The Design of a SQL Interface for a NoSQL Database

Mary Holstege, PhD, Principal Engineer

Nov 7, 2012

@mathling

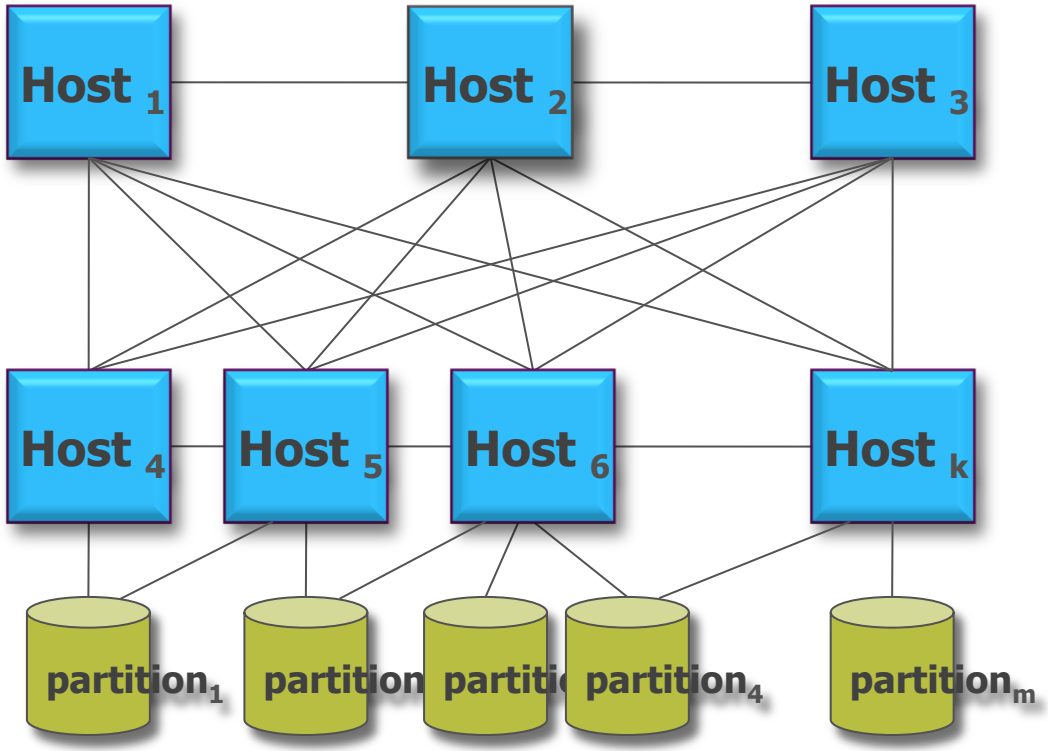
Topics

- MarkLogic: Enterprise NoSQL Database
- SQL over NoSQL, What's The Point?
- How Does It Work?
- Technical Nitty Gritty
- Q&A

MarkLogic

NoSQL Database

- ✓ Shared-nothing
- ✓ Clustered
- ✓ Non-relational
- ✓ Schema-free
- ✓ Scalable

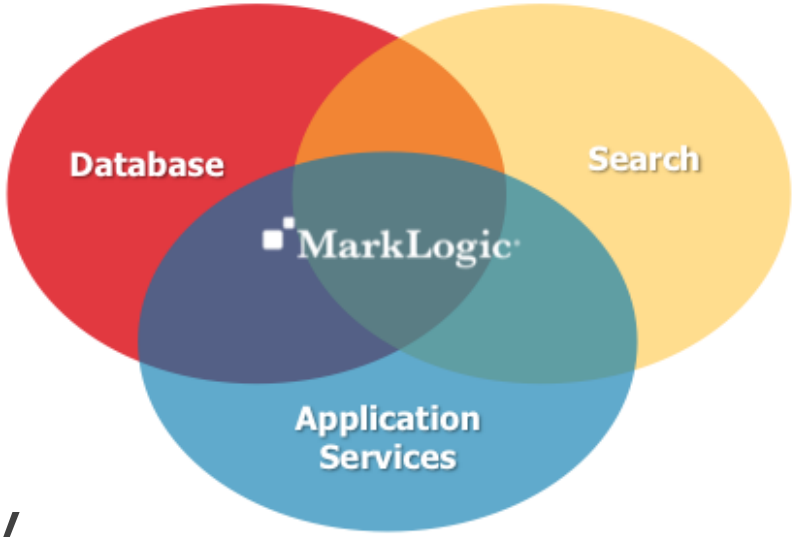


MarkLogic

MarkLogic

Enterprise NoSQL Database

- ✓ ACID
- ✓ Real-time full-text search
- ✓ Automatic failover
- ✓ Replication
- ✓ Point in-time recovery
- ✓ Government-grade security



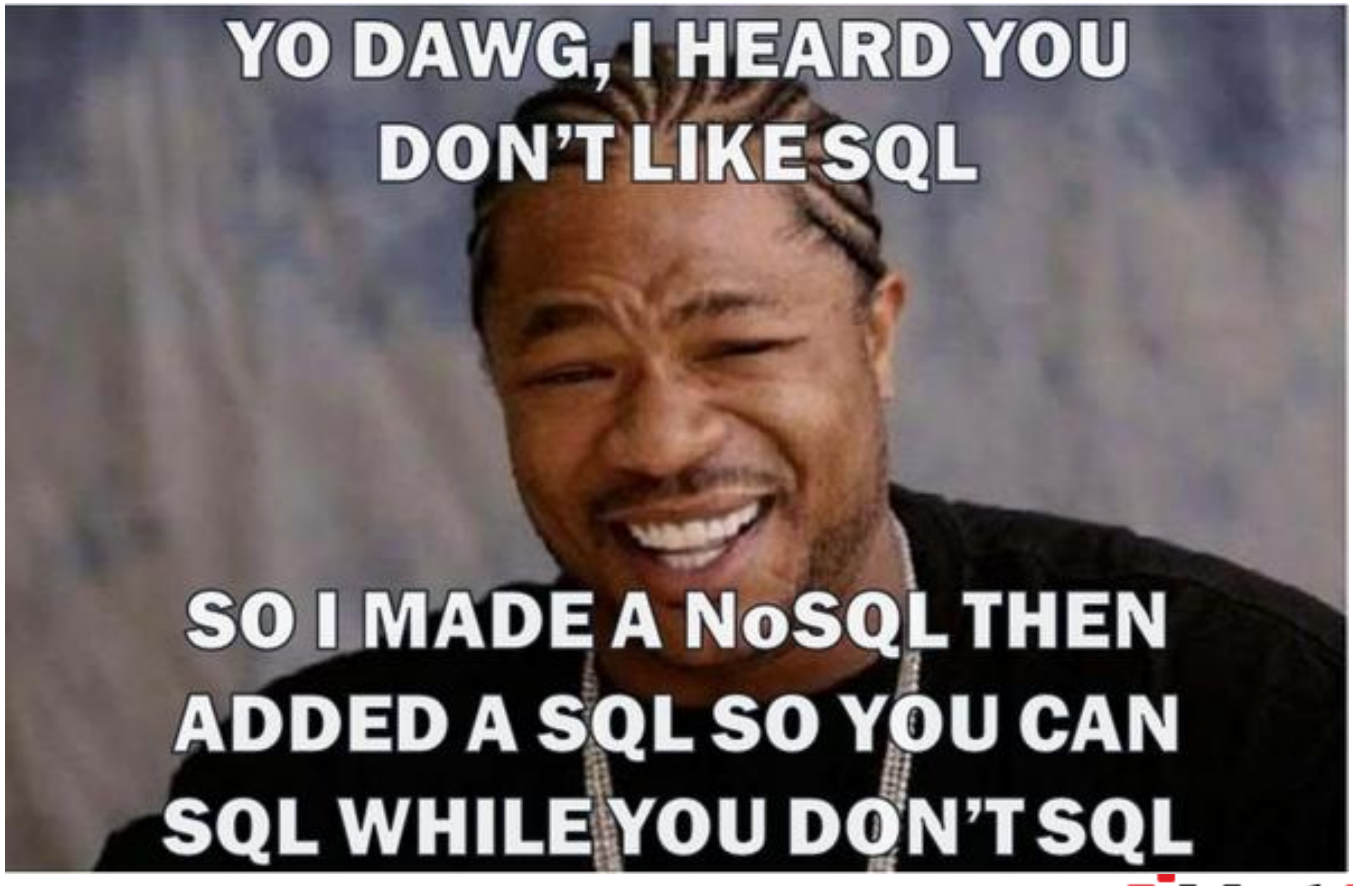
MarkLogic

Keep This In Mind

- Non-relational data model
  - Documents (XML, JSON, binary, text)
- Rich "query" language (XQuery+extension functions)
  - Really a complete language for application development
- Search engine core
  - Full-text
  - Hierarchical, structure
  - Geospatial
  - Values

MarkLogic

What's the Point?



MarkLogic

MarkLogic and BI Tools

- Use familiar relational tools with non-relational data
  - Such as BI tools
  - Standard connection – no code, no custom integration
- All the benefits of a BI tool – data analysis, visualization - with an operational Big Data database

MarkLogic

Structured and Unstructured Data

In 1962, Annan started working as a Budget Officer for the World Health Organization, an agency of the United Nations. From 1974 to 1976, he worked as the Director of Tourism in Ghana. Annan then returned to work for the United Nations as an Assistant Secretary General in three consecutive positions.

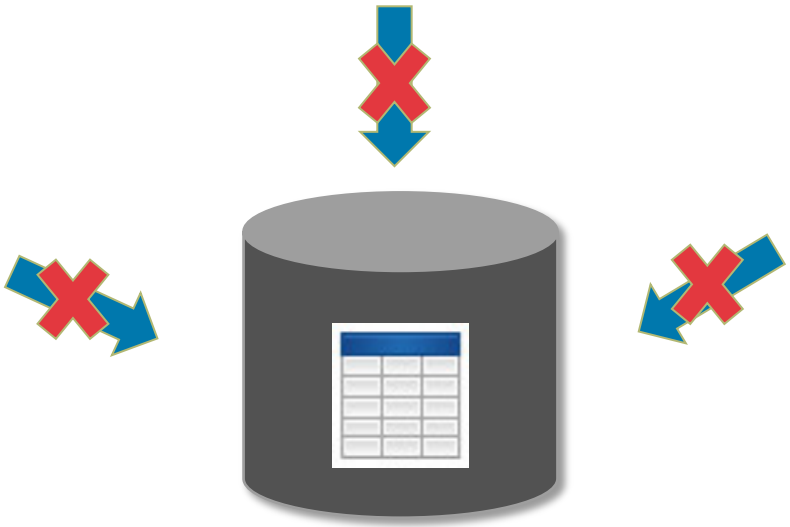
Personal Info
Aliases
Phone numbers
Bank accounts
Credit cards
Vehicles

- MarkLogic's XML data model was designed to handle rich structured and unstructured data

MarkLogic

Structured and Unstructured Data

In 1962, Annan started working as a Budget Officer for the World Health Organization, an agency of the United Nations. From 1974 to 1976, he worked as the Director of Tourism in Ghana. Annan then returned to work for the United Nations as an Assistant Secretary General in three consecutive positions.



Personal Info
Aliases
Phone numbers
Bank accounts
Credit cards
Vehicles

- Richness of unstructured content does not fit naturally into a relational model

MarkLogic

XML vs Tables and Views

- Data is stored in MarkLogic as XML
  - Rich, powerful way to represent complex data
- BI tools expect to see relational tables and views
  - Rows and columns, accessible via SQL

How Can This Possibly Work?

MarkLogic

How Can This Possibly Work?

- Focus on "structured" pieces of the data
- Create an in-memory column index on each piece
- Create a view over column indexes

```
<?xml version="1.0" encoding="UTF-8"?>
<message list="org.codehaus.grails.user" id="3mbfddak67aiefea"
  date="2010-05-17T01:00:21.627923-08:00">
  <headers>
    <from personal="Ian Roberts">i.roberts@dcs.shef.ac.uk</from>
    <to personal="Grails User">user@grails.codehaus.org</to>
    <subject>How to inject a session-scoped service into another service
    </subject>
    </headers>
  <body type="text/plain; charset=us-ascii">
    <para>
    <url>http://ldaley.com/56/scoped-services-in-grails</url> Covers the
    same thing, but has a little bit more detail WRT testing etc.</para>
    <para>Note rather than inject the application context you can also do
<function>myServiceProxy</function>(org.springframework.aop.scope.
ScopedProxyFactoryBean){ targetBeanName = 'myService'  proxyTarget
Class = true}</para>
    <para>Or see <url>http://ldaley.com/42/proxies-in-grails</url> for an
intro to proxies.</para>
    <footer type="signature" depth="1" hash="19865208979997851 97">--
<name>Ian Roberts</name>  | Department of Computer Science
<email>i.roberts@dcs.shef.ac.uk</email>
<affiliation>University of Sheffield, UK</affiliation></footer>
  </body>
</message>
```

MarkLogic

```
<?xml version="1.0" encoding="UTF-8"?>
<message list="org.codehaus.grails.user" id="3mbfddak67aiefea"
                                    0:21.627923-08:00">
                                 Roberts">i.roberts@dcs.shef.ac.uk</from>
    <to personal="Grails User">us                                  </to>
    <subject>How to inject a sess                                another service
    </subject>
    </headers>
  <body type="text/plain; charset=us-ascii">
    <para>
    <url>http://ldaley.com/56/scoped-services-in-grails</url> Covers the
    same thing, but has a little bit more detail WRT testing etc.</para>
    <para>Note rather than inject the application context you can also do
<function>myServiceProxy</function>(org.springframework.aop.scope.
ScopedProxyFactoryBean){ targetBeanName = 'myService'  proxyTarget
Class = true}</para>
    <para>Or see <url>http://ldaley.com/42/proxies-in-grails</url> for an
intro to proxies.</para>
    <footer type="signature" depth="1" hash="19652089799785197">--
<name>Ian Roberts</name>  | Department of Computer Science
<email>i.roberts@dcs.shef.ac.uk</email>
<affiliation>University of Sheffield, UK</affiliation></footer>
  </body>
</message>
```

List

org.codehaus.grails.user

From

i.roberts@dcs.shef.ac.uk

MarkLogic

List	
org.codehaus.grails.user	rails.user
/mm/57	org.codehaus.grails.user
/mm/99	org.ruby-lang.ruby-core
…	…

DOC	From
/mm/21	i.roberts@dcs.shef.ac.uk
/mm/57	j.doe@UoL.org
/mm/99	john.louis@coll.ipsw.uk
…	…

- Every document has a unique identifier in the database
- We can map the document identifier to the list name
- We can do that for every list attribute in every document
- … and we can do that for any piece-of-structure in the database
- These look like columns!

MarkLogic

DOC	List	From
/mm/21	org.codehaus.grails.user	i.roberts@dcs.shef.ac.uk
/mm/57	org.codehaus.grails.user	j.doe@UoL.org
/mm/99	org.ruby-lang.ruby-core	john.louis@coll.ipsw.uk
…	…	…

- Now we can stitch the two "columns" together
  - Using *co-occurrence*

DOC	List	From	Subject	URL	Name	Affiliation
/mm/21	grails	i.roberts	How to ..	http://ldaley.com/56	Ian Roberts	University of Sheffield
/mm/21	grails	i.roberts	How to ..	http://ldaley.com/42	Ian Roberts	University of Sheffield
/mm/57	grails	j.doe	Tips on ..	http://grails.org/88	John Doe	University of Life
/mm/99	ruby	john.louis	New ..	http://ruby.org/33		
…	…	…	…	…	…	…

- Now we can stitch the two "columns" together
  - Using *co-occurrence*
- When we do that for several structured pieces, it starts to look like a table

MarkLogic as a In-memory Columnar Database

- Combine existing capabilities:
  - In-memory distributed indexes
  - Co-occurrence (extended)
- MarkLogic as an in-memory columnar database

DOC	List	From	Subject	URL	Name	affiliation
/mm/21	grails	i.roberts	How to ..	http://ldaley.com/56	Ian Roberts	University of Sheffield
/mm/21	grails	i.roberts	How to ..	http://ldaley.com/42	Ian Roberts	University of Sheffield
/mm/57	grails	j.doe	Tips on ..	http://grails.org/88	John Doe	University of Life
/mm/99	ruby	john.louis	New ..	http://ruby.org/33		
...	...	...	...	...	...	...

MarkLogic

How Does It Work?

MarkLogic

Mapping Unstructured Data

Indexes have typed values, but text is text

- Rich set of types
  - Numbers (decimal, float, double, …)
  - Strings (string, name, uri, …)
  - Date/time (date, time, timestamp, …)
  - Geospatial (points)
- But maybe the value has the wrong type
  - `Not a real date <date>2001/755</date> ⇒ (nothing)`
- Maybe there is an empty value
  - `<message list="" id="3mbfddak67aiefea"... ⇒ ""`
- Maybe there is no value at all
  - `<p>There is no function here.</p> ⇒ (nothing)`

MarkLogic

Mapping Unstructured Data

Indexes can select specific elements or attributes

- Named element
  - `<function>myServiceProxy</function>`

    `⇒ "myServiceProxy"`
- Named attribute of an element
  - `<message list="org.codehaus.grails.user"...`

    `⇒ "org.codehouse.grails.user"`
- Multiple values OK
  - `<function>myServiceProxy</function> and`
    `<function>myServiceStub</function>`

    `⇒ "myServiceProxy","myServiceStub"`

MarkLogic

Mapping Unstructured Data

Indexes can select concatenated values

- Concatenated values of specific elements
  - `<caption>A <b>good</b> example</caption>`

    `⇒ "A good example"`
- Maybe some children should be skipped
  - `<caption>A good example<footnote>Please exclude this text</footnote></caption>`

    `⇒ "A good example"`
- Conditional inclusion/exclusion based on attribute values
  - `<p role="example">This one.</p><p>Not that one.</p>`

    `⇒ "This one."`

MarkLogic

Mapping Unstructured Data

Indexes can select complex paths

- Selecting a targeted structural relationship: `section/title`
  - `<chapter><title>Chapter One</title><section><title>Section A</title>...`

    ⇒ `"Section A"`
- Selecting multiple related elements: `/info/(person|org)`
  - `<person>John Smith</person> is at <org>IBM</org>`

    ⇒ `"John Smith","IBM"`
- Selecting based on comparison predicates: `person[@age>18]`
  - `<person age="47">John Smith</person> and his son <person age="12">Jack Smith</person>`

    ⇒ `"John Smith"`

MarkLogic

Mapping Unstructured Data

Indexes can select correlated parts (geospatial)

- Paired attributes
  - `<place lat="0.1" lng="24.24"/>`

    ⇒ `point(0.1,24.24)`

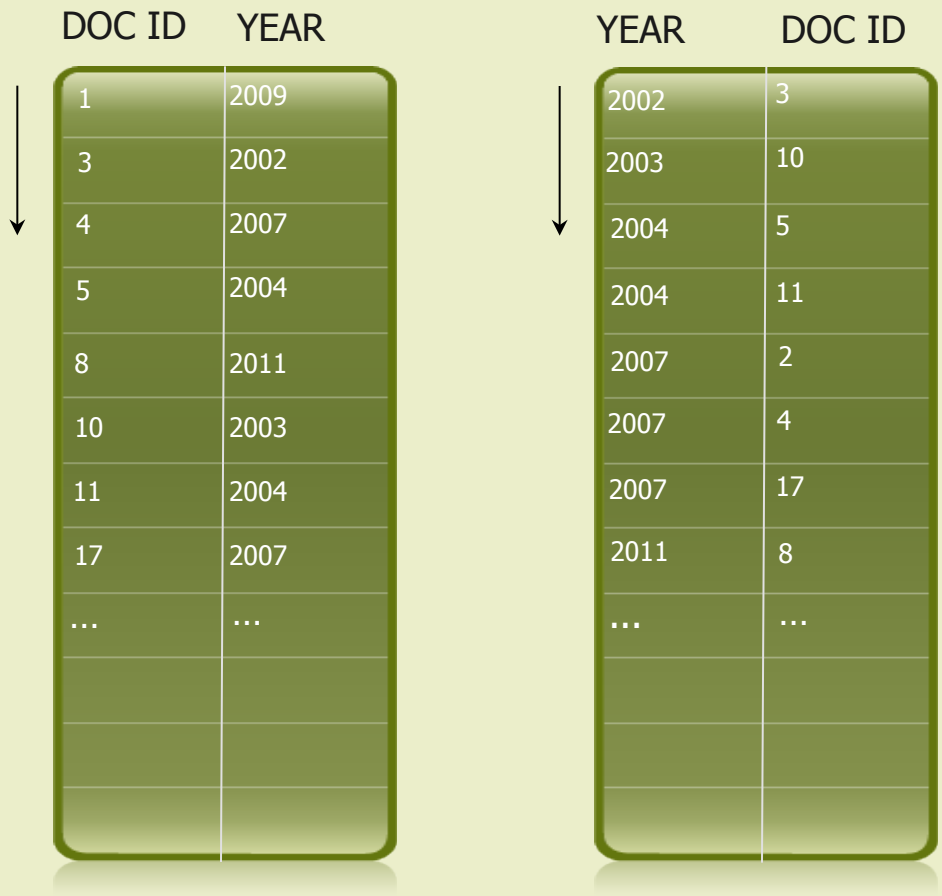
- Paired elements
  - `<pos><long>-110.44</long><lat>39.2</lat></pos>`

    ⇒ `point(39.2,-110.44)`

MarkLogic

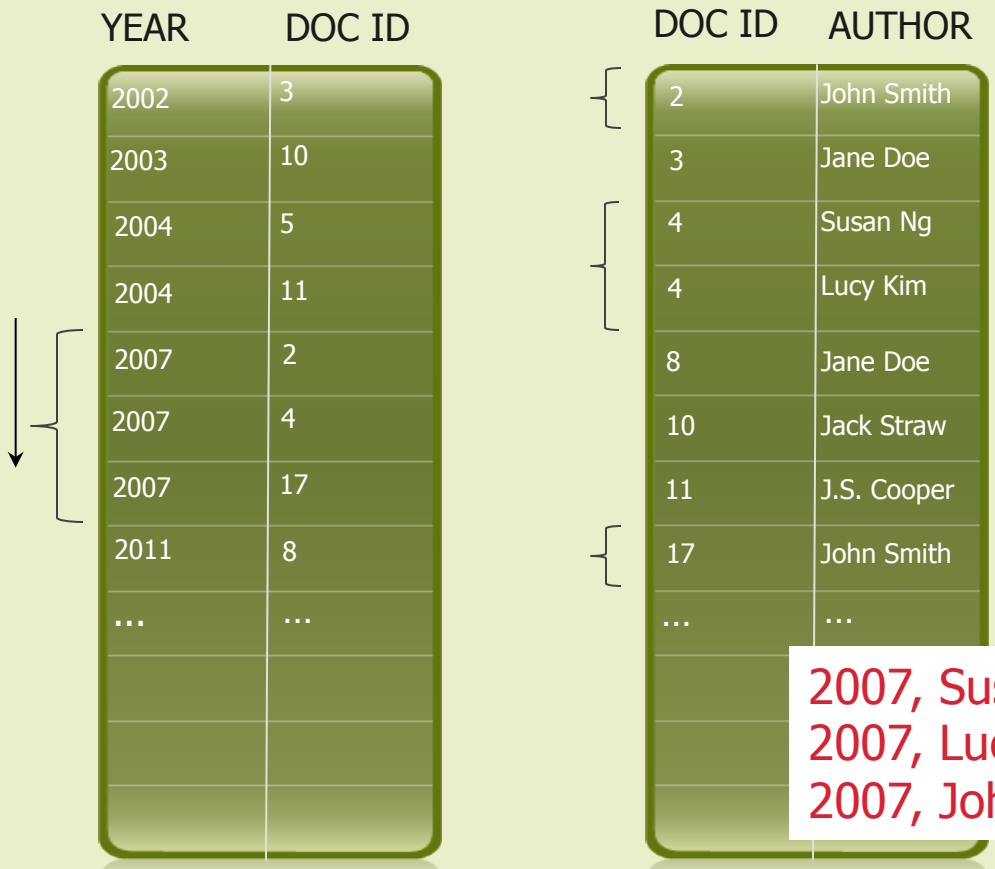
Indexes

Map document ids to values, and values to document ids in a compact distributed in-memory representation



Co-occurrences

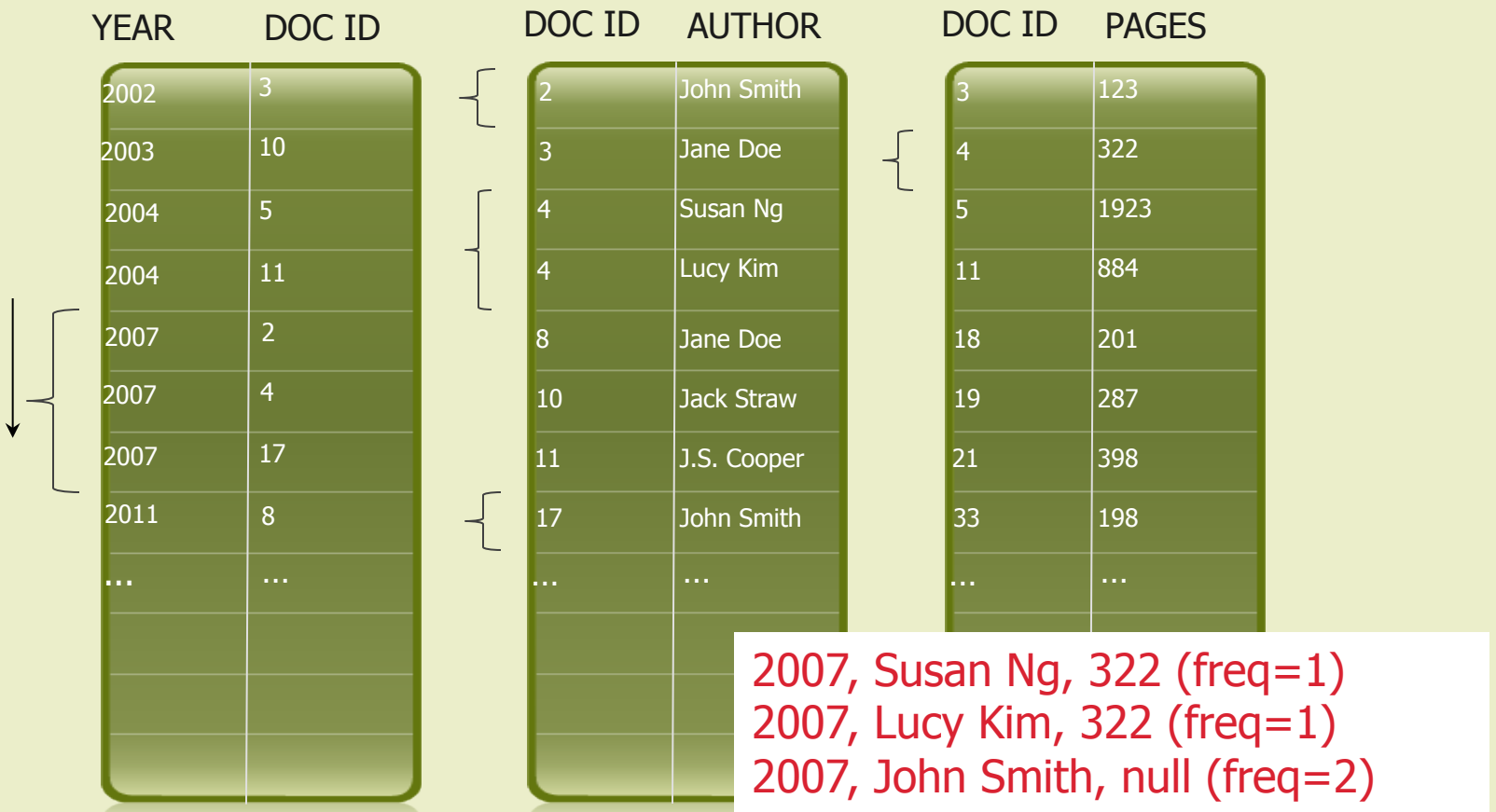
Correlate values in year index and author index



2007, Susan Ng (freq=1)
2007, Lucy Kim (freq=1)
2007, John Smith (freq=2)

Co-occurrence

Correlate year values and author values and pages values



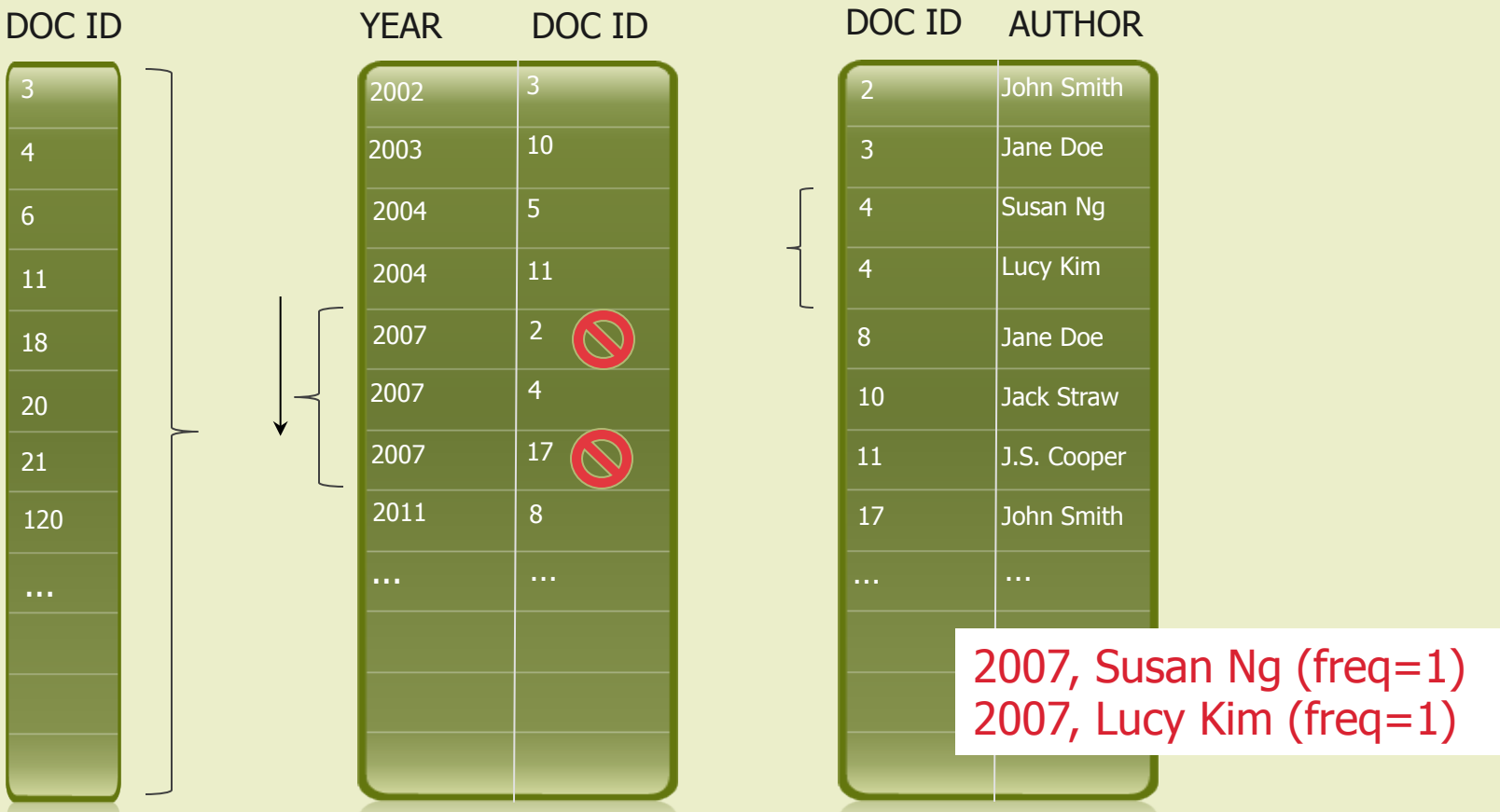
Full-Text Search As Part Of A Query

```
SELECT
  year, author
  FROM books
  WHERE title MATCH 'run'
```

- Leverage MarkLogic's fast full-text search
  - Search entire document or within individual columns
  - Using tools' custom SQL or pass-through SQL capability
- Pre-filter the data sent to the SQL tool
  - Select a subset of your data for analysis

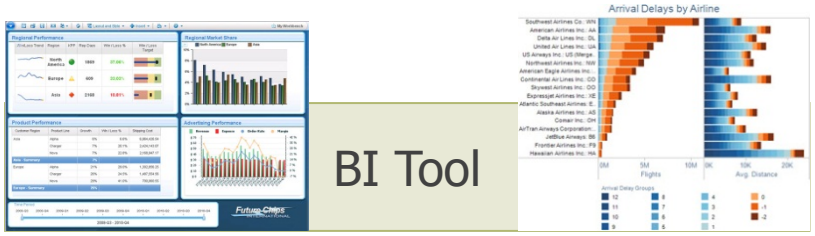
Constrained Co-occurrence

Correlate year values and author values, constrained by full-text search



Overall Architecture

Connecting to MarkLogic over ODBC



BI Tool

ODBC driver manager

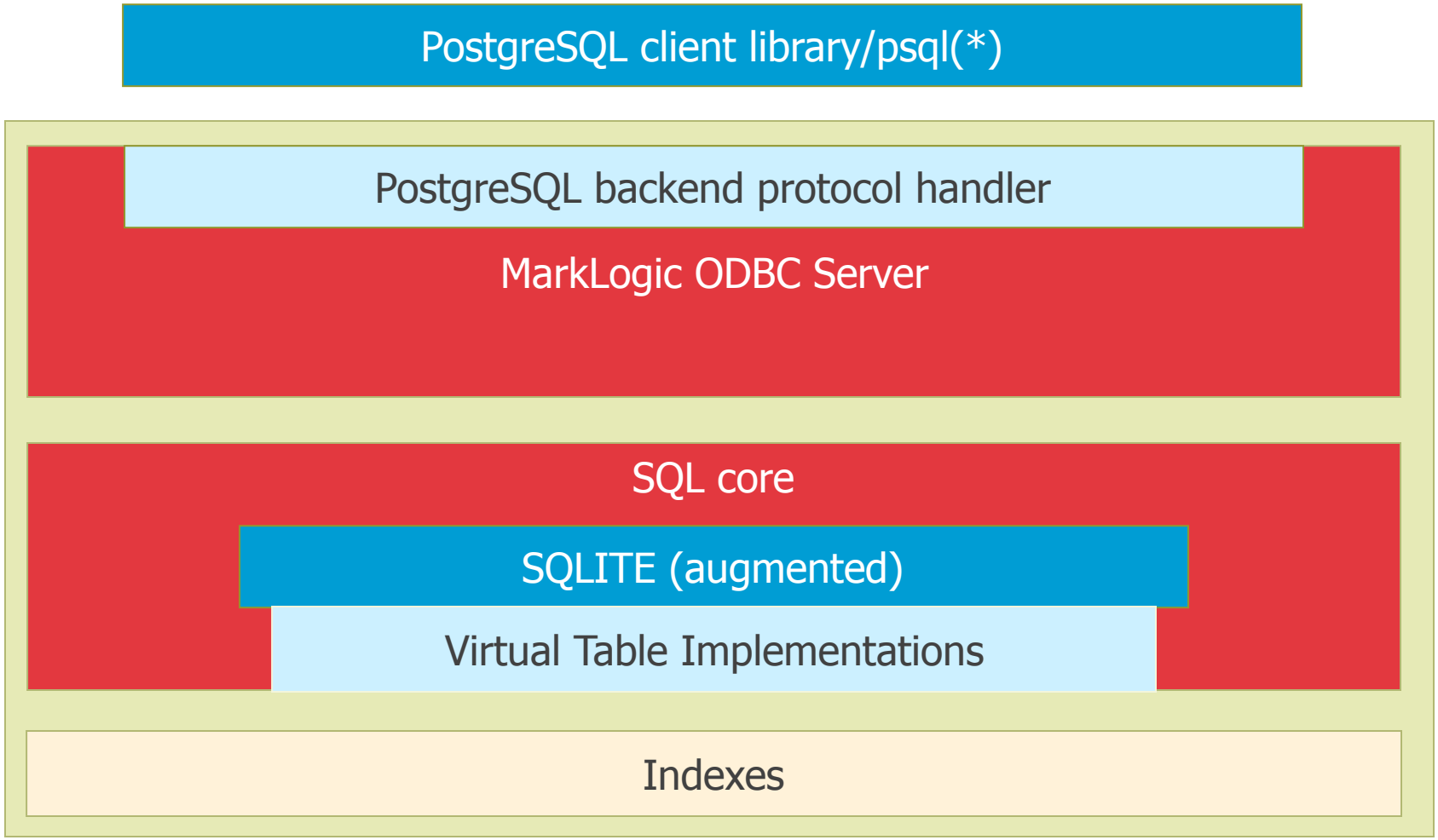
ODBC driver

TCP/IP

MarkLogic ODBC server

MarkLogic Server

A Closer Look



MarkLogic

PostgreSQL

- Client library mature, well supported by ODBC tools
- Protocol well-documented
- Hoped to use as-is

- Why not PostgreSQL all the way?
  - Not embeddable
  - Not threadsafe

MarkLogic

SQLITE

- Trivially embeddable
- Multi-thread friendly
- Useful extension points

- Why not use SQLITE all the way?
  - Designed purely for embedding
  - No ODBC drivers

MarkLogic

Sticky Bits

Issues

System tables

- PostgreSQL client libraries hardcode system table access
- SQLITE doesn't have system tables

- Created modified versions of client library and psql
- Added virtual table interface for system tables

MarkLogic

Exposing Built-in (XQuery) Functions

```
SELECT
  fn_replace(url,"http://([^/]+)/.*","$1")
  FROM emails
  WHERE subject MATCH "answer"
```

```
www.marklogic.com
stackoverflow.com
mars.jpl.nasa.gov
```

MarkLogic

Issues

SQLITE types

- Resolving three different type systems

- Manifest vs definitive typing
  - XQuery functions expect definitive types
  - PostgreSQL protocol expects definitive types
  - SQLITE does manifest typing

MarkLogic

Performance

Preserve scalability and performance in SQL context

- Push work to distributed memory-mapped indexes
  - We have aggregate framework already
- Want search constraints pre-filtering co-occurrences
  - Complexity of constraint not an issue
- We expect sparse relations
  - Performance of co-occurrence goes number of indexes correlated

MarkLogic

Issues

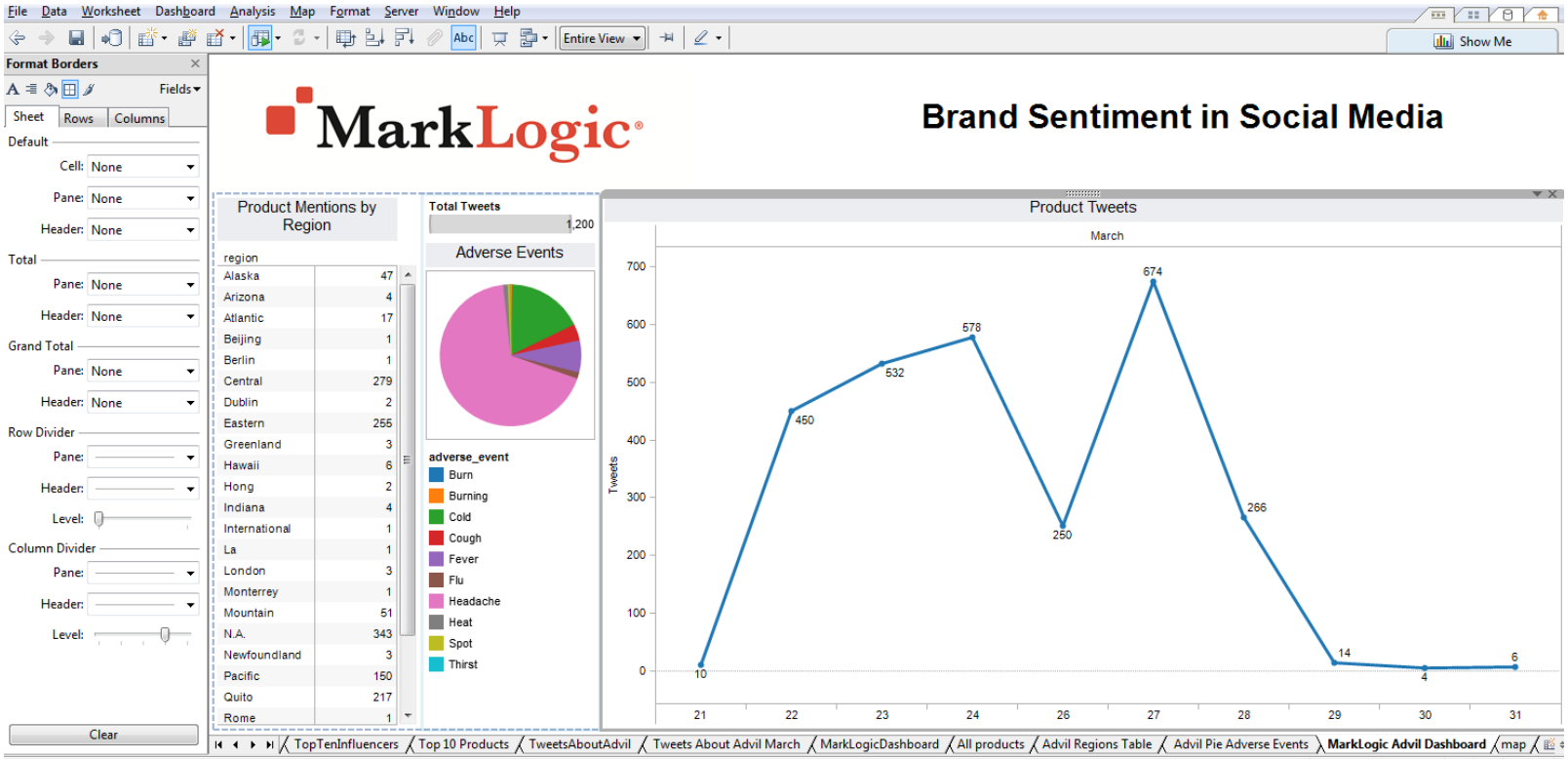
SQLITE optimizer assumes simplistic virtual tables

- Doesn't push limit, offset, or distinct to virtual tables
- Doesn't do aggregate push-down to virtual tables
- Doesn't do OR-clause optimization for MATCH
- Assumes that virtual tables always return full relation

- Augmented virtual table interface, optimizer, and VM to handle these things

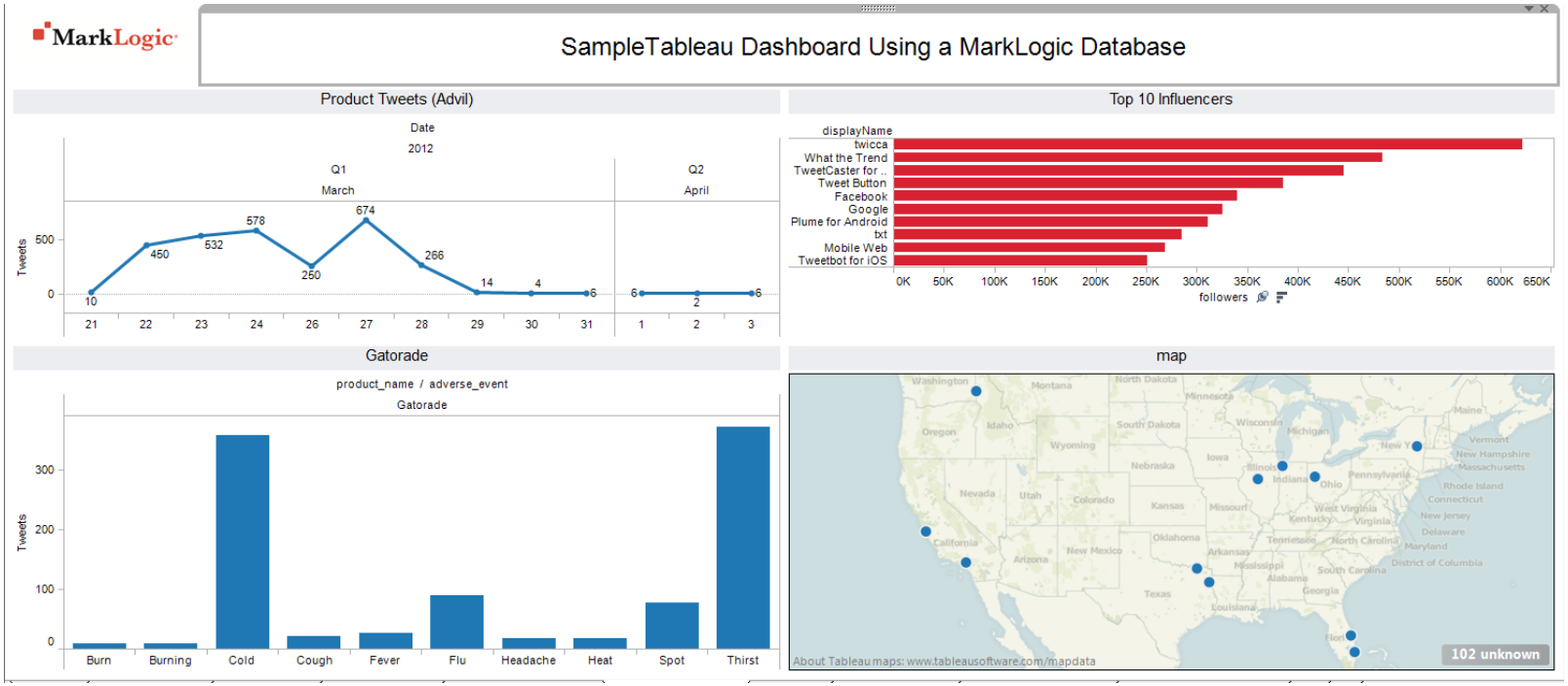
MarkLogic

Tableau



MarkLogic

Tableau



MarkLogic

In Conclusion

- Have Big Data your way
  - Native MarkLogic APIs
    - XQuery, Java, REST, HTTP, …
  - SQL/ODBC
- Integration with SQLITE core and PostgreSQL client
  - Some challenges
  - Mostly straightforward
- Try it out!
  - Free license available at developer.marklogic.com

MarkLogic

Q&A

Interaction with SQLITE

- Virtual table interface provides access to range index views
- Wire MATCH operator to MarkLogic full-text search
- Virtual table interface exposes system tables
- sqlite3_collation_required hook exposes built-in collations
- sqlite3_trace and sqlite3_progress_handler hooks connect to server timeout and logging systems
- Bridge functions link built-in XQuery functions to SQL context via sqlite3_create_function_v2

MarkLogic

Extensions to SQLITE

Additions to Virtual Table interface

- Added xDestroy method to xFindFunction to clean up function context object
- Added xAggregate/xAcceptAggregate methods to allow push-down of aggregates
- Added xRequiredColumn method to communicate full set of columns needed out of relation, not just those in constraints
- Added xFirstIndex method and extended or-clause optimization to virtual table
- Extended information passed to xBestIndex to allow for better/ pushed-down distinct, limit, offset, and or-clause optimization

MarkLogic

Extensions to SQLITE

Miscellaneous

- Added new opcode VAggregate that pushed aggregate to virtual table
- Added additional datatypes (specifically unsigned int, long)
- Added pointer to executing statement to cursor, and method for obtaining it
- ISO 8601 date parsing
- Propagated errors from vtab out
- Extended or-clause optimization to MATCH
- Fixed SOUNDEX implementation
- Better error handling from built-in aggregates
- Additional pragmas to get at function information
- Extended table info pragma to get schema and table names directly