Build a Real-Time Analytics Database

a choose your own adventure journey

YOU'RE THE STAR OF THE STORY!
CHOOSE FROM WAY TOO MANY POSSIBLE ENDINGS!
MOST OF WHICH ARE BAD!

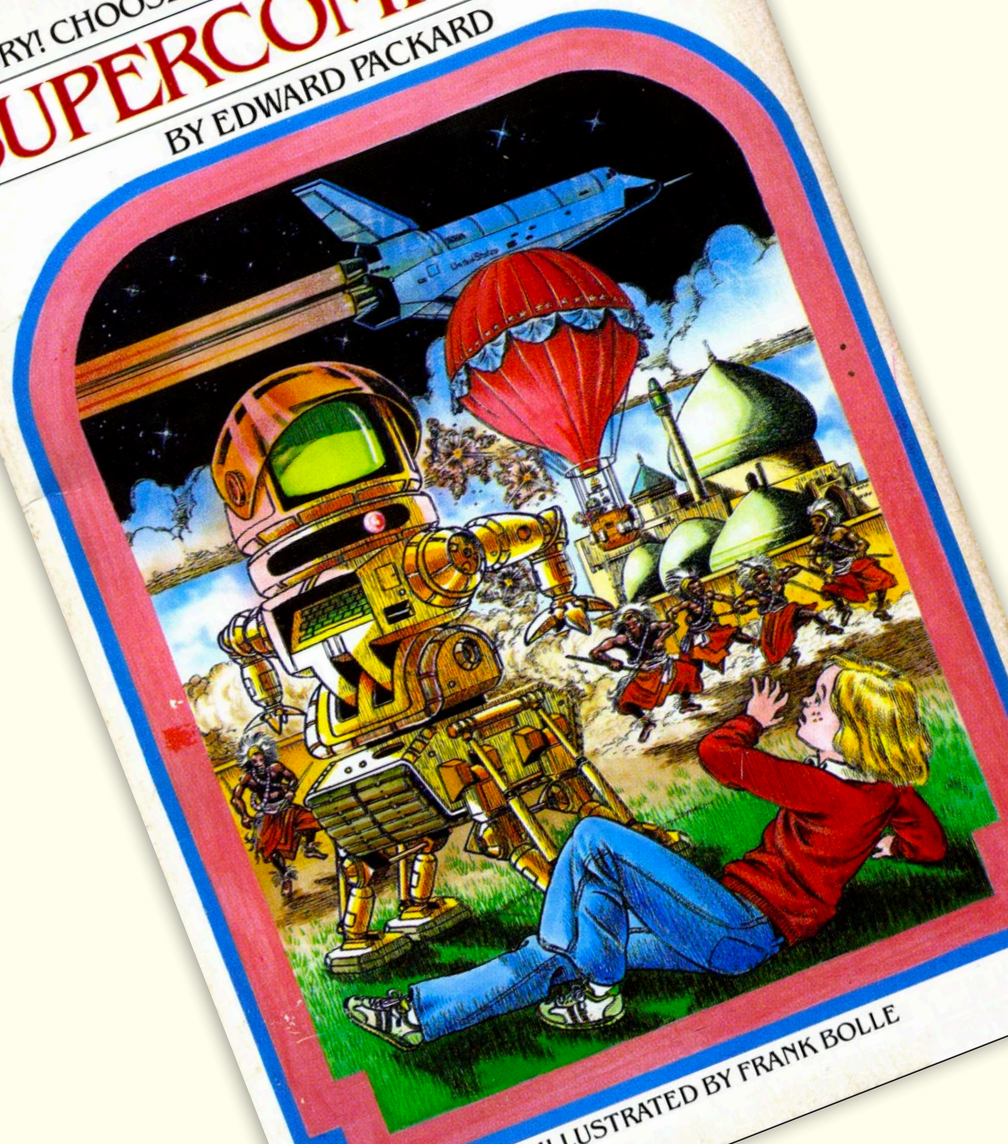
Tim Berglund
@tlberglund

CHOOSE YOUR OWN ADVENTURE® 39

YOU'RE THE STAR OF THE STORY! CHOOSE FROM 22 POSSIBLE ENDINGS

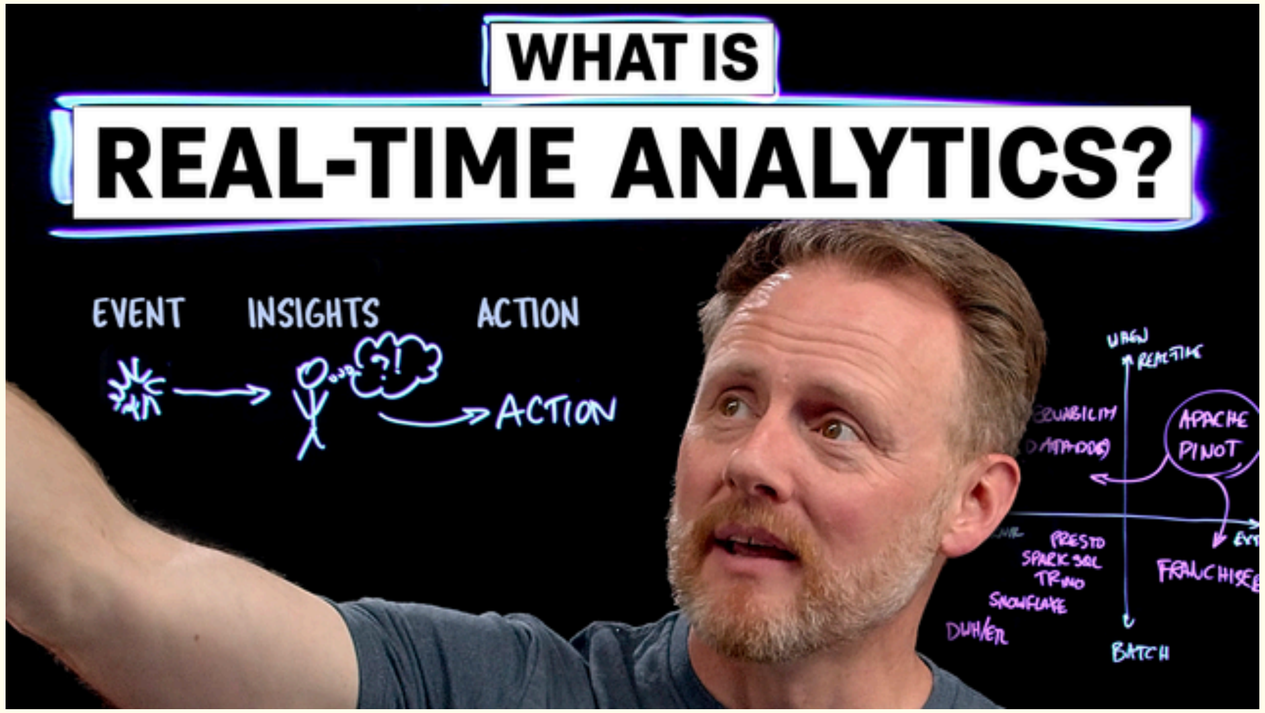
SUPERCOMPUTER

BY EDWARD PACKARD



ILLUSTRATED BY FRANK BOLLE

24618-X ★ IN U.S. $1.95 ★ A BANTAM BOOK


WHAT IS
REAL-TIME ANALYTICS?
EVENT
INSIGHTS
ACTION
ACTION
APACHE PINOT
PRESTO
SPARK SQL
TRINO
SNOWFLAKE
BATCH

What do you want to do with your data?

1. I want to remember what has happened
2. I want to keep track of things
3. I want to understand what is happening

What do you want to do with your data?

1. **Remember**
2. Keep track
3. Understand

You want a distributed log

- Apache Kafka
- Apache Pulsar
- AWS Kinesis

What do you want to do with your data?

1. Remember
2. **Keep track**
3. Understand

You want an OLTP database

- CRUD over entities
- Postgres
- MySQL
- Oracle I guess
- Focused on “this thing”

What do you want to do with your data?

1. Remember
2. Keep track
3. **Understand**

You want an OLAP database

- Measurements
- Dimensions
- Insights
- Focused on “these things”

How do you want to represent your data?

1. Key/value pairs
2. Documents
3. Tuples

How do you want to represent your data?

1. **Key/value pairs**
2. Documents
3. Tuples

You have died.

- Okay, not really
- But this is not a good OLAP option
- Maybe we'll revisit this later...

How do you want to represent your data?

1. Key/value pairs
2. **Documents**
3. Tuples

Interesting choice!

- Schema flexibility
- Soooo many more choices to make
- MongoDB

How do you want to represent your data?

1. Key/value pairs
2. Documents
3. **Tuples**

You are a normal person

- Inheriting a tradition
- Implies tables
- The relational algebra


MARVEL® COMICS GROUP
BY THE COMICS CODE AUTHORITY
75¢
150
SEPT
02454
SPECIAL DOUBLE-SIZED 150TH ANNIVERSARY ISSUE!
THE INVINCIBLE
IRON MAN®
TRAPPED IN THE TIME-LOST LAND OF KING ARTHUR-- IRON MAN BATTLES
DOCTOR DOOM!
09
71486 02454


TRAPPED IN THE
TIME-LOST LAND
OF KING ARTHUR--
IRON MAN BATTLES
DOCTOR DOOM!

How do you want to serialize your data?

1. By row
2. By column

How do you want to serialize your data?

1. **By row**
2. By column

Whole documents/tuples

- Batch up the field K/V pairs, stream the bytes
- Very OLTP-aligned: focused on the one thing
- Lots of OLAP gets done this way!

How do you want to serialize your data?

1. By row
2. **By columns**

Column Databases

- OLAP wants “these things,” not “this thing”
- You are usually aggregating just one measurement
- Besides, it’s more compressible!

What query language do you want to use?

1. None
2. A custom API
3. A custom language
4. SQL

What query language do you want to use?

1. **None**
2. API
3. Custom
4. SQL

You have died. Again.

- This was Hadoop
- Map/Reduce programming was not a net improvement to human flourishing

What query language do you want to use?

1. None
2. **API**
3. Custom
4. SQL

Kafka Streams

- Or solutions like it
- Limited to the given language bindings
- Sometimes *very good*

What query language do you want to use?

1. None
2. API
3. **Custom**
4. SQL

Your own query language!

- This is fine if you're Mongo
- You will still be writing your docs with SQL examples until the heat death of the universe
- Also it takes a lot of energy

What query language do you want to use?

1. None
2. API
3. Custom
4. **SQL**

You are a normal person

- You might try to avoid this
- You will probably fail
- You are again inheriting a substantial tradition

How do you want to organize storage?

Wait, the question is actually...

Can things change?

1. Yes, my data is mutable
2. No, my data is (pretty much) immutable

Can things change?

1. **Mutable data**
2. Immutable data

Implied storage architecture

- Pages: read/modify/write
- LSM trees: log/flush/compact

Can things change?

1. Mutable data
2. **Immutable data**

Still can't really answer the question. It implies another question about dimensionality, which often presents as a question about...indexing.

How many dimensions do tables have?

1. Just one
2. Lots

How many dimensions do tables have?

1. **Just one**
2. Lots

Then you only need one index

- Kafka/KafkaStreams
- Kafka/ksqlDB
- Cassandra
- RocksDB
- Any old K/V store will do

But wait...isn't this OLAP?

- I thought you said k/v stores were not good OLAP databases
- And we *did* choose OLAP at the start of our journey
- Analytical data is usually highly dimensional: I clicked on that site, but I am male, live in Colorado, use Brave 1.39, OSX 12.4, etc.
- If it's *not* all that dimensional, I can pre-aggregate and use whatever fast key/value store I want

How many dimensions do tables have?

1. Just one
2. **Lots**

Now you need indexes. This is its own lengthy side quest.

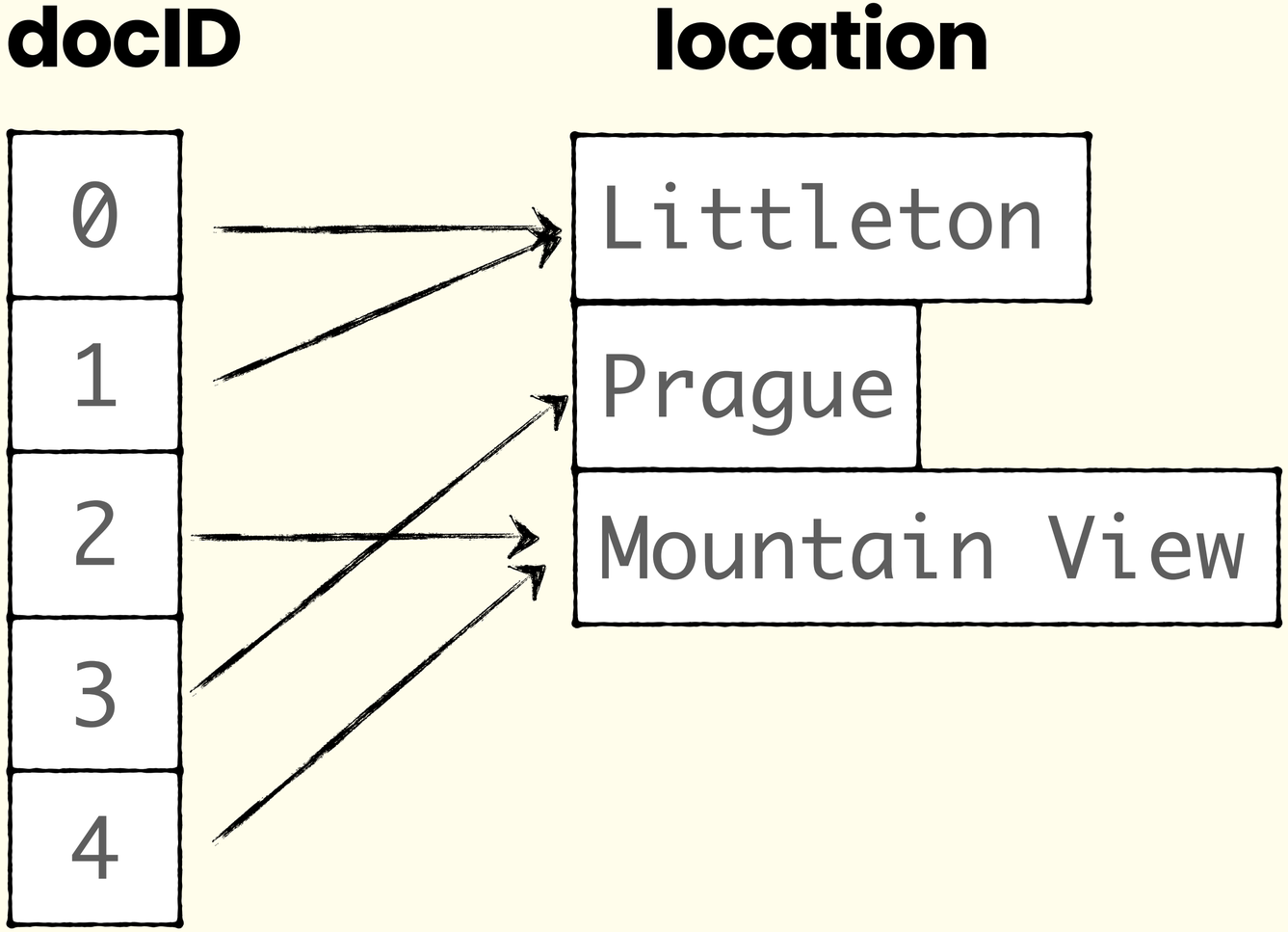
Indexes

Forward Index

- “Where is row X?”
- Remember that this is a column database. Bits of rows are stored everywhere.
- The real question is: where is *this column* for this row?
- Wait…what’s a row?
- `docID`

Forward Index

```
0, Littleton
1, Littleton
2, Mountain View
3, Prague
4, Mountain View
```

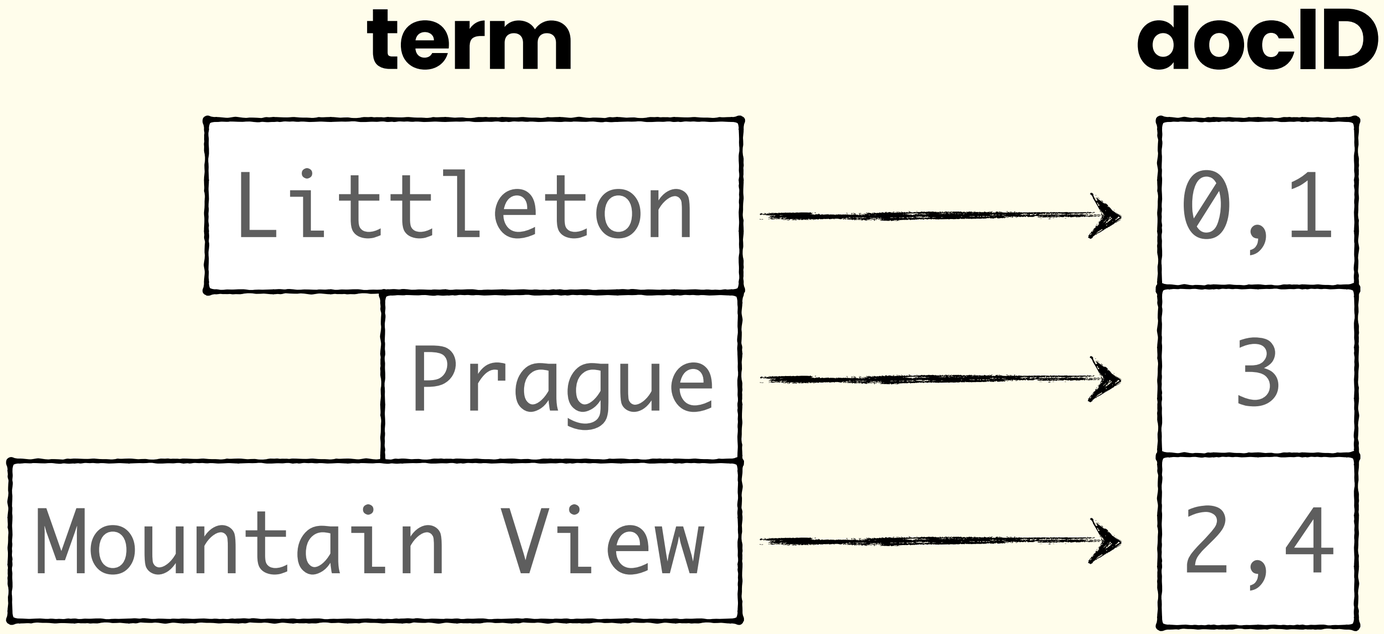


Inverted Index

- “What rows does this value occur in?”
- Give it a value, it gives you a list of docIDs

Inverted Index

```
0, Littleton
1, Littleton
2, Mountain View
3, Prague
4, Mountain View
```



Bloom Filter Index

- Helps predict whether to check a particular segment for a docID
- If it says no, the answer is certainly no
- If it says yes, the answer *might be* yes
- Maintains a space-efficient, in-memory bitmap
- Only works on dictionary-encoded columns
- Accelerates equality predicates only

Text Index

- Exact-match term queries are supported by the inverted index
- Text BLOB columns often need regex, phrase, and fuzzy matches
- Supported expressions:
  - Phrase: ‘“PETG filament”’
  - Term: ‘filament’
  - Boolean: ‘“PETG filament” AND “red”’
  - Prefix: ‘filam*’
  - Regex: ‘/P[L|E][A|T]G? filament/‘

```
SELECT COUNT(*)
FROM Inventory
WHERE TEXT_MATCH ('description', '<search_expression>')
```

Geospatial Index

- Based on the H3 library from Uber
- Hexagon-based decomposition of geospace
- Support arbitrary points, polygons
- Distance, within, contains as predicates
- https://docs.pinot.apache.org/basics/indexing/geospatial-support

```
SELECT address, ST_DISTANCE(location_st_point, ST_Point(-122, 37, 1))
FROM starbucksStores
WHERE ST_DISTANCE(location_st_point, ST_Point(-122, 37, 1)) < 5000
limit 1000
```

JSON Index

```
{
  "name": "adam",
  "age": 30,
  "country": "us",
  "addresses":
  [
    {
      "number" : 112,
      "street" : "main st",
      "country" : "us"
    },
    {
      "number" : 2,
      "street" : "second st",
      "country" : "us"
    },
    {
      "number" : 3,
      "street" : "third st",
      "country" : "ca"
    }
  ]
}
```

Simple key lookup

```
SELECT ...
FROM personnel
WHERE JSON_MATCH(person, '"$.name"=''adam''')
```

JSON Index

```
{
  "name": "adam",
  "age": 30,
  "country": "us",
  "addresses":
  [
    {
      "number" : 112,
      "street" : "main st",
      "country" : "us"
    },
    {
      "number" : 2,
      "street" : "second st",
      "country" : "us"
    },
    {
      "number" : 3,
      "street" : "third st",
      "country" : "ca"
    }
  ]
}
```

Chained key lookup

```
SELECT ...
FROM personnel
WHERE JSON_MATCH(person, '"$.addresses[*].number"=112')
```

JSON Index

```
{
  "name": "adam",
  "age": 30,
  "country": "us",
  "addresses":
  [
    {
      "number" : 112,
      "street" : "main st",
      "country" : "us"
    },
    {
      "number" : 2,
      "street" : "second st",
      "country" : "us"
    },
    {
      "number" : 3,
      "street" : "third st",
      "country" : "ca"
    }
  ]
}
```

Nested filter expression

```
SELECT ...
FROM personnel
WHERE JSON_MATCH(person,
    '"$.name"=''adam'' AND "$.addresses[*].number"=112')
```

JSON Index

```
{
  "name": "adam",
  "age": 30,
  "country": "us",
  "addresses":
  [
    {
      "number" : 112,
      "street" : "main st",
      "country" : "us"
    },
    {
      "number" : 2,
      "street" : "second st",
      "country" : "us"
    },
    {
      "number" : 3,
      "street" : "third st",
      "country" : "ca"
    }
  ]
}
```

Array access

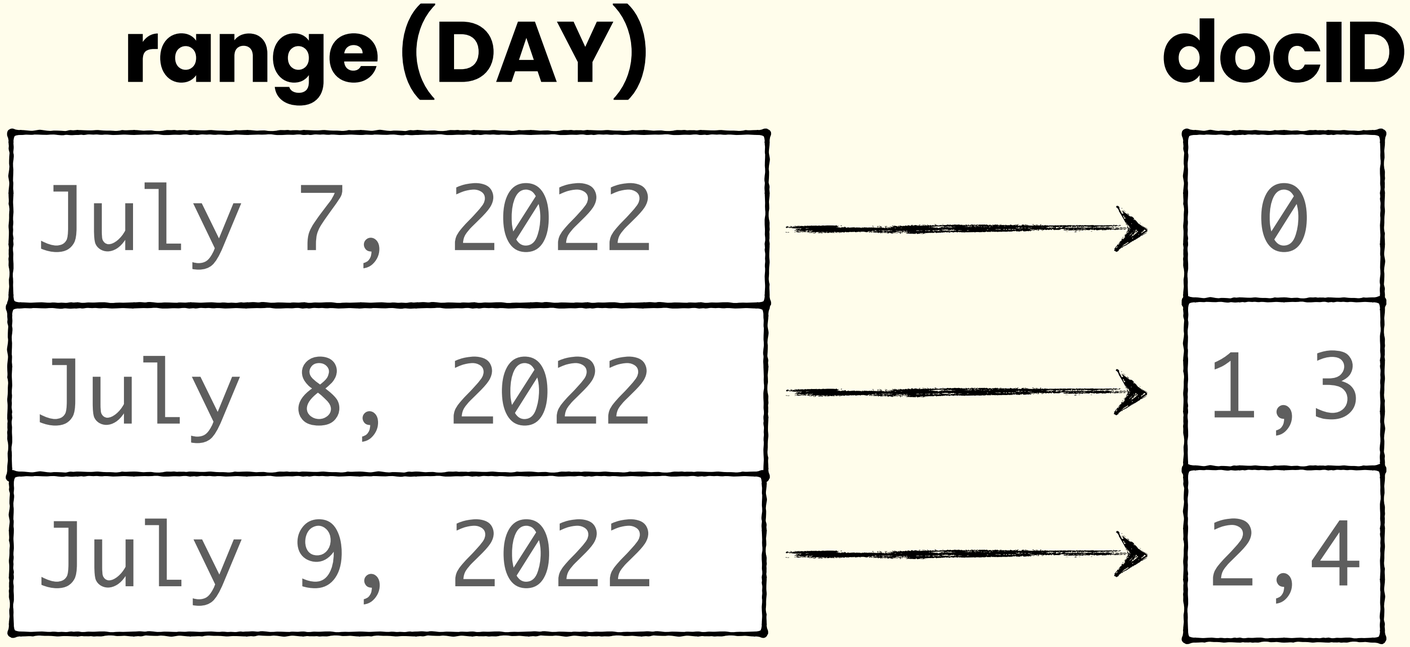
```
SELECT ...
FROM personnel
WHERE JSON_MATCH(person, '"$.addresses[0].number"=112')
```

Timestamp Index

- Applies to columns of type `TIMESTAMP` (shockingly)
- Like an inverted index, but uses date ranges for keys
- Ranges (“granularity”) are configurable at index definition time

Timestamp Index

```
0, 20220707T1449
1, 20220708T0543
2, 20220709T0128
3, 20220708T2150
4, 20220709T1507
```



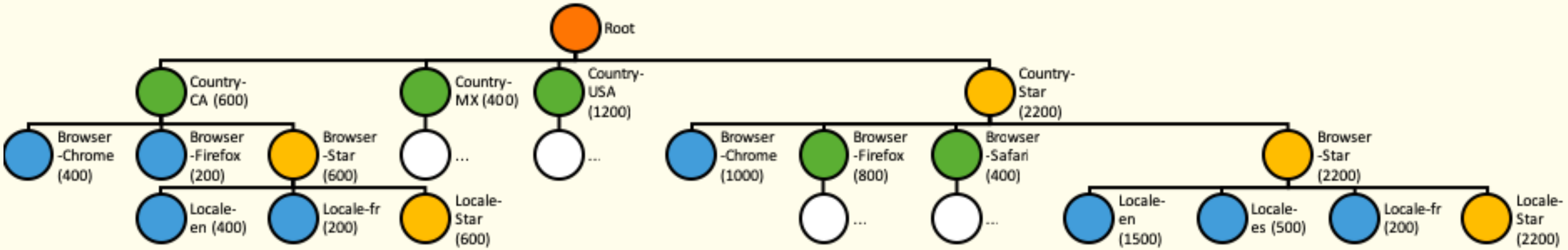
StarTree Index

- We often want to compute aggregates predicated on multiple dimensions
- This is what a pivot table does in a spreadsheet
- The StarTree index is like writing a pivot table to disk

The data set

Country	Browser	Locale	Impressions
CA	Chrome	en	400
CA	Firefox	fr	200
MX	Safari	es	300
MX	Safari	en	100
USA	Chrome	en	600
USA	Firefox	es	200
USA	Firefox	en	400

The Tree Itself



Documents in the Index

Country	Browser	Locale	SUM__Impressions
CA	Chrome	en	400
CA	Firefox	fr	200
MX	Safari	en	100
MX	Safari	es	300
USA	Chrome	en	600
USA	Firefox	en	400
USA	Firefox	es	200
CA	*	en	400
CA	*	fr	200
CA	*	*	600
MX	Safari	*	400
USA	Firefox	*	600
USA	*	en	1000
USA	*	es	200
USA	*	*	1200

How fast do you need it?

1. Dashboard speed
2. UI speed

How fast do you need it?

1. **Dashboard speed**
2. UI speed

Reporting and dashboards

- Queries that take seconds are fine
- Batch ingest is fine

How fast do you need it?

1. Dashboard speed
2. **UI speed**

Powering the UI with analytics

- 100ms is a long time
- Streaming ingest is required

How do you want to scale?

1. Single-process (I don’t)
2. Distributed (I do)

How do you want to scale?

1. **Single-process**
2. Distributed

You have chosen a good life

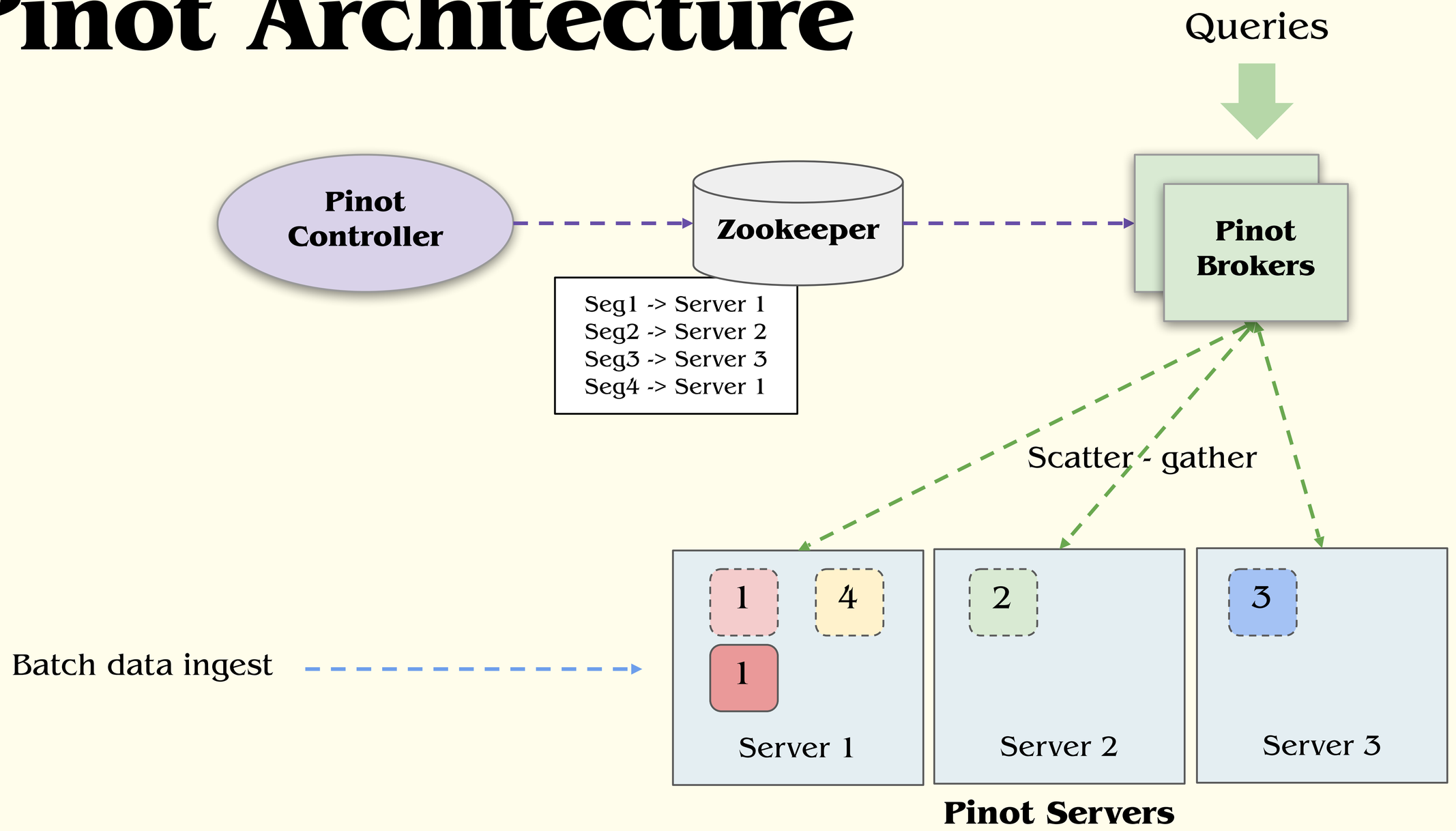
- You own the storage
- You serialize a single stream of reads and mutations
- You'll never get all that big
- Unless you replicate and shard, but I just said you chose a good life

How do you want to scale?

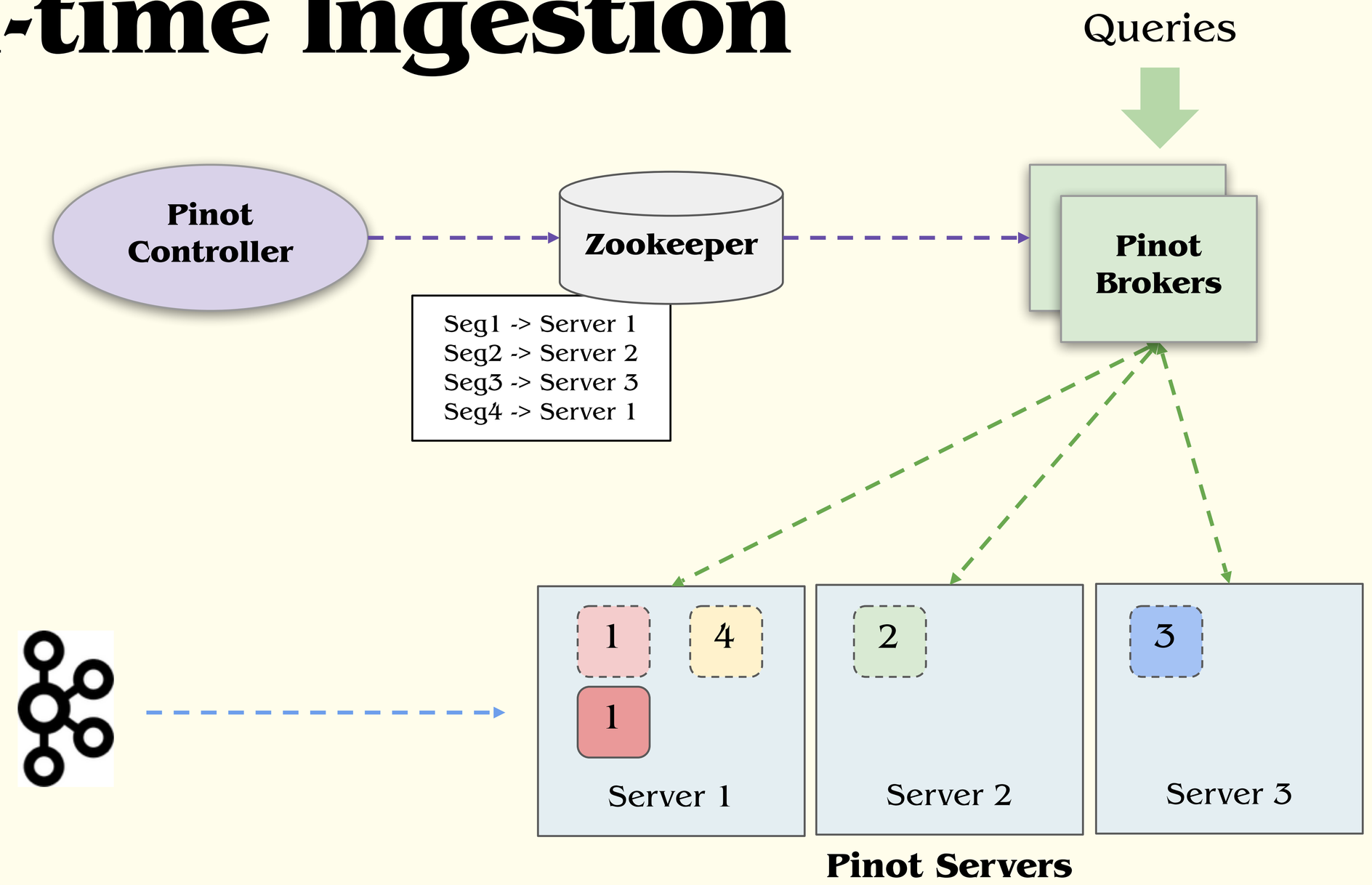
1. Single-process
2. **Distributed**

You have chosen to scale compute and storage. You have so many options. Best just to talk about how Pinot works.

Pinot Architecture

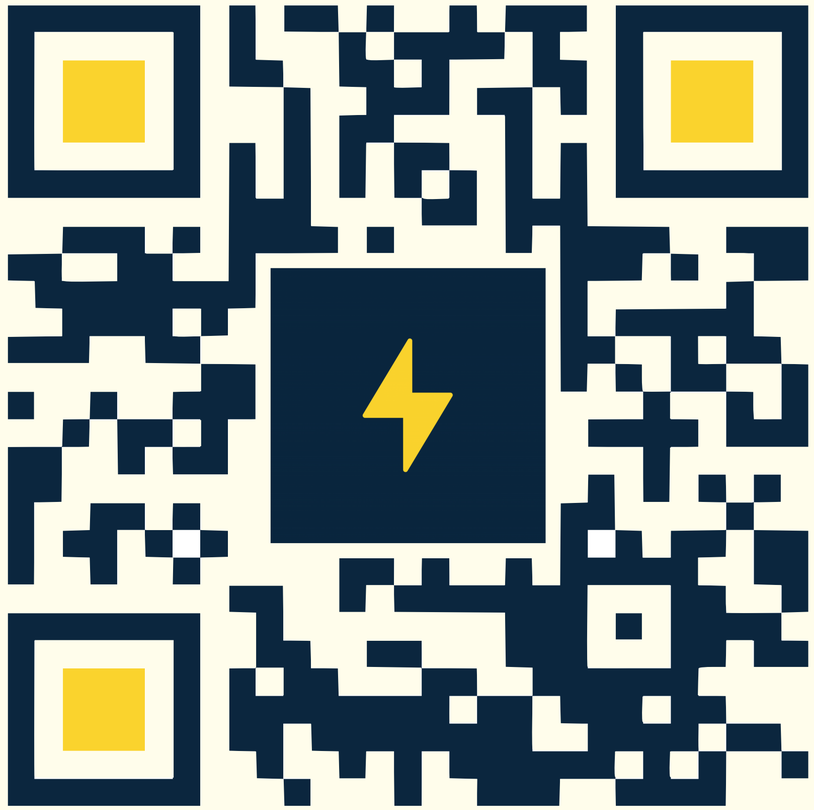


Real-time Ingestion



Real-time analytics

- Latency
- Concurrency
- Freshness
- Apache Pinot



Thank you!

https://stree.ai/slack

@tlberglund