Introduction to Clickhouse & How to use it.

We have recently moved to Clickhouse to handle Analytics. This document give brief overview about it.

Architecture Fundamentals

PostgreSQL (Row-Based)

- Stores data row by row
- Optimized for OLTP (transactional workloads)
- Excellent for updates, deletes, and complex transactions
- Traditional B-tree indexes on every row

ClickHouse (Column-Based)

- Stores data column by column
- Optimized for OLAP (analytical workloads)
- Excellent for large scans and aggregations
- Sparse indexes (one entry per 8,192 rows by default)
- Designed for **append-only** workloads with **immutable** data

It is very fast for aggregations.

e.g Let's find total calls done on hotline in 20 Days of October.

This query takes 16 seconds on Jobfinder.

```
select count(1) from recruiter_call_logs where
"startedAt">'2025-10-01'
and "startedAt"<'2025-10-20'

```

Similiar query takes 1 second on Clickhouse.

```
select count(1) from jobfinder.recruiter_call_logs where
"startedAt">'2025-10-01'
and "startedAt"<'2025-10-20'
```

- It warehouse, you will find data from both Jobfinder, Warehouse_RDS present. You can select schema in Postgres databases you can select Different schemas and join among them. e.g
  - You can use a single query to process data over both Warehouse RDS and jobfinder tables.
- It is very fast in analytics.
- There is a lag of 1 minute between changes in your source postgres databases and Clickhouse.

How to write better queries in Clickhouse.

NEVER Use SELECT *

ClickHouse reads data column by column. Every column you select adds I/O overhead.

❌ Don't Do This:

sql

```sql
SELECT *
FROM jobfinder.recruiter_call_logs
WHERE "startedAt"<'2025-10-20'
```

✅ Do This Instead:

sql

```sql
SELECT "callerNumber", "receiverNumber", "duration"
FROM jobfinder.recruiter_call_logs
WHERE "startedAt"<'2025-10-20'
```

Use PREWHERE Instead of WHERE

WHERE: Reads ALL columns from disk, then filters

PREWHERE: Reads filter columns first, then only reads remaining columns for matching rows

```sql
SELECT "callerNumber", "receiverNumber", "duration"
FROM jobfinder.recruiter_call_logs
PREWHERE "startedAt"<'2025-10-20'
```

Use uniqExact(x) to count total unique values instead of COUNT(DISTINCT x)

```sql
select uniqExact( distinct("id")) from jobfinder.recruiter_call_logs
prewhere
"startedAt">'2025-10-19'
and "startedAt"<'2025-10-20'
```

Rules for joins.

- Always ensure the **smallest table is on the right-hand side of the Join.**
  - The right table is loaded into memory, so keep it small.
- Filter BEFORE joining
- Use IN Instead of INNER JOIN (When Possible)
  - For simple lookups, `IN` clauses can be faster than JOINs

```
-- less efficient as join created intermediate rows.
-- takes 11 seconds.
select uniqExact(co."phone_number")
from jobfinder.candidates_olap co
join jobfinder.recruiter_call_logs rcl
on rcl."receiverNumber" = co."phone_number"
prewhere rcl."createdAt">'2025-01-01'
SETTINGS use_query_cache = 0


-- MORE EFFICIENT.
-- takes 4 seconds.
select uniqExact(co."phone_number")
from jobfinder.candidates_olap co
prewhere "phone_number" in (select "receiverNumber" from
jobfinder.recruiter_call_logs rcl where rcl."createdAt">'2025-01-01')
SETTINGS use_query_cache = 0
```

Use hasToken for word matching when.

- Searching for complete words/tokens
- String has natural delimiters (spaces, underscores, dots)
- You need "word boundary" matching

Use Like when.

- You need partial word matching
- Pattern is in the middle of a word
- You need complex pattern matching with `%` wildcards

This splits sentences into tokens by using delimiters and then find on them.

```
-- Don't use this.
select uniqExact( ("id")) from jobfinder.recruiter_call_logs
prewhere
"startedAt">'2025-01-01'
and"startedAt"<'2025-10-20'
and "tag" LIKE '%Service%'


-- Use this
select uniqExact( ("id")) from jobfinder.recruiter_call_logs
prewhere
"startedAt">'2025-01-01'
and"startedAt"<'2025-10-20'
and hasToken("tag", 'Service')
```

Operation	PostgreSQL Way	ClickHouse Way	Speed Improvement
Count distinct	`COUNT(DISTINCT id)`	`uniq(id)`	5-10x faster
Select all columns	`SELECT *` acceptable	Never use	5-20x faster
Filtering	`WHERE`	`PREWHERE`	2-5x faster
String search	`LIKE '%text%'`	`hasToken()` or `position()`	10-50x faster
Multiple JOINs	Optimized	Avoid if possible	Consider denormalization

PostgreSQL	ClickHouse	Description
`DATE_TRUNC('day', ts)`	`toStartOfDay(ts)`	Start of day
`DATE_TRUNC('month', ts)`	`toStartOfMonth(ts)`	Start of month
`DATE_TRUNC('hour', ts)`	`toStartOfHour(ts)`	Start of hour
`EXTRACT(YEAR FROM ts)`	`toYear(ts)`	Extract year
`EXTRACT(MONTH FROM ts)`	`toMonth(ts)`	Extract month
`EXTRACT(DAY FROM ts)`	`toDayOfMonth(ts)`	Extract day
`EXTRACT(DOW FROM ts)`	`toDayOfWeek(ts)`	Day of week (1-7)