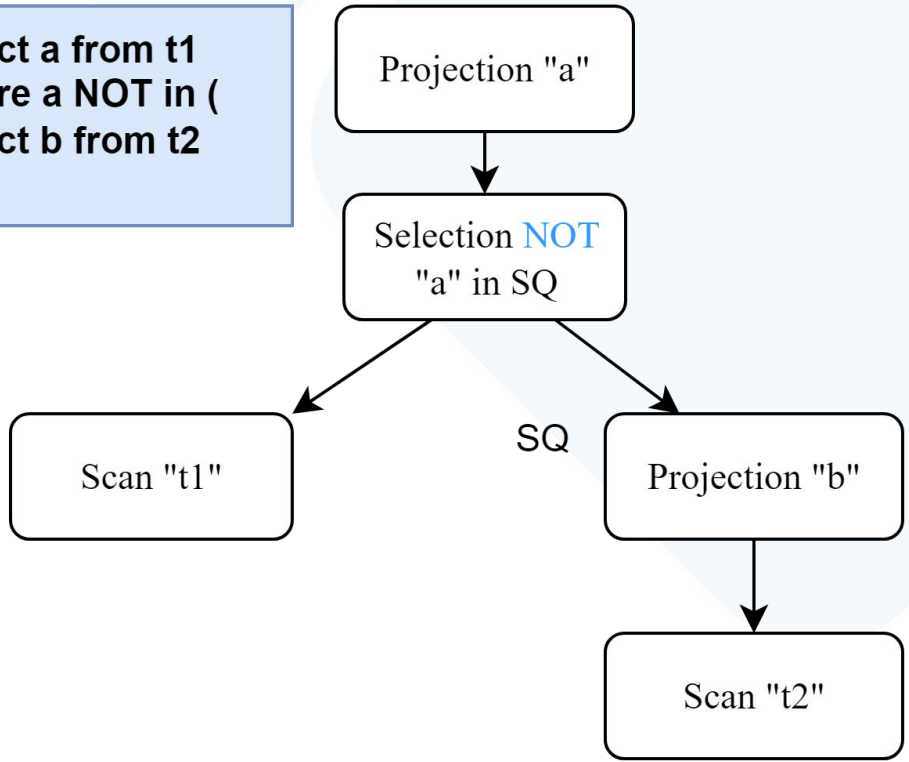
Different filter

PICODATA

t1 sharded by "a"

t2 sharded by "b"

```
select a from t1
where a NOT in (
select b from t2
)
```



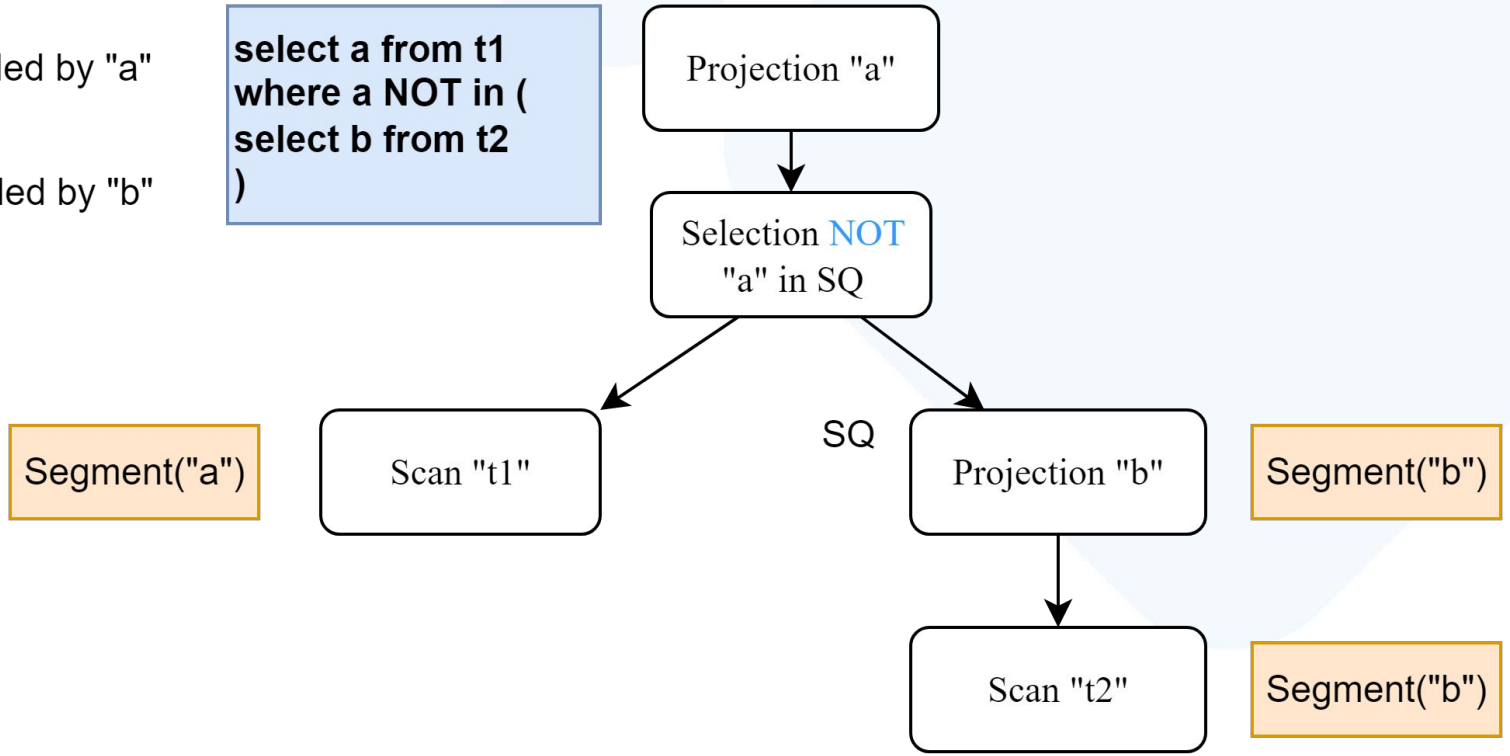
Different filter

PICODATA

t1 sharded by "a"

t2 sharded by "b"

```
select a from t1
where a NOT in (
select b from t2
)
```



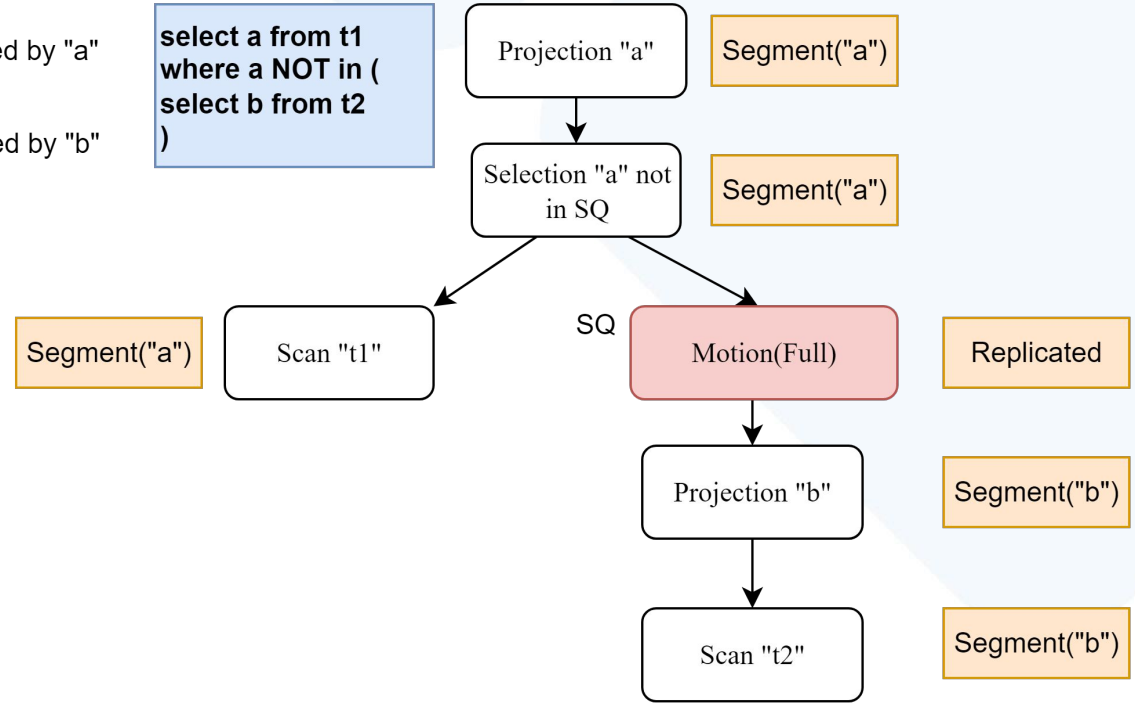
Motion with Full policy

PICODATA

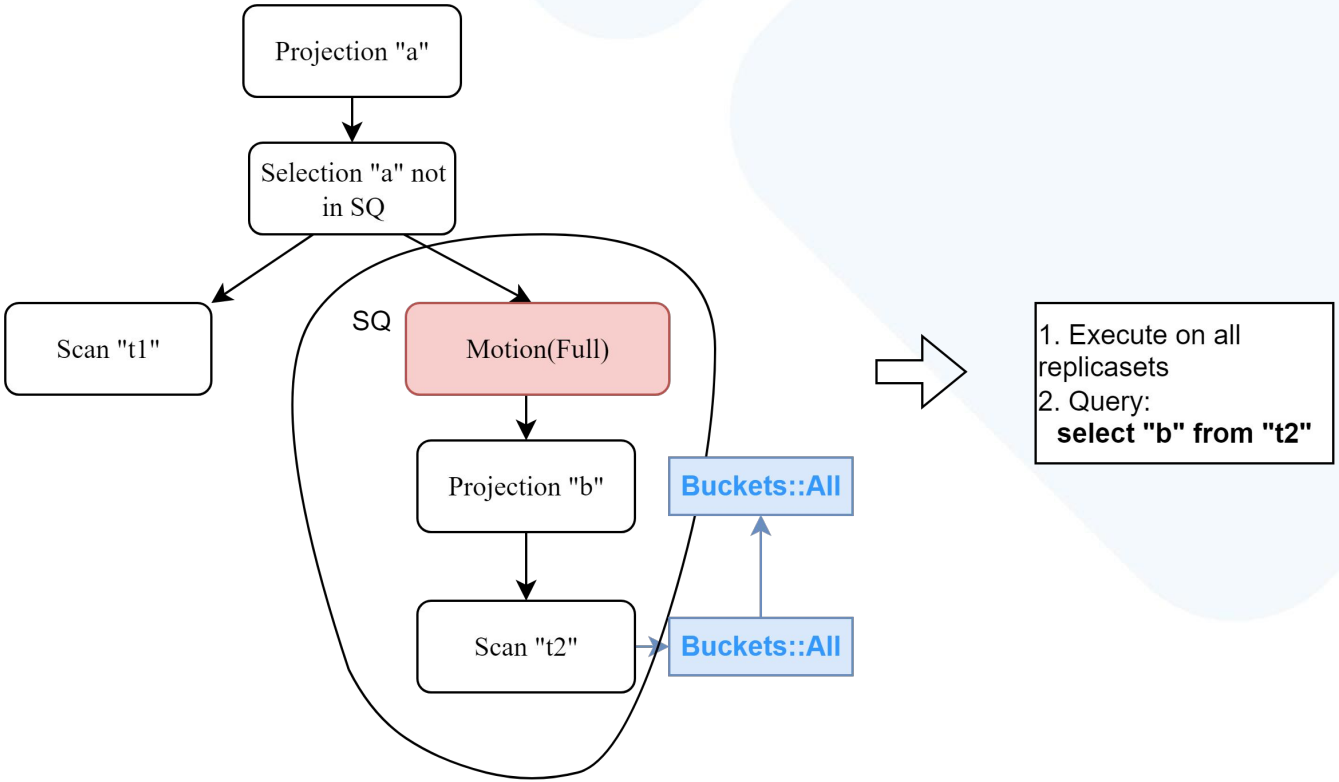
t1 sharded by "a"

t2 sharded by "b"

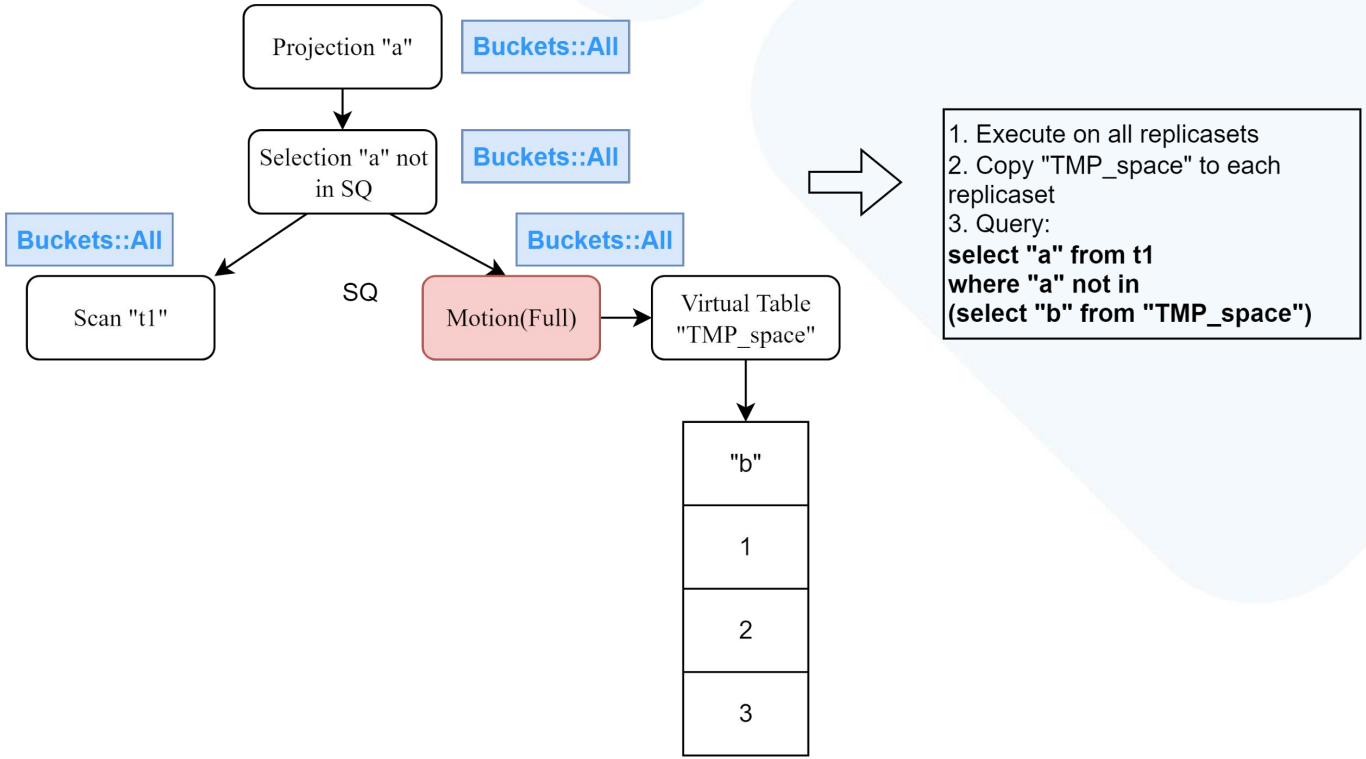
```
select a from t1
where a NOT in (
select b from t2
)
```



Motion with Full policy

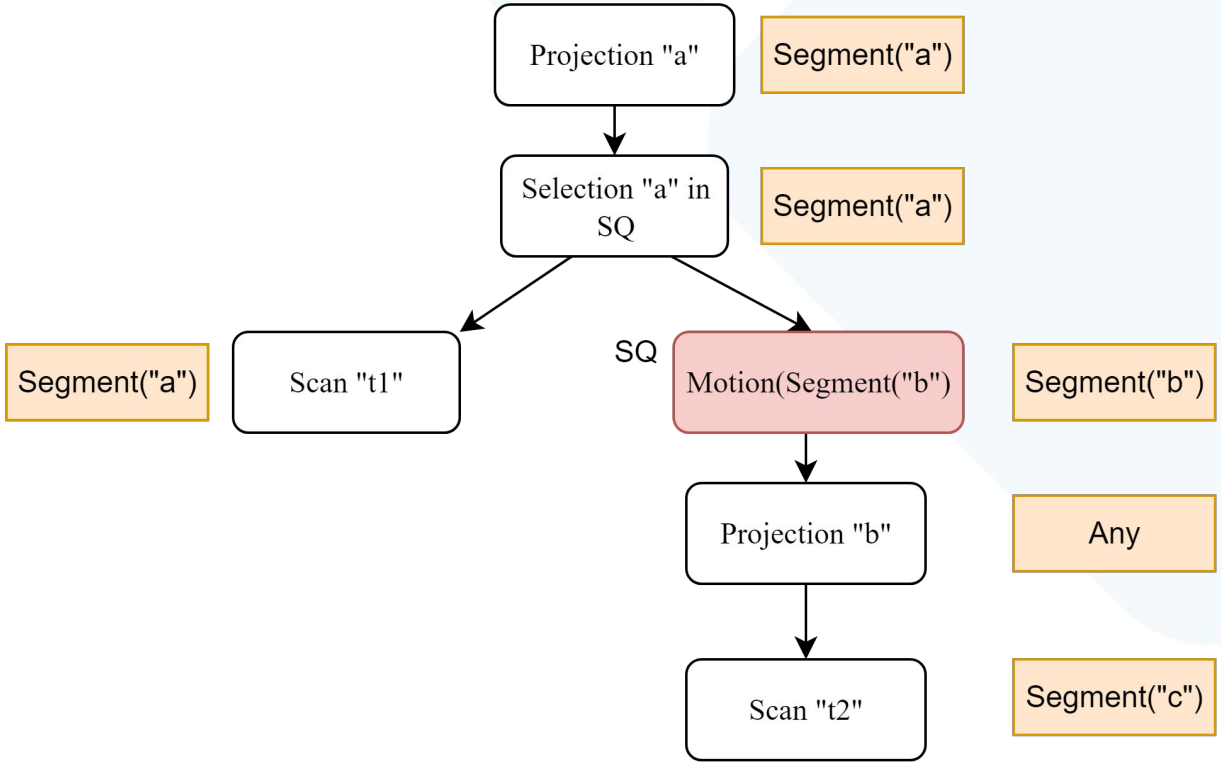


Motion with Full policy



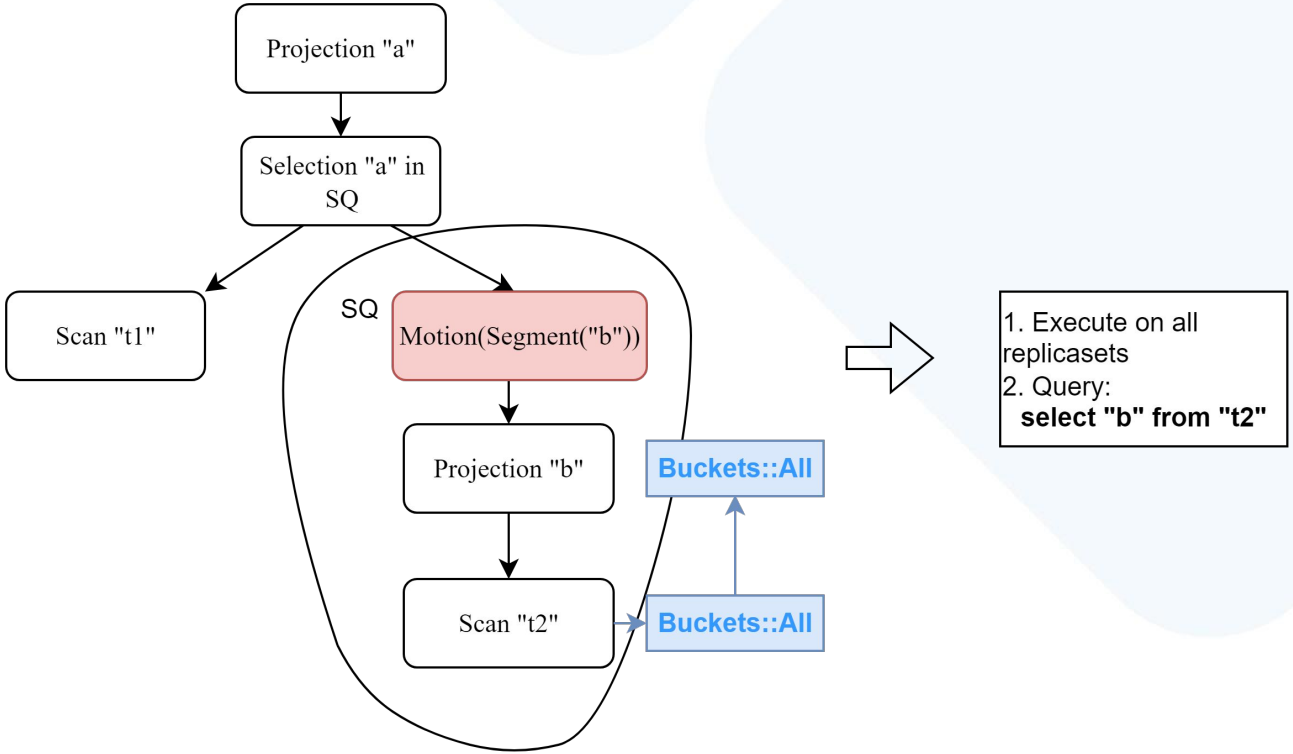
Case 3: t1(a), t2(c)

Motion with Segment policy

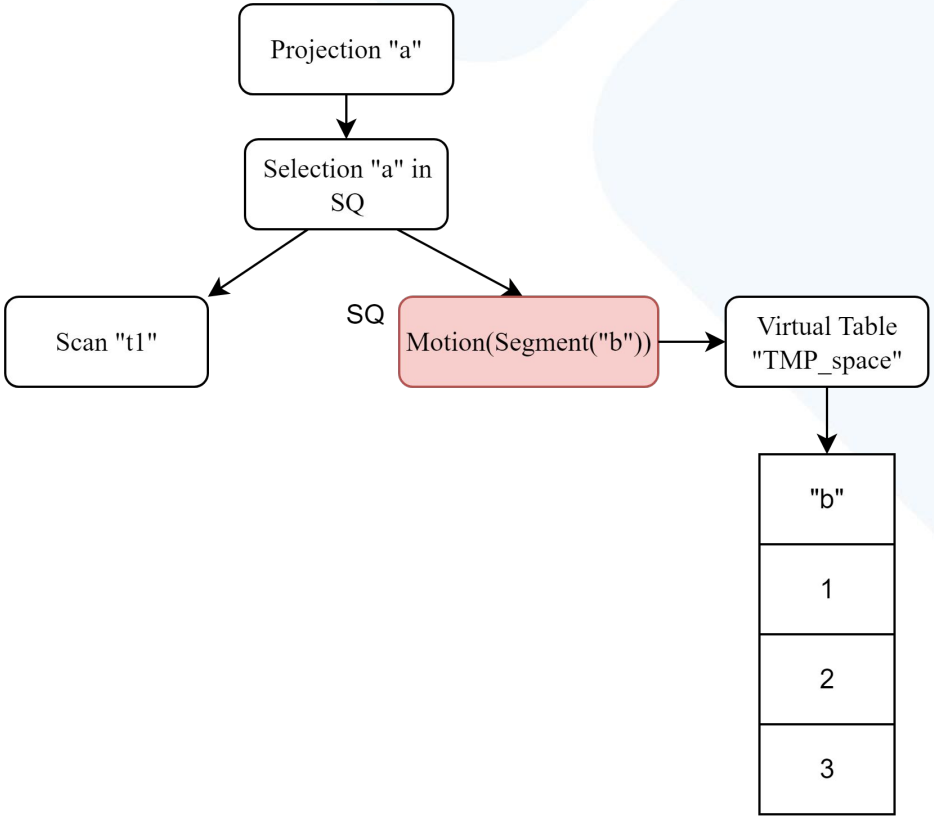


Motion(Segment): execution

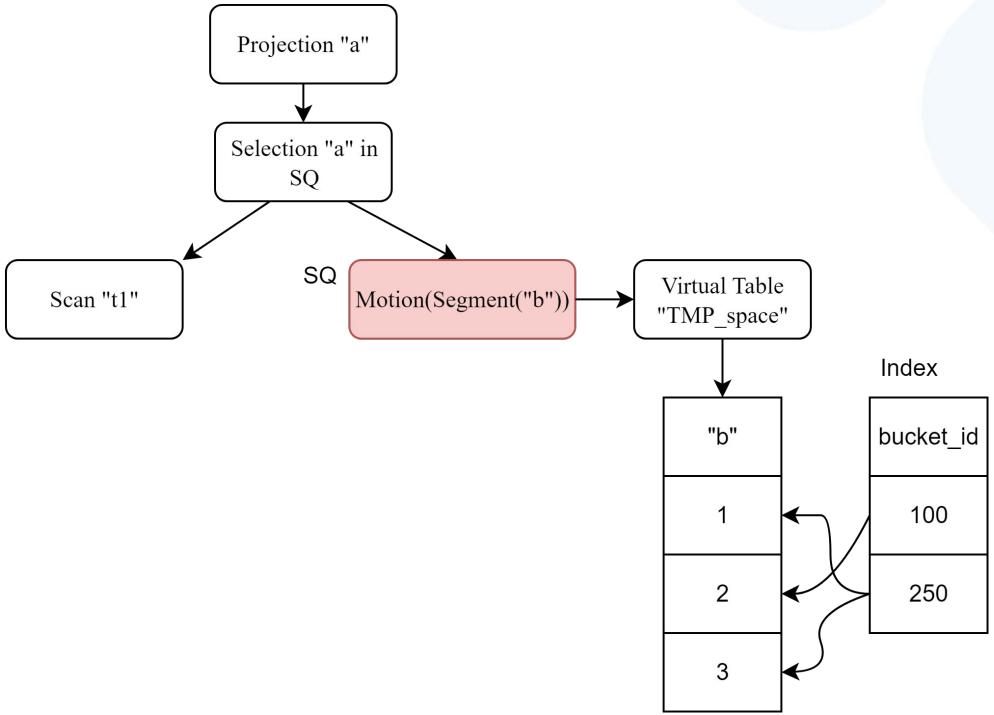
PICODATA



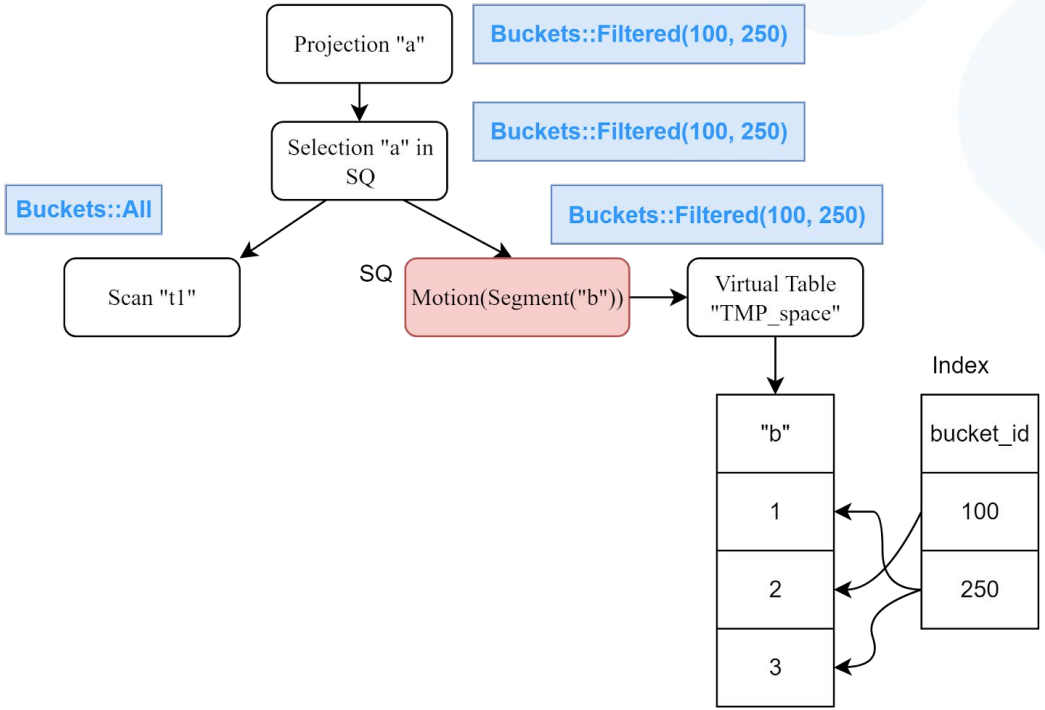
Motion(Segment): Linking

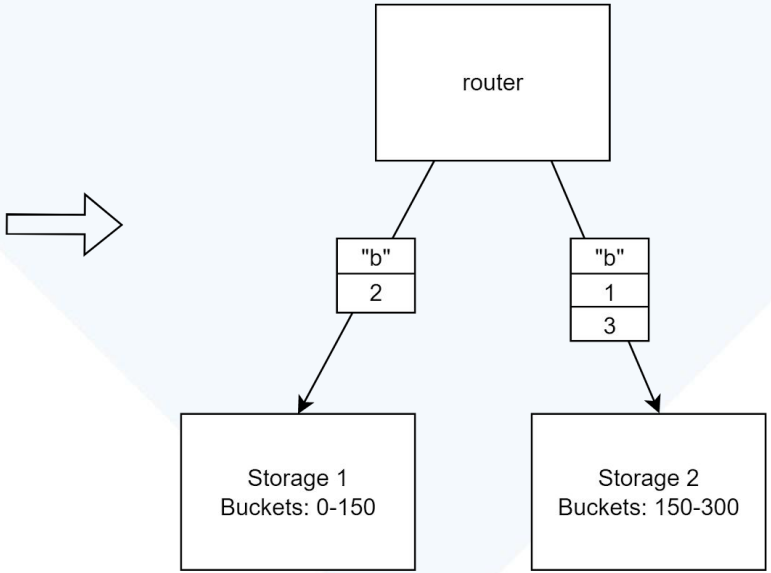


Motion(Segment): Index



Motion(Segment): sending





DML query

DML

PICODATA

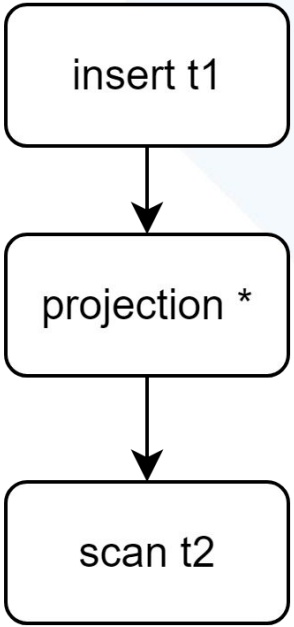
```
insert into "t1"
select * from "t2"
```

t1

a	b	bucket_id

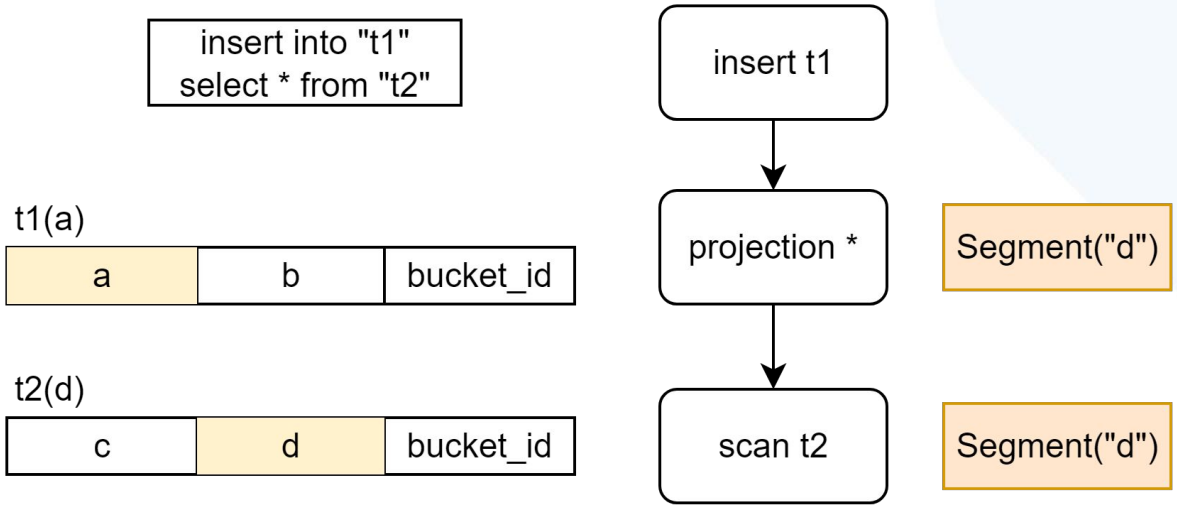
t2

c	d	bucket_id



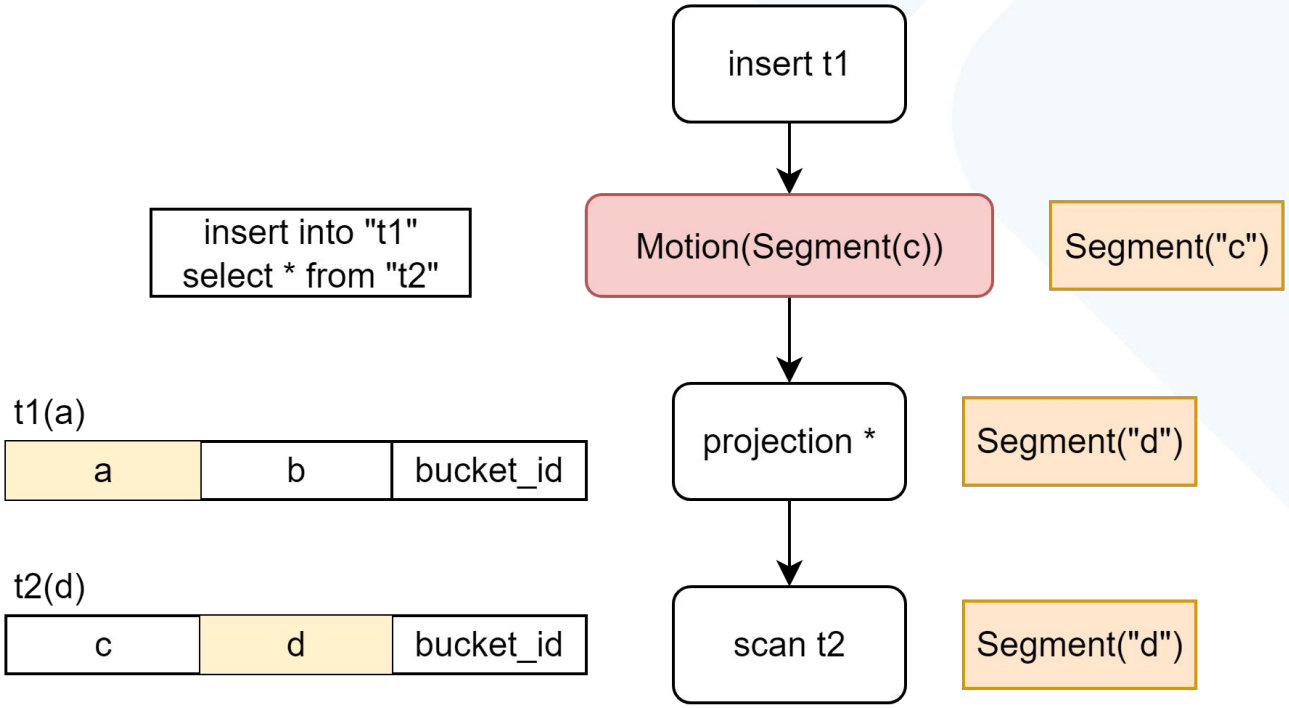
Do we need a motion?

PICODATA



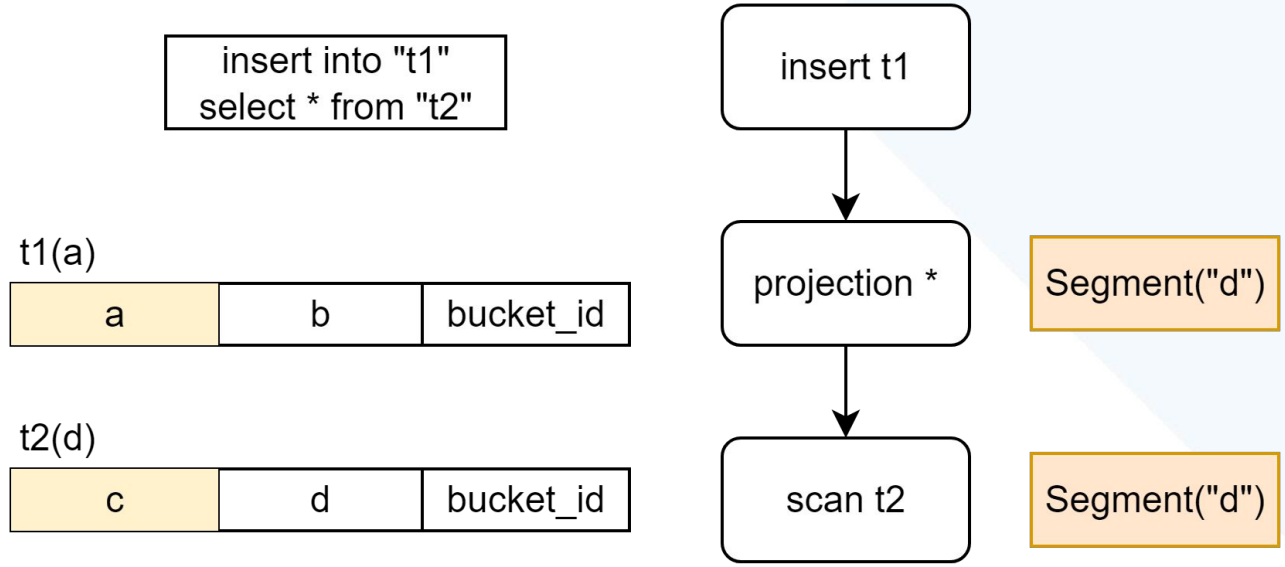
Motion with Segment policy

PICODATA



Motion with Segment policy

PICODATA

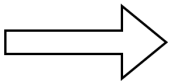


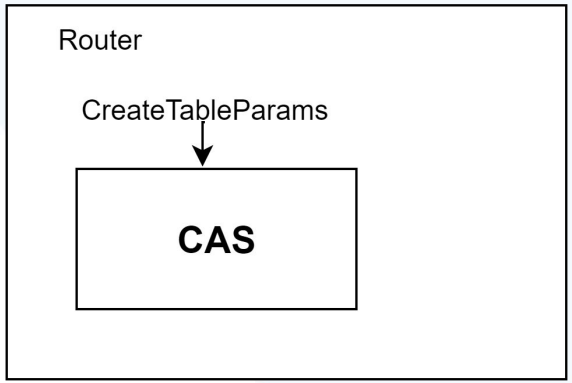
DDL query

DDL

PICODATA

```
create table "t" (
        "a" integer not null,
        "b" int not null,
        primary key ("b","a"))
using memtx
distributed by ("a", "b")
option (timeout = 3)
```





Thanks for your attention!

Glossary

PICODATA

1. IR (intermediate representation) - plan of a query in picosql library.
2. Executor - module responsible for plan (IR) execution in picosql
3. VDBE - internal representation of sql query in Tarantool
4. Virtual Table - temporary table during distributed query execution
5. Router - node where IR is built and where Executor works.
6. Storage - node where data is stored and where execution happens.