

Not Only SQL

Rob Kellington

TRUST SCIENCE[®]

Caveat

- SQL and RDBMS Systems are Fantastic!
- Nearly **every** project I have worked on has been based on one of these databases
 - DB2
 - Oracle
 - SQL Server
 - SQLite
 - PostgreSQL

What is SQL?

- A “SQL” Database
 - Language - <https://en.wikipedia.org/wiki/SQL:2016>
 - Database Engine
 - Storage System
- An RDBMS combines these

Why SQL?

- SQL is a language specifically designed to deal with data
- Many product options - licences and open source
- Amazing integration with supporting tools & tech
- Resources easy to find
- Best choice for most data management solutions

Why not SQL?

- Data Variety - data types and complex structures
- Data Volume & Velocity
- Performance & Scalability
- Specific use cases
- Plus ... alternatives continue to improve

Our Challenges

1. Complex API JSON Documents
2. Logs
3. Billing

Plus - we build using a serverless strategy

JSON Data

- Supporting Analytics and Machine Learning Pipeline
- Multiple documents - inputs, intermediate, results
- Relational model would require *many* tables
 - This is a challenge with complex data warehouses as well
- Dealing with schema changes & versioning
- Context and processing rules/flags

JSON Data

- Use AWS S3 as our canonical truth for data
- Different bucket/structure for different use cases
- Allows flexibility and operational benefits
- S3 provides versioning - no updates
- PostgreSQL for indexing each S3 file
 - uses JSON datatype in Postgresql for flexibility

JSON Data

```
{ } JSON S3 Structure Example.json × { } JSON Schema Example.json AWS Cost & Usage.sql
Users > Rob > Desktop > { } JSON S3 Structure Example.json
1  <S3 Bucket>
2  /<client>
3      /a3a9e470-90c8-11e9-b5b7-f37c87eba0bd (requestId)
4
5  /PipelineStep1
6      /PipelineStep1/Source1/a3a9e470-90c8-11e9-b5b7-f37c87eba0bd.json
7      /PipelineStep1/Source2/a3a9e470-90c8-11e9-b5b7-f37c87eba0bd.json
8      /PipelineStep1/data/a3a9e470-90c8-11e9-b5b7-f37c87eba0bd-thisInfo.json
9      /PipelineStep1/data/a3a9e470-90c8-11e9-b5b7-f37c87eba0bd-thatInfo.json
10     /PipelineStep1/data/a3a9e470-90c8-11e9-b5b7-f37c87eba0bd-otherDataInfo.json
11     /PipelineStep1/data/a3a9e470-90c8-11e9-b5b7-f37c87eba0bd-derivedInfo.json
12     /PipelineStep1/data/a3a9e470-90c8-11e9-b5b7-f37c87eba0bd-PRIVATEInfo.json
13
14 /PipelineStep2
15     /PipelineStep2/a3a9e470-90c8-11e9-b5b7-f37c87eba0bd-centralityInfo.json
16
17 /PipelineStep3
18     /PipelineStep3/a3a9e470-90c8-11e9-b5b7-f37c87eba0bd-request.json
19     /PipelineStep3/a3a9e470-90c8-11e9-b5b7-f37c87eba0bd-results.json
20
21 /PipelineStep4
22     /PipelineStep4/identity/fac5c5d219dc7112af565cddb3b273da9b757bbeb74f39c5b936091790e1.
23     /PipelineStep4/model/fac5c5d219dc7112af565cddb3b273da9b757bbeb74f39c5b936091790e1442!
24     /PipelineStep4/request/a3a9e470-90c8-11e9-b5b7-f37c87eba0bd.json
25     /PipelineStep4/request/raw/a3a9e470-90c8-11e9-b5b7-f37c87eba0bd.json
26     /PipelineStep4/result/a3a9e470-90c8-11e9-b5b7-f37c87eba0bd-results.json
27     /PipelineStep4/result/mining-identity-tks-hash.json
28
29 /PipelineStep5
30     /PipelineStep5/cleanerRequest/a3a9e470-90c8-11e9-b5b7-f37c87eba0bd-model1a.json
31     /PipelineStep5/cleanerRequest/a3a9e470-90c8-11e9-b5b7-f37c87eba0bd-model2.json
32     /PipelineStep5/cleanerRequest/a3a9e470-90c8-11e9-b5b7-f37c87eba0bd-model5.json
33     /PipelineStep5/cleanerRequest/a3a9e470-90c8-11e9-b5b7-f37c87eba0bd-model3.json
34     /PipelineStep5/formatterRequest/a3a9e470-90c8-11e9-b5b7-f37c87eba0bd.json
35     /PipelineStep5/predictorRequest/a3a9e470-90c8-11e9-b5b7-f37c87eba0bd-model1a.json
36     /PipelineStep5/predictorRequest/a3a9e470-90c8-11e9-b5b7-f37c87eba0bd-model2.json
```

```
{ } JSON S3 Structure Example.json { } JSON Schema Example.json × AWS Cost & Usage.sql
Users > Rob > Desktop > { } JSON Schema Example.json > { } dataContext > { } clientConfig > { } callbacks > abc web
1  {
2      "schemaVersion": "2.0.0",
3      "scoring": {
4          "type": "Scoring",
5          "package": "Regular",
6          "useCase": "Lending-Payday",
7          "jurisdiction": {
8              "country": "CAN",
9              "state": "ON"
10         },
11         "generateReport": true
12     },
13     "control": {
14         "tags": [
15             "mobile"
16         ],
17         "callbackHeader": "X-API-Key: 9tXKB8aZoLiqWvdvasdfqr4gZakMhw1JbhrvcP7f54nJ",
18         "environment": "prod",
19         "correlationId": 1571937202686
20     },
21     "source": "Portal",
22     "user": "bill@client.com",
23     "batchName": "single",
24     "sendMobileInvite": false,
25     "data": {
26         "person": {
27             "firstName": "mary",
28             "lastName": "jones",
29             "emails": [
30                 {
31                     "primary": "some.name@hotmail.com"
32                 }
33             ],
34             "phones": [
35                 {
36                     "home": ""
```

JSON Data

- Supporting Analytics
 - “Research Data Extractor”
 - Uses PostgreSQL for search
 - Copies to new bucket individual files
 - Tools can load/process as logical table
 - Python Pandas, Spark Dataframes, Presto SQL

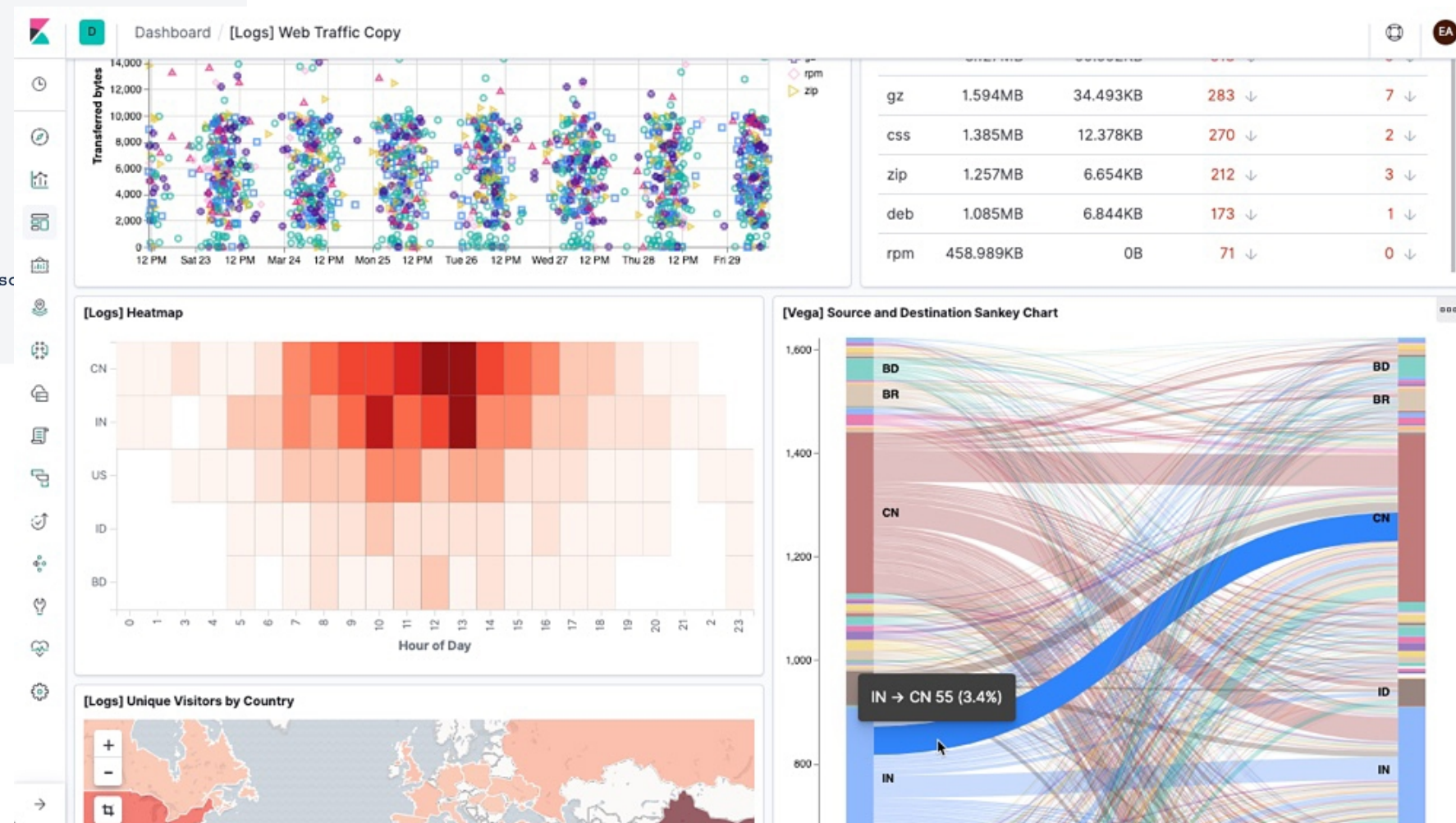
Log Data

- AWS - logs from all processes go to CloudWatch
 - Challenge to track and investigate
- Stream CloudWatch Logs Data to Elasticsearch
- NonSQL Query - but great for text search
- ElasticSearch & Kibana (variation of ELK stack)

Log Data

Fetch Items of a Given RequestId:

```
1 method: POST
2 endpoint: /<index>/_search
3
4 {
5   "query": {
6     "bool": {
7       "must": {
8         "bool": {
9           "should": [
10            {
11              "match": {
12                "scoringRequestId": "<RequestId>"
13              }
14            }
15          ]
16        }
17      }
18    },
19    "sort": [
20      {
21        "timestamp": { "order": "asc" }
22      } # In most cases, user prefer to read the history log in asc
23    ],
24  },
25 }
```



Billing Data

- AWS AWS Cost & Usage Report
 - Very large files - 195 columns, 200k records per month
- Rarely accessed
 - Standard queries once a month and occasional inquiry
- Process into Parquet files
 - columnar, compressed files in S3
- Athena - managed Presto service
- Access - standard SQL

Billing Data - Athena



Services ▾

Resource Groups ▾



rob.kellington@trustscience.co... ▾

Oregon ▾

Support ▾

Athena

Query Editor

Saved Queries

History

AWS Glue Data Catalog

Workgroup : primary

Settings

Tutorial

Help

What's new 10+

Database



athenacurcfn_daily_costand_usage ▾

Filter tables and views...

▼ Tables (2)

Create table

▶ cost_and_usage_data_status

▼ dailycostandusage (Partitioned)

identity_line_item_id (string)

identity_time_interval (string)

bill_invoice_id (string)

bill_billing_entity (string)

bill_bill_type (string)

bill_payer_account_id (string)

bill_billing_period_start_date (timestamp)

bill_billing_period_end_date (timestamp)

line_item_usage_account_id (string)

line_item_line_item_type (string)

line_item_usage_start_date (timestamp)

line_item_usage_end_date (timestamp)

line_item_product_code (string)

line_item_usage_type (string)

line_item_operation (string)

✓ New query 1

✓ New query 2 ✕ +

```
1 select count(*)
2 from dailycostandusage
3 where year = '2019'
4 and month = '10'
5 -- 198,366
```

Run query

Save as

Create ▾

(Run time: 2.81 seconds, Data scanned: 0 KB)

Format query

Clear

Use Ctrl + Enter to run query, Ctrl + Space to autocomplete

Results



	_col0
1	198366

Other NoSQL

- Key Value
 - DynamoDB
- Document
 - Mongo
- Column
 - Google's BigTable
- Graph
 - Neo4J
- Analytics Engines
 - Spark, Hadoop
- Blockchain
 - Hyperledger Fabric and Ethereum, Ledger Database

Summary

- SQL is great
 - But lots of alternatives exist that may work better for some of your use cases
- Cloud / Managed services make NoSQL options easier
 - S3 storage, RDS PostgreSQL, Athena, ElasticSearch/Kibana
-