DATA ANALYTICS IN BROWSER WITH ALASQL

Presented By:

Gaurab Patra
Co-Founder & CTO
Flurgo

Bhagyajit Jagdev
Lead Engineer
Flurgo

WHY THIS TALK

By the time this talk will be over
data is generated across the world

12 B MB

Solving data analysis use cases on any devices that run Javascript

Data are created outside the traditional data center

The cloud is extended to the edge

It won’t be cloud versus edge; it will be cloud with edge

Activating data at the edge - ask better questions and get more timely answers

Mobile, laptop, smart watch, smart home appliances, gaming console and more

This talk does not draw any comparison with similar databases.

BACKGROUND

When someone shares an analysis implemented in JavaScript

You’re not just seeing a static snapshot of their work; you’re running it live in your browser

What is AlaSQL

- Lightweight easy-to-use client-side in-memory SQL database designed to work in browsers and Node.js
- Opensource
- Strong focus on query speed and datasource flexibility for relational data
- Handles schemaless data, and graph data as well
- Handles both traditional relational tables and nested JSON data (NoSQL)
- Export, store, and import data from localStorage, IndexedDB, and Excel.

Go beyond passive reading

You’re not just seeing a static snapshot of their work; you’re running it live in your browser

History

Written by: Andrey Gershun
Initial commit: October 26, 2014
Major milestone (0.1.4): foreign keys, unique/not-null/check constraints
Version (0.4.0): Support for Typescript syntax

ECOSYSTEM

JavaScript is the richest medium we've ever had for communication

Open and Collaborative

Edit and run realtime

Rapid Prototyping

Share exploratory views of data to answer questions, and explain concepts

AlaSQL Fitment

- Where persistent storage is not required
- Speed of processing is important
- Rapid quering is important
- Don't need the results to be permanent
- In-memory SQL for joining, filtering, grouping data
- Query the server-side database for once, bring the data back locally, and run any filter sort search on it
- Processing a local file
- Pre-process large date on edge
- ETL
- BI

OPERATIONS

Few features of AlaSQL are fundamentally important for the topic we are discussion today - running analysis on edge

- Well supported
- Extensible
- Ability to execute SQL against data sets (JSON or Arrays)
- Fast in-memory SQL data processing for BI and ERP applications on fat clients
- Easy ETL and options for persistence by data import/manipulation/export of several formats
- All major browsers, Node.js, and mobile applications
- All structured SQL operators are available
- Fast
- In-built compilation
- Query optimization
- Indexing
- Complex and efficient join operations

FUNDAMENTALS

- Capability to add custom javascript functions where standard SQL statements are not sufficient
- This can be useful both for selecting data as well as pre-processing outputs
- Flexible - import/export and query directly on data stored in Excel (both xls and .xlsx), CSV, JSON, TAB, IndexedDB, LocalStorage, and SQLite files.
- Ability to execute SQL against data sets (JSON or Arrays)
- In-memory which makes queries and such faster
- For persistent storage - use alasql on top of it for processing
- Create compiled statements and functions
- WHERE expressions are pre-filtered for joins
- Joined tables are pre-indexed
- AlaSQL uses hash tables for its indexes. Upon index creation, all entries in the table are hashed and stored in a JavaScript object.

ENGINE

AlaSQL enables us to convert SQL language into an Abstract Syntax Tree (AST) from a parsed SQL statement

Statement

```
var ast = alasql.parse('SELECT * FROM table1 WHERE a = b AND a->fn(b->c) > 0');
console.log(ast.statements[0].where);
```

Tree Structure


```
{"expression":
   {"left":
      {"left": {"columnid":"a"},"op":"=", "right":{"columnid":"b"}},
     "op":"AND",
     "right":{"left":
                    {"left":{"columnid":"a"},
                     "op":"->",
                     "right":
                         {"funcid":"fn","args":[
                           {"left":{"columnid":"b"},
                            "op":"->",
                            "right":"c"}
                          ]}},
              "op":">",
              "right":{"value":0}}}}
```