



an
Introduction
to
Apache
Pinot

Tim Berglund, StarTree
@tlberglund

Revolution



DWH



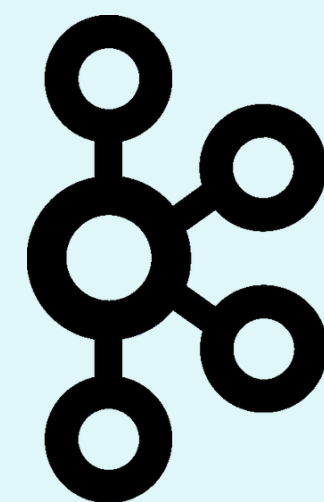
ETL



Monoliths didn't manage
complexity

Batch was too slow for
changing consumer
expectations

DIFFERENTIATED APPLICATION CODE



EVENT SUBSTRATE

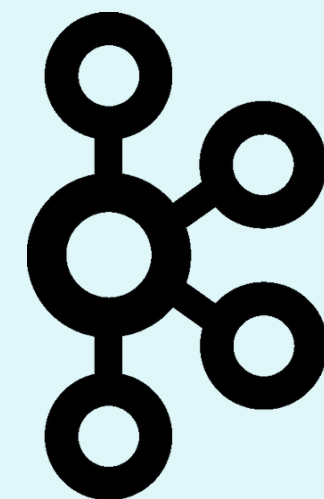
DIFFERENTIATED APPLICATION CODE

Kafka Streams

Kafka Connect+Your Fave DB

Flink

Bespoke solutions

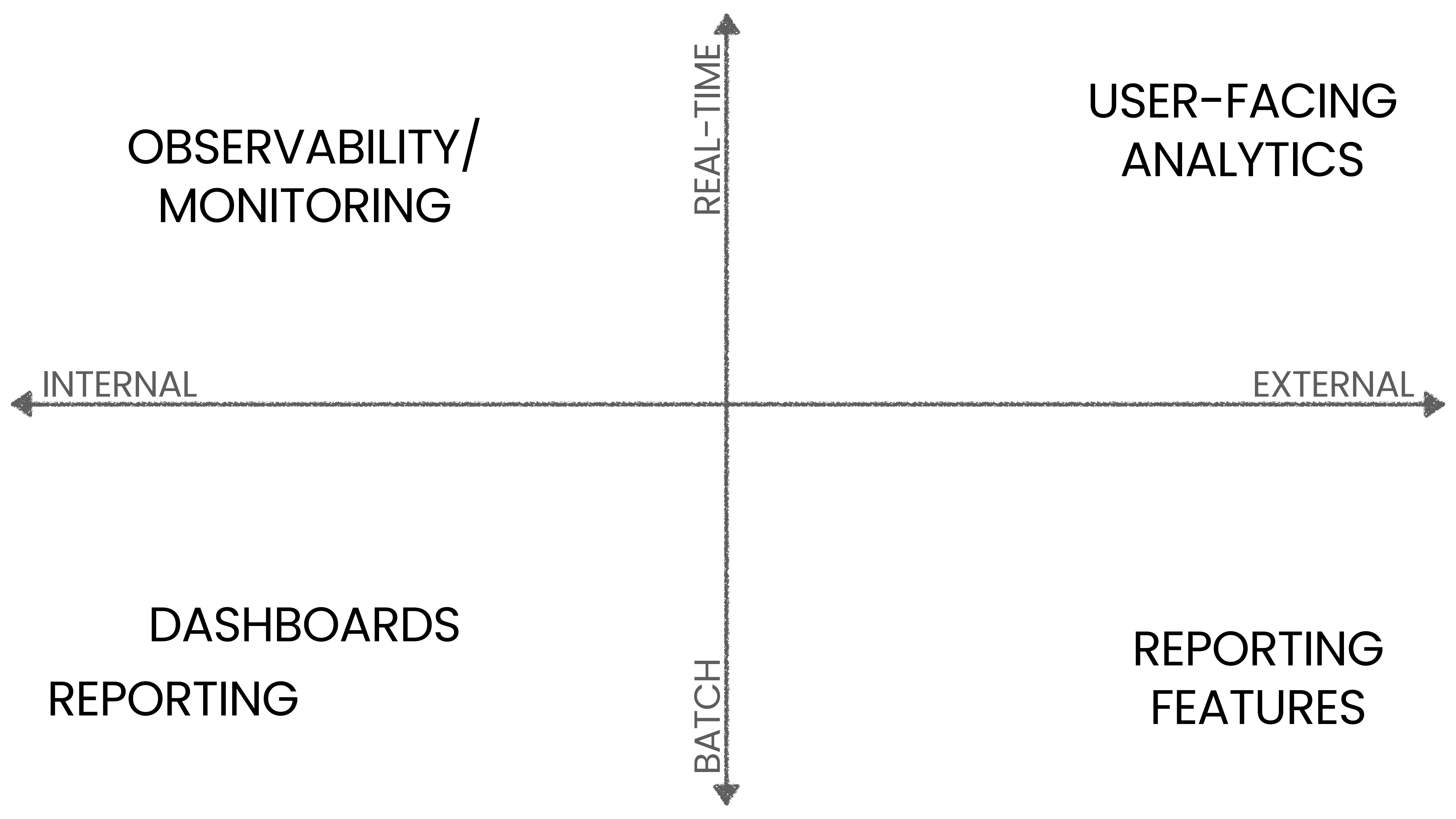


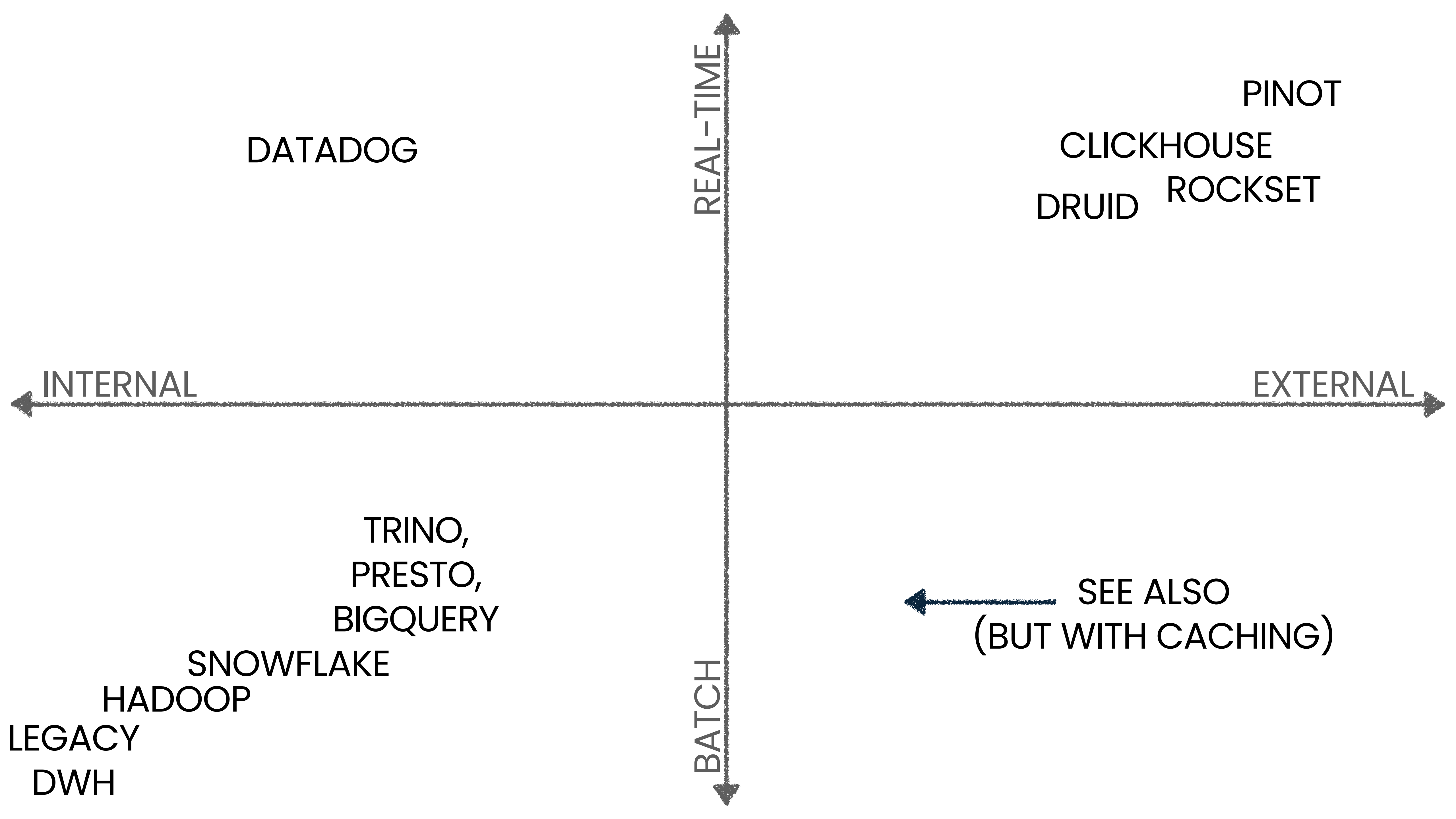
EVENT SUBSTRATE

We need to see what has
happened.

The real-time revolution is not
complete.

a Taxonomy of Analytics





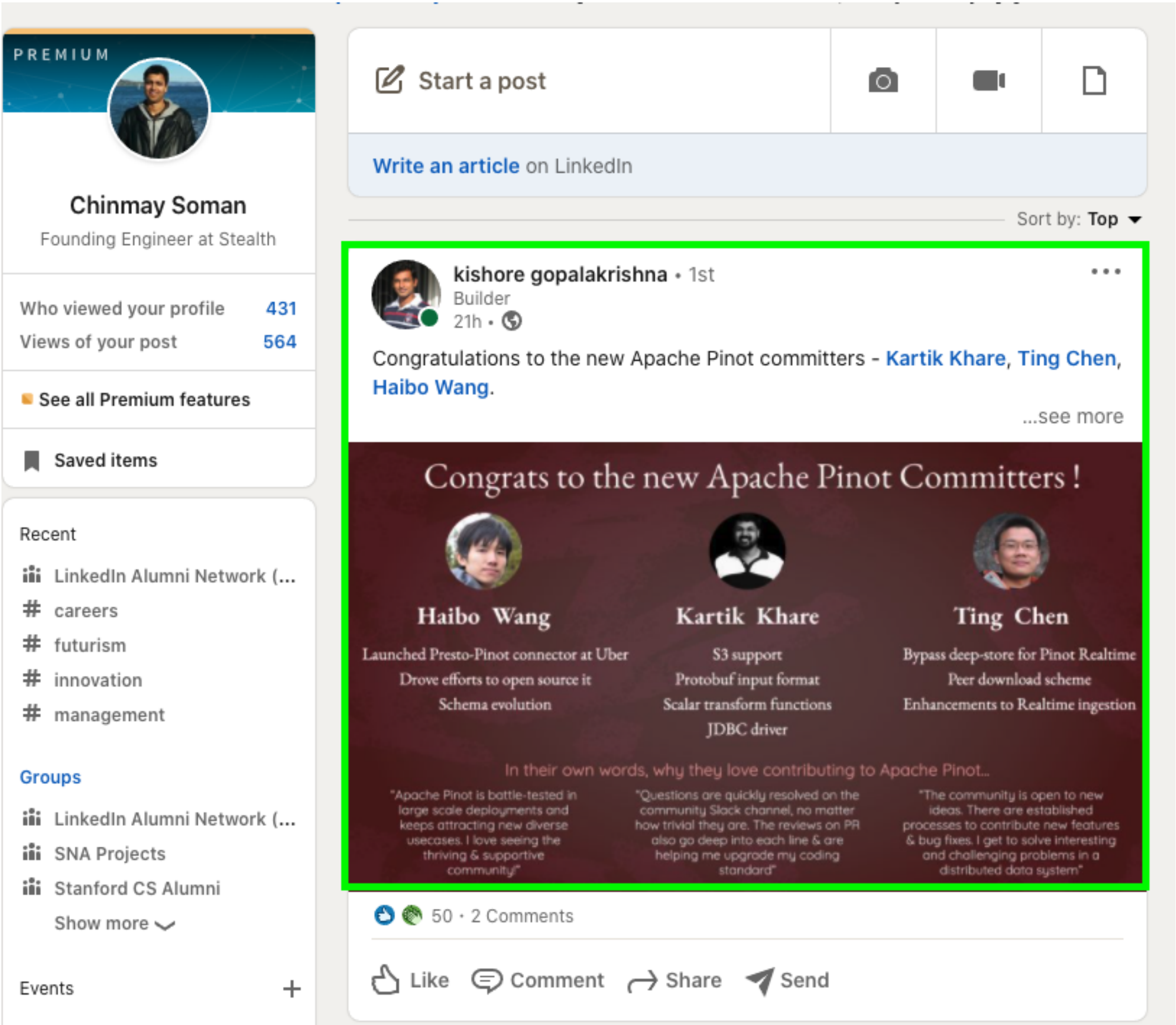
**Who does real-time
analytics?**

Who Viewed My Profile?



Total users	700 Million+
QPS	100,000s
Latency SLA	< 100 ms p99th
Freshness	Seconds

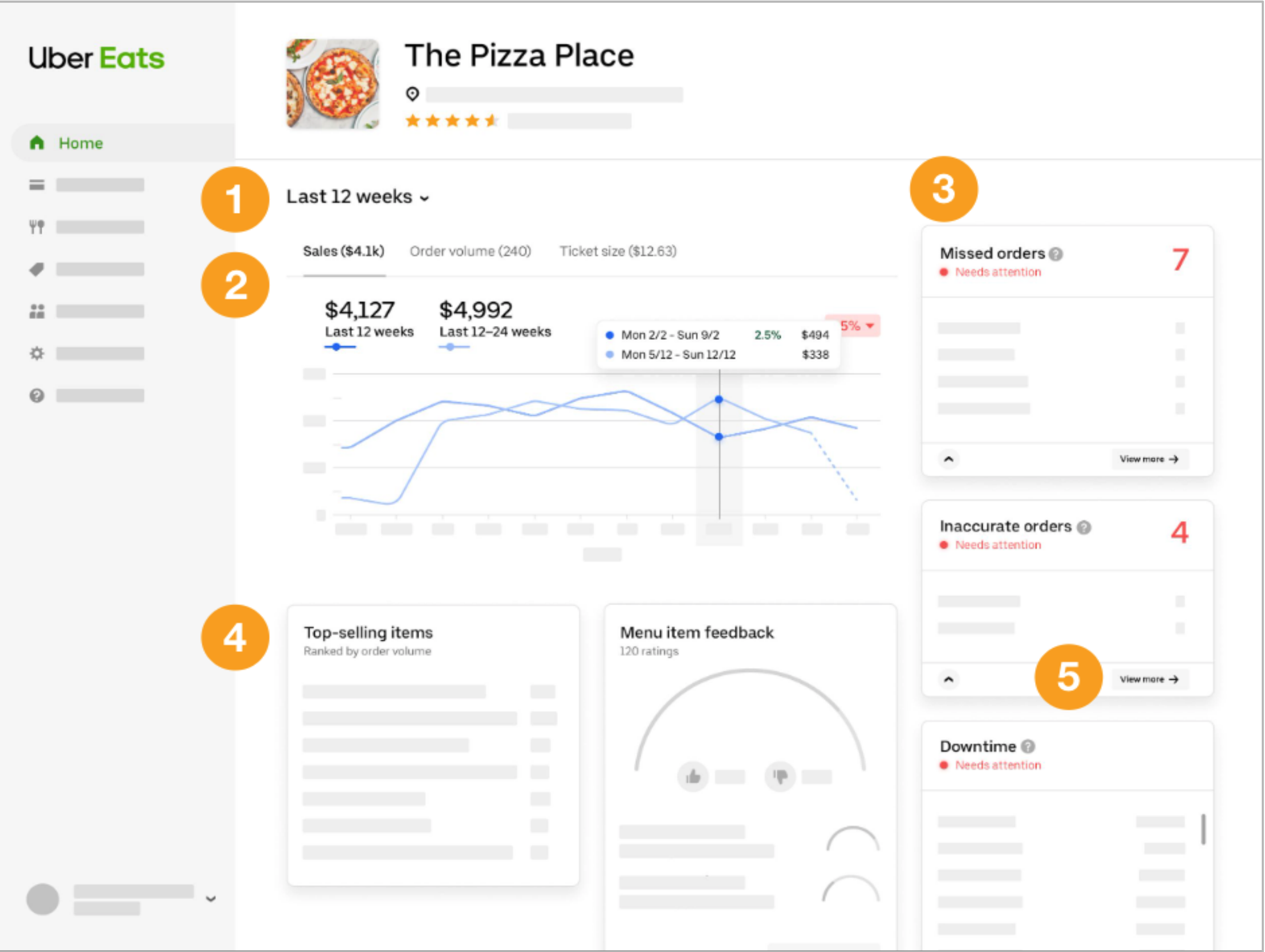
LinkedIn Feed



```
SELECT sum(count) from T
WHERE memberId = X
AND item in (list of 500-1500 items)
AND time >= (now - 14 days)
GROUP BY action, item, position, time
```

Total users	700 Million
QPS	100,000
Latency SLA	< 100 ms p99th
Freshness	Seconds

Uber Eats Restaurant Manager



- Identify surge in realtime
- Detect missed/inaccurate orders in realtime

Total users	500000+
QPS	100s
Latency SLA	< 100 ms p99th
Freshness	Seconds - Minutes

Data Model

Query Language

Pinot Architecture

Batch Ingestion

```
[
  ...,
  {
    "ip": "111.173.165.103",
    "userid": 10,
    "remote_user": "-",
    "time": "3271",
    "_time": 3271,
    "request": "GET /site/user_status.html HTTP/1.1",
    "status": "406",
    "bytes": "1289",
    "referrer": "-",
    "agent": "Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/
59.0.3071.115 Safari/537.36"
  },
  ...
]
```

Batch Ingestion

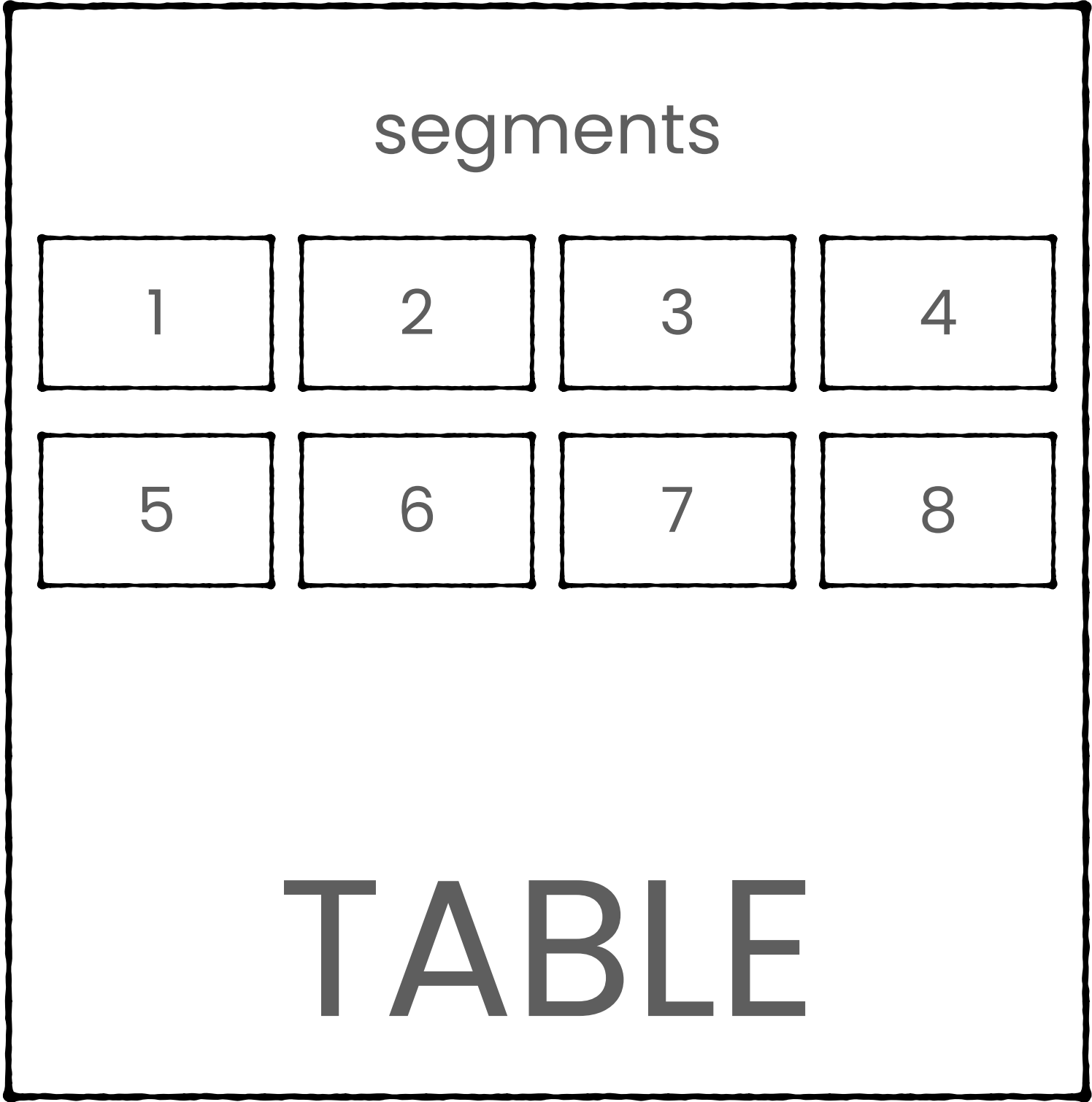
[
 {...},
 {...},
 {...},
 {...}
]



TABLE

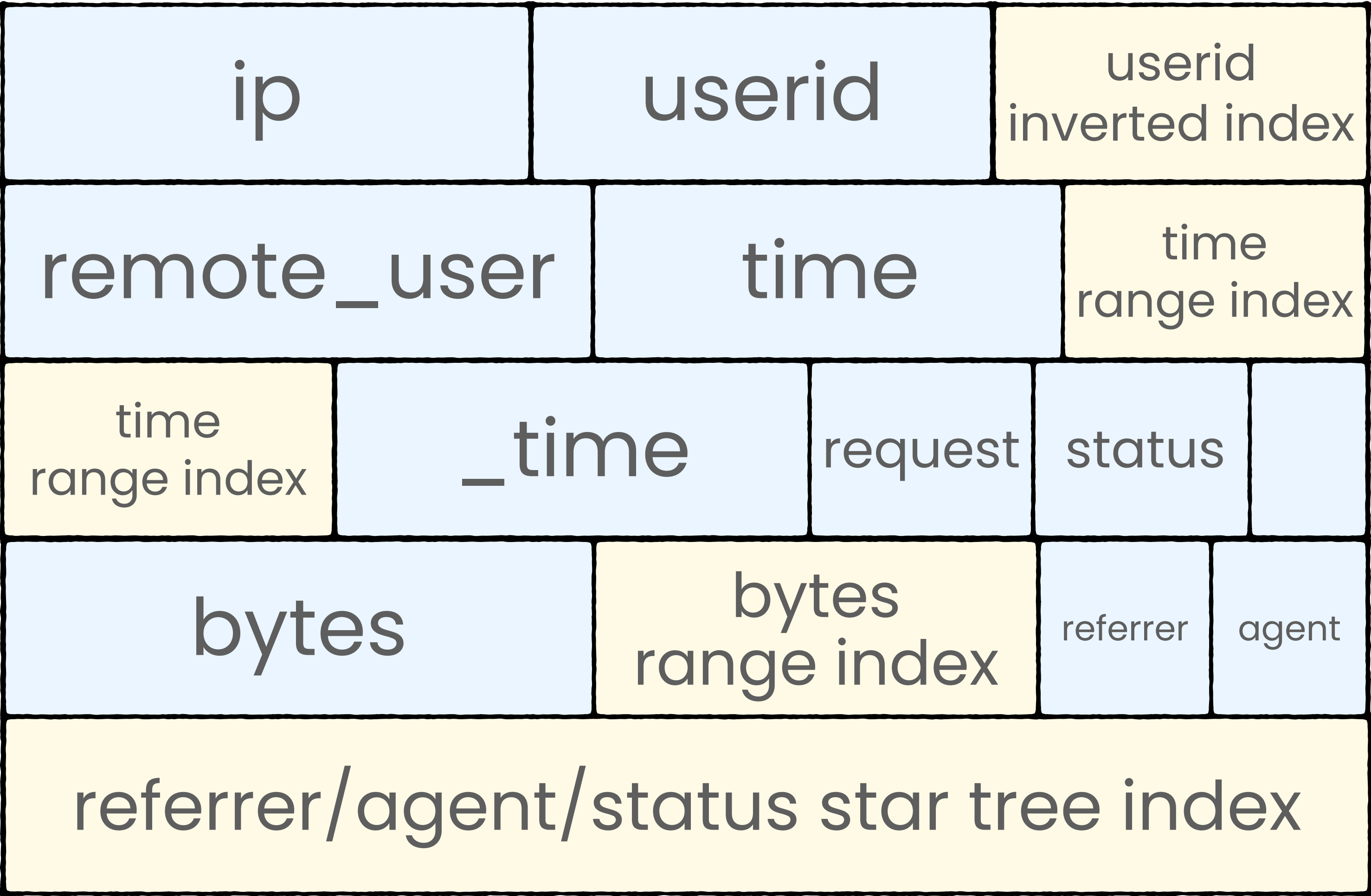
Batch Ingestion

[
{ . . . },
{ . . . },
{ . . . },
{ . . . }
]

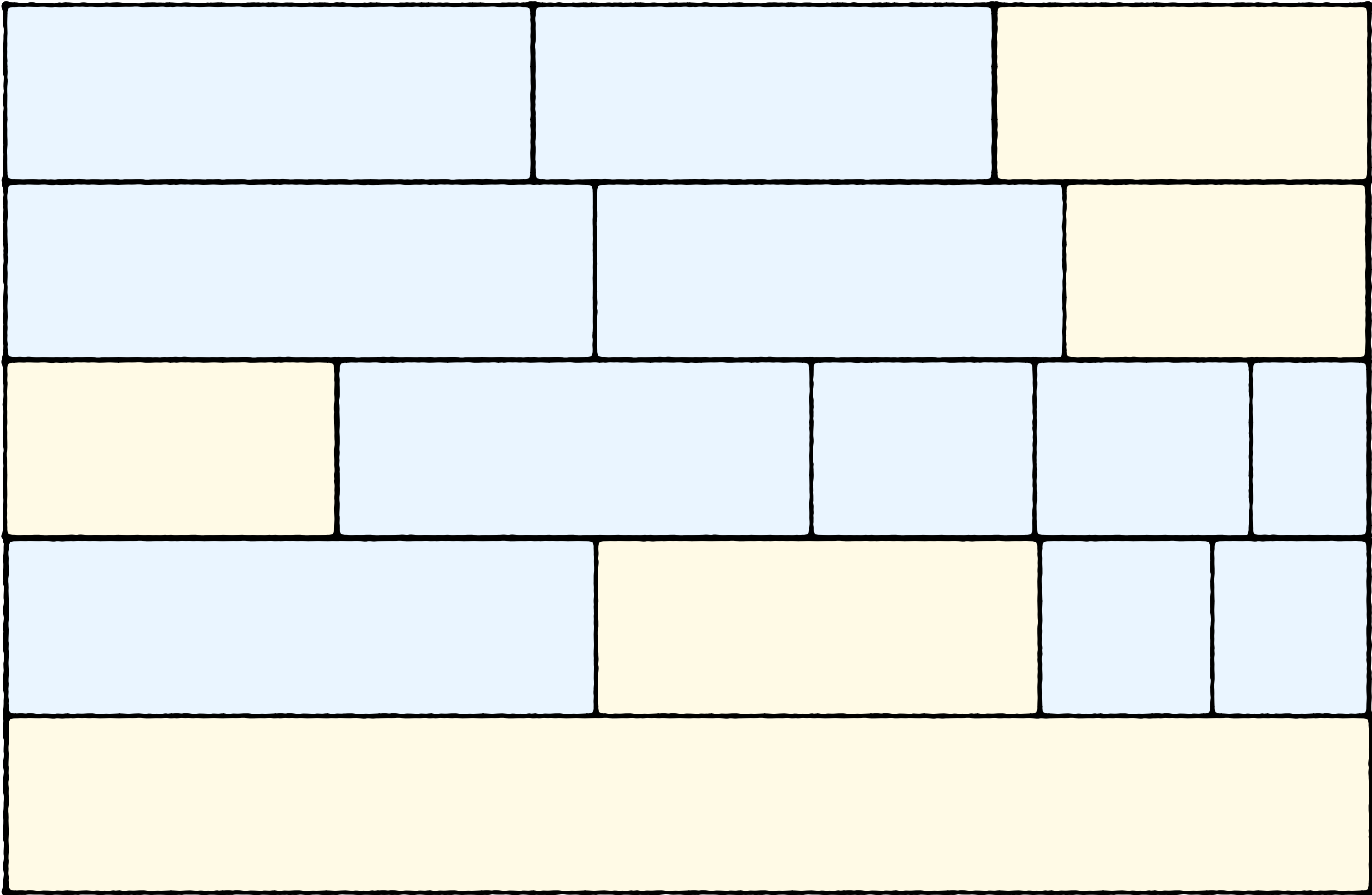


Segment Structure

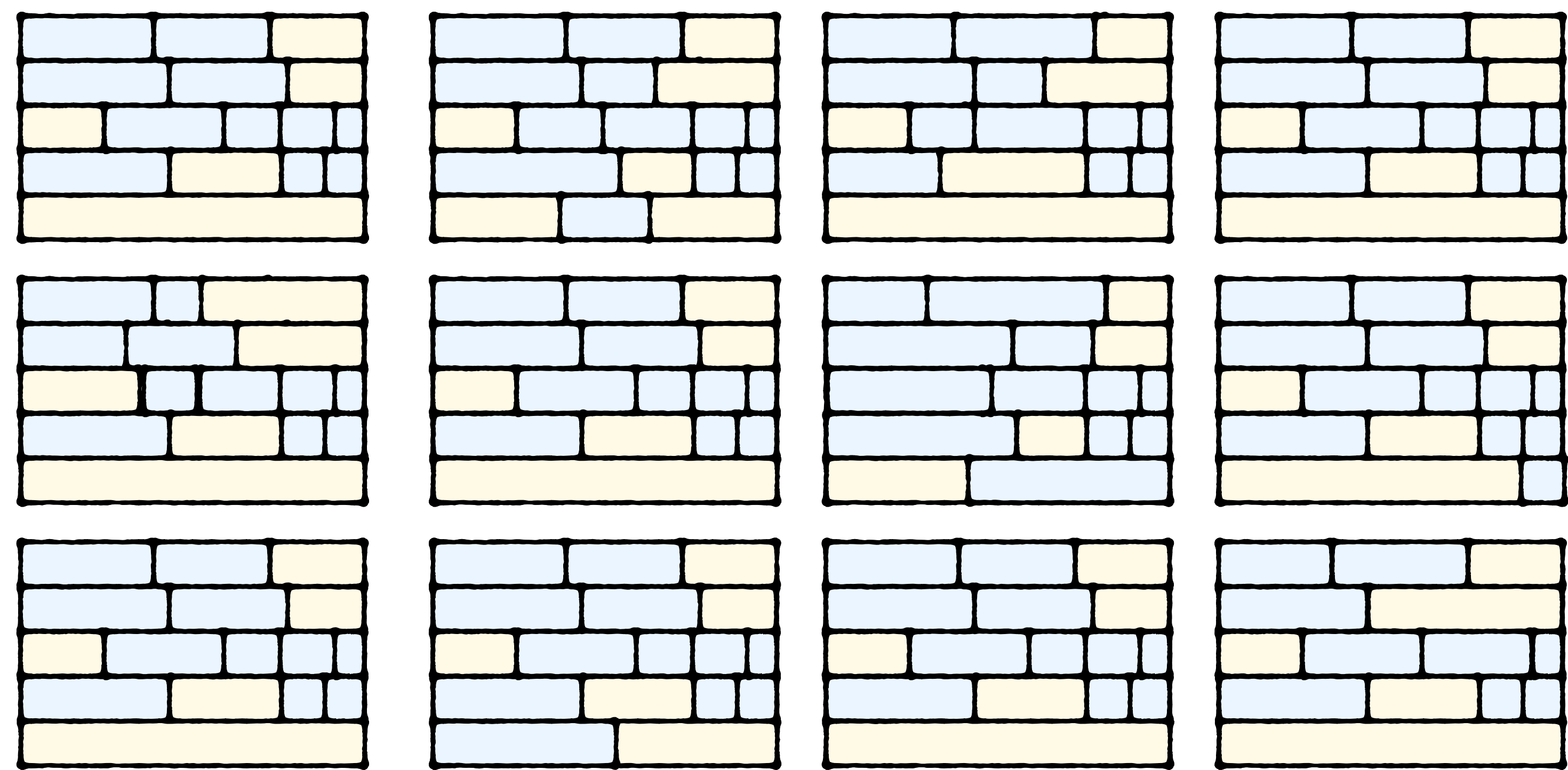
```
....,  
{  
  "ip": "111.173.165.103",  
  "userid": 10,  
  "remote_user": "-",  
  "time": "3271",  
  "_time": 3271,  
  "request": "GET",  
  "status": "406",  
  "bytes": "1289",  
  "referrer": "-",  
  "agent": "Mozilla/5.0"  
},  
....,
```



Segments



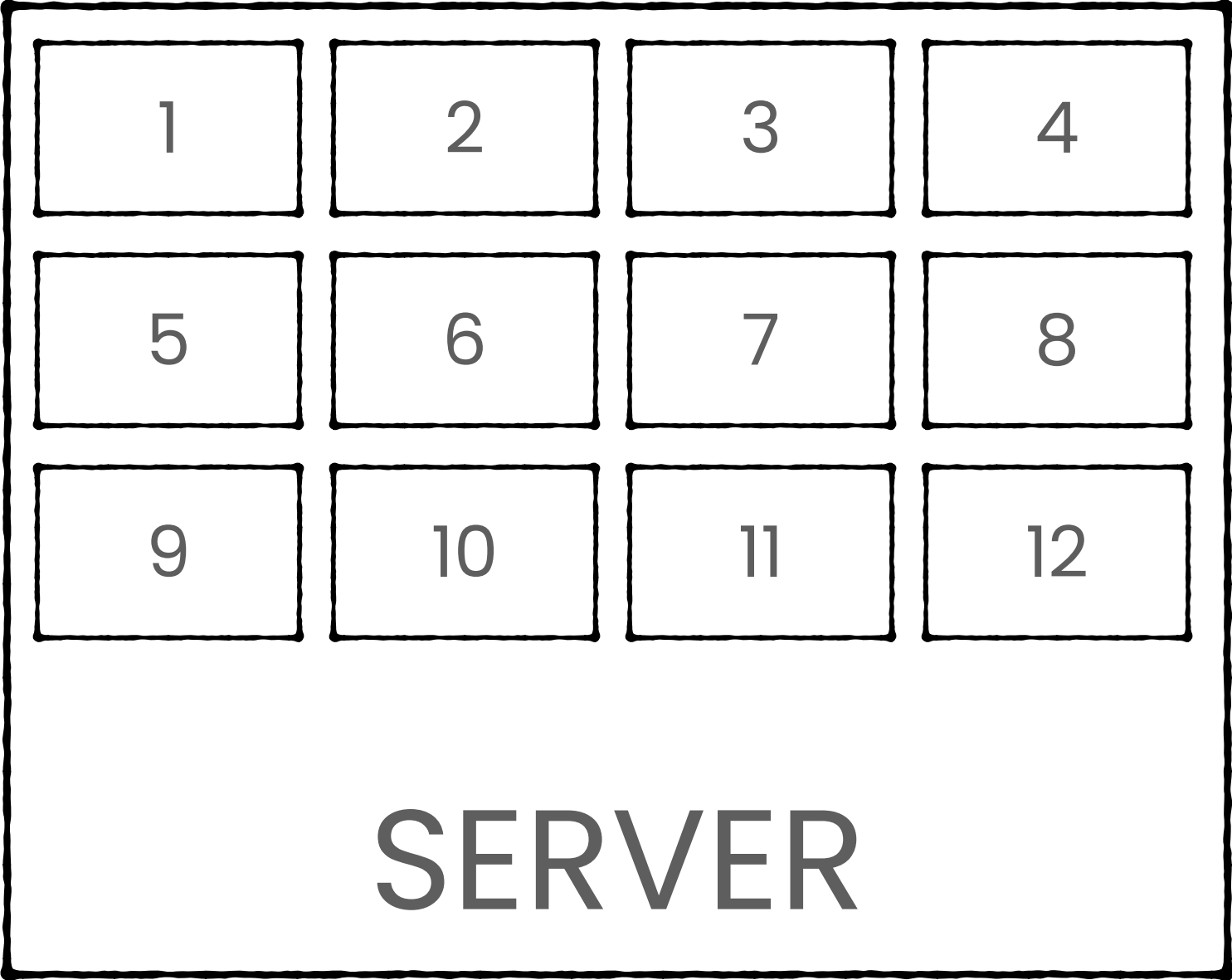
Segments



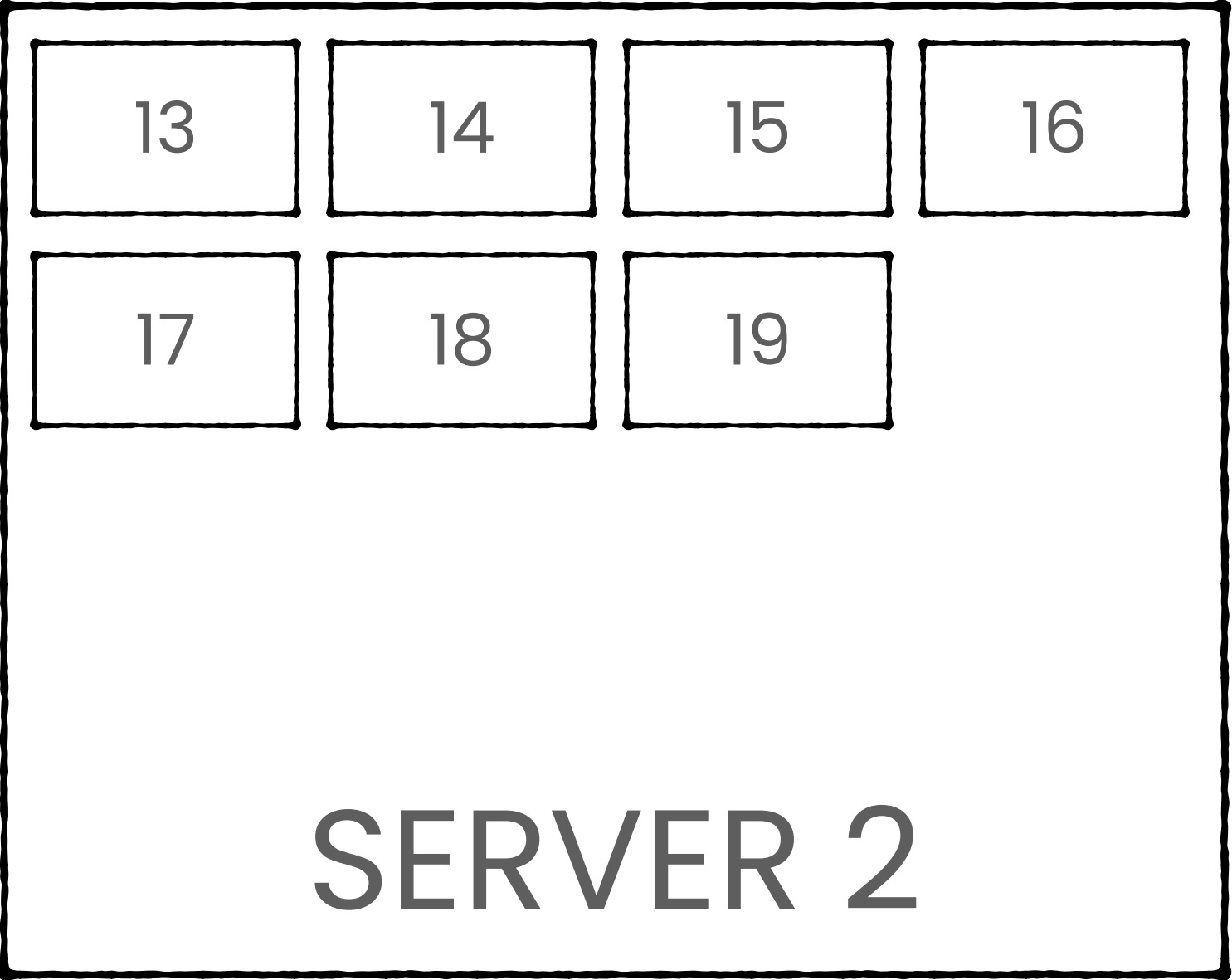
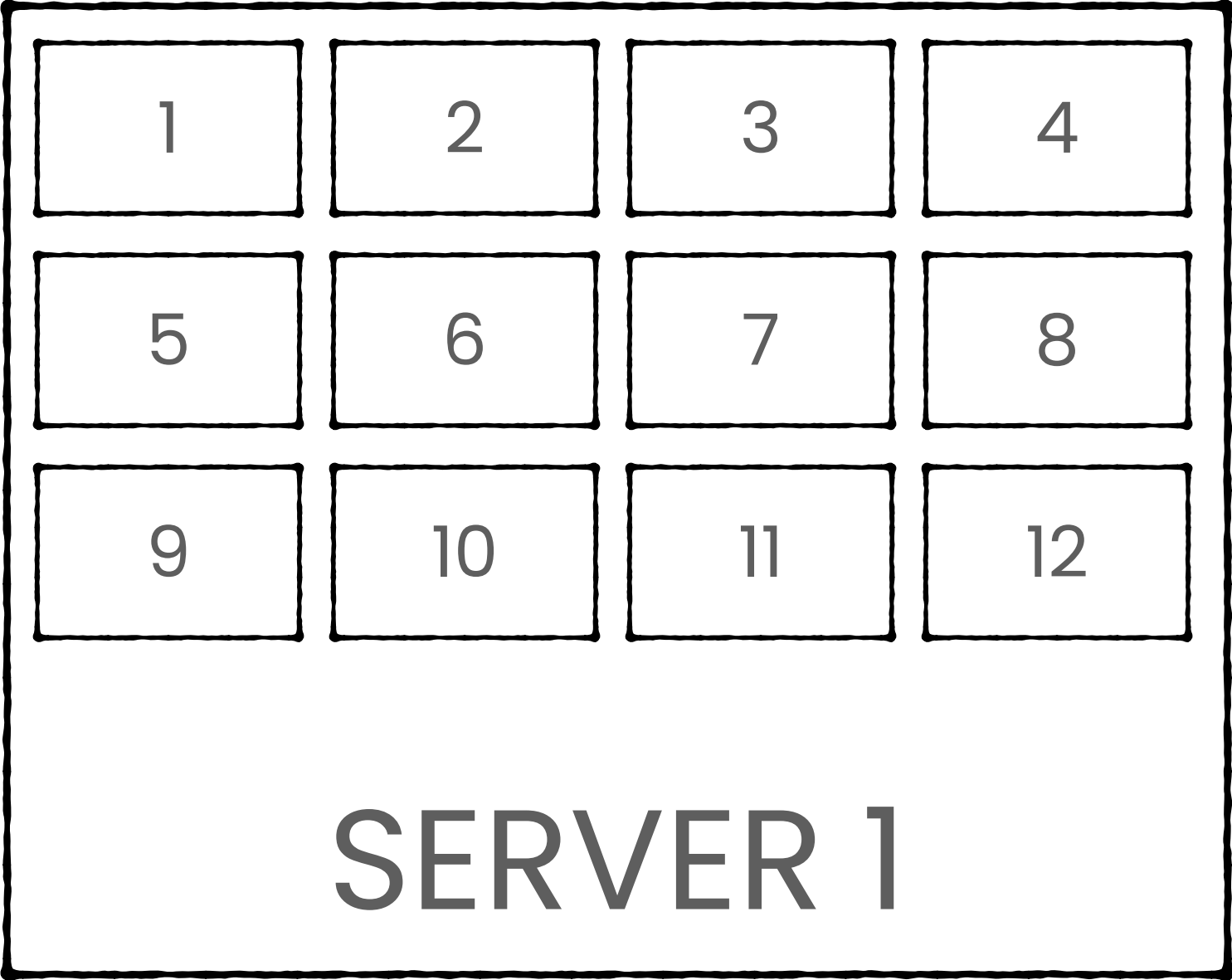
Segments

1	2	3	4
5	6	7	8
9	10	11	12

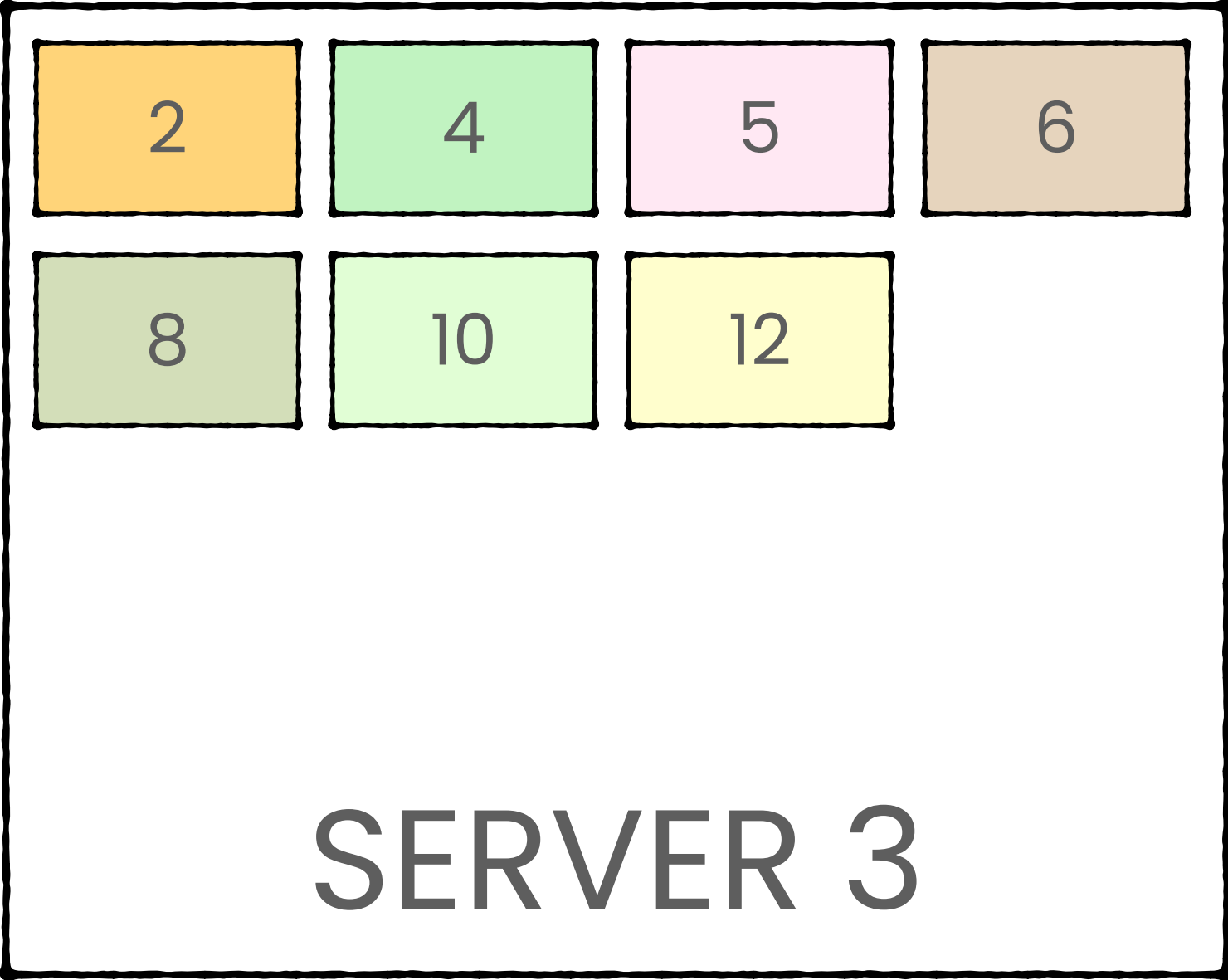
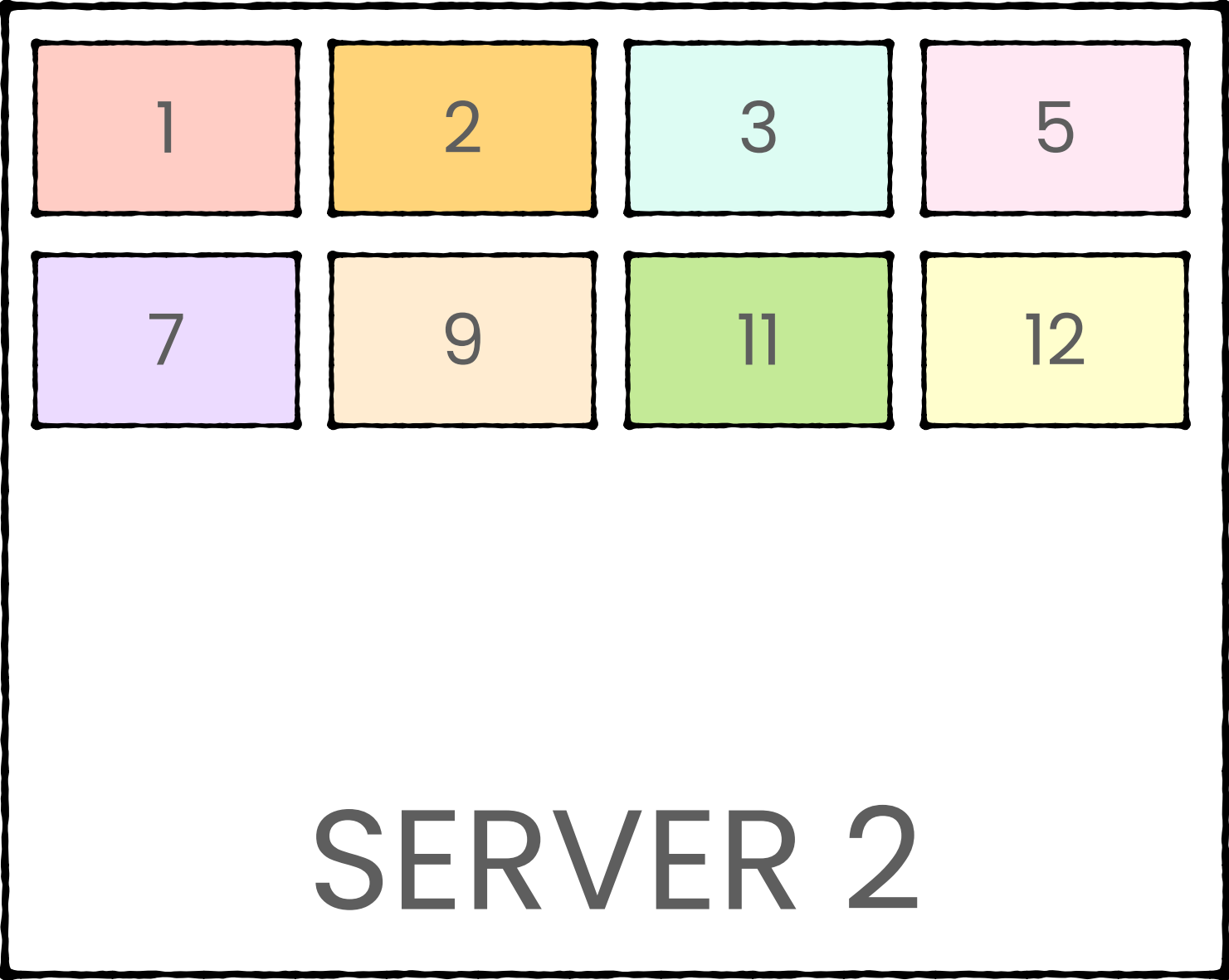
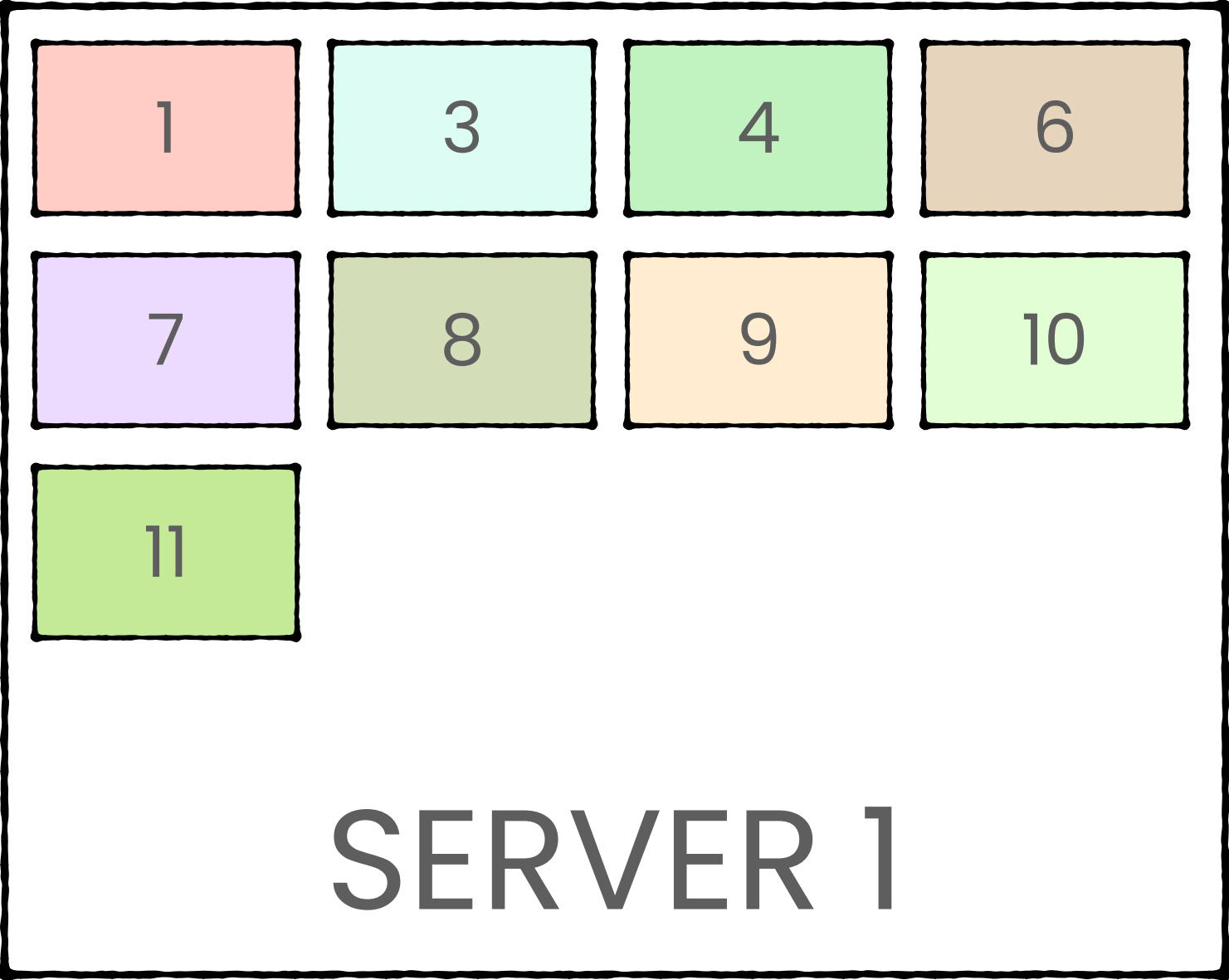
Servers



Servers

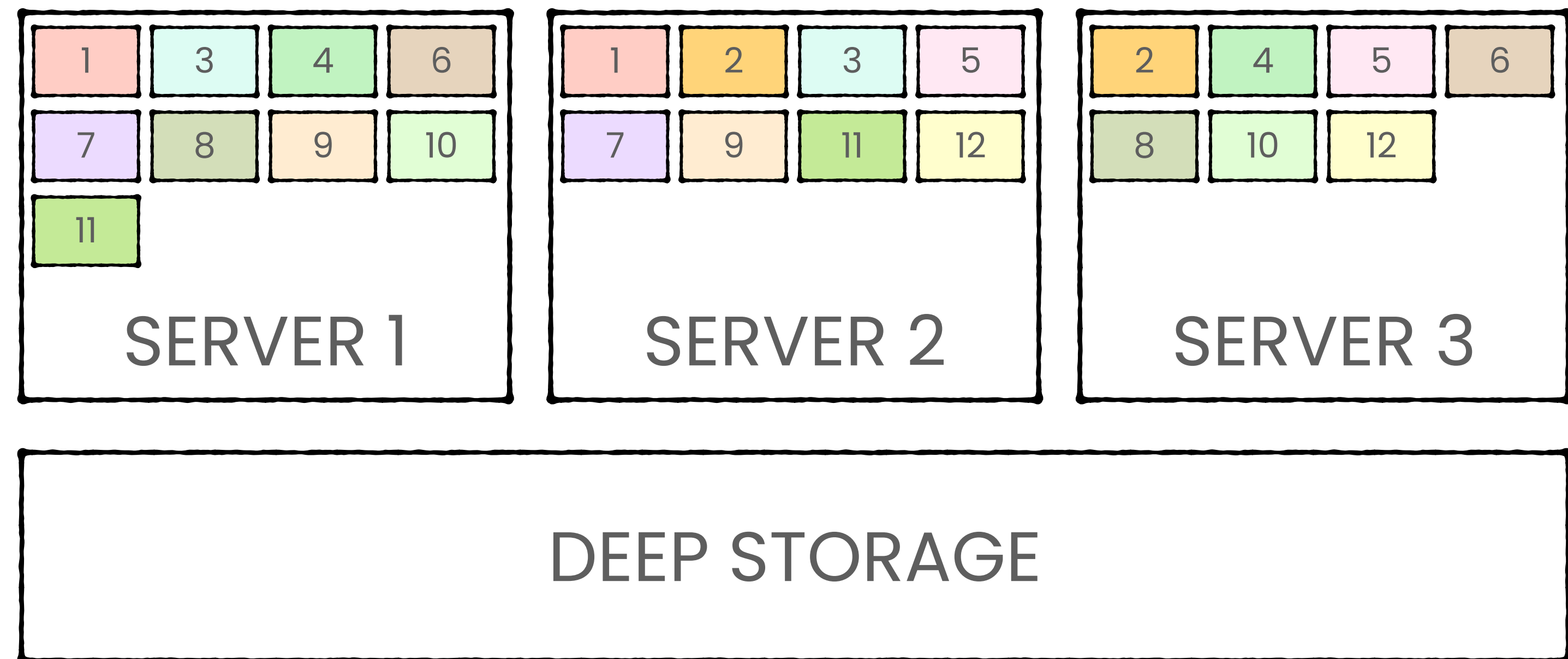


Replication

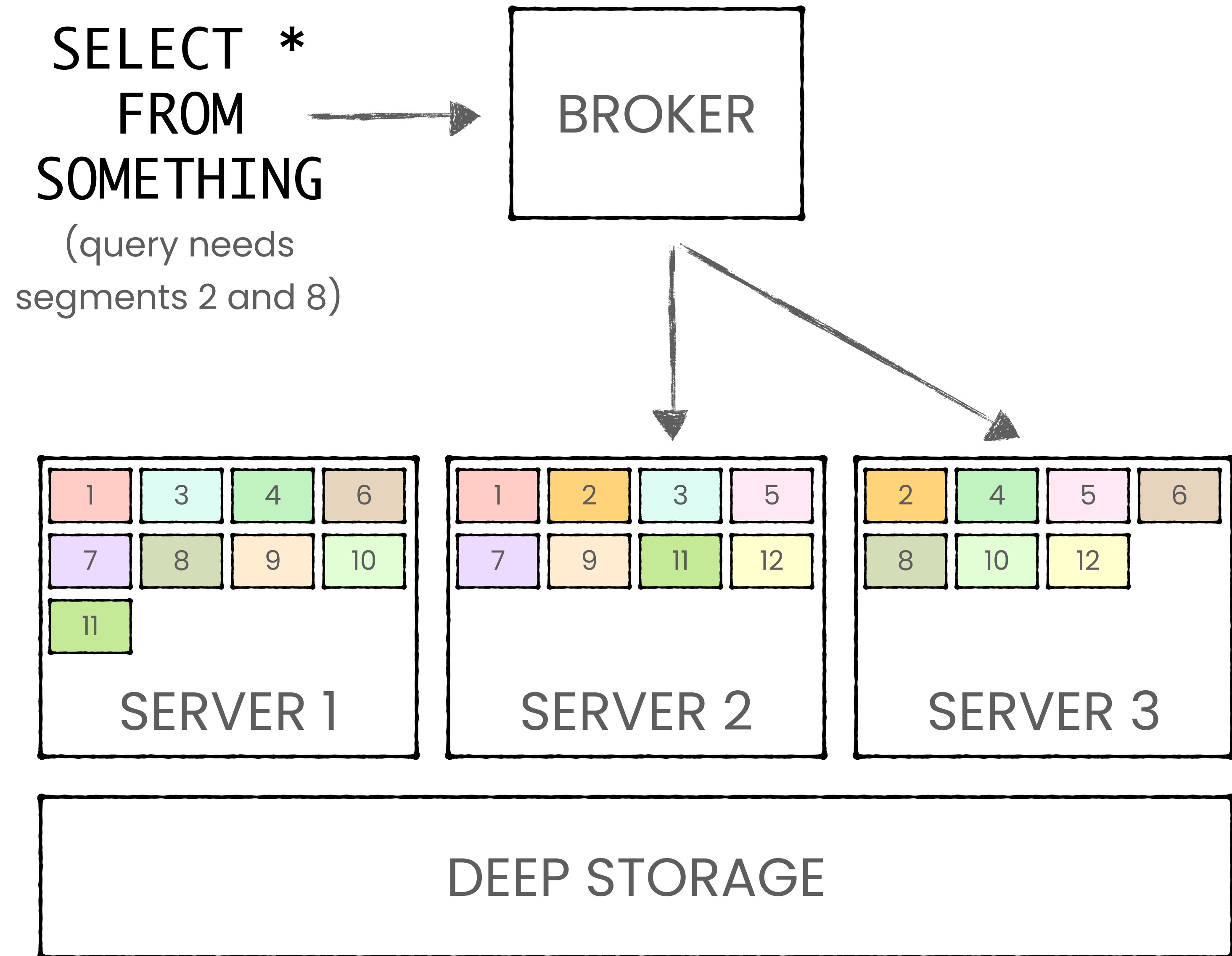


Servers

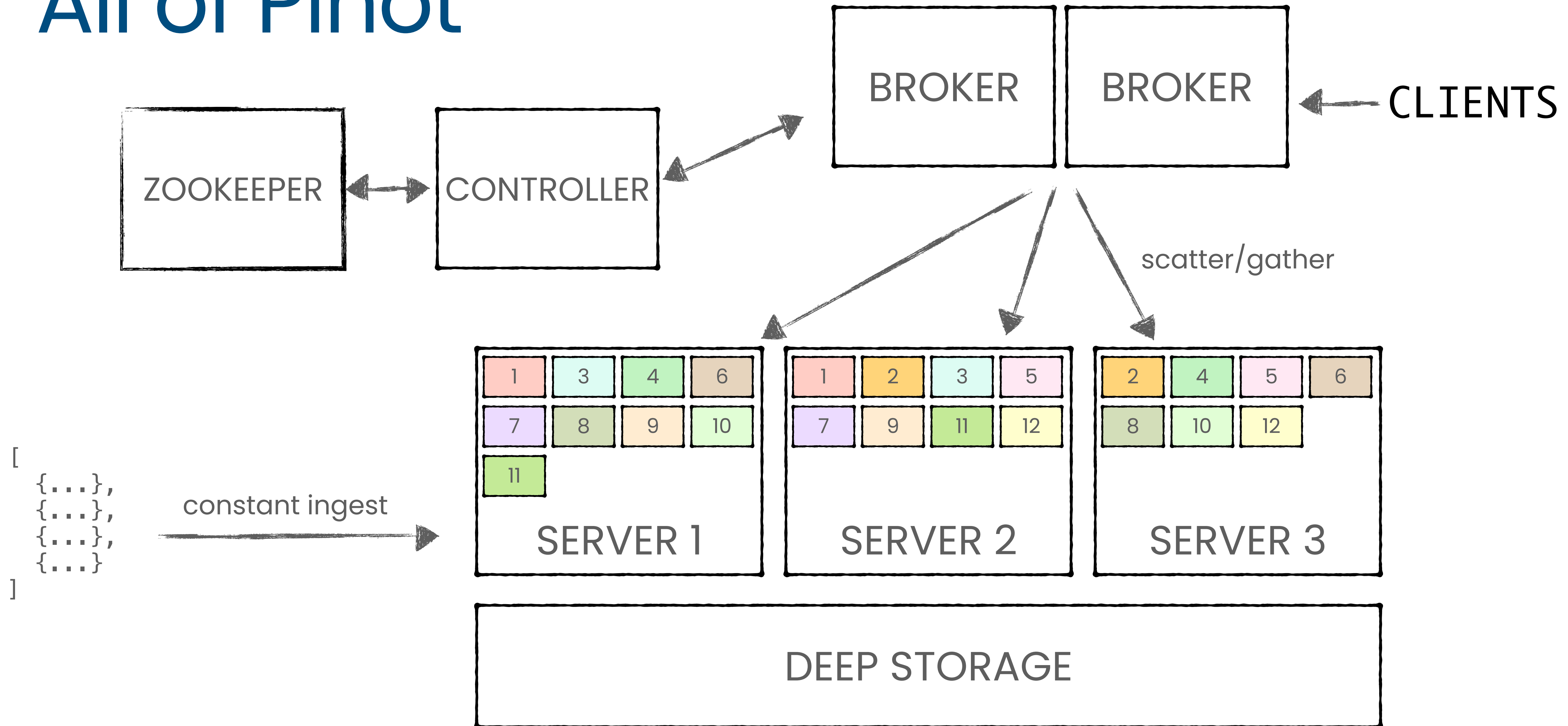
- Segment storage
- Actual processing of queries
- Scatter/gather among segments
- Servers are the *real* heroes here
- Segments can be put in the cloud



Brokers



All of Pinot



Controller

- Home for the Apache Helix cluster manager
- Cache of global metadata (configs and schemas) from ZooKeeper
- Config admin endpoint
- Server/segment mapping
- Endpoint for segment uploads

“But wait, what about streaming data?”

—Pretty much everybody

Batch Ingestion

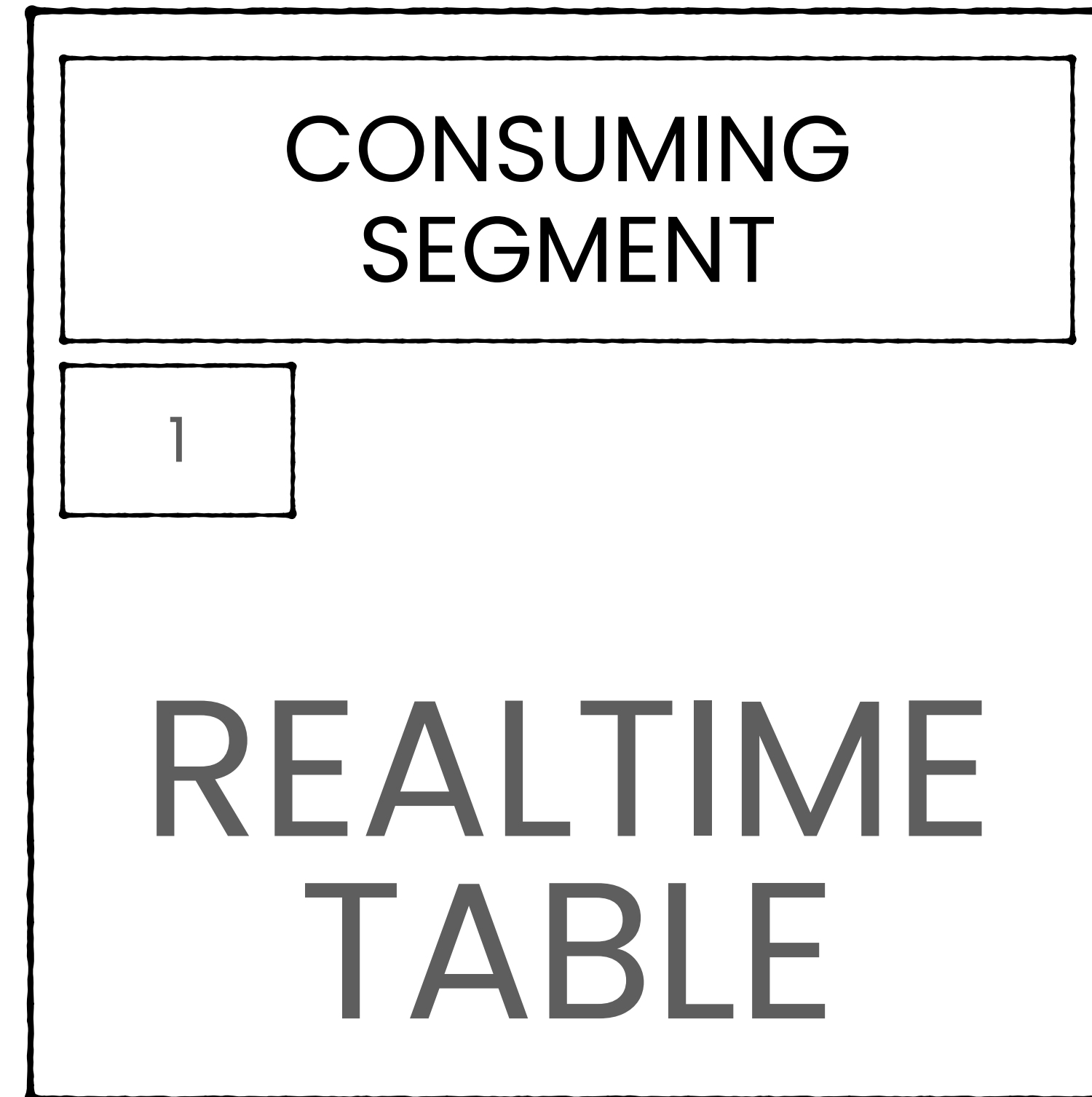
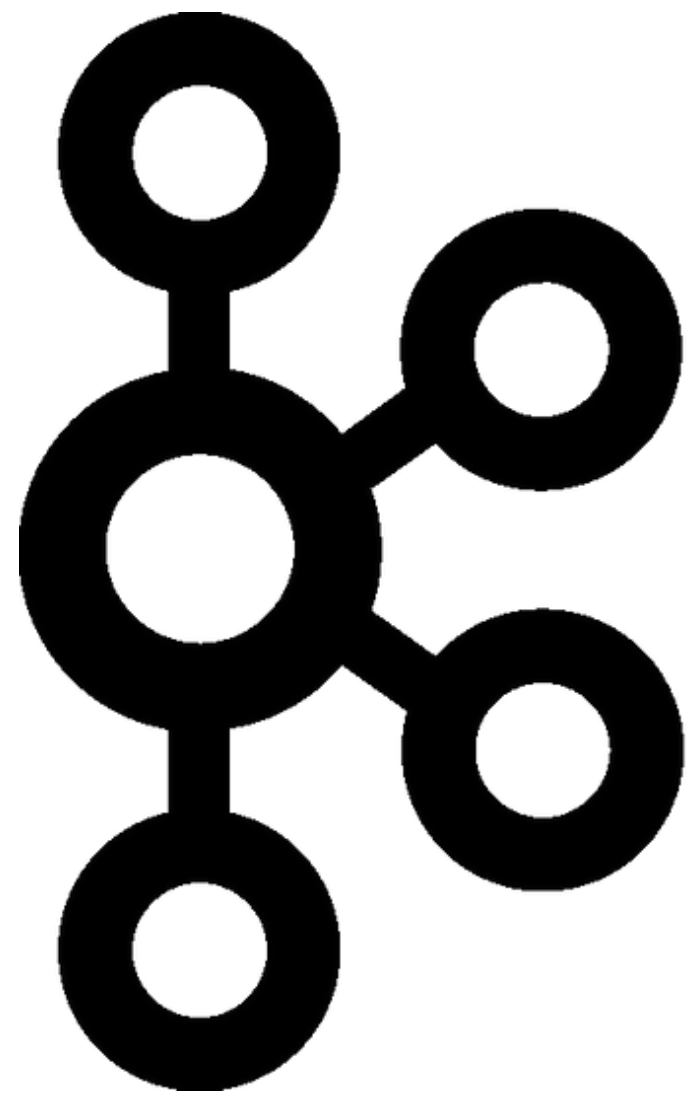
```
....,  
{  
  "ip": "111.173.165.103",  
  "userid": 10,  
  "remote_user": "-",  
  "time": "3271",  
  "_time": 3271,  
  "request": "GET",  
  "status": "406",  
  "bytes": "1289",  
  "referrer": "-",  
  "agent": "Mozilla/5.0"  
},  
....,
```



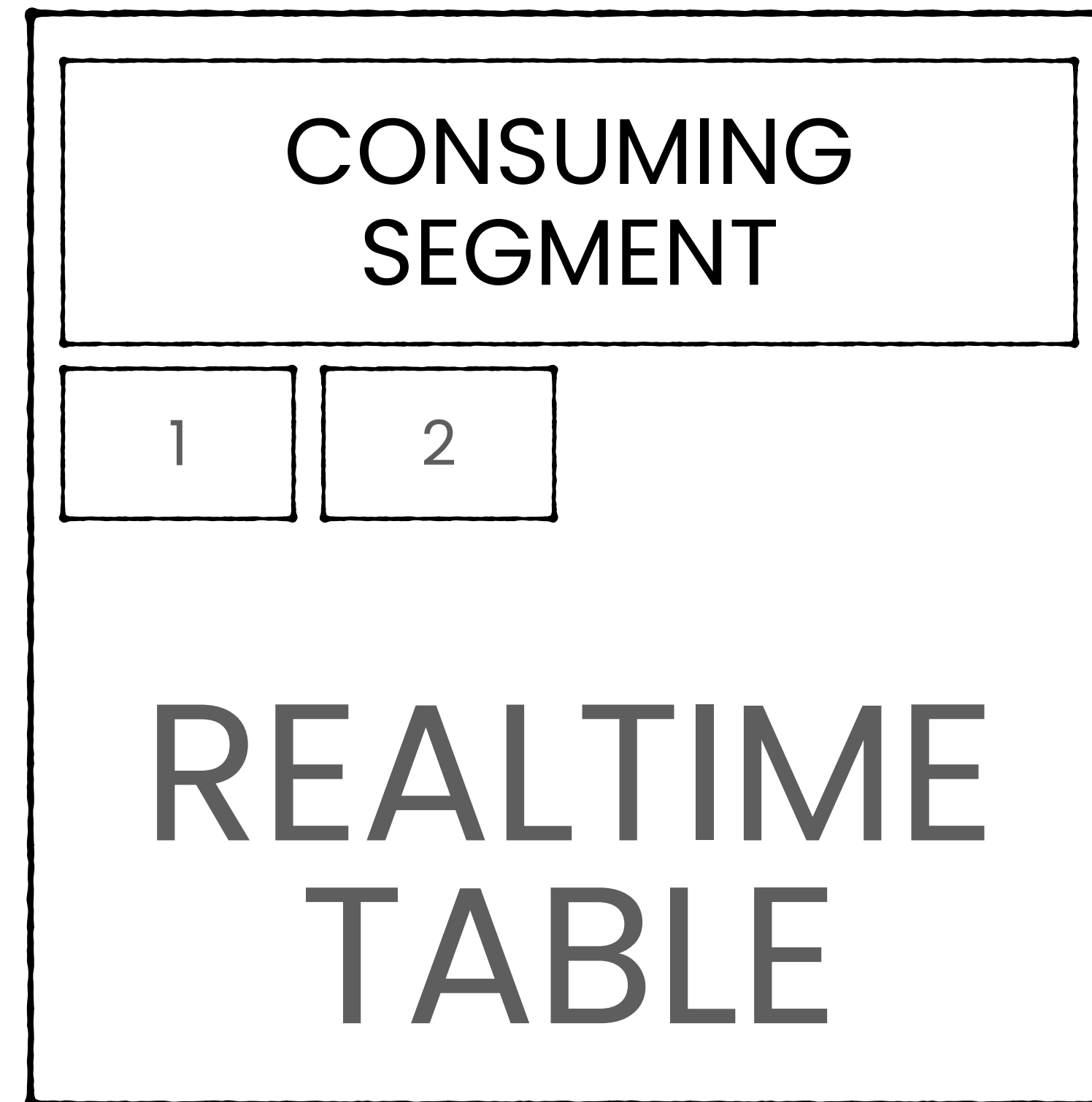
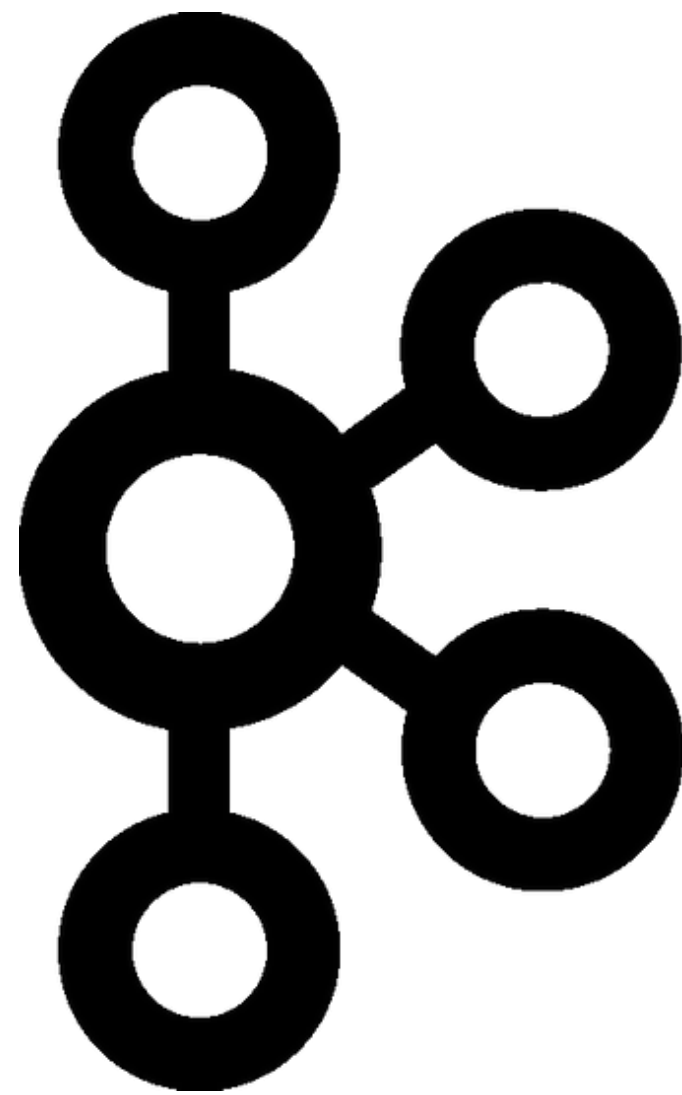
1	2	3	4
5	6	7	8

OFFLINE
TABLE

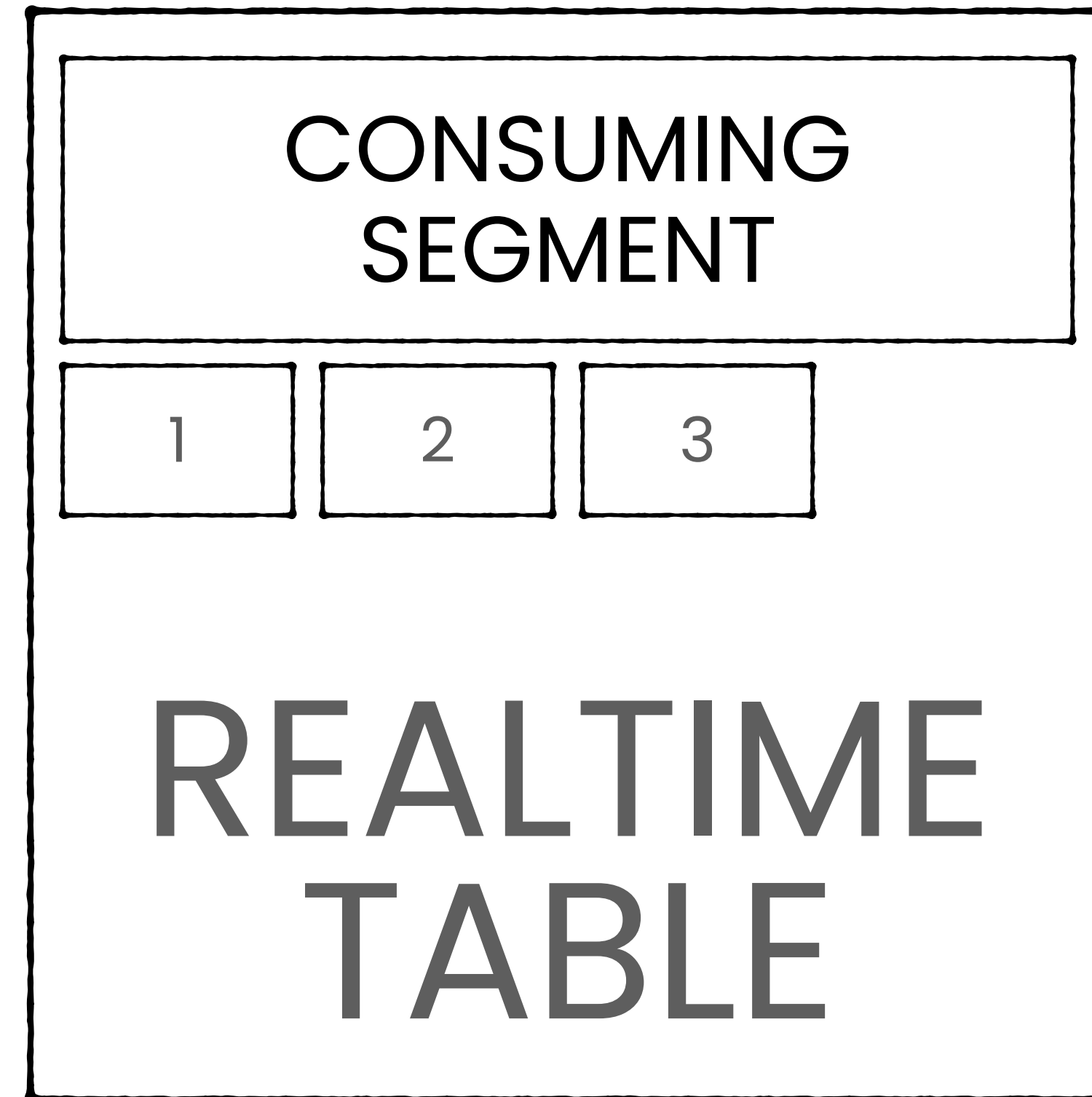
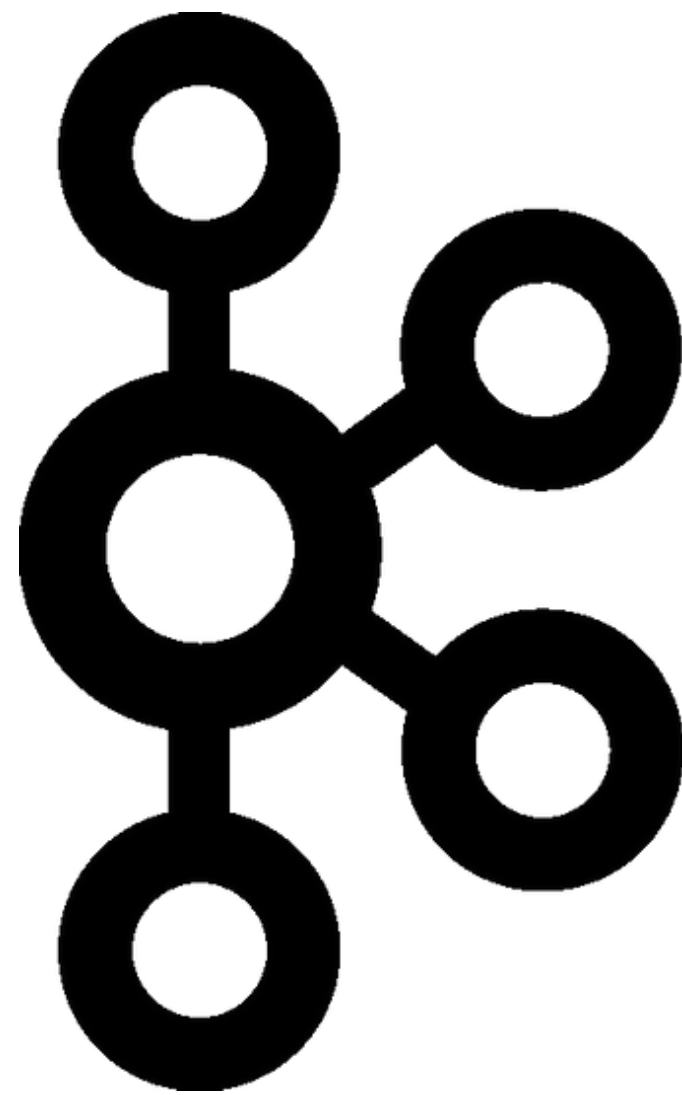
Realtime Ingestion



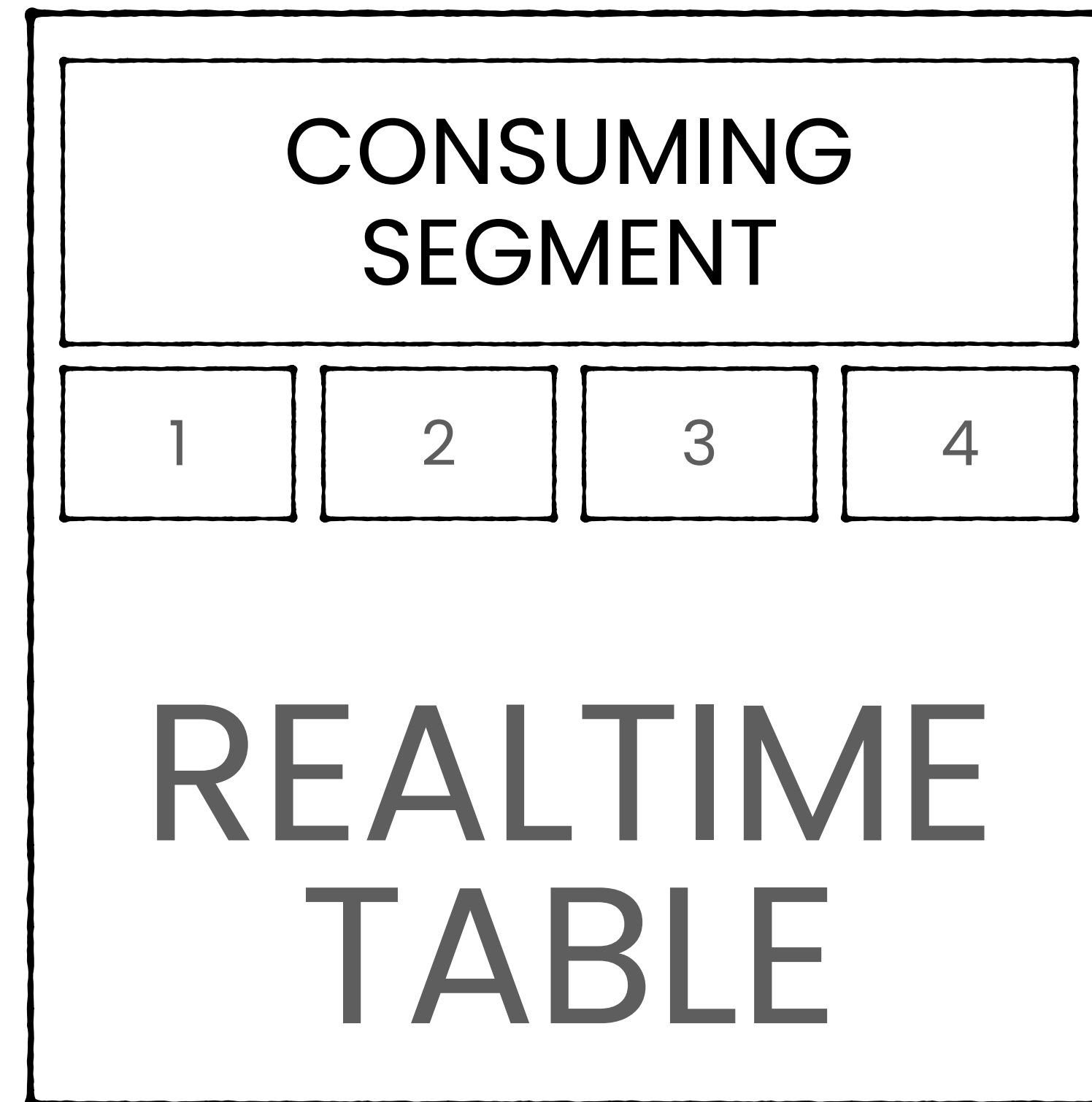
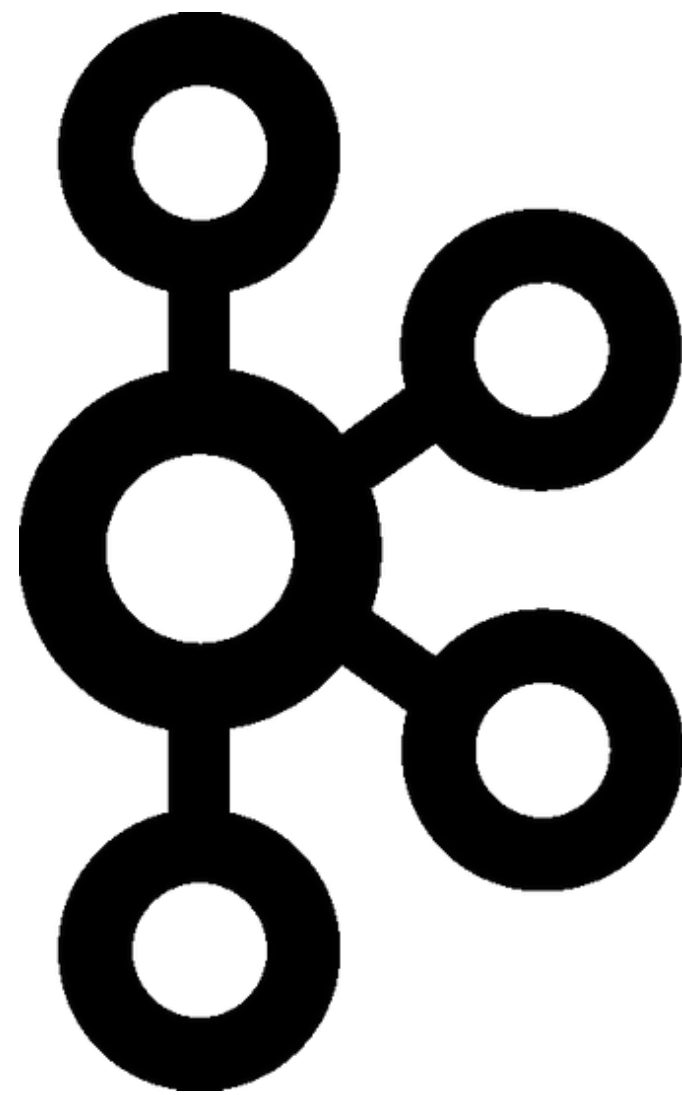
Realtime Ingestion



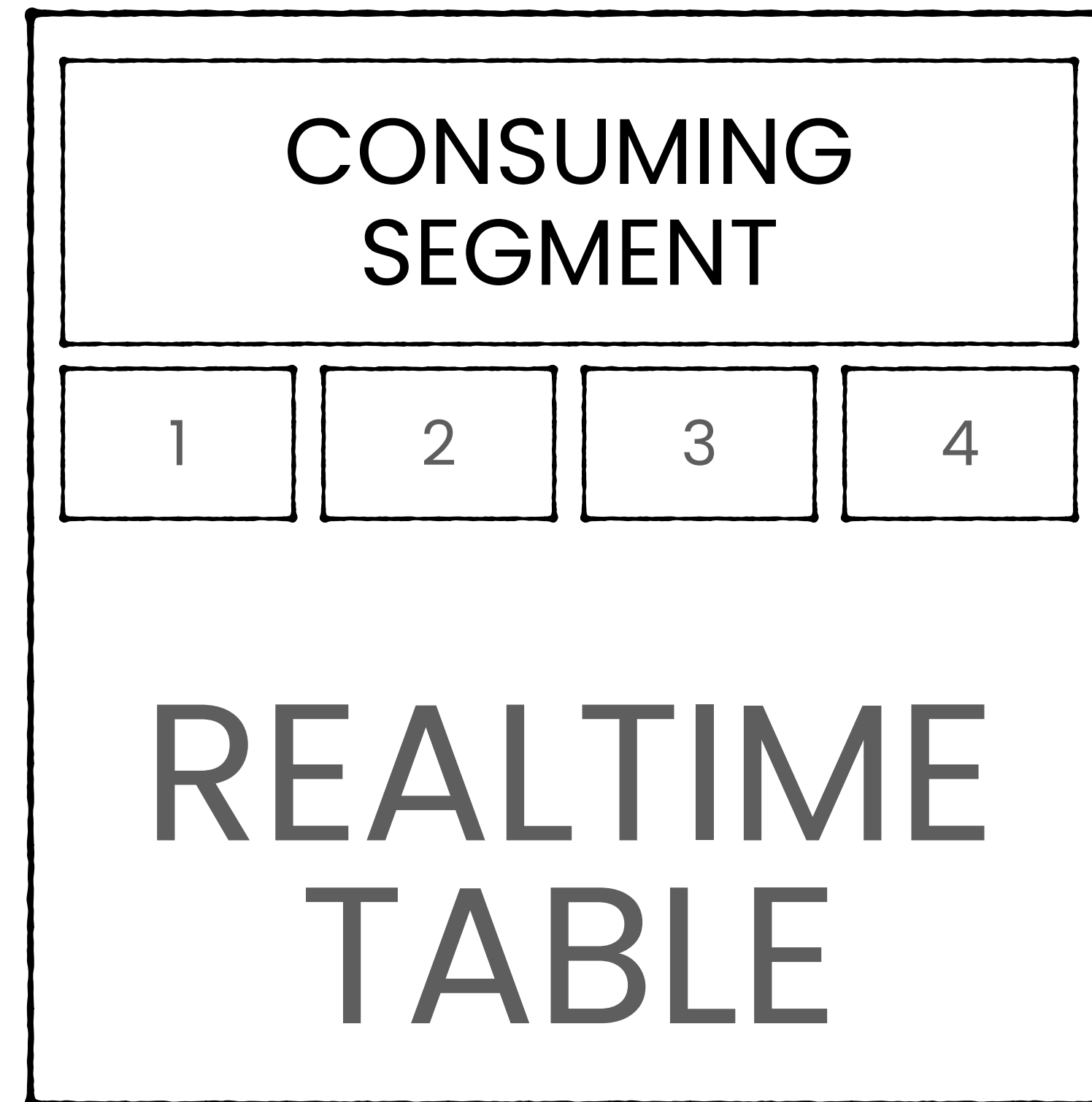
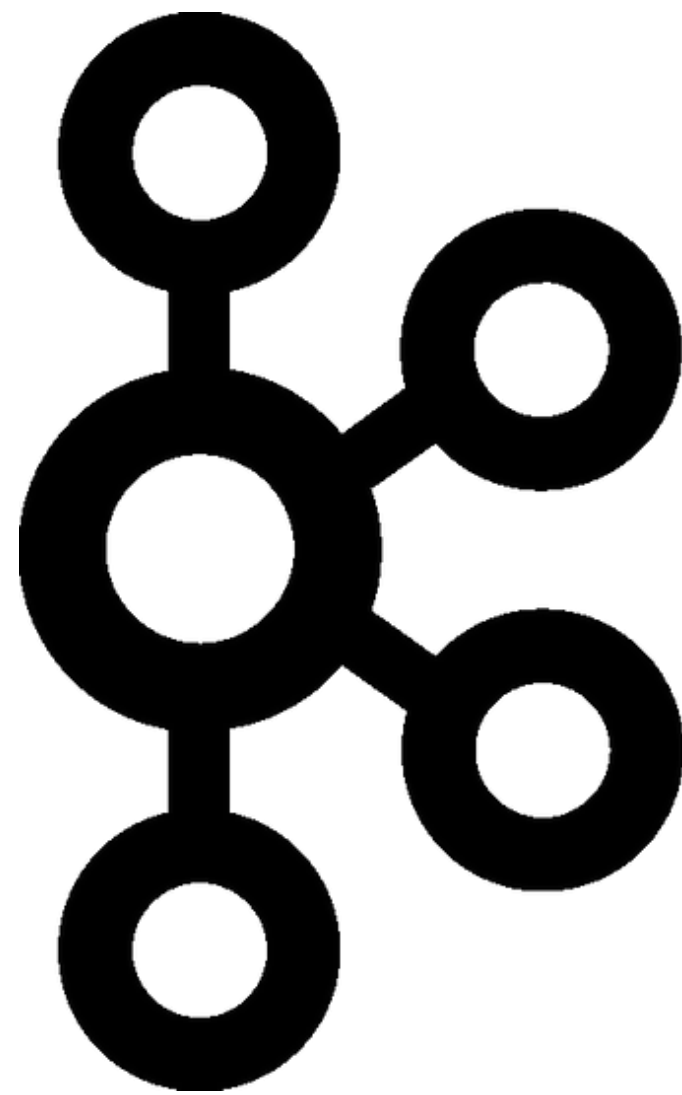
Realtime Ingestion



Realtime Ingestion



Realtime Ingestion



Indexes

Indexes

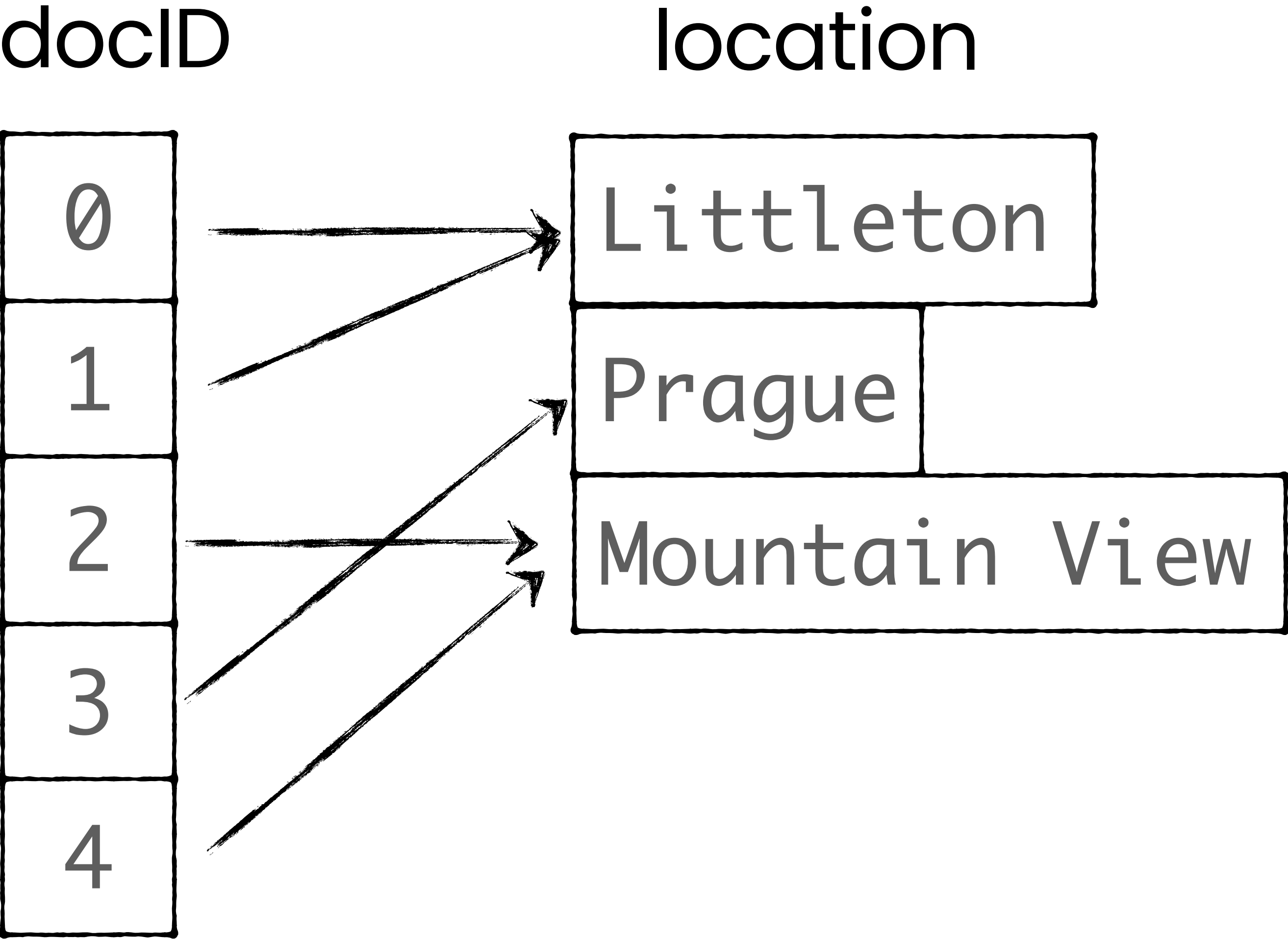
- Forward
- Inverted
- Bloom
- Range
- Text
- Geospatial
- JSON
- Timestamp
- Star Tree

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

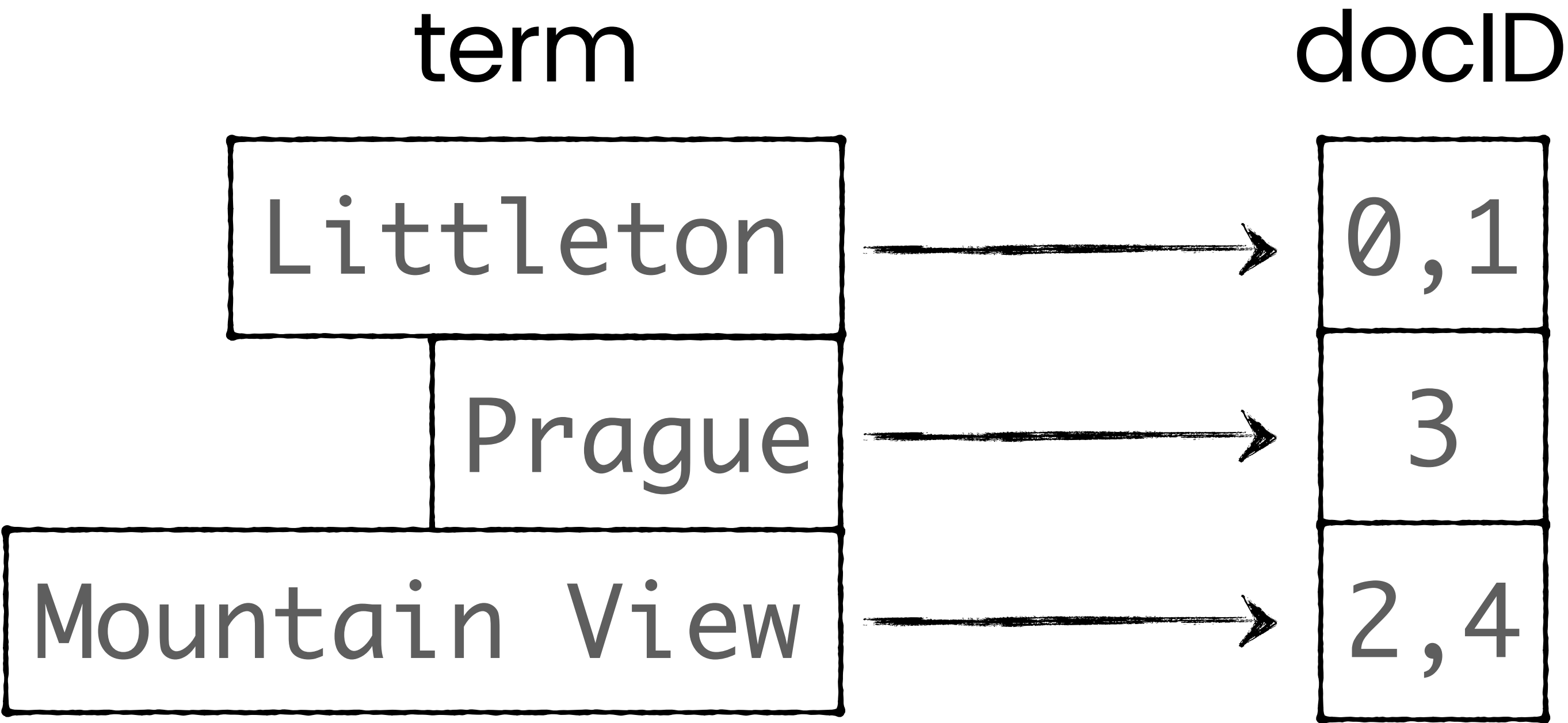


Inverted Index

- “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"
    }
  ]
}
```

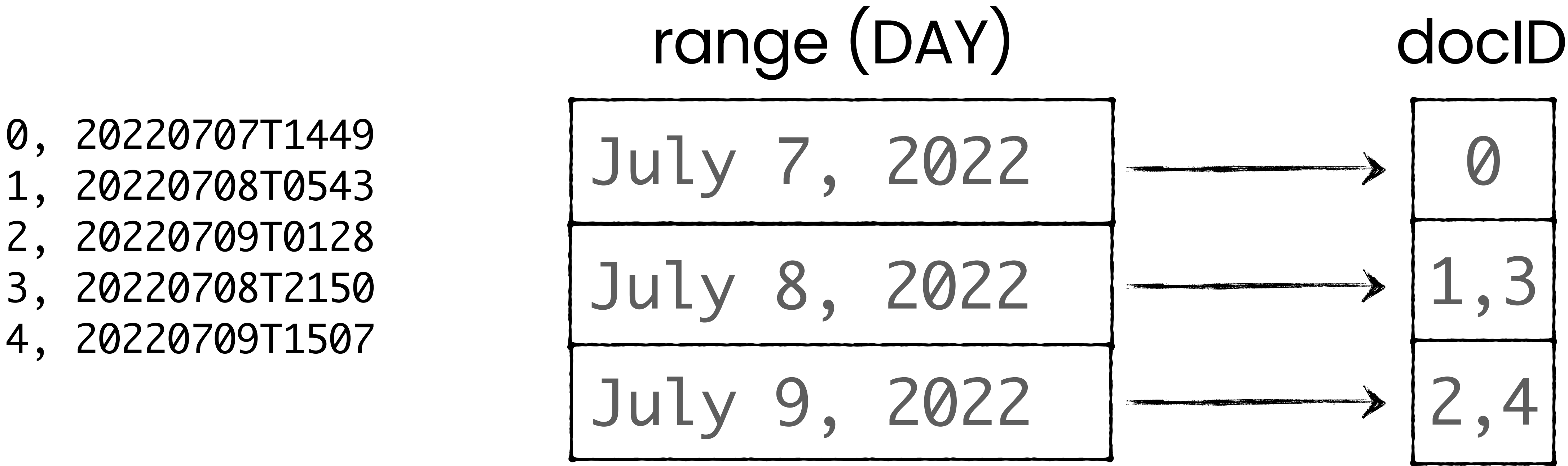
Array access

```
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



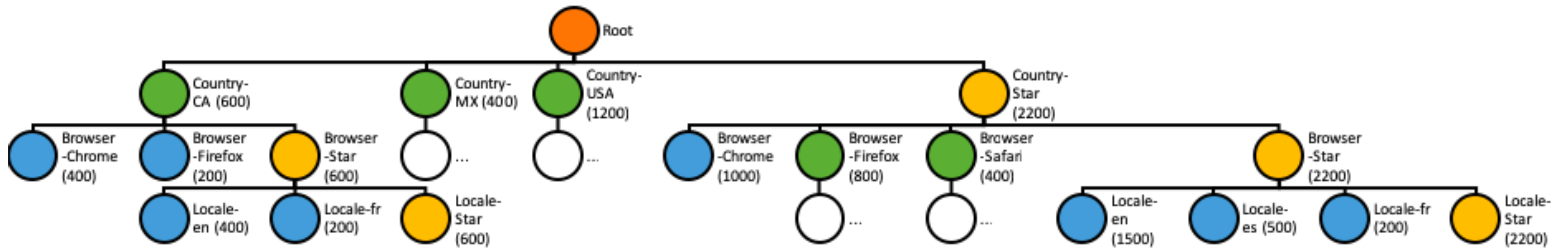
StarTree Index

- 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

Ingest

Ways to Ingest Data

- Batch
 - Command line/standalone
 - Spark
- Realtime
 - Kafka
 - Kinesis
 - Pulsar

Batch Ingest

- Behold the big data

```
studentID,firstName,lastName,gender,subject,score,timestampInEpoch  
200, Lucy, Smith, Female, Maths, 3.8, 1570863600000  
200, Lucy, Smith, Female, English, 3.5, 1571036400000  
201, Bob, King, Male, Maths, 3.2, 1571900400000  
202, Nick, Young, Male, Physics, 3.6, 1572418800000
```

Batch Ingest

- Create a schema file

```
{
  "schemaName": "transcript",
  "dimensionFieldSpecs": [
    {
      "name": "studentID",
      "dataType": "INT"
    },
    {
      "name": "firstName",
      "dataType": "STRING"
    },
    {
      "name": "lastName",
      "dataType": "STRING"
    },
    {
      "name": "gender",
      "dataType": "STRING"
    },
    {
      "name": "subject",
      "dataType": "STRING"
    }
  ]
}
```


Batch Ingest

- Create a table config

```
{
  "tableName": "transcript",
  "tableType": "OFFLINE",
  "segmentsConfig": {
    "replication": 1,
    "timeColumnName": "timestampInEpoch",
    "timeType": "MILLISECONDS",
    "retentionTimeUnit": "DAYS",
    "retentionTimeValue": 365
  },
  "tenants": {
    "broker": "DefaultTenant",
    "server": "DefaultTenant"
  },
  "tableIndexConfig": {
    "loadMode": "MMAP"
  },
  "ingestionConfig": {
    "batchIngestionConfig": {
      "segmentIngestionType": "APPEND",
      "segmentIngestionFrequency": "DAILY"
    }
  },
  "metadata": {}
}
```


Batch Ingest

- Run the command-line tool

```
bin/pinot-admin.sh AddTable \\  
  -tableConfigFile /path/to/table-config.json \\  
  -schemaFile /path/to/table-schema.json -exec
```

Streaming Ingest

- Behold the data, create the schema
- Create the table config
- Those things point to Kafka
- Ingest beings!

Upserts



Changes



Changes



Does not
change



The Metaphysics of Upserts

- Things change
- Events do not change
- Don't tell Heraclitus, but things also maintain continuous identity over time (i.e., they have primary keys)

Batch Ingestion

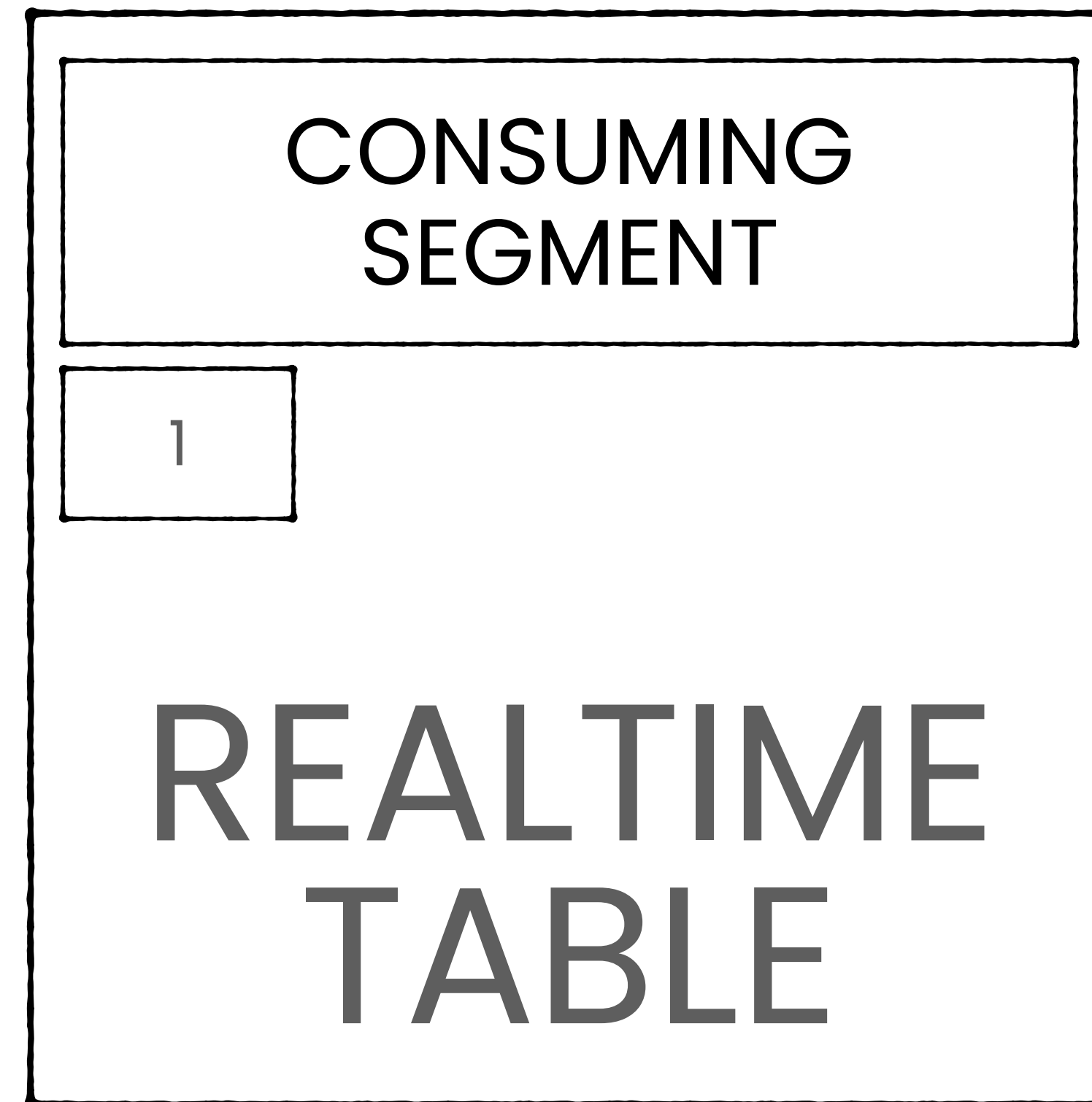
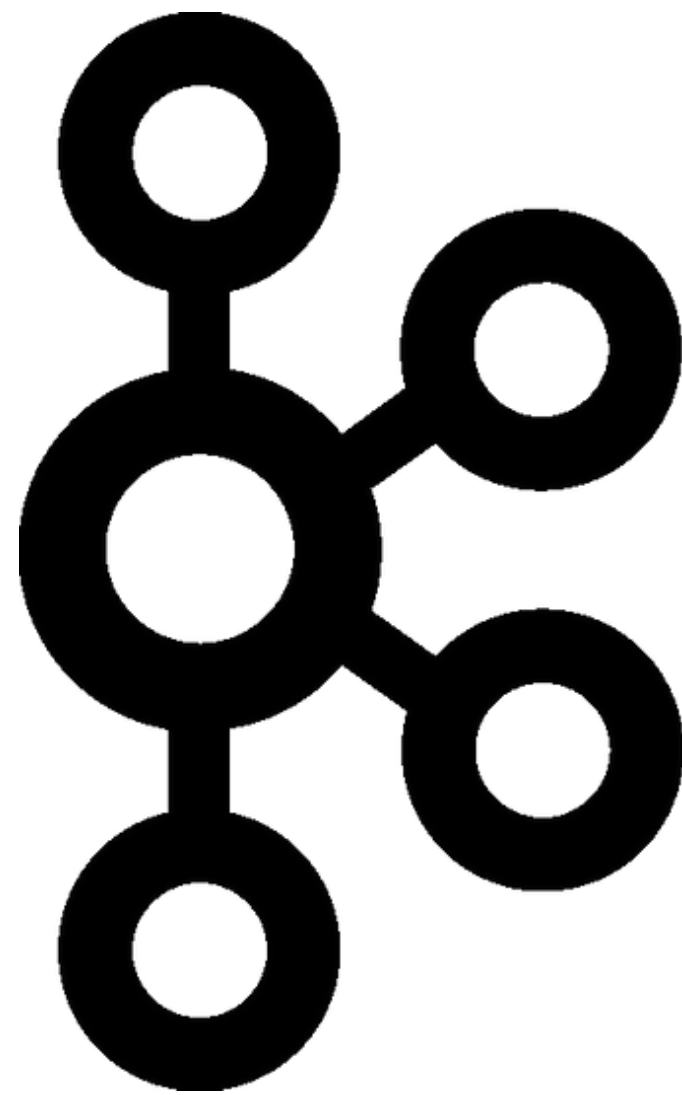
```
....,  
{  
  "ip": "111.173.165.103",  
  "userid": 10,  
  "remote_user": "-",  
  "time": "3271",  
  "_time": 3271,  
  "request": "GET",  
  "status": "406",  
  "bytes": "1289",  
  "referrer": "-",  
  "agent": "Mozilla/5.0"  
},  
....,
```



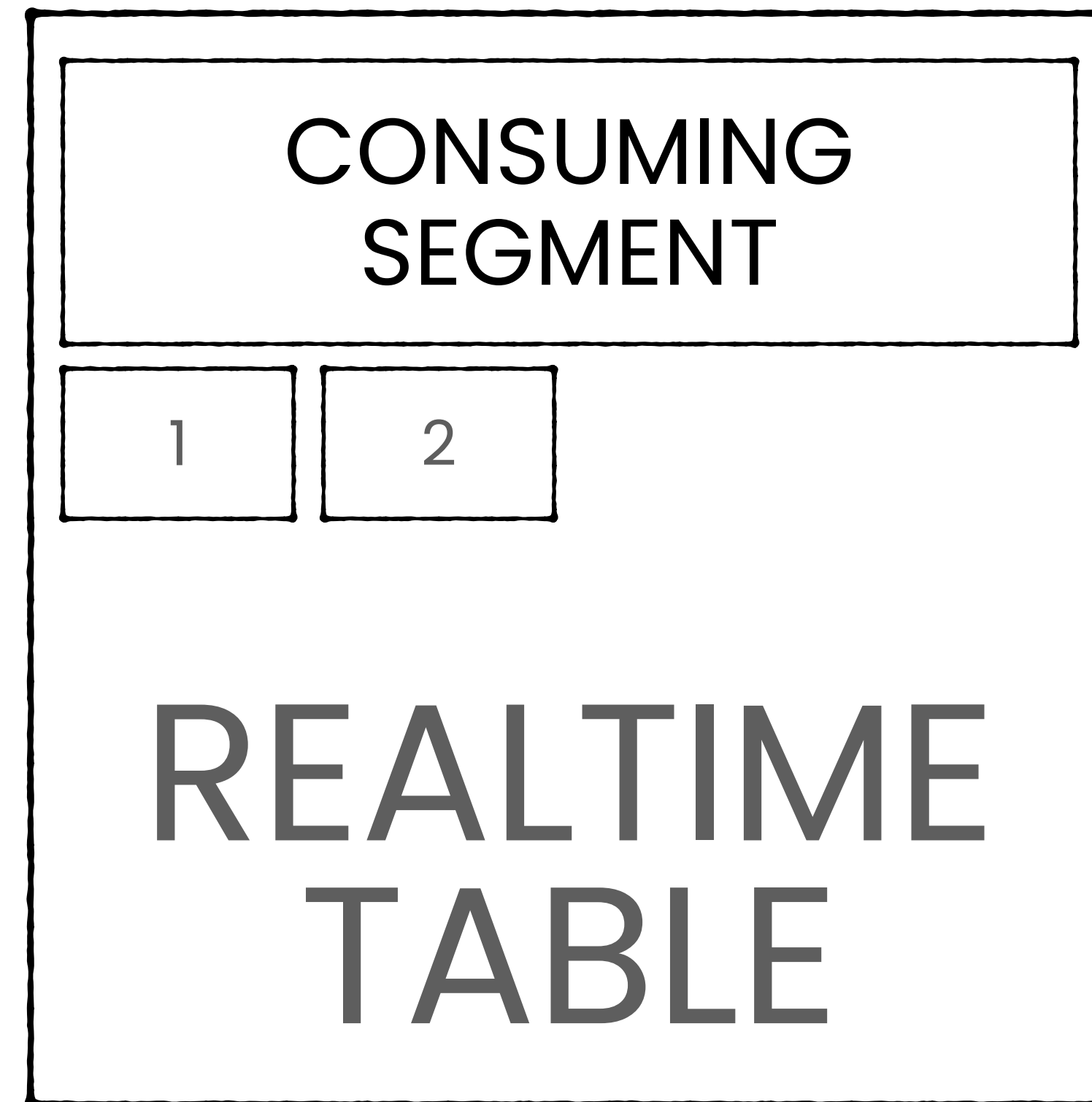
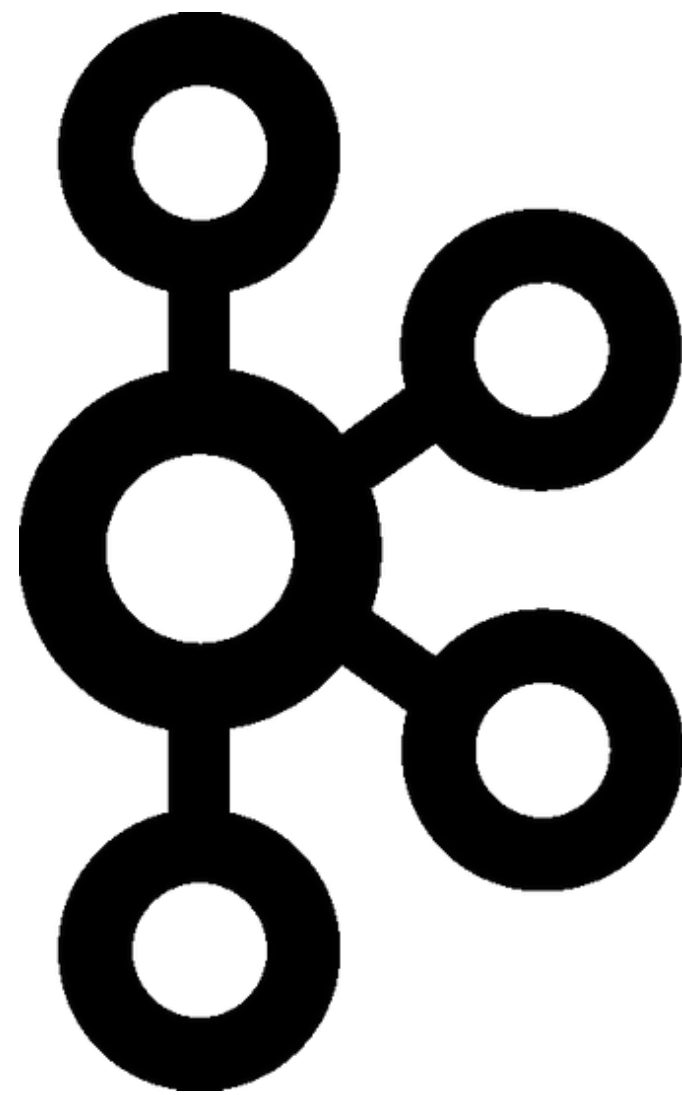
1	2	3	4
5	6	7	8

OFFLINE
TABLE

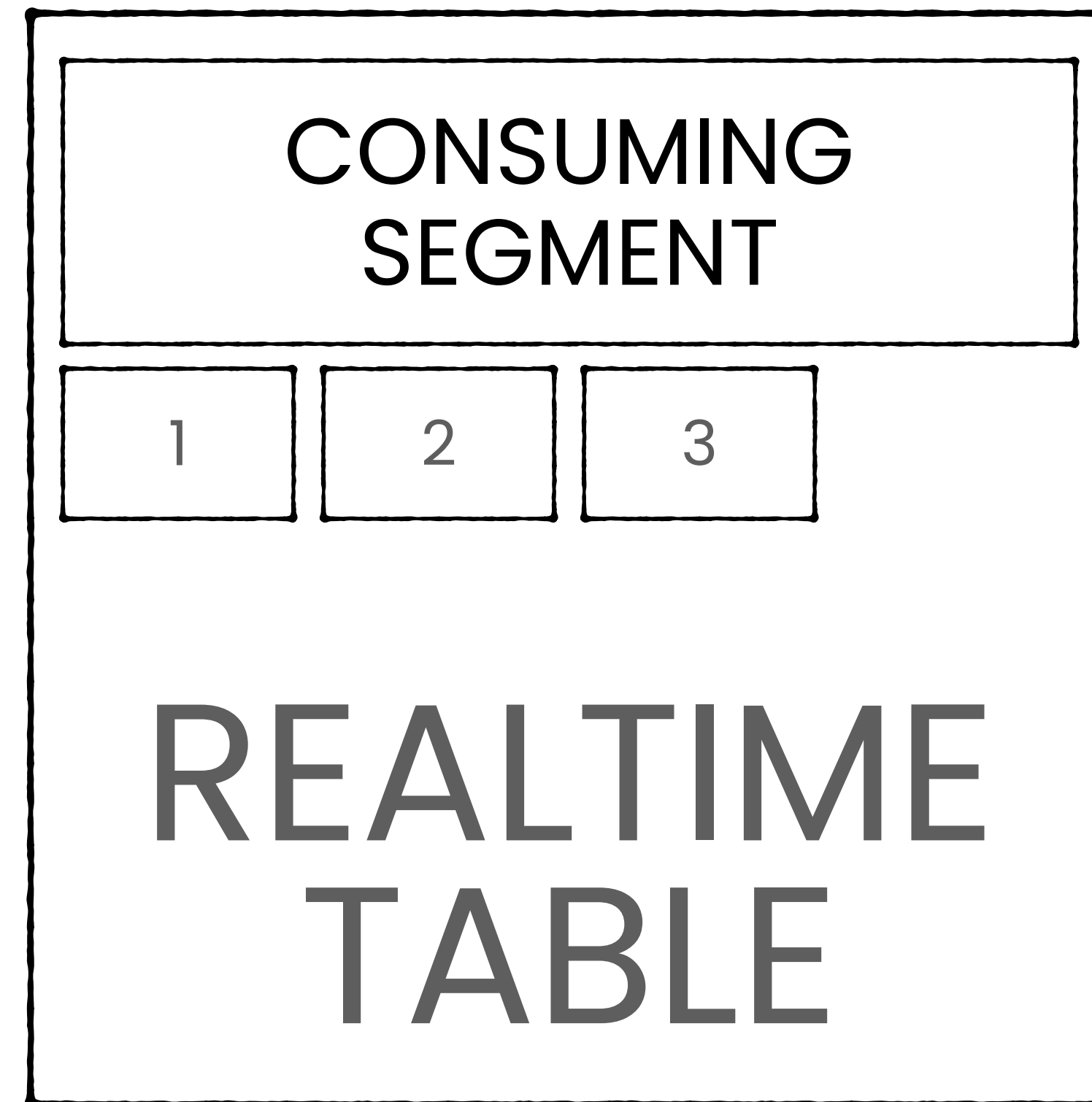
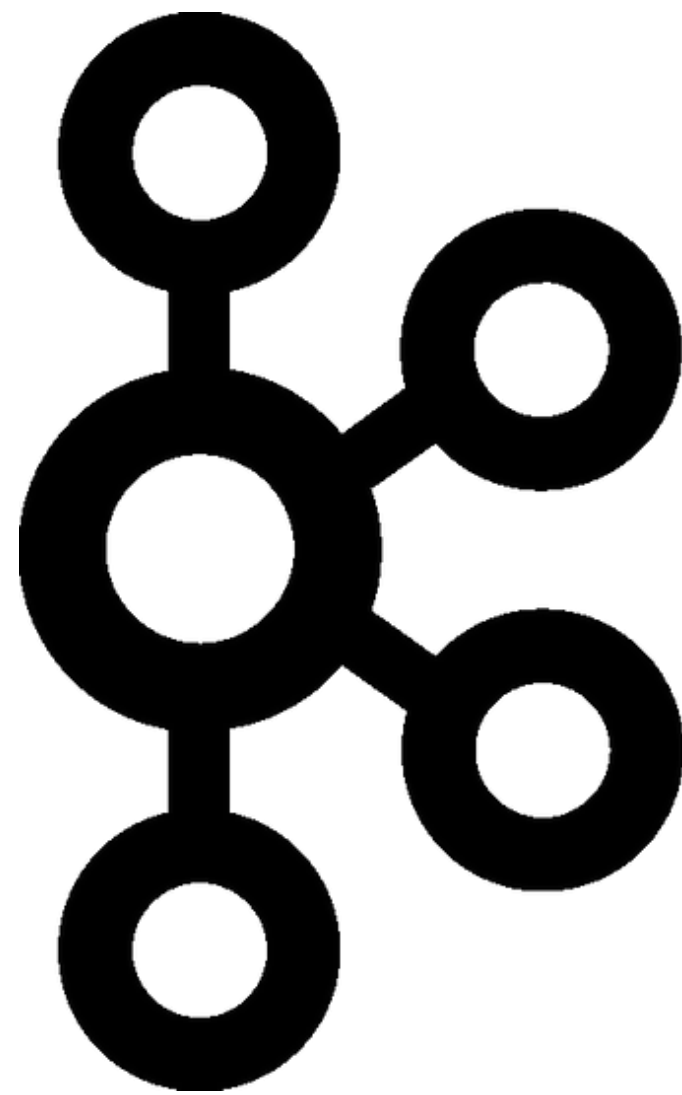
Realtime Ingestion



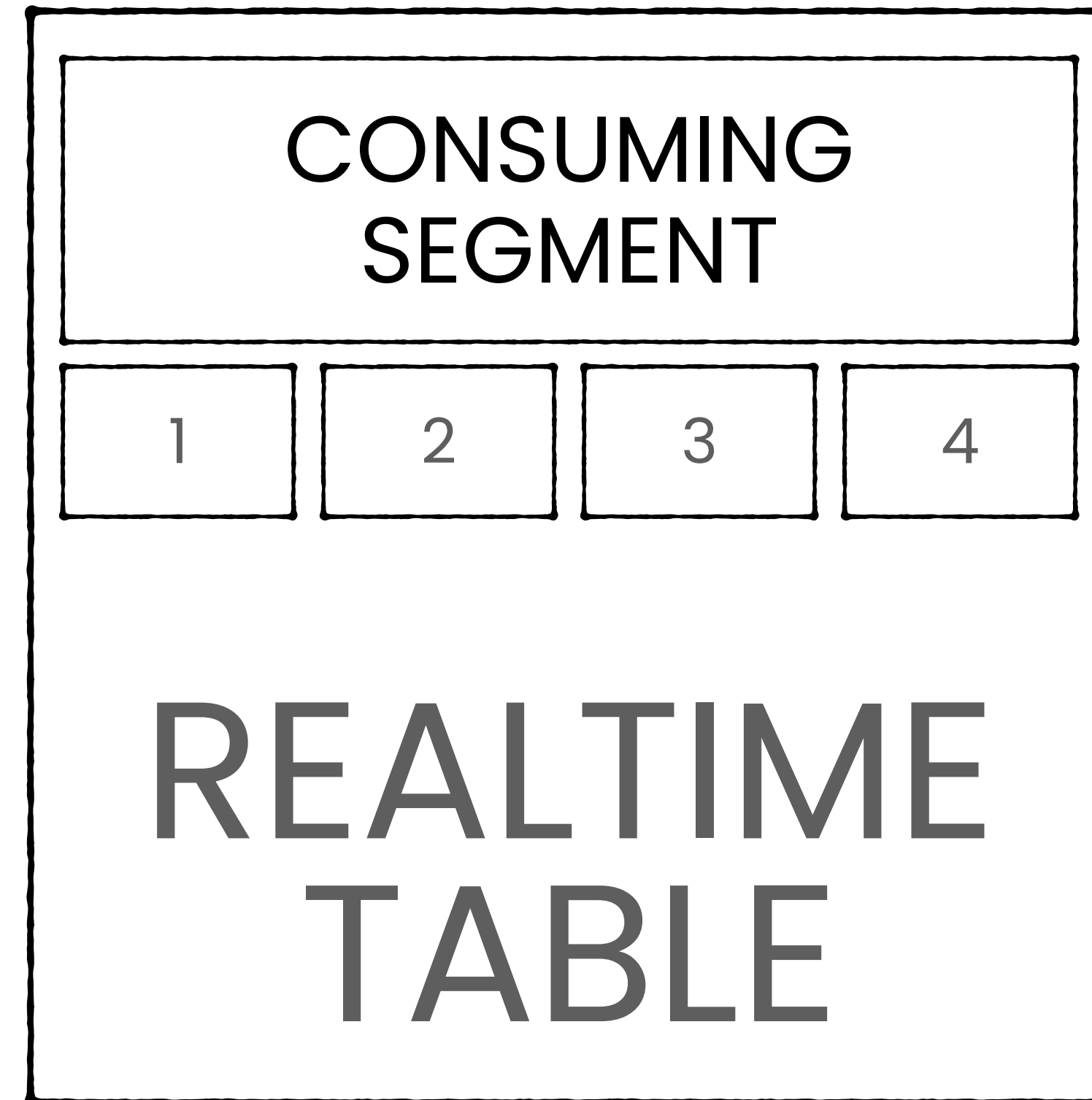
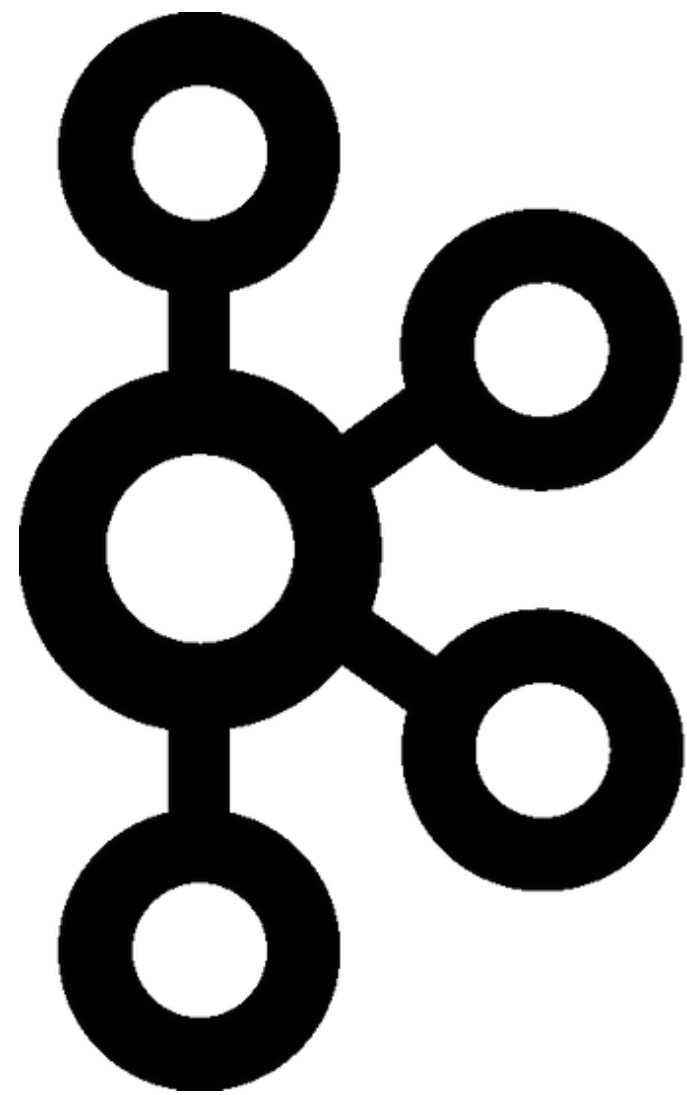
Realtime Ingestion



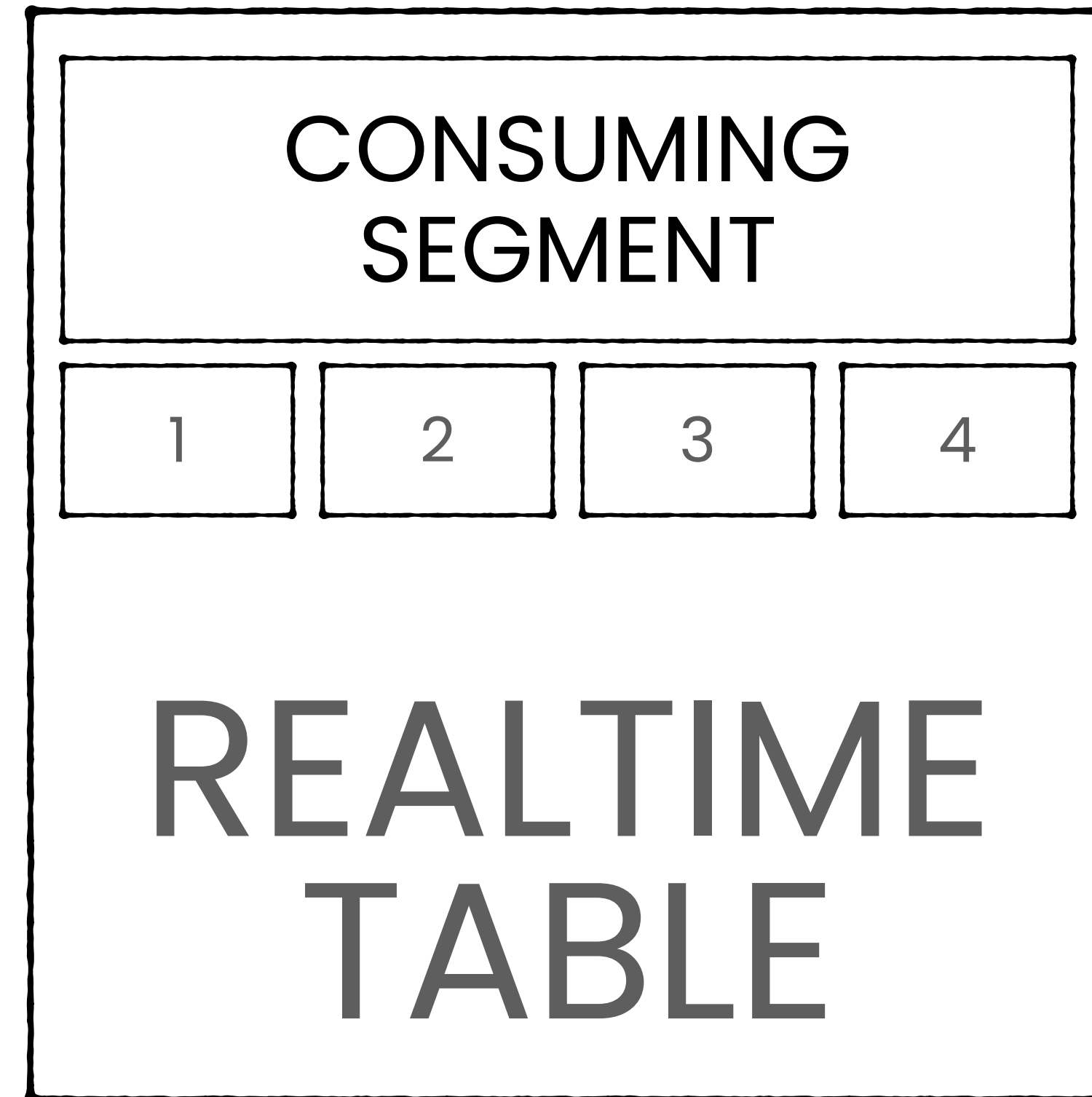
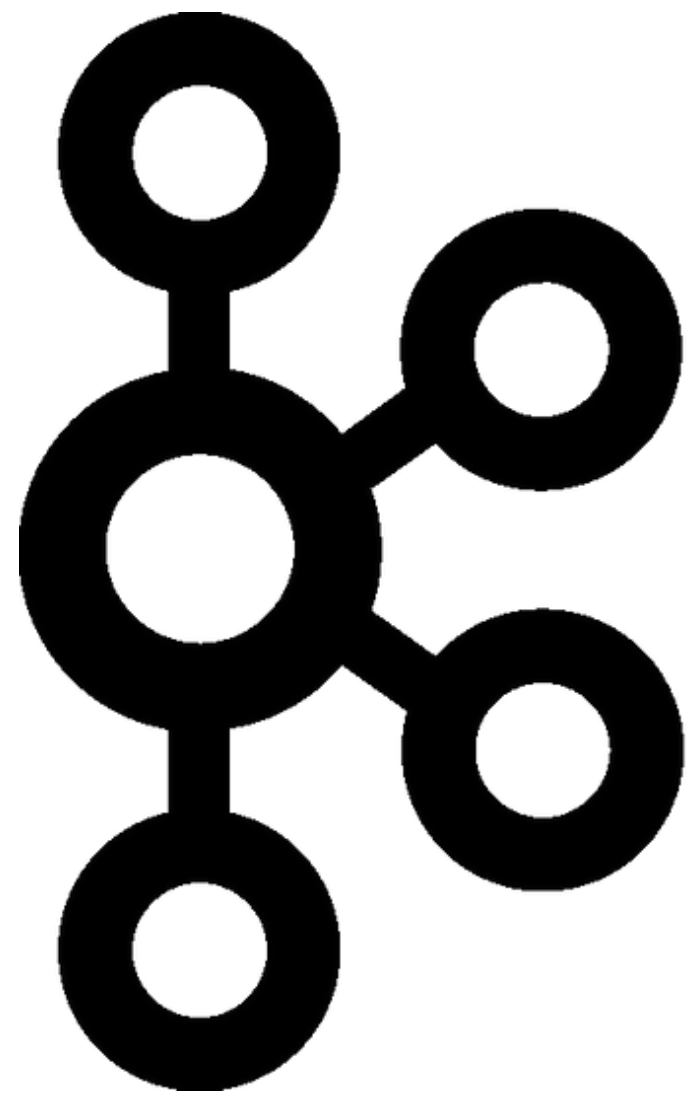
Realtime Ingestion



Realtime Ingestion



Realtime Ingestion



Upserts

1:A	2:A	3:B	4:C	3:D	1:B	5:Z	6:F	3:Q
-----	-----	-----	-----	-----	-----	-----	-----	-----

```
{  
  "primaryKeyColumns": ["event_id"]  
}
```

Upserts

1:A	2:A	3:B	4:C	3:D	1:B	5:Z	6:F	3:Q
-----	-----	-----	-----	-----	-----	-----	-----	-----

Upserts

1:A	2:A	3:B	4:C	3:D	1:B	5:Z	6:F	3:Q
-----	-----	-----	-----	-----	-----	-----	-----	-----

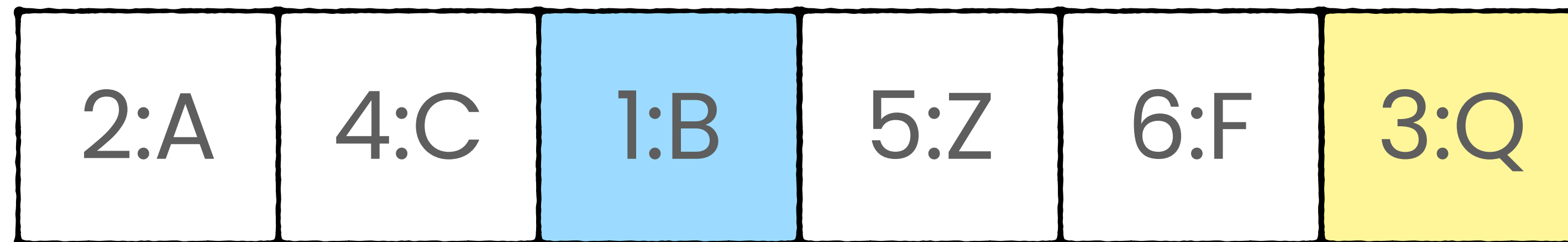
Full Upsert Mode

```
{  
  "upsertConfig": {  
    "mode": "FULL"  
  }  
}
```

1:A	2:A	3:B	4:C	3:D	1:B	5:Z	6:F	3:Q
-----	-----	-----	-----	-----	-----	-----	-----	-----

Full Upsert Mode

```
{  
  "upsertConfig": {  
    "mode": "FULL"  
  }  
}
```



Partial Upsert Mode

```
{  
  "upsertConfig": {  
    "mode": "PARTIAL",  
    "partialUpsertStrategies": {  
      "event_id": "APPEND"  
    }  
  }  
}
```

1:A	2:A	3:B	4:C	3:D	1:B	5:Z	6:F	3:Q
-----	-----	-----	-----	-----	-----	-----	-----	-----

Partial Upsert Mode

```
{  
  "upsertConfig": {  
    "mode": "PARTIAL",  
    "partialUpsertStrategies": {  
      "event_id": "APPEND"  
    }  
  }  
}
```

2:A	3:B	4:C	3:D	1:{A,B}	5:Z	6:F	3:Q
-----	-----	-----	-----	---------	-----	-----	-----

Partial Upsert Mode

```
{  
  "upsertConfig": {  
    "mode": "PARTIAL",  
    "partialUpsertStrategies": {  
      "event_id": "APPEND"  
    }  
  }  
}
```

2:A	4:C	1:{A,B}	5:Z	6:F	3:{B,D,Q}
-----	-----	---------	-----	-----	-----------

Partial Upsert Mode

- Modes
 - **OVERWRITE**: replace last value
 - **INCREMENT**: arithmetically add to the last value
 - **APPEND**: add to an unordered set of values (repetition allowed)
 - **UNION**: add to an unordered set of values (repetition not allowed)
 - **IGNORE**: keep the last value
- Comparison columns
 - Time by default
 - Configurable to any column name (timestamp type)

The real-time revolution is not
complete.

Thank you!

<https://stree.ai/slack>

@tlberglund

