QINGCLOUD

Continue to use ClickHouse as TSDB

邰翀

青云QingCloud 数据库研发工程师

Content

- Look back: Why we choose it
- Now: How we do
- Future: What we do

QINGCLOUD

Why we choose it

Why we choose it

商务分析

股票交易

系统监控

自动驾驶

Why we choose it

不断的汇总日成交量从
而制定商业规划

商务分析

股票交易

MySQL Table Locks

MySQL Temporary Objects

MySQL Sorts

MySQL Select Types

系统监控

自动驾驶

QINGCLOUD

Why we choose it

不断的汇总日成交量从
而制定商业规划

商务分析

不断收集市场变化信
息预测股价涨跌

股票交易

系统监控

自动驾驶

Why we choose it

不断的汇总日成交量从
而制定商业规划

商务分析

不断收集市场变化信
息预测股价涨跌

股票交易

不断收集CPU、
Memory等系统指标预
测系统未来趋势

系统监控

自动驾驶

QINGCLOUD

Why we choose it

不断的汇总日成交量从而制定商业规划

商务分析

不断收集市场变化信息预测股价涨跌

股票交易

不断收集CPU、Memory等系统指标预测系统未来趋势

系统监控

不断收集温度，坐标，方向，速度等指标，优化路线和驾驶方式

自动驾驶

QINGCLOUD

Why we choose it

- 上述业务数据特点:
  - (1) 数据多
  - (2) 旧数据趋于不变
  - (3) 新数据更有价值
  - (4) 数据总是随时间变化而不断变化

Why we choose it

- 解决方案
  - (1) Row-Orient Database
  - (2) Column-Orient Database
  - (3) Time-Series-Orient Database

INSERT INTO ...

Time	Name	Age	Humidity	HeartRate	Localtion	...	Temperature
2019/10/10/ 10:00:00	Tracy	22	45%	95	116.29860 40.13091	...	11
2019/10/10/ 10:00:00	Tom	26	45%	92	121.55687 31.31908	...	20
...	...	...	...	...	...	...	...
2019/10/11/ 11:00:01	Tracy	22	45%	90	116.30101 31.31673	...	11
2019/10/11/ 11:00:01	Tom	26	45%	96	121.54794 31.32318	...	21

Why we choose it

- Row-Orient Database

Time	Name	Age	Humidity	HeartRate	…	Temperature
2019/10/10/ 10:00:00	Tracy	22	45%	95	…	11
2019/10/10/ 10:00:00	Tom	26	45%	92	…	20
…	…	…	…	…	…	…
2019/10/11/ 11:00:01	Tracy	22	45%	90	…	11
2019/10/11/ 11:00:01	Tom	26	45%	96	…	21

Why we choose it

SELECT HeartRate FROM …
WHERE Time BETWEEN … AND … AND Name = “Tom”

Time	Name	Age	Humidity	HeartRate	…	Temperature
2019/10/10/ 10:00:00	Tracy	22	45%	95	…	11
2019/10/10/ 10:00:00	**Tom**	**26**	**45%**	**92**	**…**	**20**
…	…	…	…	…	…	…
2019/10/11/ 11:00:01	Tracy	22	45%	90	…	11
2019/10/11/ 11:00:01	**Tom**	**26**	**45%**	**96**	**…**	**21**

Red : Data needed **Green** : Data Scaned

QINGCLOUD

Why we choose it

- Column-Orient Database

Time	Name	Age	Humidity	HeartRate	Localtion	Temperature	…
2019/10/10/ 10:00:00	Tracy	22	45%	95	116.29860 40.13091	11	…
2019/10/10/ 10:00:00	Tom	26	45%	92	121.55687 31.31908	20	…
…	…	…	…	…	…	…	…
2019/10/11/ 11:00:01	Tracy	22	45%	90	116.30101 31.31673	11	…
2019/10/11/ 11:00:01	Tom	26	45%	96	121.54794 31.32318	21	…

Why we choose it

SELECT HeartRate FROM …
WHERE Time BETWEEN … AND … AND Name = “Tom”

Time	Name	Age	Humidity	HeartRate	Localtion	Temperature	…
2019/10/10/ 10:00:00	**Tracy**	22	45%	**95**	116.29860 40.13091	11	…
2019/10/10/ 10:00:00	**Tom**	26	45%	**92**	121.55687 31.31908	20	…
…	**…**	…	…	**…**	…	…	…
2019/10/11/ 11:00:01	**Tracy**	22	45%	**90**	116.30101 31.31673	11	…
2019/10/11/ 11:00:01	**Tom**	26	45%	**96**	121.54794 31.32318	21	…

Red : Data needed **Green** : Data Scaned

QINGCLOUD

Why we choose it

- Time-Series Database

Name	Age	Metric	Time Interval	00:00	00:01	…	59:59
Tracy	22	Humidity	2019/10/10/ 10	45%	…	…	…
Tracy	22	HeartRate	2019/10/10/ 10	95	…	…	…
…	…	…	…	…	…	…	…
Tom	26	Humidity	2019/10/10/ 10	45%	…	…	…
Tom	26	HeartRate	2019/10/10/ 10	92	…	…	…
…	…	…	…	…	…	…	…
Tracy	22	Humidity	2019/10/10/ 11	…	45%	…	…
Tracy	22	HeartRate	2019/10/10/ 11	…	90	…	…
…	…	…	…	…	…	…	…
Tom	26	Humidity	2019/10/10/ 11	…	45%	…	…
Tom	26	HeartRate	2019/10/10/ 11	…	96	…	…
…	…	…	…	…	…	…	…

Why we choose it

SELECT HeartRate FROM …
WHERE Time BETWEEN … AND … AND Name = “Tom”

Name	Age	Metric	Time Interval	00:00	00:01	…	59:59
Tracy	22	Humidity	2019/10/10/ 10	45%	…	…	…
Tracy	22	HeartRate	2019/10/10/ 10	95	…	…	…
…	…	…	…	…	…	…	…
Tom	26	Humidity	2019/10/10/ 10	45%	…	…	…
Tom	**26**	**HeartRate**	**2019/10/10/ 10**	**92**	**…**	**…**	**…**
…	…	…	…	…	…	…	…
Tracy	22	Humidity	2019/10/11/ 11	…	45%	…	…
Tracy	22	HeartRate	2019/10/10/ 11	…	90	…	…
…	…	…	…	…	…	…	…
Tom	26	Humidity	2019/10/10/ 11	…	45%	…	…
Tom	26	HeartRate	2019/10/10/ 11	…	96	…	…
…	…	…	…	…	…	…	…

Red : Data needed **Green** : Data Scaned

Why we choose it

没有最好的解决方案

QINGCLOUD

Why we choose it

没有最好的解决方案

小孩子才做选择

“好的”我们都想要！

QINGCLOUD

How we do

How we do

- ClickHouse 实现方式
  - (1) Column-Orient Model
  - (2) Time-Series-Orient Model

QINGCLOUD

How we do

Column-Orient Model

```
CREATE TABLE demonstration.insert_view
(
  `Time` DateTime,
  `Name` String,  `Age` UInt8, ...,
  `HeartRate` UInt8, `Humidity` Float32, ...
) ENGINE = MergeTree()
PARTITION BY toYYYYMM(Time)
ORDER BY (Name, Time, Age, ...);
```

Time	Name	Age	Humidity	HeartRate	Localtion	Temperature	...
2019/10/10/ 10:00:00	Tracy	22	45%	95	116.29860 40.13091	11	...
2019/10/10/ 10:00:00	Tom	26	45%	92	121.55687 31.31908	20	...
...	...	...	...	...	...	...	...
2019/10/11/ 11:00:01	Tracy	22	45%	90	116.30101 31.31673	11	...
2019/10/11/ 11:00:01	Tom	26	45%	96	121.54794 31.32318	21	...

QINGCLOUD

How we do

► Column-Orient Model

```
CREATE TABLE demonstration.insert_view
(
    `Time` DateTime,
    `Name` LowCardinality(String), `Age` UInt8, ...,
    `HeartRate` UInt8, `Humidity` Float32, ...
) ENGINE = MergeTree()
PARTITION BY toYYYYMM(Time)
ORDER BY (Name, Time, Age, ...);
```

Time	Name Dict	Name index	Age	Humidity	HeartRate	Localtion	Temperature	...
2019/10/10/ 10:00:00	Tracy,1	1	22	45%	95	116.29860 40.13091	11	...
2019/10/10/ 10:00:00	Tom,2	2	26	45%	92	121.55687 31.31908	20	...
...		...	...	...	...	...	...	...
2019/10/11/ 11:00:01		1	22	45%	90	116.30101 31.31673	11	...
2019/10/11/ 11:00:01		2	26	45%	96	121.54794 31.32318	21	...

QINGCLOUD

How we do

- Column-Orient Model

CPU : **Intel Skylake** 8 **core**

Memory : 64 **GB**

Disk : 500**GB SSD**

Data Set : **TSBS**, 12 **Hours**, 40000 **Drivers**, 10 **Metrics** ≈ 16.9 **billion Rows**

How we do

- Column-Orient Model

```
:) SELECT value
FROM benchmark.tags
WHERE (metric_name = 'cpu-usage_user')
AND
((created_at >= '2016-01-01 08:00:00')
AND
(created_at <= '2016-01-01 09:00:00'))
ORDER BY toStartOfMinute(created_at) DESC
LIMIT 5
```

```
┌─value─┐
│     4 │
│     4 │
│     4 │
│     4 │
│     4 │
└───────┘
```

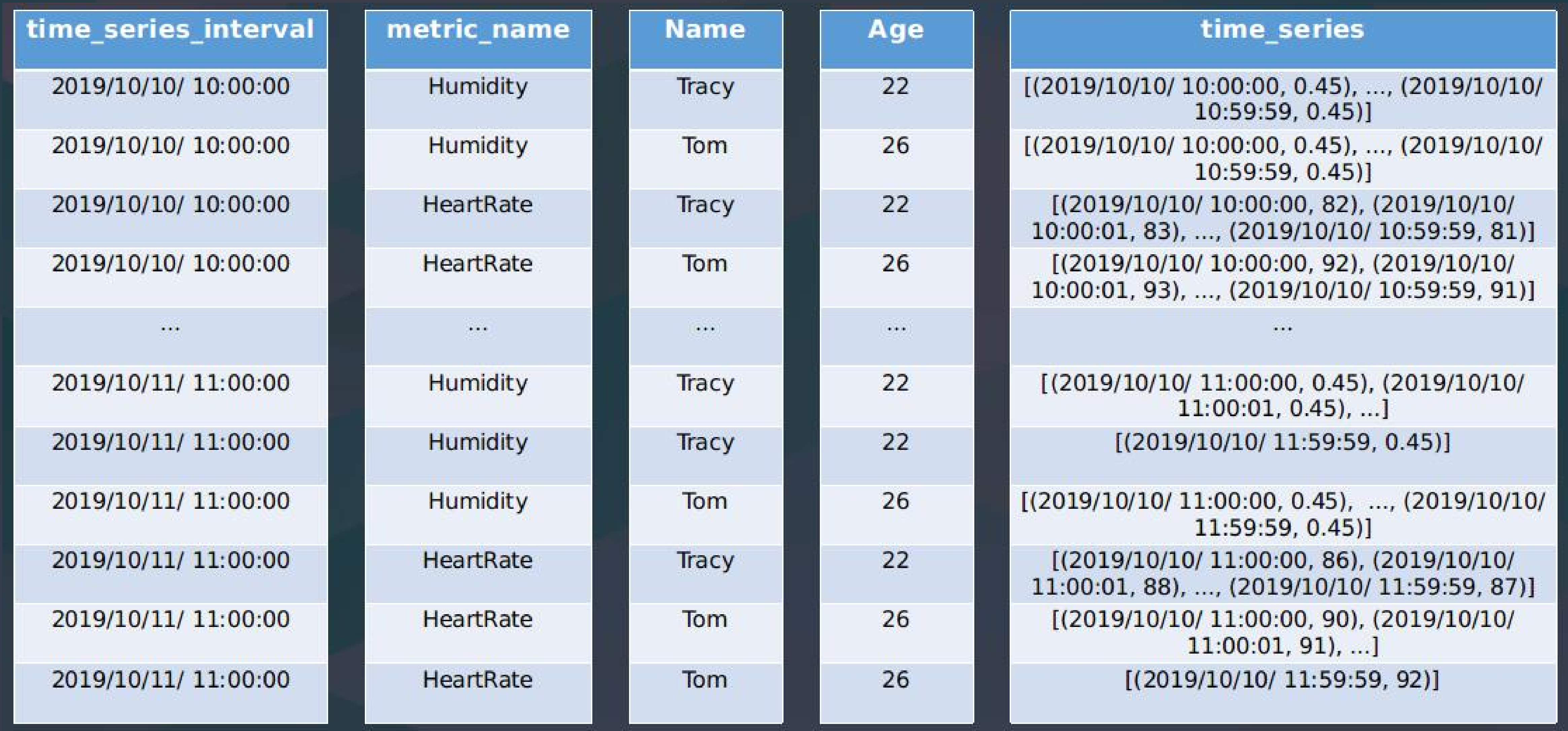
```
5 rows in set. Elapsed: 0.854 sec.
Processed 144.06 million rows,
5.19 GB (168.64 million rows/s.,
6.07 GB/s.)
```

QINGCLOUD

How we do

- Time-Series-Orient Model

```
CREATE TABLE demonstration.test
(
    `time_series_interval` DateTime,
    `metric_name` String,
    `Name` String, `Age` UInt8, ...,
    `time_series` AggregateFunction(
        groupArray, Tuple(DateTime, Float64))
) ENGINE = AggregatingMergeTree()
PARTITION BY toYYYYMM(time_series_interval)
ORDER BY (metric_name, time_series_interval)
```



time_series_interval	metric_name	Name	Age	time_series
2019/10/10/ 10:00:00	Humidity	Tracy	22	[(2019/10/10/ 10:00:00, 0.45), ..., (2019/10/10/ 10:59:59, 0.45)]
2019/10/10/ 10:00:00	Humidity	Tom	26	[(2019/10/10/ 10:00:00, 0.45), ..., (2019/10/10/ 10:59:59, 0.45)]
2019/10/10/ 10:00:00	HeartRate	Tracy	22	[(2019/10/10/ 10:00:00, 82), (2019/10/10/ 10:00:01, 83), ..., (2019/10/10/ 10:59:59, 81)]
2019/10/10/ 10:00:00	HeartRate	Tom	26	[(2019/10/10/ 10:00:00, 92), (2019/10/10/ 10:00:01, 93), ..., (2019/10/10/ 10:59:59, 91)]
...	...	...	...	...
2019/10/11/ 11:00:00	Humidity	Tracy	22	[(2019/10/10/ 11:00:00, 0.45), (2019/10/10/ 11:00:01, 0.45), ...]
2019/10/11/ 11:00:00	Humidity	Tracy	22	[(2019/10/10/ 11:59:59, 0.45)]
2019/10/11/ 11:00:00	Humidity	Tom	26	[(2019/10/10/ 11:00:00, 0.45), ..., (2019/10/10/ 11:59:59, 0.45)]
2019/10/11/ 11:00:00	HeartRate	Tracy	22	[(2019/10/10/ 11:00:00, 86), (2019/10/10/ 11:00:01, 88), ..., (2019/10/10/ 11:59:59, 87)]
2019/10/11/ 11:00:00	HeartRate	Tom	26	[(2019/10/10/ 11:00:00, 90), (2019/10/10/ 11:00:01, 91), ...]
2019/10/11/ 11:00:00	HeartRate	Tom	26	[(2019/10/10/ 11:59:59, 92)]

QINGCLOUD

How we do

► Time-Series-Orient Model

```
CREATE TABLE demonstration.test
(
    `time_series_interval` DateTime,
    `metric_name` LowCardinality(String),
    `Name` LowCardinality(String), `Age` UInt8, ...,
    `time_series` AggregateFunction(
        groupArray, Tuple(DateTime, Float64))
) ENGINE = AggregatingMergeTree()
PARTITION BY toYYYYMM(time_series_interval)
ORDER BY (metric_name, time_series_interval)
```

time_series_interval	dict	metric_name	dict	Name	Age	time_series
2019/10/10/ 10:00:00	Humidity	1	Tracy	1	22	[(2019/10/10/ 10:00:00, 0.45), ..., (2019/10/10/ 10:59:59, 0.45)]
2019/10/10/ 10:00:00	HeartRate	2	Tom	2	26	[(2019/10/10/ 10:00:00, 0.45), ..., (2019/10/10/ 10:59:59, 0.45)]
2019/10/10/ 10:00:00	...	1	...	1	22	[(2019/10/10/ 10:00:00, 82), (2019/10/10/ 10:00:01, 83), ..., (2019/10/10/ 10:59:59, 81)]
2019/10/10/ 10:00:00		2		2	26	[(2019/10/10/ 10:00:00, 92), (2019/10/10/ 10:00:01, 93), ..., (2019/10/10/ 10:59:59, 91)]
...		...		...	...	...
2019/10/11/ 11:00:00		1		1	22	[(2019/10/10/ 11:00:00, 0.45), (2019/10/10/ 11:00:01, 0.45), ...]
2019/10/11/ 11:00:00		1		1	22	[(2019/10/10/ 11:59:59, 0.45)]
2019/10/11/ 11:00:00		1		2	26	[(2019/10/10/ 11:00:00, 0.45), ..., (2019/10/10/ 11:59:59, 0.45)]
2019/10/11/ 11:00:00		2		1	22	[(2019/10/10/ 11:00:00, 86), (2019/10/10/ 11:00:01, 88), ..., (2019/10/10/ 11:59:59, 87)]
2019/10/11/ 11:00:00		2		1	26	[(2019/10/10/ 11:00:00, 90), (2019/10/10/ 11:00:01, 91), ...]
2019/10/11/ 11:00:00		2		2	26	[(2019/10/10/ 11:59:59, 92)]

QINGCLOUD

How we do

Now :

```
INSERT INTO demonstration.test SELECT ..., 'HeartRate',
groupArrayState(Tuple('2019-10-11 11:11:00', 87));
```

AND :

```
SELECT ..., metric_name, groupArrayMerge(time_series)
FROM demonstration.test ...;
```

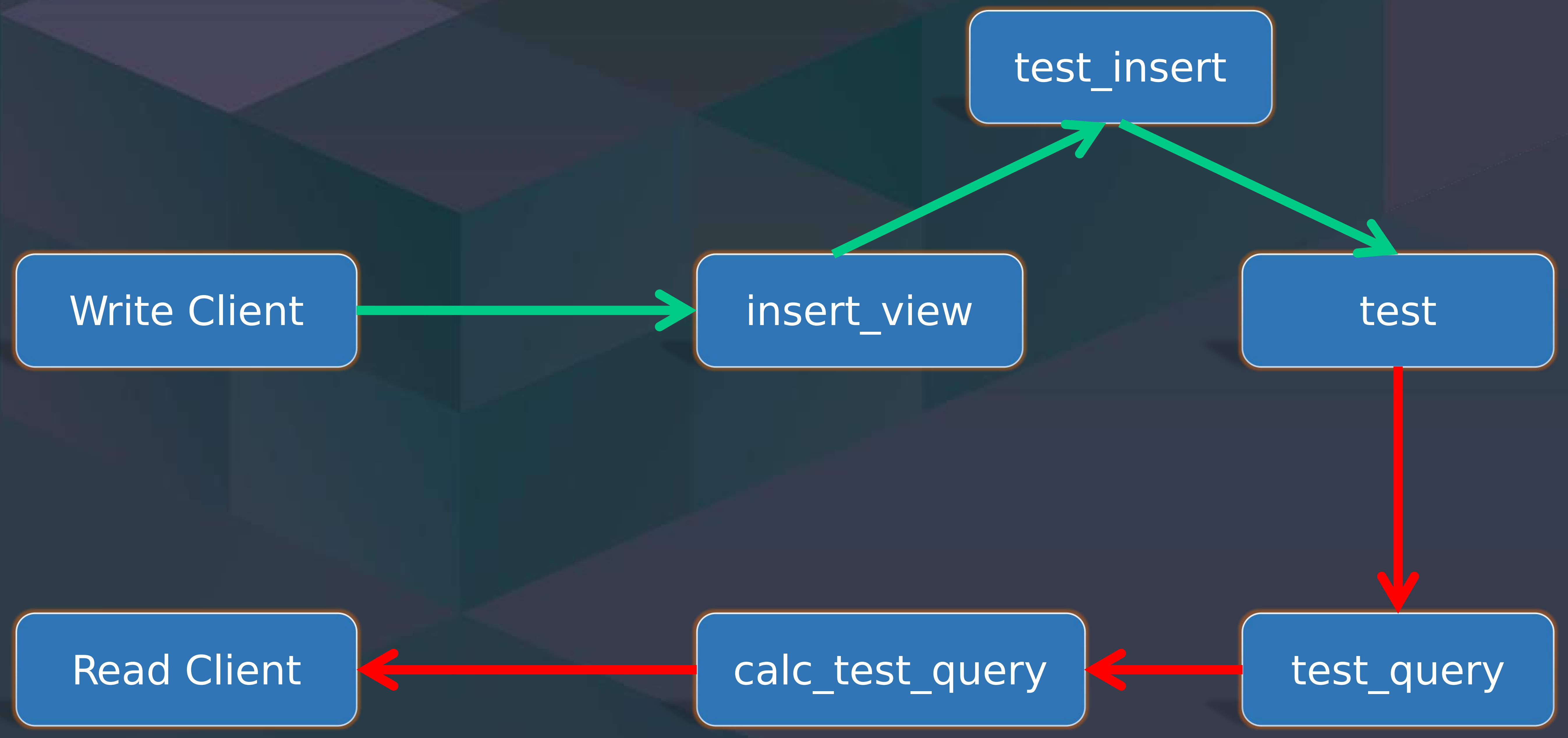
How we do

It's veeeeeery complicated!!!

So...

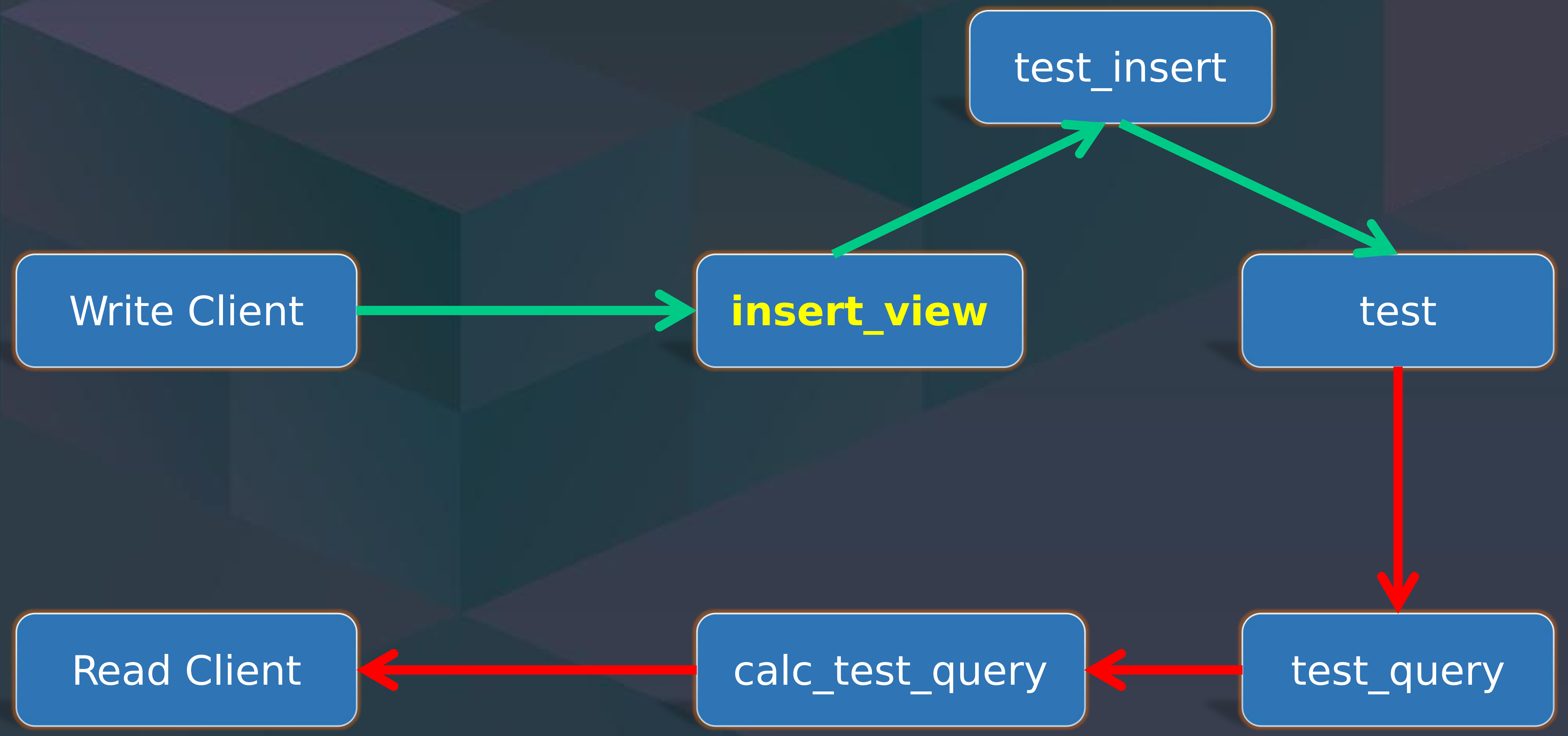
How we do

- Time-Series-Orient Model



How we do

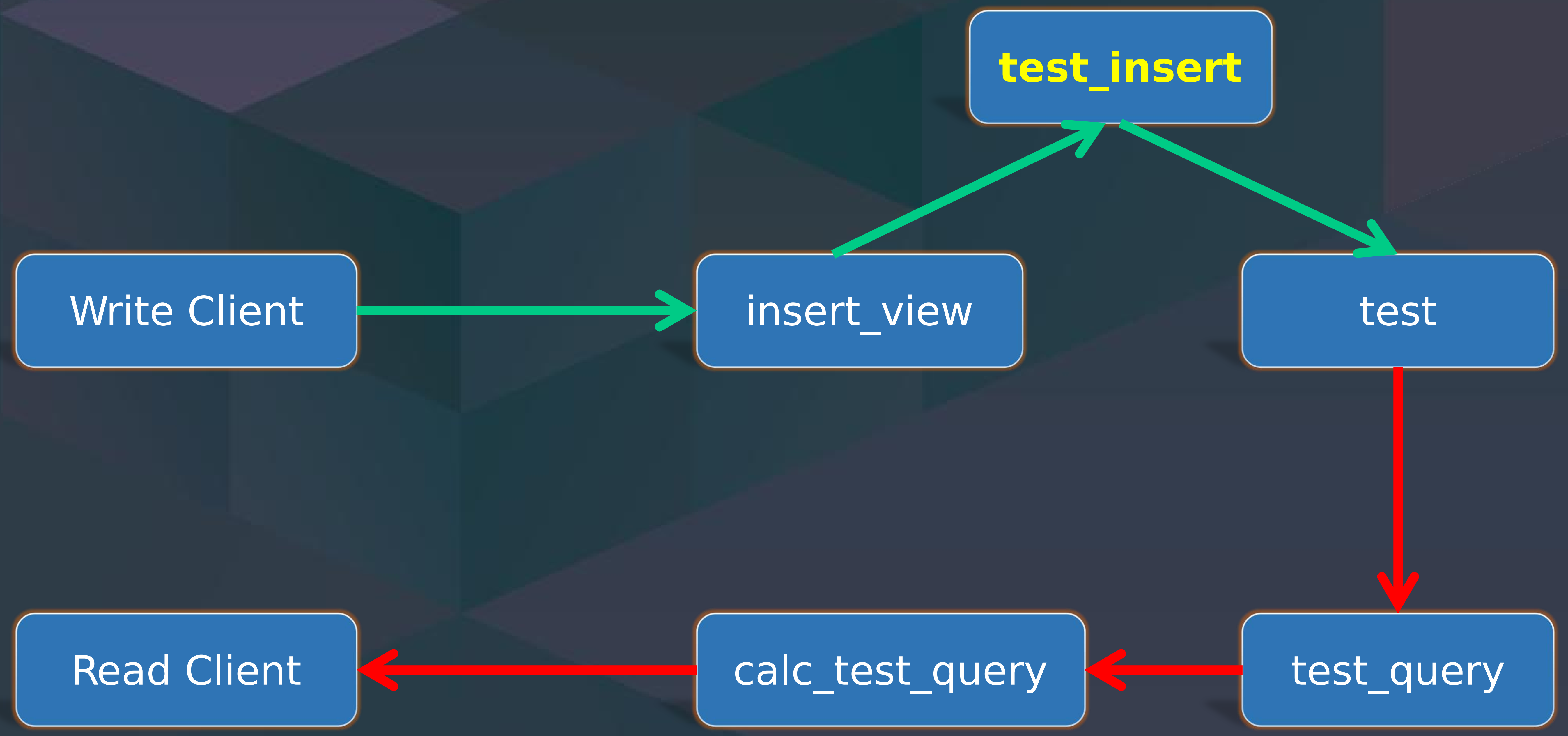
- Time-Series-Orient Model



```
CREATE TABLE demonstration.insert_view
(
    `Time` DateTime,
    `metric_name` String,
    `Name` String, `Age` UInt8, ...,
    `value` Float64
)
ENGINE = Null
```

How we do

- Time-Series-Orient Model

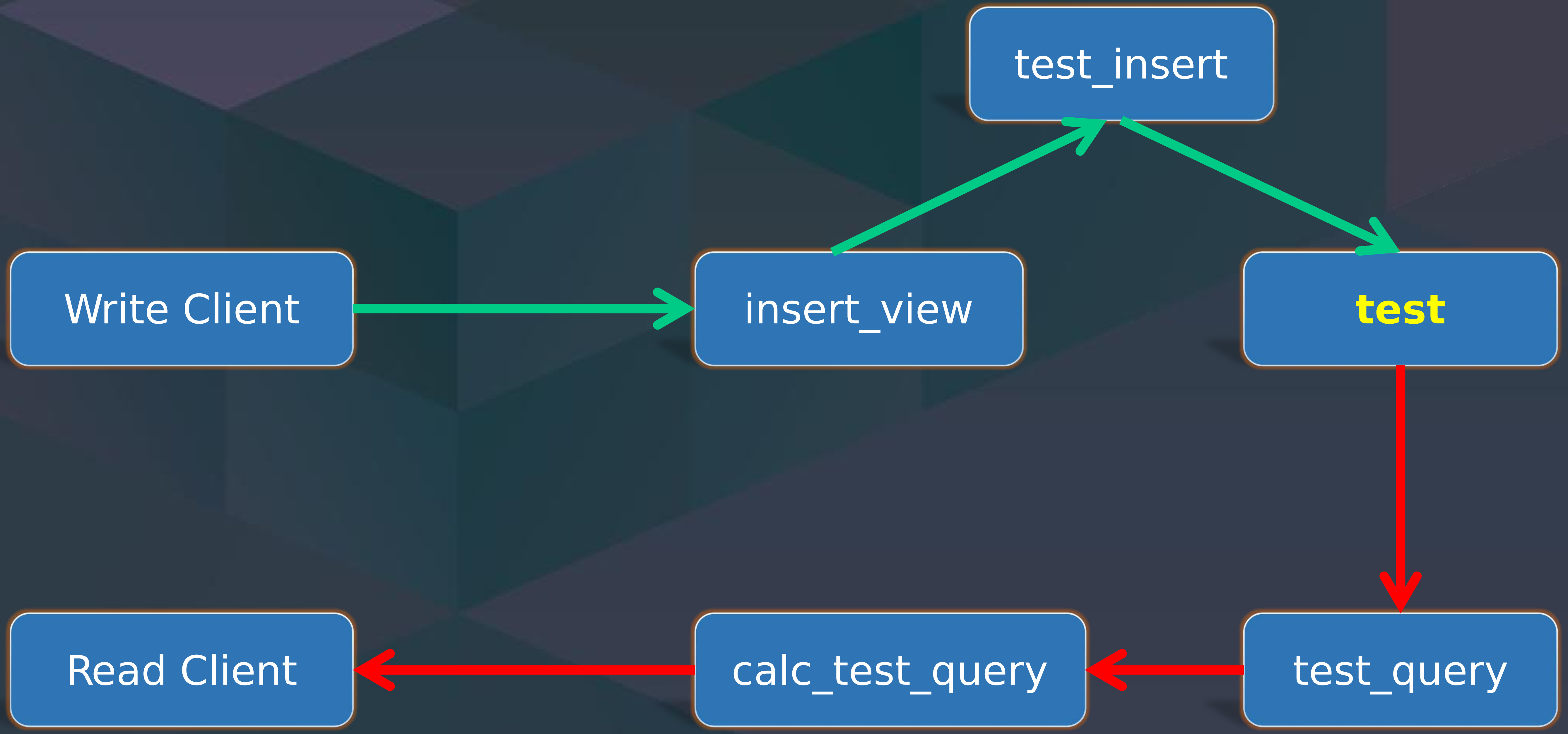


```
CREATE MATERIALIZED VIEW demonstration.test_insert
TO demonstration.test AS SELECT
    toStartOfInterval(Time, toIntervalMinute(30))
        AS time_series_interval,
    metric_name, Name, Age, ...,
    groupArrayState((Time, value)) AS timeseries
FROM demonstration.insert_view
GROUP BY time_series_interval, metric_name, Name, Age
```

QINGCLOUD

How we do

- Time-Series-Orient Model

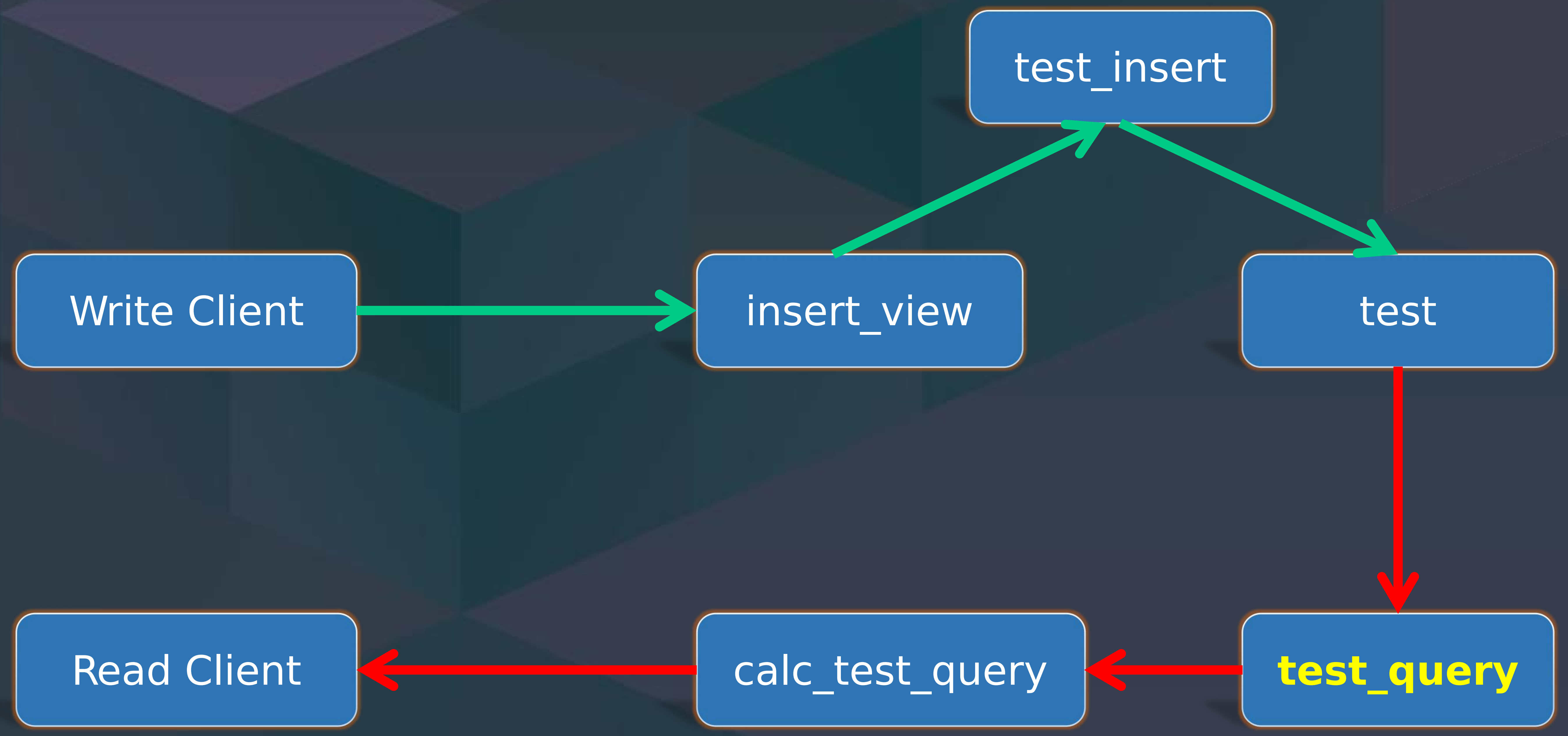


```
CREATE TABLE demonstration.test
(
    `time_series_interval` DateTime,
    `metric_name` LowCardinality(String),
    `Name` LowCardinality(String),  `Age` UInt8, ...
     `time_series` AggregateFunction(
          groupArray, Tuple(DateTime, Float64))
) ENGINE = AggregatingMergeTree()
PARTITION BY toYYYYMM(time_series_interval)
ORDER BY (metric_name, time_series_interval,
Name, ...)
```

QINGCLOUD

How we do

- Time-Series-Orient Model

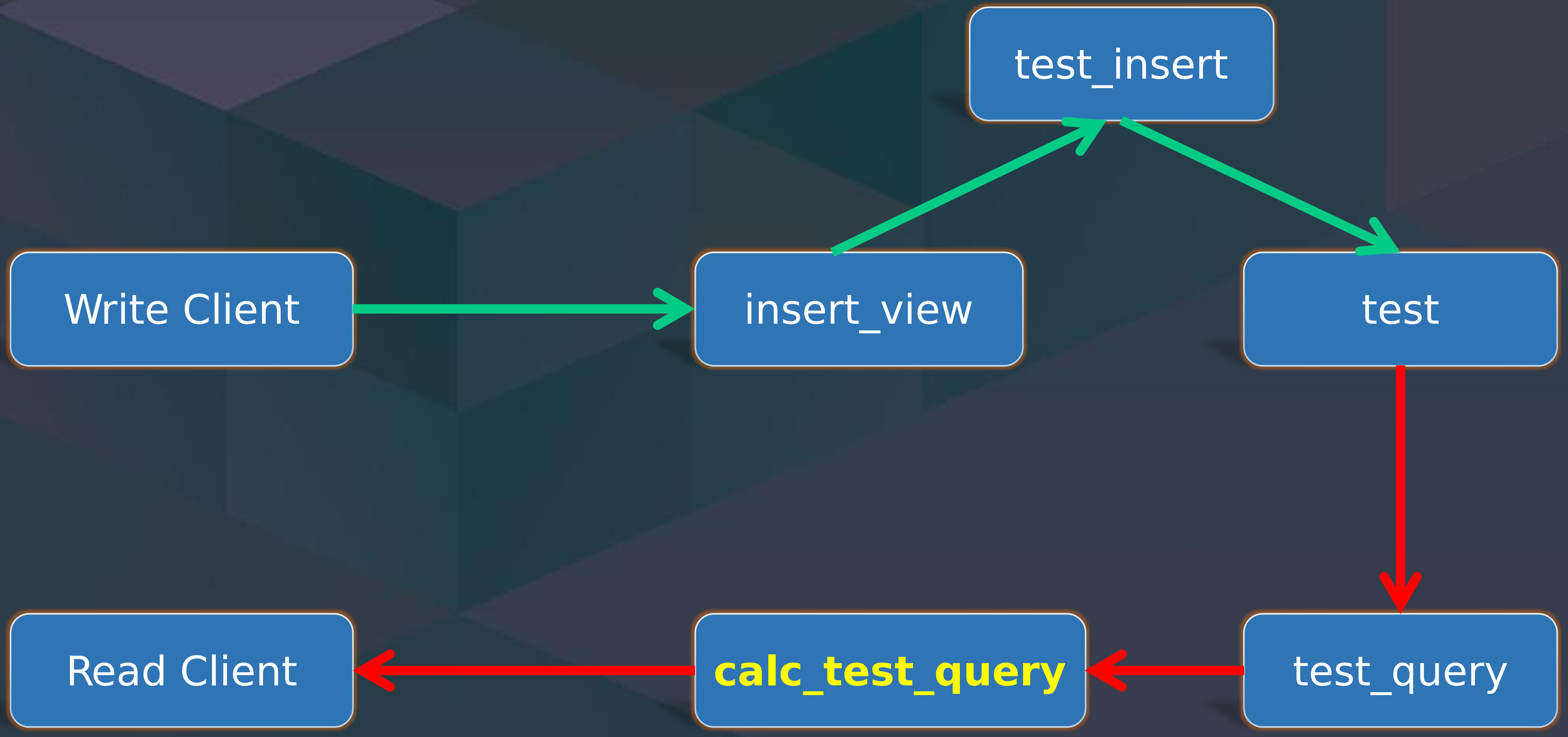


```
CREATE VIEW demonstration.test_query AS
SELECT
    metric_name,  Name, Age, ...,
    finalizeAggregation(timeseries) AS timeseries
FROM demonstration.test
```

QINGCLOUD

How we do

- Time-Series-Orient Model



```
CREATE VIEW demonstration.calc_test_query AS
SELECT
    metric_name,  Name , Age,
    timeseries.1 AS date_time,  timeseries.2 AS value
FROM demonstration.test_query
ARRAY JOIN timeseries AS timeseries
```

How we do

Now :

```
INSERT INTO demonstration.insert_view(..., metric_name,
date_time, value ) VALUES(...);
```

AND :

```
SELECT ..., metric_name, date_time, value FROM
demonstration.calc_test_query;
```

How we do

- Time-Series-Orient Model

CPU：**Intel Skylake** 8 **core**

Memory：64 **GB**

Disk：500**GB SSD**

Data Set：**TSBS**, 12 **Hours**, 40000 **Drivers**, 10 **Metrics** ≈ 19.6 **billion Rows**

How we do

- Time-Series-Orient Model

```
:) SELECT value
FROM benchmark.calc_tags_query
WHERE (metric_name = 'cpu-usage_user')
AND
((created_at >= '2016-01-01 08:00:00')
AND
(created_at <= '2016-01-01 09:00:00'))
ORDER BY toStartOfMinute(created_at) DESC
LIMIT 5
```

```
┌─value─┐
│     4 │
│     4 │
│     4 │
│     4 │
│     4 │
└───────┘

5 rows in set. Elapsed: 1.565 sec.
Processed 281.69 thousand
rows, 11.06 GB (180.01 thousand
rows/s., 7.07 GB/s.)
```

QINGCLOUD

What we do

What we do

- Support JSONB DataType for tags & value
  - Support LowCardinality(JSONB)
  - Support BoolFilter skip index with JSONB data type
- Support TimeSeriesMergeTree Table Engine
  - Support Multiple Streams for AggregationFunction
  - Support TimeSeriesAggregateFunction(store sum, min, max, avg)
  - Support convert sum(time_series) to sum(time_series.sum)

QINGCLOUD

QingCloud ChronusDB

青云 QingCloud 自研的一款高性能、具备强大 分析能力的时序数据库产品

高性能并发读写

- 千万数据点并发实时写入
- 引入辅助索引，加快数据检索速度

低成本存储

- 列式存储结合高效的编码
- Delta、XOR 等适合时序场景的压缩算法
- 通过 Rollup 功能，对历史数据做聚合，减少数据量

稳定可扩展

- 分布式架构
- 数据多副本存储
- 服务高可用

QINGCLOUD

Thanks For You