CRATEDB: A SEARCH ENGINE OR A DATABASE? BOTH!

HOW WE BUILT A SQL DATABASE ON TOP OF ELASTICSEARCH AND LUCENE

Maximilian Michels
@stadtlegende
max@crate.io
mxm@apache.org

WHY ARE WE TALKING ABOUT THIS?

- Traditional databases are well-researched and there are plenty of them (Postgres, MySQL, Oracle…)
  - Scalable search using these can be tricky
- Search engines are databases optimized for search and scale (Lucene, Solr, Elasticsearch)
  - You can’t typically use SQL with Search Engines
- Why not stick with an mature query language standard which everybody knows?

"A scalable SQL database optimized for search without the NoSQL bullshit."

CRATEDB IN A NUTSHELL

- Since 2014: https://github.com/crate/crate
- Apache 2.0 licensed (community edition)
- Built using Elasticsearch, Lucene, Netty, Antlr, …
- SQL-99 compatible
- REST / Postgres Wire Protocol / JDBC / Python …

WHAT TO EXPECT

- What is great about CrateDB
  - Easy to setup
  - No funny APIs / SQL
  - Great scale out - Massive reads / writes
  - Container aware
- Not so great
  - Transactions
  - Foreign keys

USING CRATEDB

CrateDB

CRATEDB IS JUST LIKE A SQL DB

- SQL is the only query API

```
CREATE TABLE fosdem.speakers (id int PRIMARY KEY,
name string)
```

```
CREATE TABLE fosdem.talks (id INT PRIMARY KEY,
title STRING, abstract STRING, speaker INT);
```

```
INSERT INTO fosdem.speakers (id, name) VALUES (1,
'max')
```

```
INSERT INTO fosdem.talks (id, title, abstract,
speaker) VALUES (1, 'Talk about CrateDB', 'bla', 1)
```

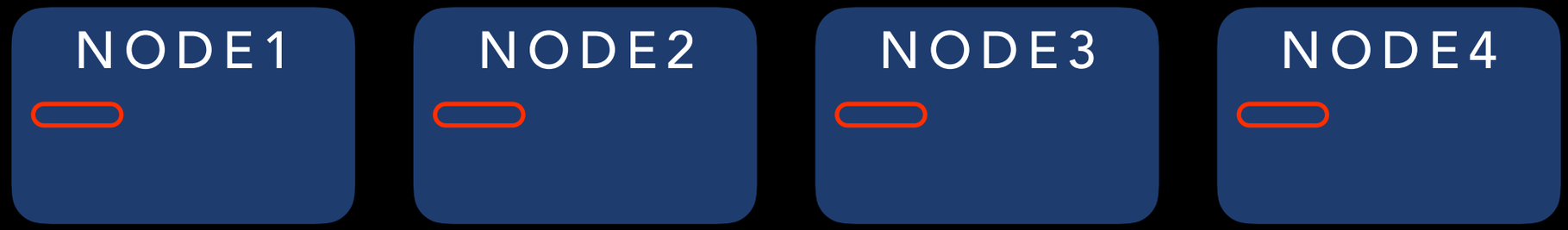
```
SELECT * FROM fosdem.talks t1 LEFT JOIN
fosdem.speakers t2 ON t1.id = t2.id
```

CrateDB

BUT THERE IS MORE

- CrateDB denormalized (no joins necessary)
- `CREATE TABLE fosdem.speakers (name STRING, talk OBJECT AS (title STRING, abstract STRING))`
- `INSERT INTO fosdem.speakers (name, talk) VALUES ('max', {title = 'CrateDB', abstract = 'Lorem ipsum'})`
- `SELECT talk['title'] as title FROM fosdem.speakers ORDER BY title`

CLUSTERING / REPLICATION



- `CREATE TABLE fosdem.speakers (name STRING, talk OBJECT AS (title STRING, abstract STRING))`
- `CLUSTERED BY name into 4 shards`

CrateDB

CLUSTERING / REPLICATION

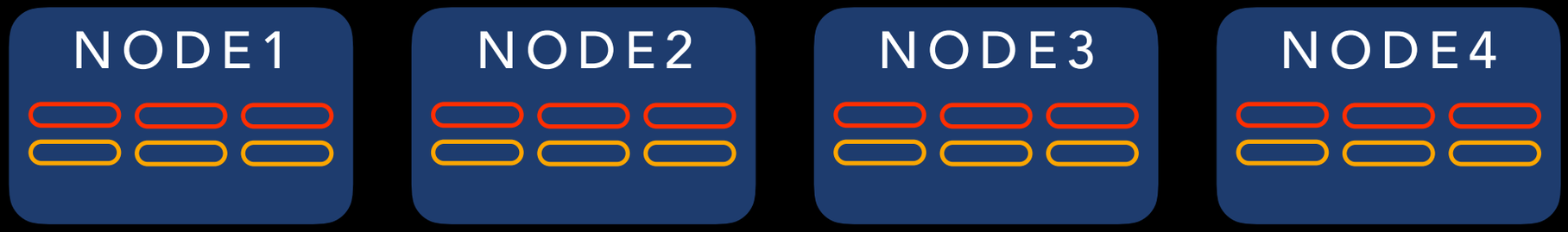
- `CREATE TABLE fosdem.speakers (name STRING, talk OBJECT AS (title STRING, abstract STRING))`
- `CLUSTERED BY name into 4 shards`
- `WITH (number_of_replicas = 1)`

PRIMARY

REPLICA

CrateDB

PARTITIONED TABLES



- `CREATE TABLE fosdem.speakers (name STRING, talk OBJECT as (title = STRING, abstract = STRING), year INT)`
- `CLUSTERED BY name into 4 shards`
- `PARTITIONED BY (year, …)`
- `WITH (number_of_replicas = 1)`

PRIMARY

REPLICA

CrateDB

MORE FEATURES

- Aggregations
- Geo search
- Text Analyzers
- UDFs
- Snapshots
- User management
- Schema / Table privileges
- SSL encryption
- MQTT Ingestion

ARCHITECTURE

CrateDB

ON THE SHOULDERS OF GIANTS

- **CrateDB:** Distributed SQL Execution Engine
  - **Antlr:** Parsing of SQL statements
  - **Netty:** REST, Postgres Wire Protocol, Web interface
  - **Lucene:** Storage, Indexing, Queries
  - **Elasticsearch:** Transport, Routing, Replication

INTRODUCTION TO Lucene

- Lucene stores documents which are CrateDB's rows
- Documents have fields
- ```
  { _id   : '123',
    name  : 'Bob',
    title : 'How I Learned to Stop Worrying
             and Love the Bomb',
    text  : 'Lorem ipsum…'
  }
  ```
- Fields are indexed for efficient lookup
- Fields have column store for efficient aggregation
  ```

INTRODUCTION TO ELASTICSEARCH

- Elasticsearch core concepts revolve around indices, shards, and replicas

- An index is a document store with n parts, called shards

- Each shard has 0 or more replicas which hold copies of the shard data

- Replicas are not only useful for fault tolerance but also increase the search performance

HOW TABLES RELATE TO INDICES AND SHARDS

- Each table in CrateDB is represented by an ES index with a mapping

- Each partition in a partitioned table is represented by an ES index

- Partition indices are created by encoding the partition value in the index name

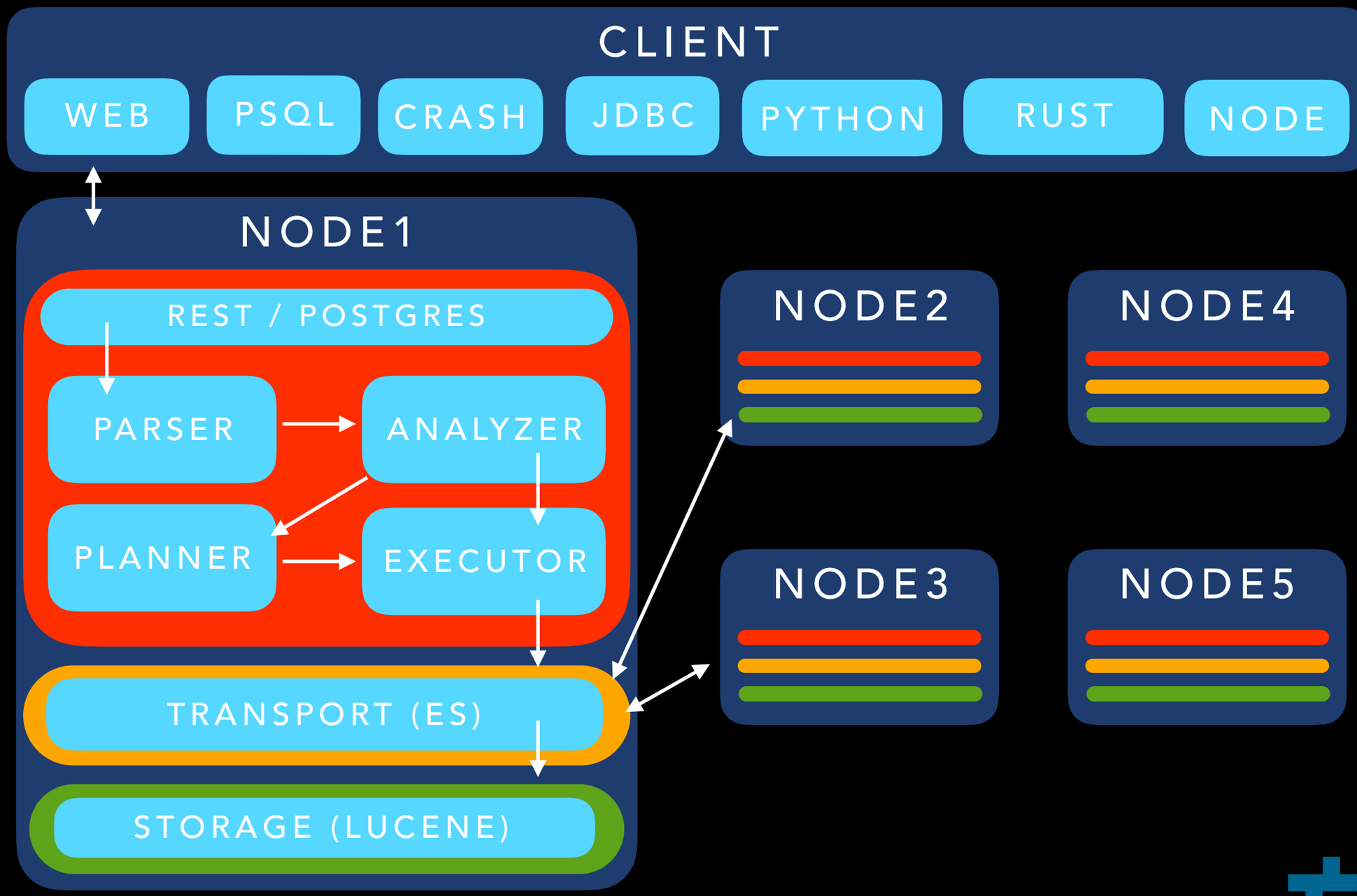
```
"properties":{
   “name”:{"type":"keyword"},
   "talks":{"dynamic":"true",
      "properties":{
         "abstract":{"type":"keyword"},
         "title":{"type":"keyword"}
      }
   }
}
```

TABLE	T1	T2			T3	…
INDEX	t1	t2.day1	t2.day2	…	t3	…
SHARD1	X	X	X	…	X	…
SHARD2	X	X	X	…	X	…
SHARD3	X				X	…
SHARD4	X					…
…						…

CrateDB

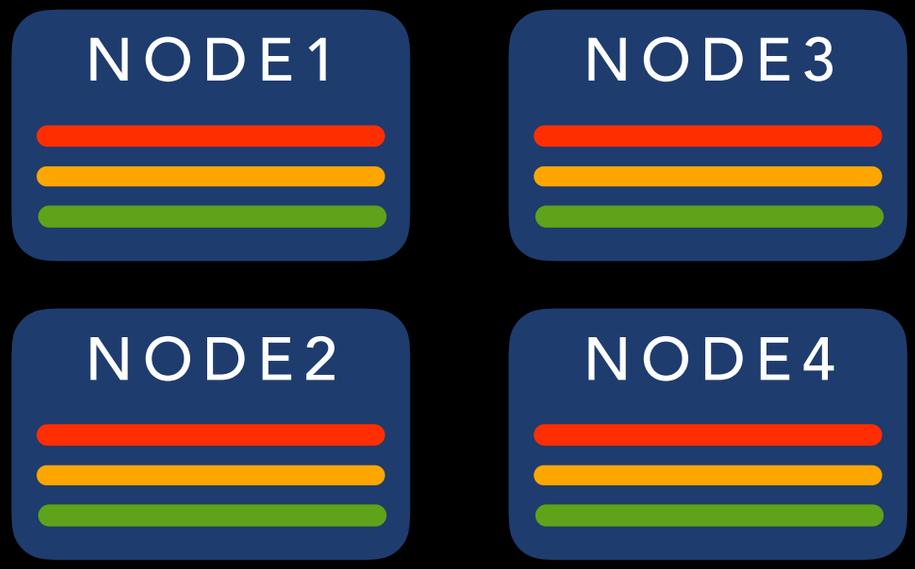
FROM QUERY TO EXECUTION

- SELECT name, count(*) as talks FROM fosdem.speakers
  WHERE room = 'hpc' AND year = 2018 GROUP BY name ORDER BY name



ARCHITECTURE HIGHLIGHTS

- Distributed storage / Distributed query execution
- Masterless
- Replication
- Only ephemeral storage needed (Container aware)
- Optimized for search: Indexing of all fields with Lucene (tuneable)



HANDS-ON

CrateDB

WHAT CAN YOU DO WITH CRATEDB?

- Monitoring (IoT, Industry 4.0, Cyber Security)
- Stream Analysis
- Text Analysis
- Time Series Analysis
- Geospatial Queries

CrateDB Web Interface

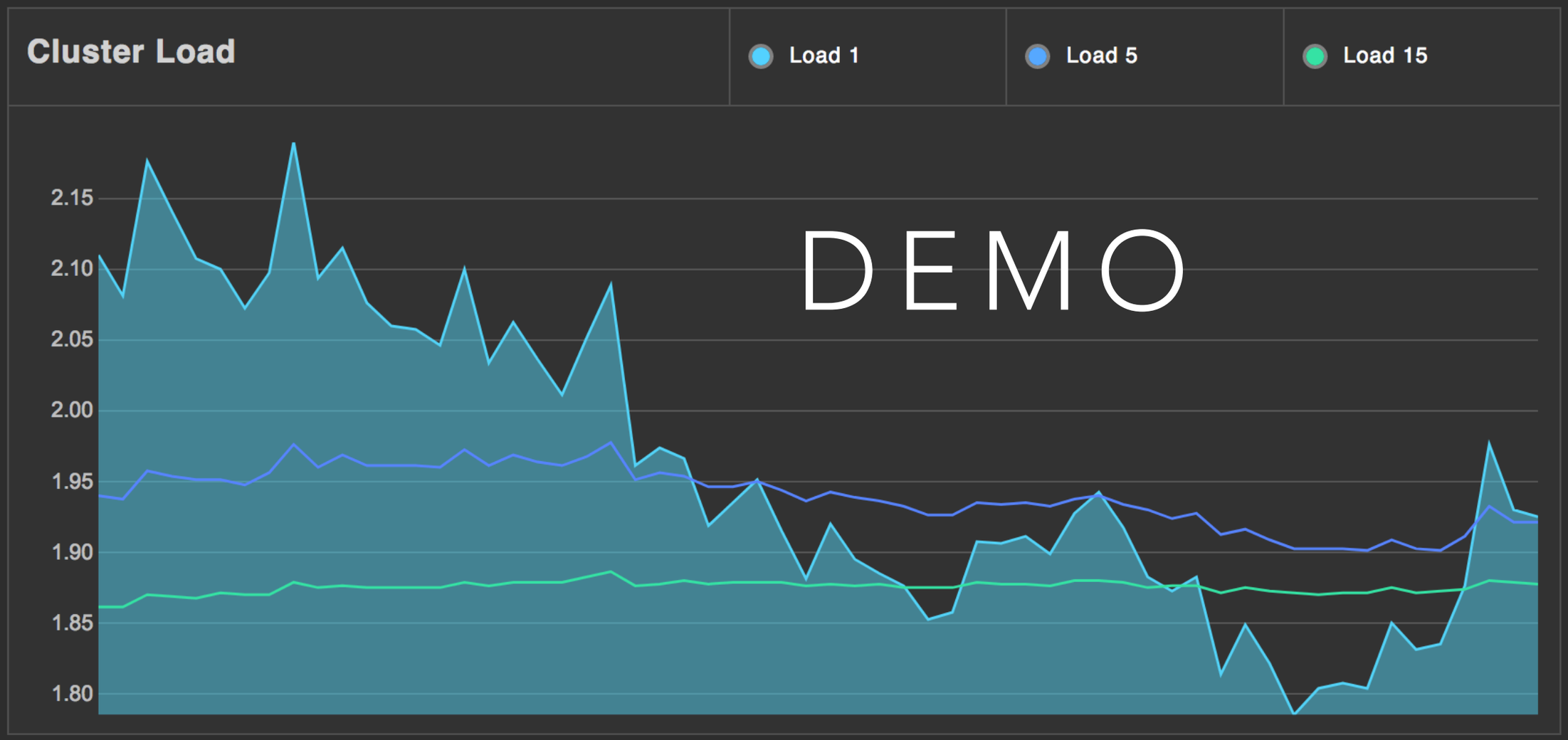
Cluster: demo

Health	Replicated Data	Available Data	Total Records
good	100.0%	100.0%	11.5 Billion

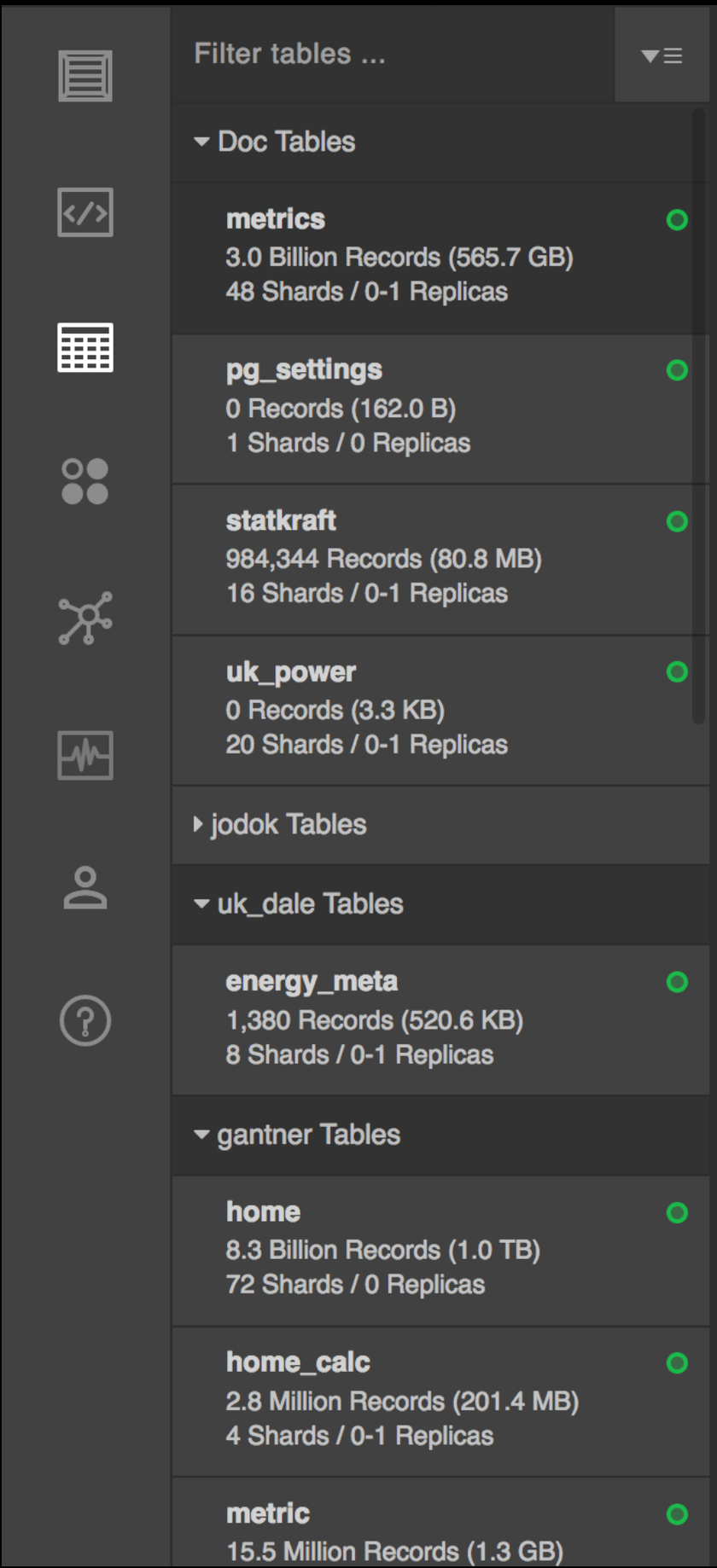
Underrepl. Records	Unavail. Records
0	0

Cluster Load

Load 1 · Load 5 · Load 15



CrateDB Web Interface



Tables

Name	Health
metrics (partitioned)	good
Configured Replicas	**Configured Shards**
0-1	48
Started Shards	**Missing Shards**
96	0
Underrepl. Shards	**Total Records**
0	3.0 Billion
Unavailable Records	**Underrepl. Records**
0	0
Size	**Recovery**
565.9 GB	100.0%

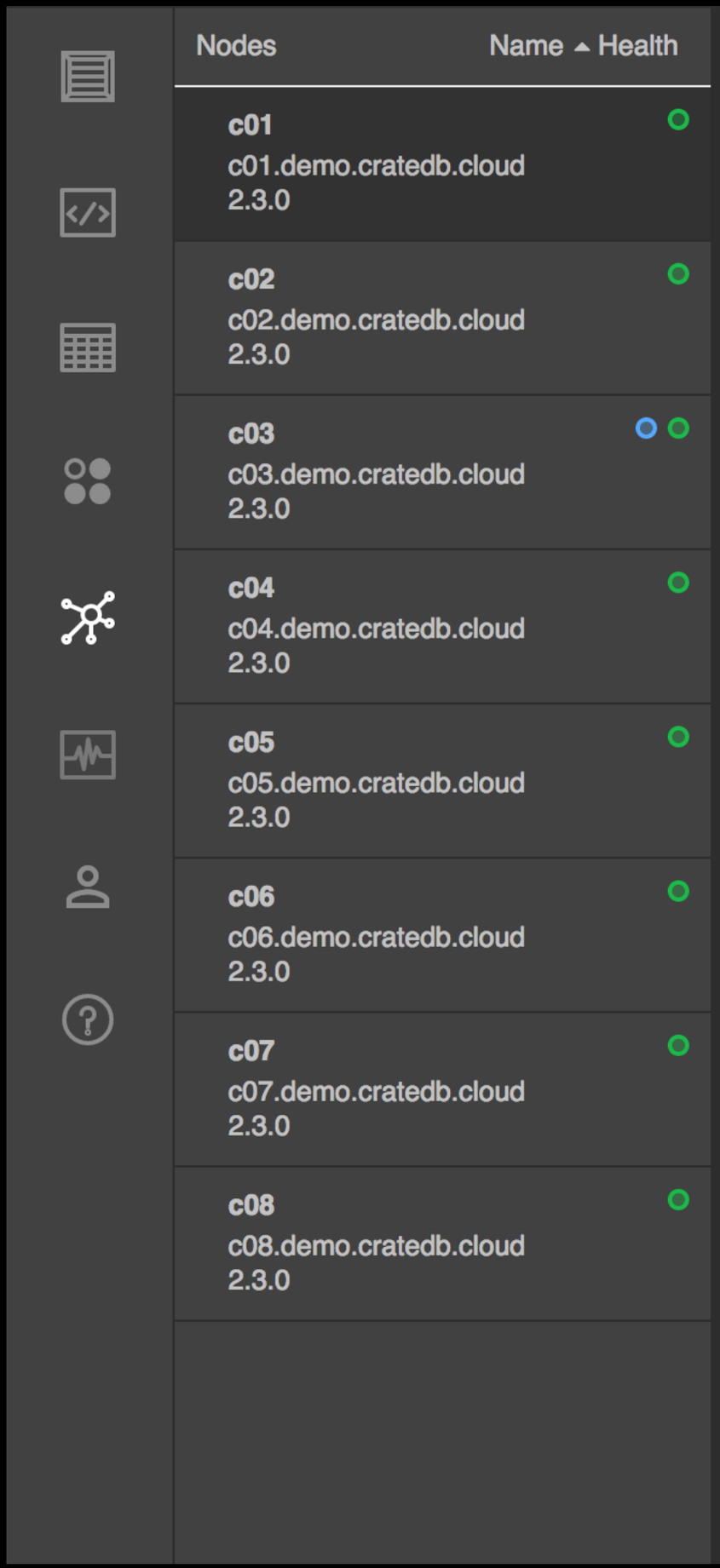
QUERY TABLE

Partitions

Partition Columns: **day__generated**

Health	Ident	Partition Values	Conf. Replicas	Conf. Shards	Started Shards	Missing Shards	Under Shards
		day__generated					
good	04732d9h6so3idpm60o30c1g	1517097600000	0-1	8	16	0	0

CrateDB Web Interface



CrateDB Web Interface

Shards

Show Shard IDs | Started Primary | Started Replica | Initializing | Relocating | Unassigned

	fhv					gantner		
	parkplatz	part		person	replicas	test	home		
	2 4 10 14	7 9 10 14	11 12 14 15	1 5 10 14	0	2 4 5 10	3	3	3
	1 3 5 7	0 1 2 15	2 3 4 10	1 11 12 13	0	3 7 11 12	4	4	4
	1 5 7 13	0 3 4 6	1 3 6 8	4 6 8 15		1 3 5 8	1	1	1
	0 8 11 15	5 11 12 15	5 7 9 13	7 8 11 12	0	0 11 12 15	0	0	0
	0 2 3	1 2 5	0 4 5	0 3	0				

CONCLUSION

CrateDB

WHAT WE HAVE LEARNED

- Elasticsearch used Lucene and Netty to built a distributed search engine
- CrateDB used Elasticsearch, Lucene, and Netty to built a distributed SQL database
- CrateDB is perfect when you
  - want or have to use SQL
  - store large amounts of structured or unstructured data
  - have many thousands of queries per second

CrateDB

SEE FOR YOURSELF!

- Try out CrateDB
  - Download from https://crate.io/download/
  - or `$ curl try.crate.io | bash`
  - or `$ docker run crate`
  - or build from source https://github.com/crate/crate
- Check out https://crate.io/docs
- Contributions welcome
  - Check out https://github.com/crate/crate/blob/master/devs/docs/index.rst
  - Check out the issues
  - Stackoverflow
  - Join our Slack channel

CrateDB

THANK YOU!

Maximilian Michels
@stadtlegende
max@crate.io
mxm@apache.org