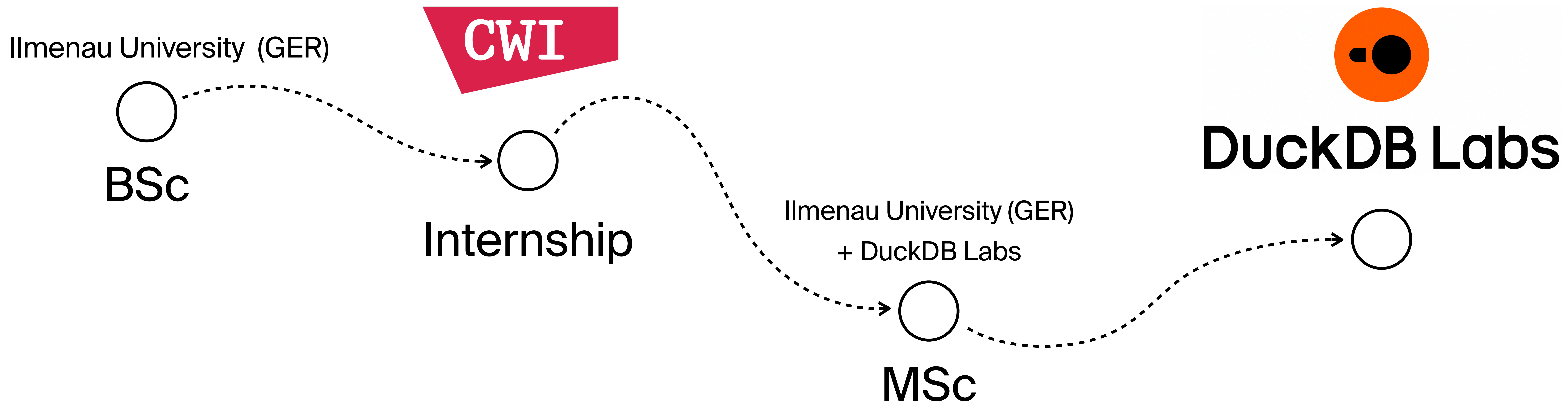
2025-09-17

Indexes are (not) all you need: DuckDB pitfalls and how to find them

Tania Bogatsch

DuckDB Labs


Ilmenau University (GER)
BSc
CWI
Internship
Ilmenau University (GER)
+ DuckDB Labs
MSc
DuckDB Labs

DuckDB pitfalls and how to find them

- DuckDB tries to choose sensible defaults
  - Check them out (docs — configuration)
- Indexes as a case study
  - Use profiling
  - Monitor DuckDB’s memory

tree-based

Indexes in DB systems

Indexes and transactional workloads

- Inherent to the highly transactional workloads of traditional DB systems
  - Queries are point lookups and single-tuple changes
  - Fixed set of queries
  - Fine-tuned system parameters

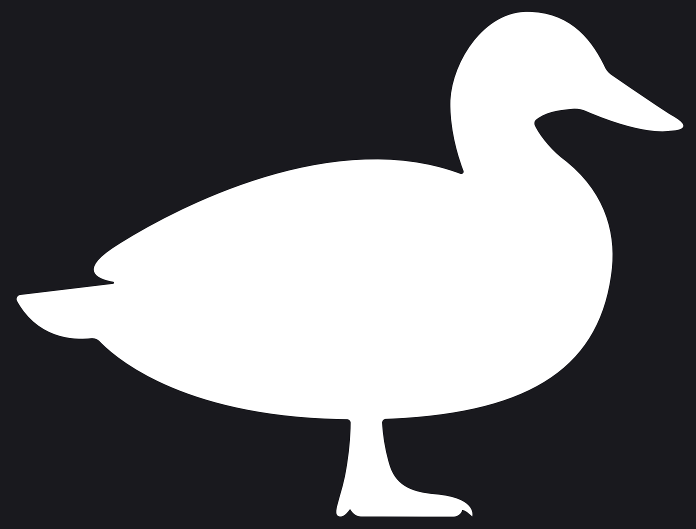
What about indexes in analytical workloads?

- Initially counter-intuitive
  - Large table scans to compute complex aggregates
    - Efficient scan performance
  - Ad hoc queries

Where do we still see and use indexes?

- Integrity constraints
- Filters
    - Zonemaps, etc. depend on clustering
    - The data set's size matters
        - Full scans to fetch a few tuples become costly
- Versatility ⚖️

Indexes in DuckDB

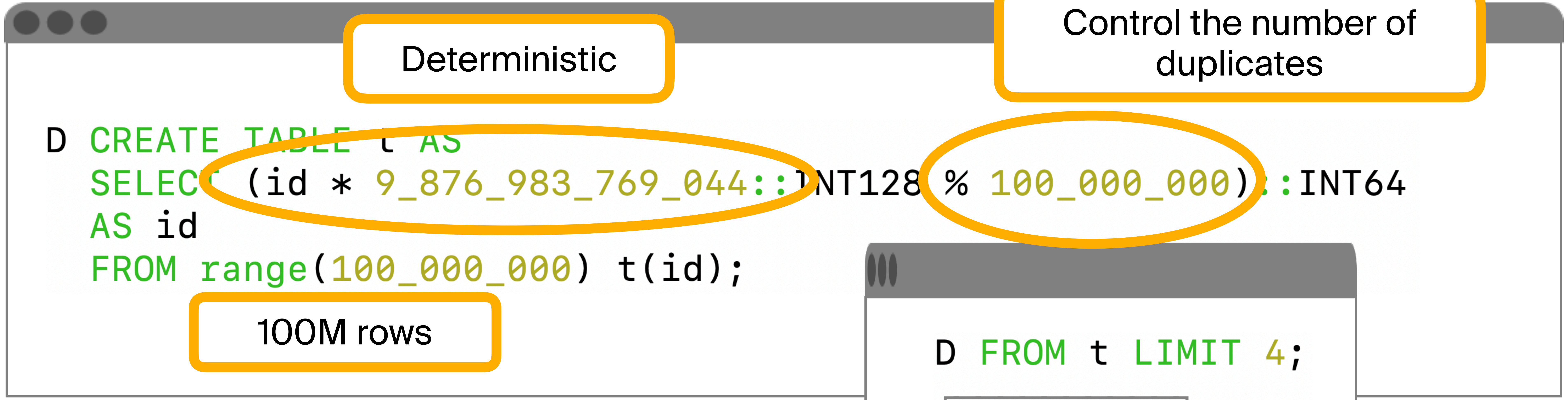


Indexes in DuckDB

- Integrity constraints
  - `PRIMARY KEY`
  - `FOREIGN KEY`
  - `UNIQUE`

- Explicit indexes
  - `CREATE [UNIQUE] INDEX name ON table (columns)`

Setup - our table

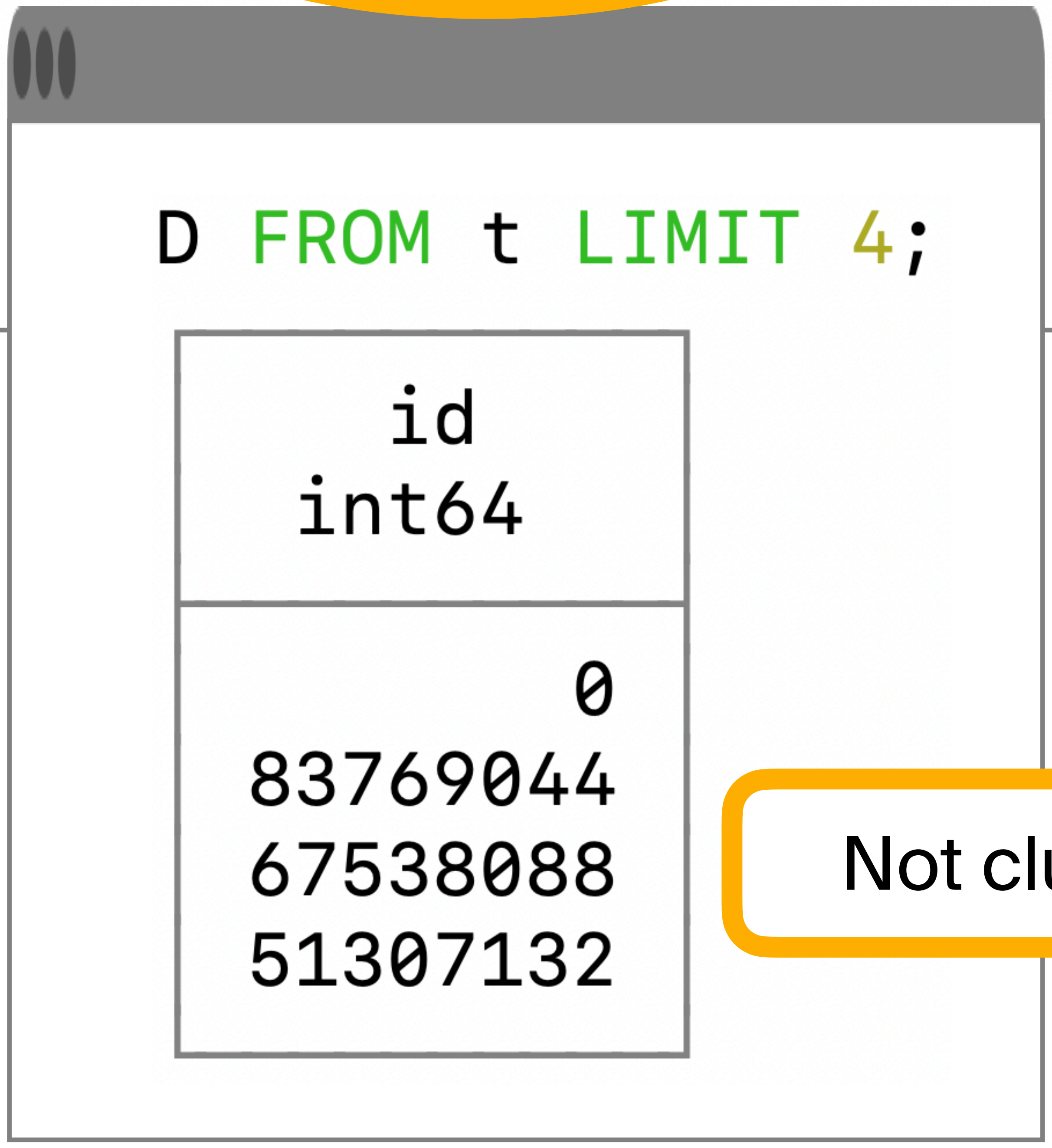


```
D CREATE TABLE t AS
  SELECT (id * 9_876_983_769_044::INT128 % 100_000_000)::INT64
  AS id
  FROM range(100_000_000) t(id);
```

Deterministic

Control the number of duplicates

100M rows



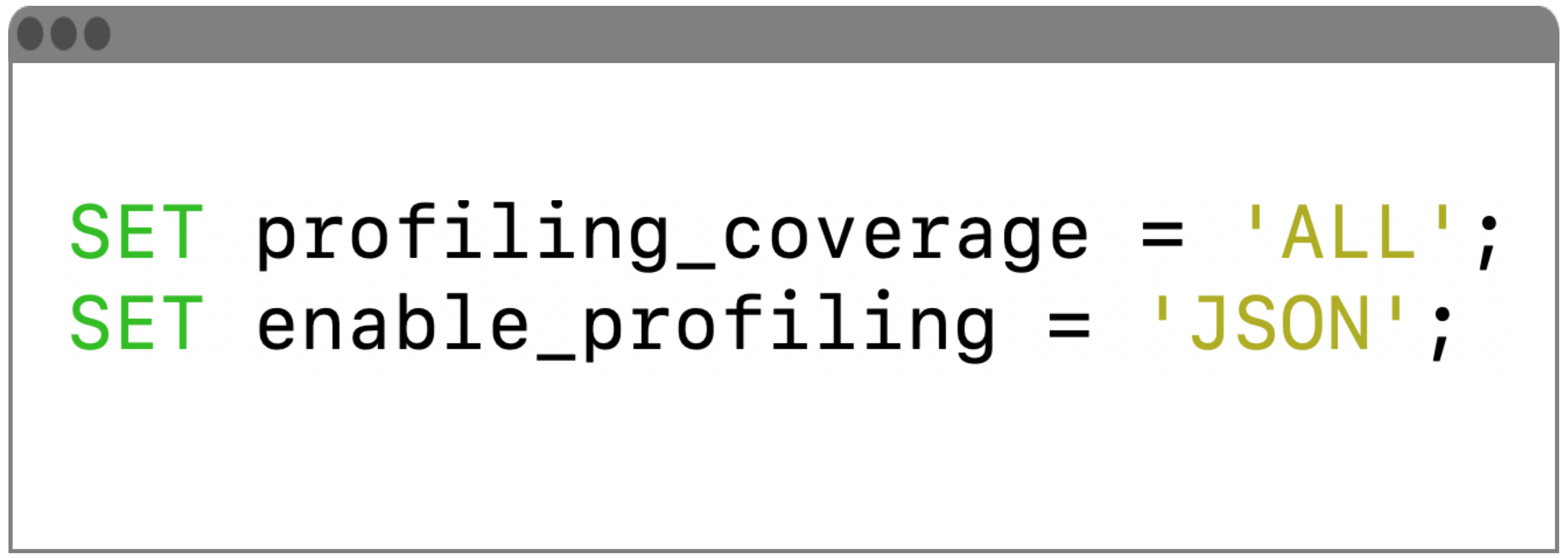
```
D FROM t LIMIT 4;
┌──────────┐
│    id    │
│  int64   │
├──────────┤
│        0 │
│ 83769044 │
│ 67538088 │
│ 51307132 │
└──────────┘
```

Not clustered

Setup - our index

```
CREATE INDEX idx ON t(id);
```

Enable profiling



```sql
SET profiling_coverage = 'ALL';
SET enable_profiling = 'JSON';
```

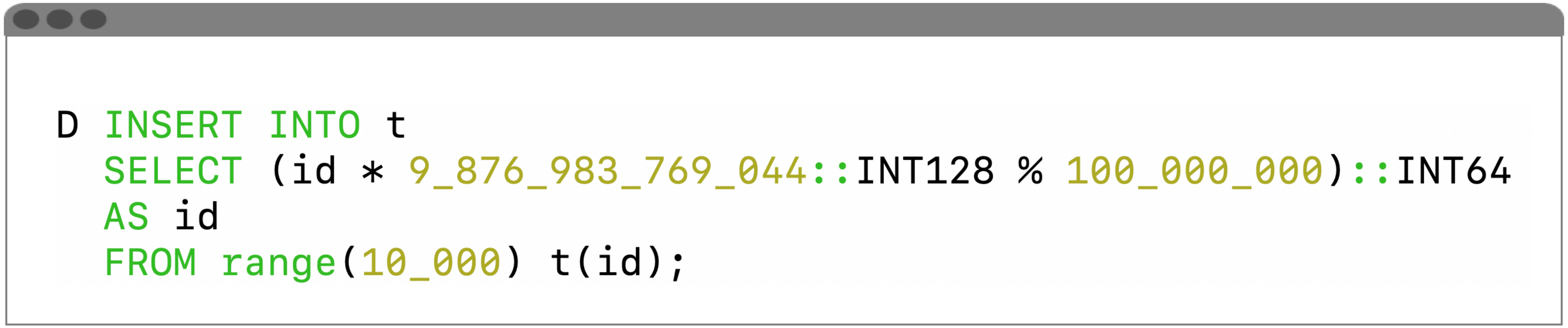
Check it out in the docs – profiling!

- Top-level metrics
  - Latency
  - Peak buffer memory
  - ...
- Per-operator metrics
  - Scan or probe
  - Operator timings
  - ...

1. Index maintenance

- Performance of 10k integer bulk append
  - No indexes
  - vs. three indexes on `t(id)`

Index maintenance



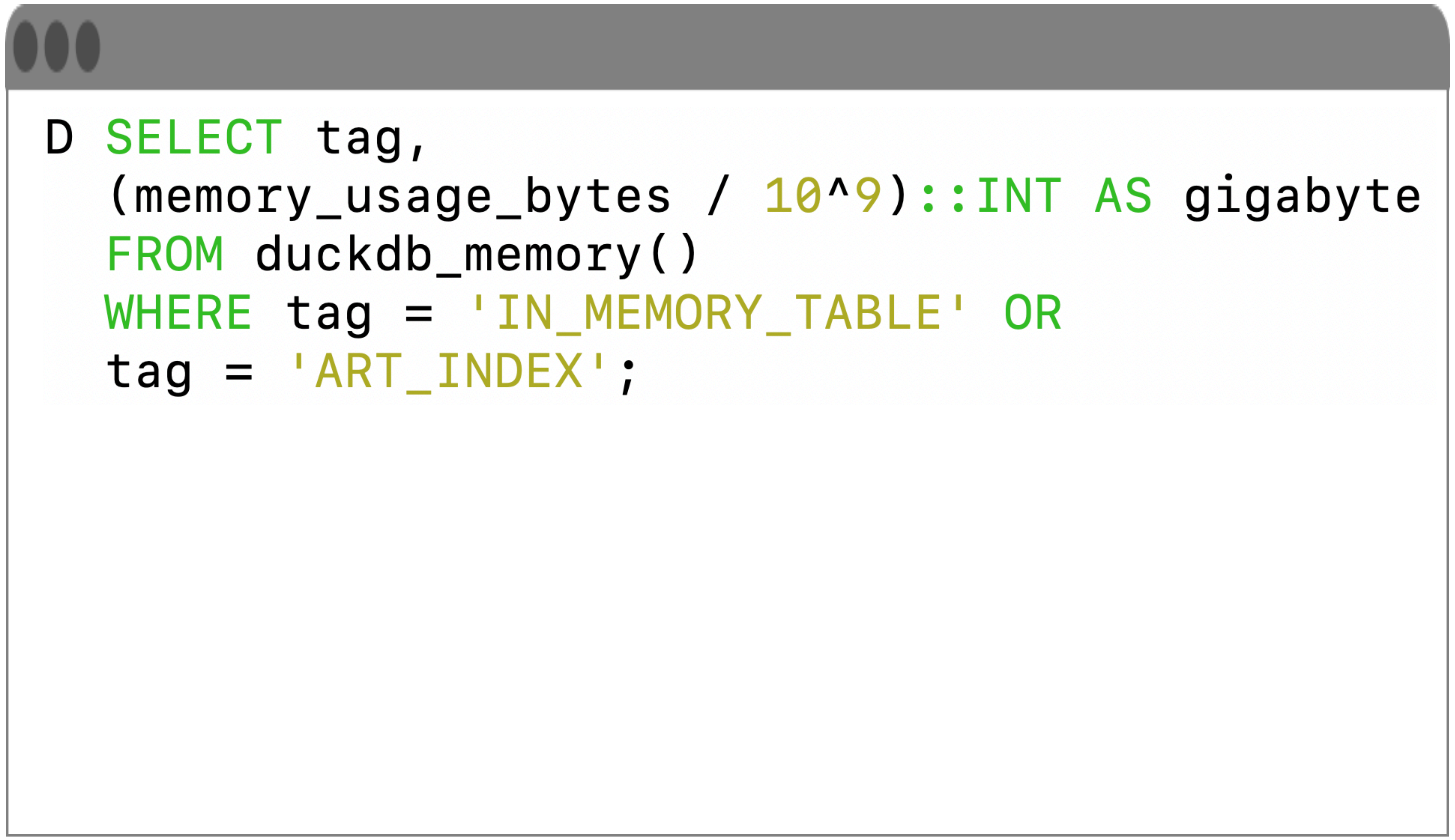
```
D INSERT INTO t
  SELECT (id * 9_876_983_769_044::INT128 % 100_000_000)::INT64
  AS id
  FROM range(10_000) t(id);
```

	No indexes	Three indexes
Latency	7 ms	400 ms

2. Index memory

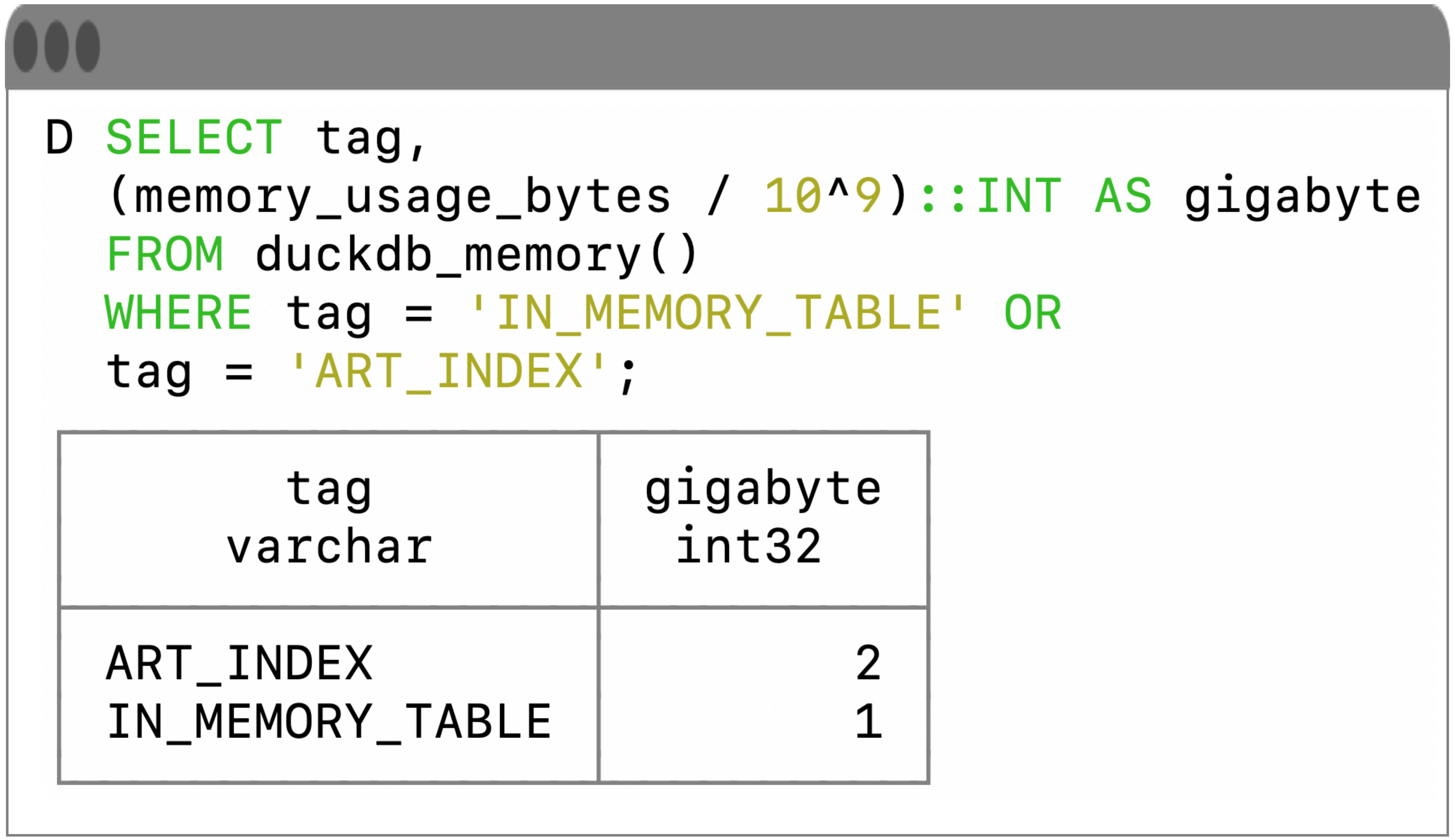
- How much memory does our index use?

2. Index memory: `duckdb_memory()`



```
D SELECT tag,
  (memory_usage_bytes / 10^9)::INT AS gigabyte
  FROM duckdb_memory()
  WHERE tag = 'IN_MEMORY_TABLE' OR
  tag = 'ART_INDEX';
```

2. Index memory: `duckdb_memory()`



```
D SELECT tag,
  (memory_usage_bytes / 10^9)::INT AS gigabyte
  FROM duckdb_memory()
  WHERE tag = 'IN_MEMORY_TABLE' OR
  tag = 'ART_INDEX';
┌─────────────────┬──────────┐
│       tag       │ gigabyte │
│     varchar     │  int32   │
├─────────────────┼──────────┤
│ ART_INDEX       │        2 │
│ IN_MEMORY_TABLE │        1 │
└─────────────────┴──────────┘
```

2. Index memory

- Index contains
  - Column data, row IDs, index metadata
- Memory not evicted yet
  - Planned for v1.6
  - With eviction
    - Active indexes create memory pressure

3. Index scans

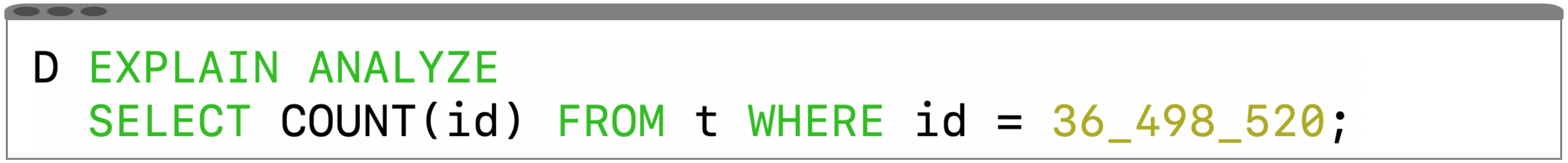
- Am I using an index scan?

`index_scan_max_count` (default 2048)

`index_scan_percentage` (default 0.001)

3. Index scans

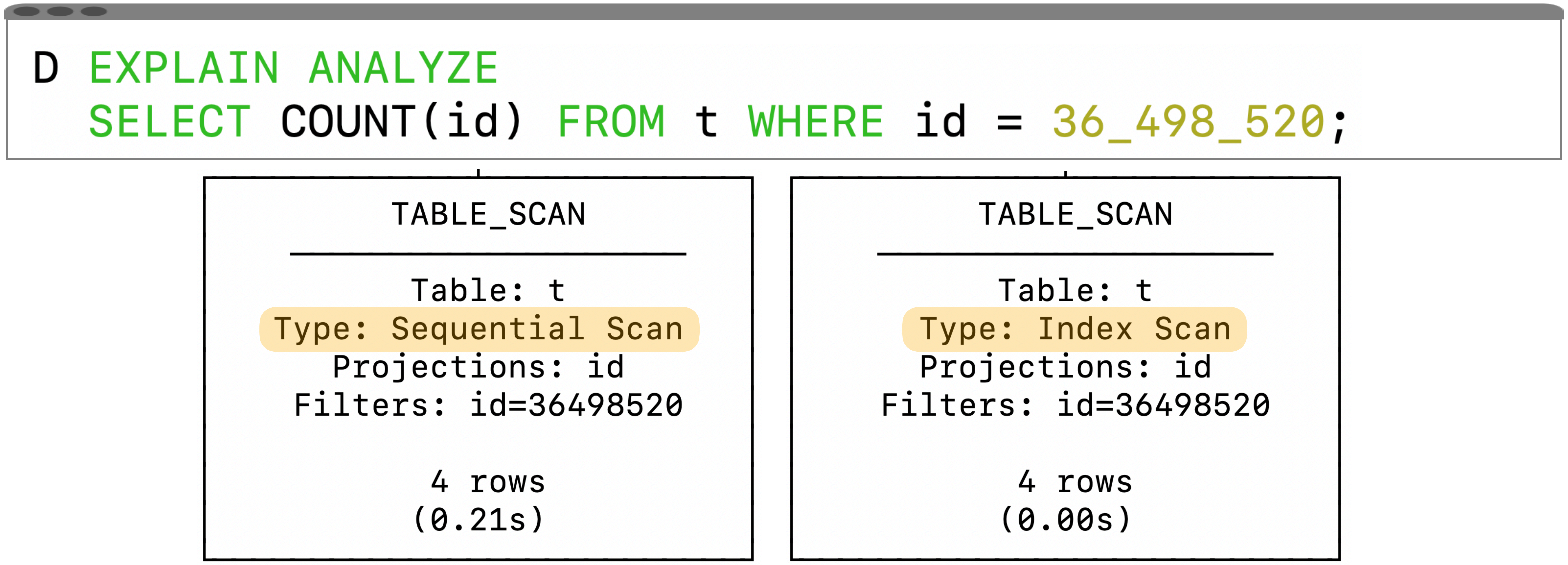
- Am I using an index scan?



```
D EXPLAIN ANALYZE
  SELECT COUNT(id) FROM t WHERE id = 36_498_520;
```

3. Index scans

- Am I using an index scan?



```
D EXPLAIN ANALYZE
  SELECT COUNT(id) FROM t WHERE id = 36_498_520;
```

```
        TABLE_SCAN
  ─────────────────────────
         Table: t
  Type: Sequential Scan
      Projections: id
   Filters: id=36498520

          4 rows
          (0.21s)
```

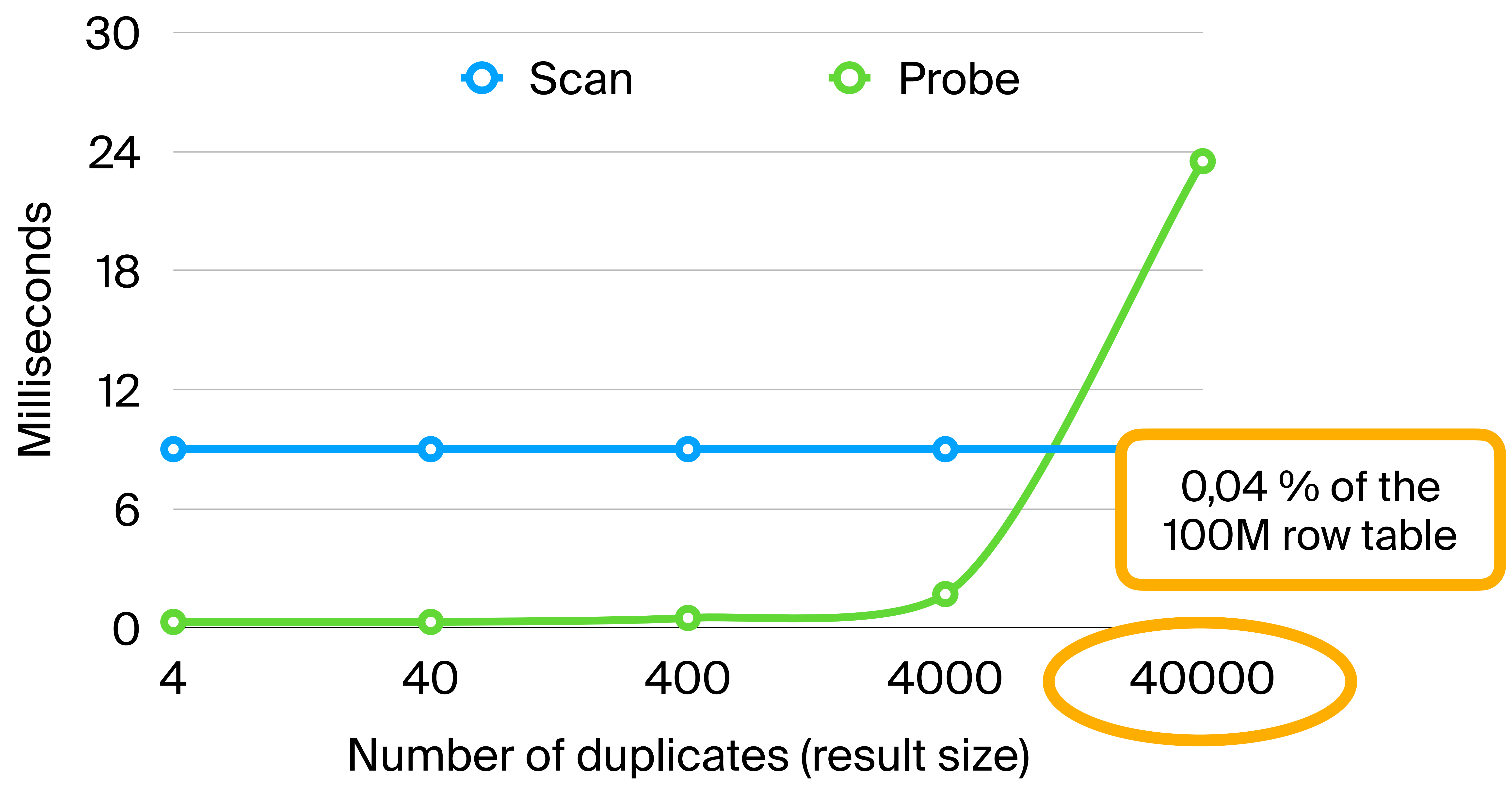
```
        TABLE_SCAN
  ─────────────────────────
         Table: t
     Type: Index Scan
      Projections: id
   Filters: id=36498520

          4 rows
          (0.00s)
```

3. Index scans

- How much faster is my index scan?

3. Index scans



Final thoughts

- Profile and benchmark your system
  - No rule of thumb
- Set up observability, if you have a pipeline
  - Profile relevant metrics
  - Monitor memory tags
- Check out
  - Our performance guide
  - DuckDB configuration