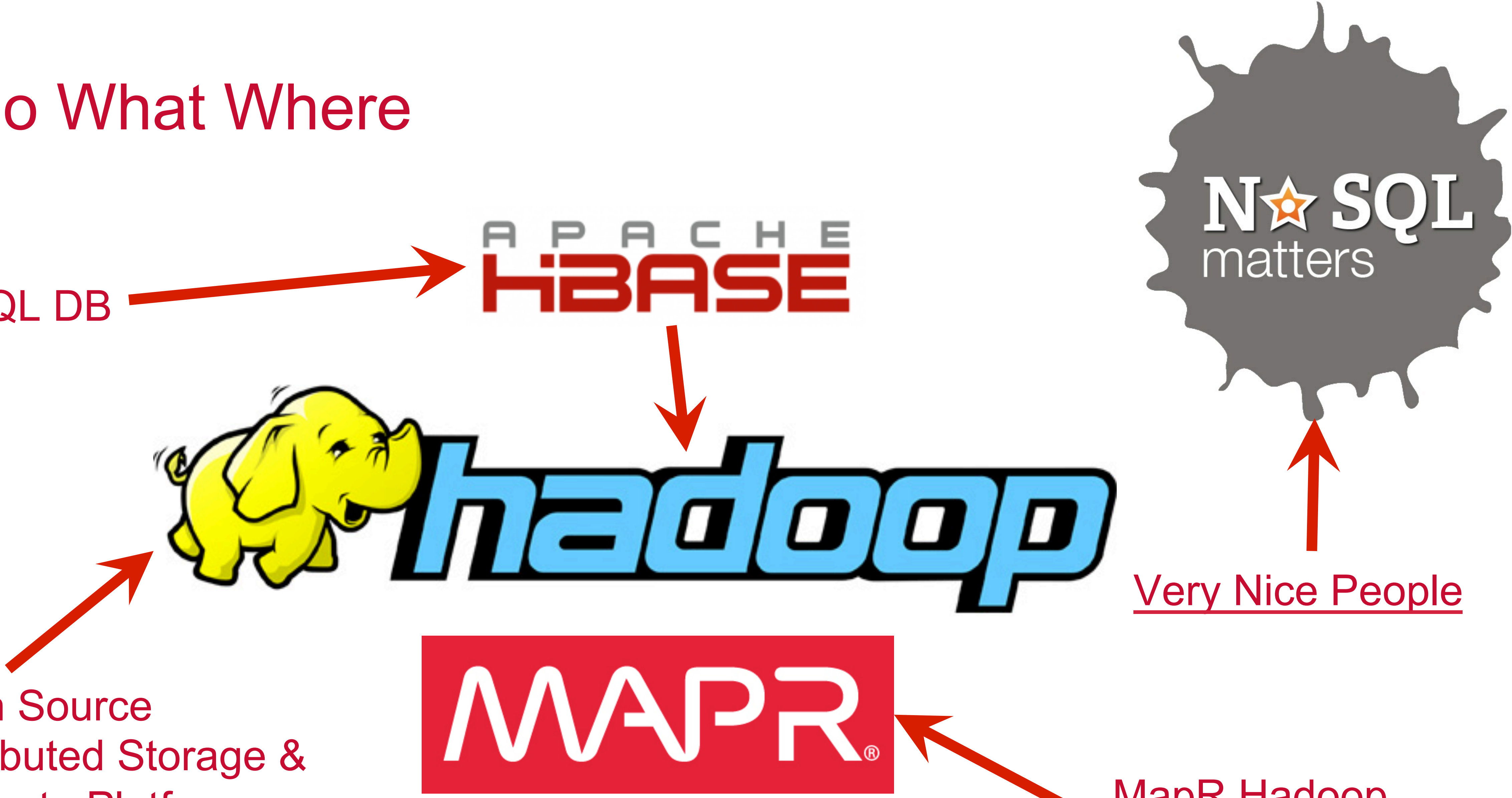
# SQL on NoSQL (and all of the data) With Apache Drill

Richard Shaw

Solutions Architect

@aggress

# Who What Where

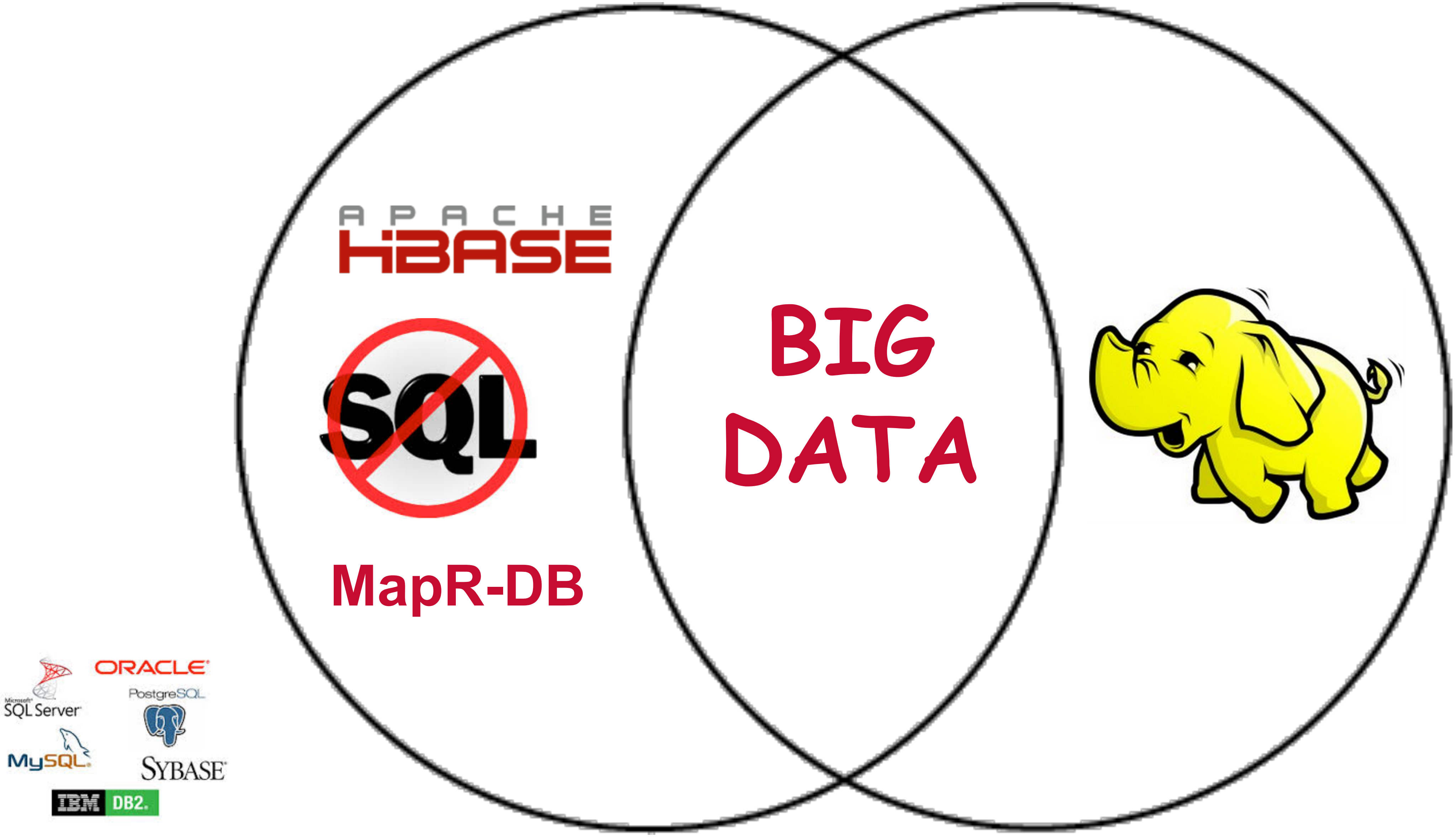


NoSQL DB

Open Source
Distributed Storage &
Compute Platform
(up to 1000s of nodes)

Very Nice People

MapR Hadoop
MapR-DB

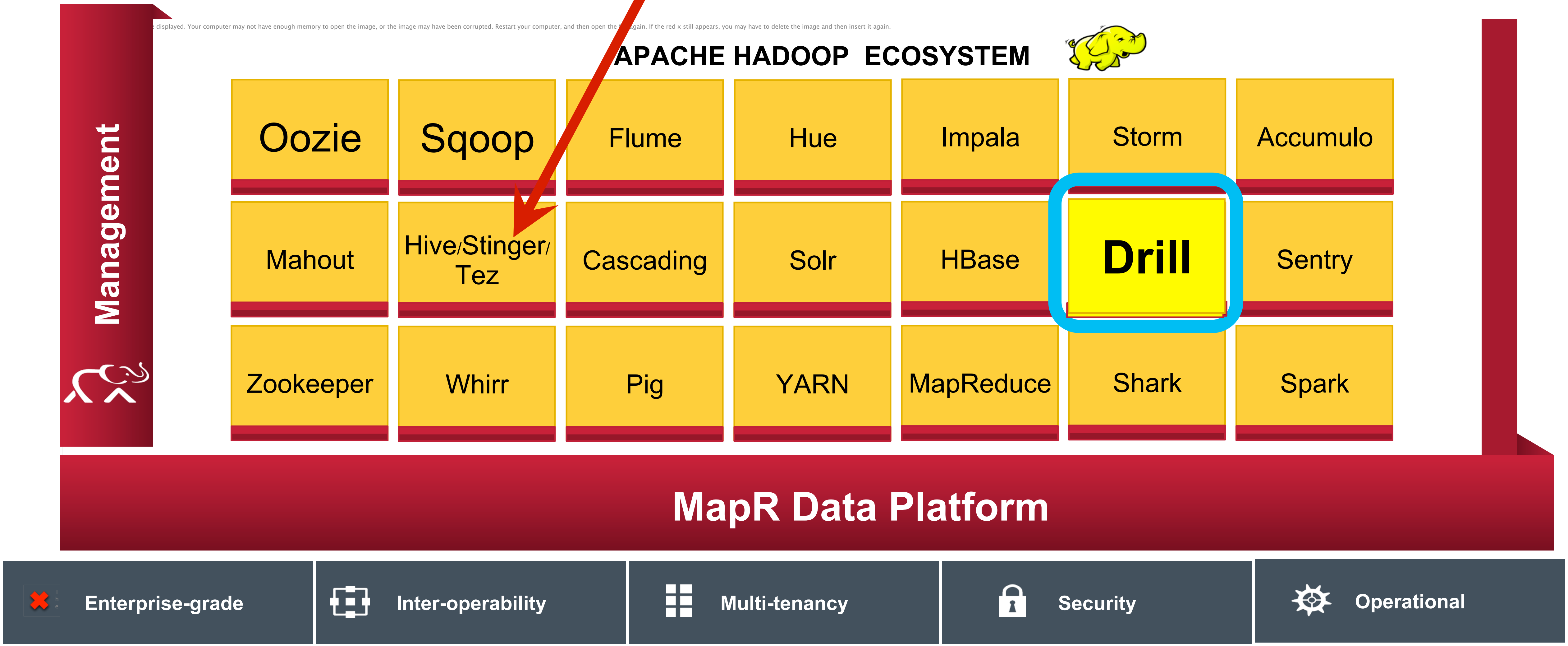


BIG DATA

MapR-DB

# Hadoop World

HiveQL to
Map Reduce



e displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

# Low Latency SQL on NoSQL and Other Stuff

# Real-World Data Modeling and Transformations

# What's Drill?

- Apache open source project
- Scale-out execution engine for low-latency SQL queries
- Unified SQL-based API for zero day analytics & operational applications
- Flexible data sources
- Data agility for NoSQL, HBase, Hadoop

Mentored by
-MapR
-Lucidworks
-Elasticsearch
-Academics

Power to Users

The Sexy Bit

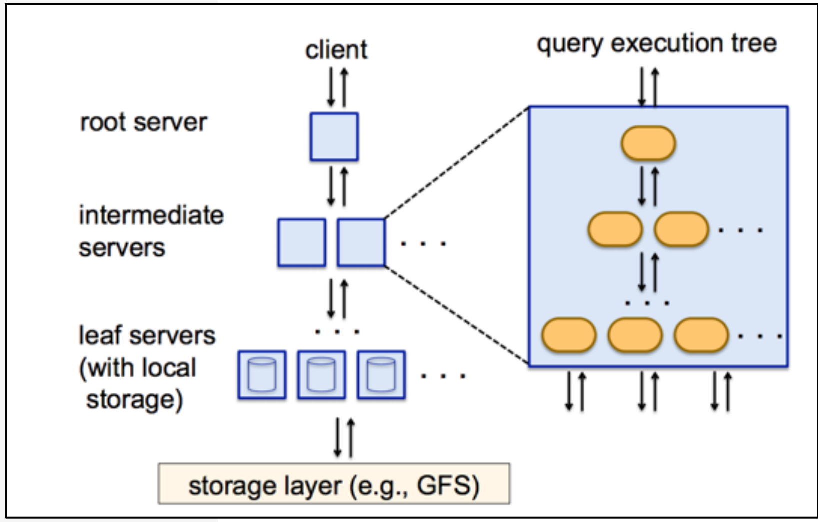
The Useful Bit

# Drill and Google Dremel

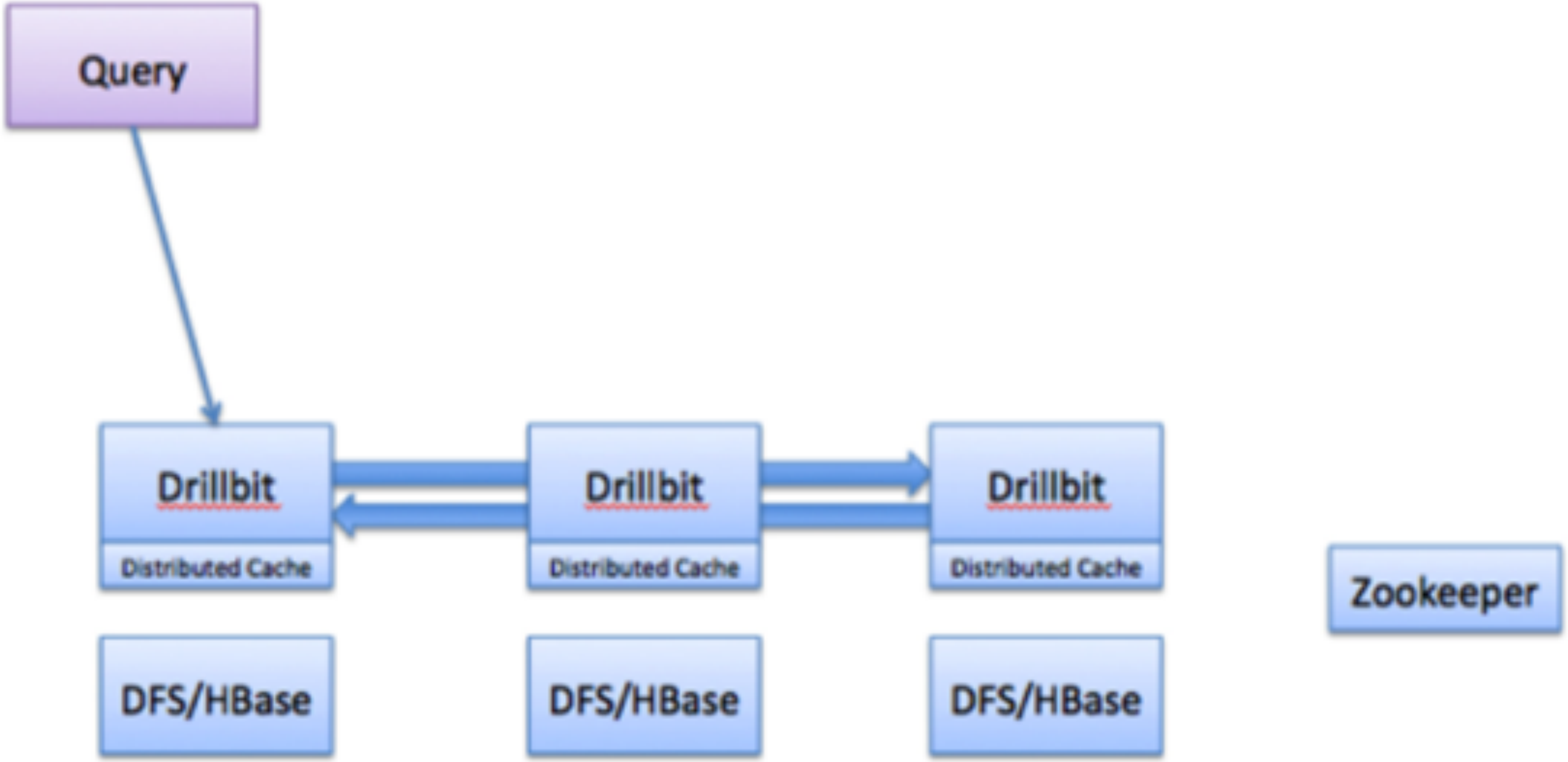
- Google Tech
- SQL querying of Google data over GFS & BigTable
- In use production use since 2006 - **8 YEARS!**
- Tens of thousand of concurrent users over PB of data
- Dremel paper released 2010

# Dremel Architecture

- Designed for columnar format data (nested data)
- Analysis in-situ -> Ad hoc version of Map Reduce
- Multi-level execution trees
- Load balancing across Tablets
- Handles contended or failed queries - redirects



# Drill Architecture



# Drill Architecture Highlights

- Flexible - Dynamic Schema Discovery & Data Model (nested etc). Many data formats
- Extensible - Java API. Work with RDBMS' and NoSQL DBs
- Performance - Distributed engine, columnar optimised

# Self-Describing Data is Ubiquitous

Flat files in a distributed file system

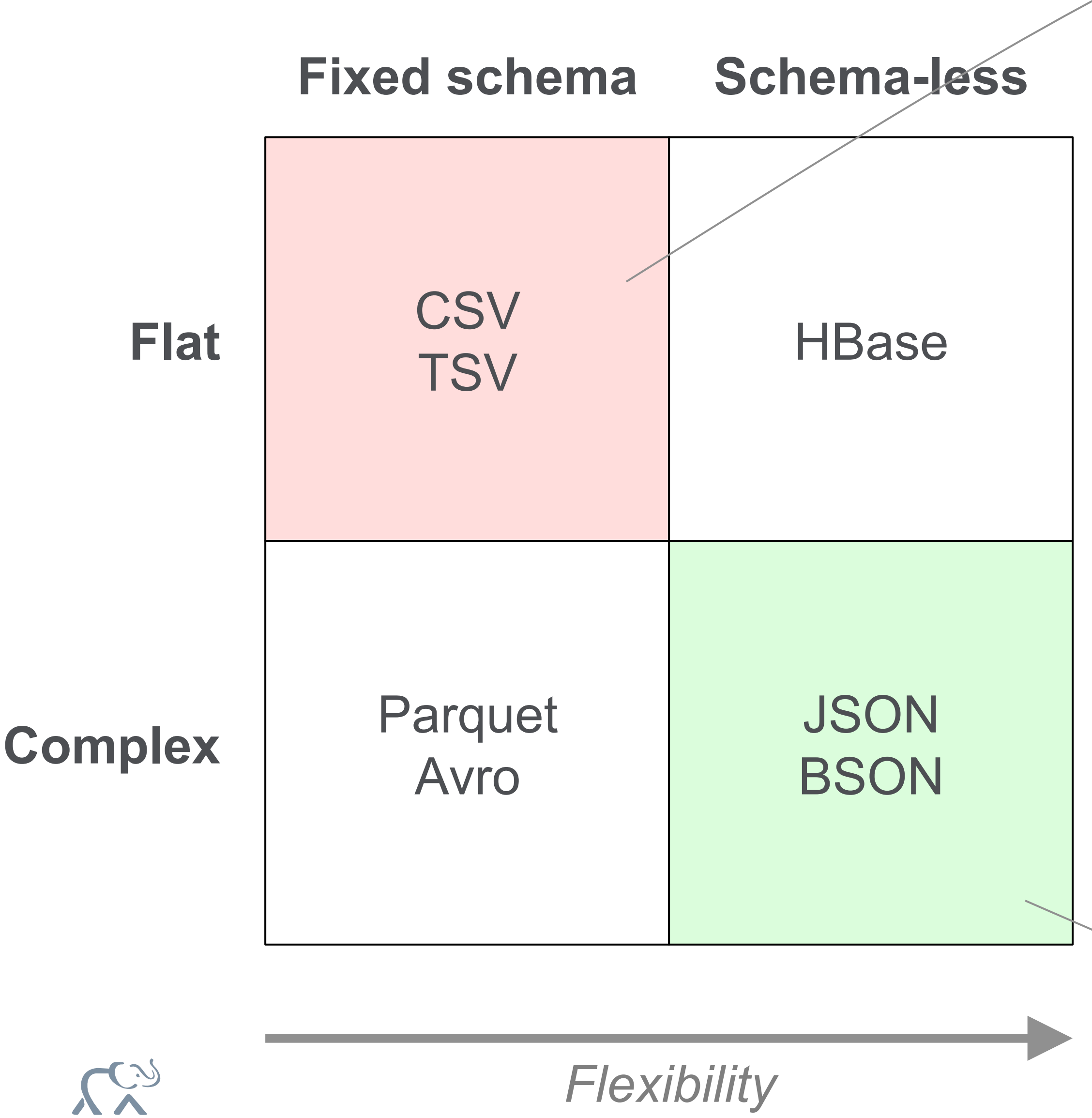
- Complex data (Thrift, Avro, protobuf)
- Columnar data (Parquet, ORC)
- Loosely defined (JSON)
- Traditional files (CSV, TSV)

Data stored in NoSQL stores

- Relational-like (rows, columns)
- Sparse data (NoSQL maps)
- Embedded blobs (JSON)
- Document stores (nested objects)

```
{
  name: {
    first: Michael,
    last: Smith
  },
  hobbies: [ski, soccer],
  district: Los Altos
}
{
  name: {
    first: Jennifer,
    last: Gates
  },
  hobbies: [sing],
  preschool: CCLC
}
```

# Drill's Data Model is Flexible



| | Fixed schema | Schema-less |
|---|---|---|
| **Flat** | CSV<br>TSV | HBase |
| **Complex** | Parquet<br>Avro | JSON<br>BSON |

*Flexibility*

This Was
The Sexy Bit

Drill Sees in JSON

RDBMS/SQL-on-Hadoop table

| Name | Gender | Age |
|---|---|---|
| Michael | M | 6 |
| Jennifer | F | 3 |
| | | |

Apache Drill table

```
{
  name: {
    first: Michael,
    last: Smith
  },
  hobbies: [ski, soccer],
  district: Los Altos
}
{
  name: {
    first: Jennifer,
    last: Gates
  },
  hobbies: [sing],
  preschool: CCLC
}
```

# Quick Tour

# Self-Service Data Exploration with Apache Drill

# Zero to Results in 2 Minutes (3 Commands)

```
$ tar xzf apache-drill.tar.gz

$ apache-drill/bin/sqlline -u jdbc:drill:zk=local

0: jdbc:drill:zk=local>
  SELECT   count(*) AS incidents, columns[1] AS category
  FROM     dfs.`/tmp/SFPD_Incidents_-_Previous_Three_Months.csv`
  GROUP BY columns[1]
  ORDER BY incidents DESC;
+------------+------------+
| incidents  |  category  |
+------------+------------+
| 8372       | LARCENY/THEFT |
| 4247       | OTHER OFFENSES |
| 3765       | NON-CRIMINAL |
| 2502       | ASSAULT    |
...
35 rows selected (0.847 seconds)
```

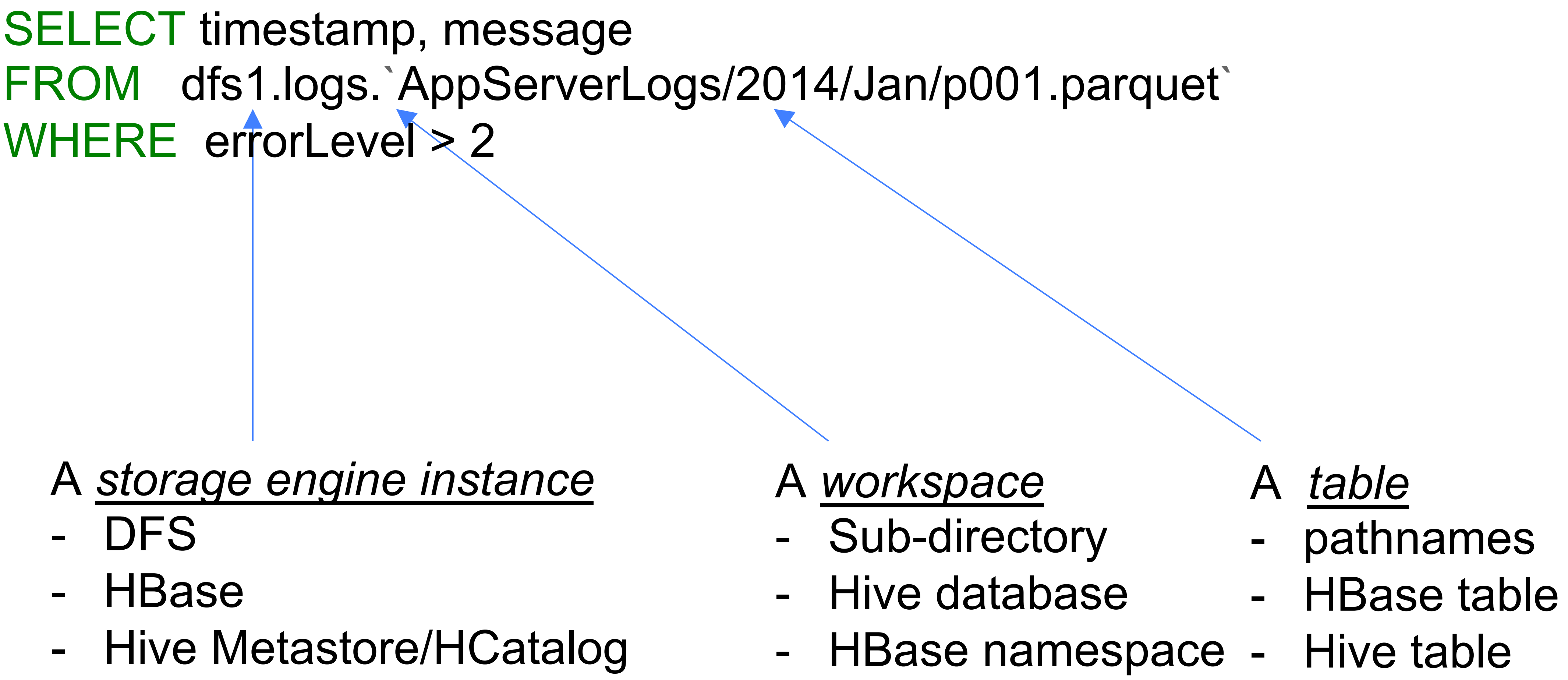
Install

Launch shell (embedded mode)

Query

Results

# Data Source is in the Query



```
SELECT timestamp, message
FROM   dfs1.logs.`AppServerLogs/2014/Jan/p001.parquet`
WHERE  errorLevel > 2
```

A *storage engine instance*

- DFS
- HBase
- Hive Metastore/HCatalog

A *workspace*

- Sub-directory
- Hive database
- HBase namespace

A *table*

- pathnames
- HBase table
- Hive table

# Data Sources

- JSON
- CSV
- ORC (ie, all Hive types)
- Parquet
- HBase tables
- … can combine them

```
Select  USERS.name,
PROF.emails.work  from
  dfs.logs.`/data/logs`  LOGS,
  dfs.users.`/profiles.json`  USERS,
where
   LOGS.uid = USERS.uid   and
   errorLevel > 5
order by  count(*);
```

# Query Directory Trees

```
# Query file: How many errors per level in Jan 2014?

SELECT   errorLevel, count(*)
FROM     dfs.logs.`/AppServerLogs/2014/Jan/part0001.parquet`
GROUP BY errorLevel;

# Query directory sub-tree: How many errors per level?

SELECT   errorLevel, count(*)
FROM     dfs.logs.`/AppServerLogs`
GROUP BY errorLevel;

# Query some partitions: How many errors per level by month from 2012?

SELECT   errorLevel, count(*)
FROM     dfs.logs.`/AppServerLogs`
WHERE    dirs[1] >= 2012
GROUP BY errorLevel, dirs[2];
```

# Works with HBase and Embedded Blobs

```
# Query an HBase table directly (no schemas)

SELECT cf1.month, cf1.year
FROM   hbase.table1;

# Embedded JSON value inside column profileBlob inside column family cf1 of
the HBase table users

SELECT profile.name, count(profile.children)
FROM (
  SELECT CONVERT_FROM(cf1.profileBlob, 'json') AS profile
  FROM hbase.users
)
```

# Combine Data Sources on the Fly

```
# Join log directory with JSON file (user profiles) to identify the name and email address for
anyone associated with an error message.

SELECT DISTINCT users.name, users.emails.work
FROM          dfs.logs.`/data/logs` logs,
          dfs.users.`/profiles.json` users
WHERE          logs.uid = users.id AND
          logs.errorLevel > 5;

# Join a Hive table and an HBase table (without Hive metadata) to determine the number of
tweets per user

SELECT  users.name, count(*) as tweetCount
FROM    hive.social.tweets tweets,
      hbase.users users
WHERE   tweets.userId = convert_from(users.rowkey, 'UTF-8')
GROUP BY tweets.userId;
```

# Use ANSI SQL with no modifications

```
# TPC-H standard query 4

SELECT
o.o_orderpriority, count(*) AS order_count
FROM orders o
WHERE o.o_orderdate >= date '1996-10-01'
    AND o.o_orderdate < date '1996-10-01' + interval '3' month
    AND EXISTS(
            SELECT * FROM lineitem l
            WHERE l.l_orderkey = o.o_orderkey
            AND l.l_commitdate < l.l_receiptdate
            )
    GROUP BY o.o_orderpriority
    ORDER BY o.o_orderpriority;
```

# Seamless integration with Apache Hive

- Low latency queries on Hive tables
- Support for 100s of Hive file formats
- Ability to reuse Hive UDFs
- Support for multiple Hive Metastores in a single query

# What’s the Worst That Can Happen

LIVE DEMO

# Think of the use cases

- Querying against many different data sets
- JDBC & ODBC connection to BI & Analytical tools
- Power to the users - Think Google Dremel
- Supporting different formats and databases

# The Present

- Drill 0.5.0 is out
- Go play with it
- https://incubator.apache.org/drill/

Very active development

10 minutes to get running

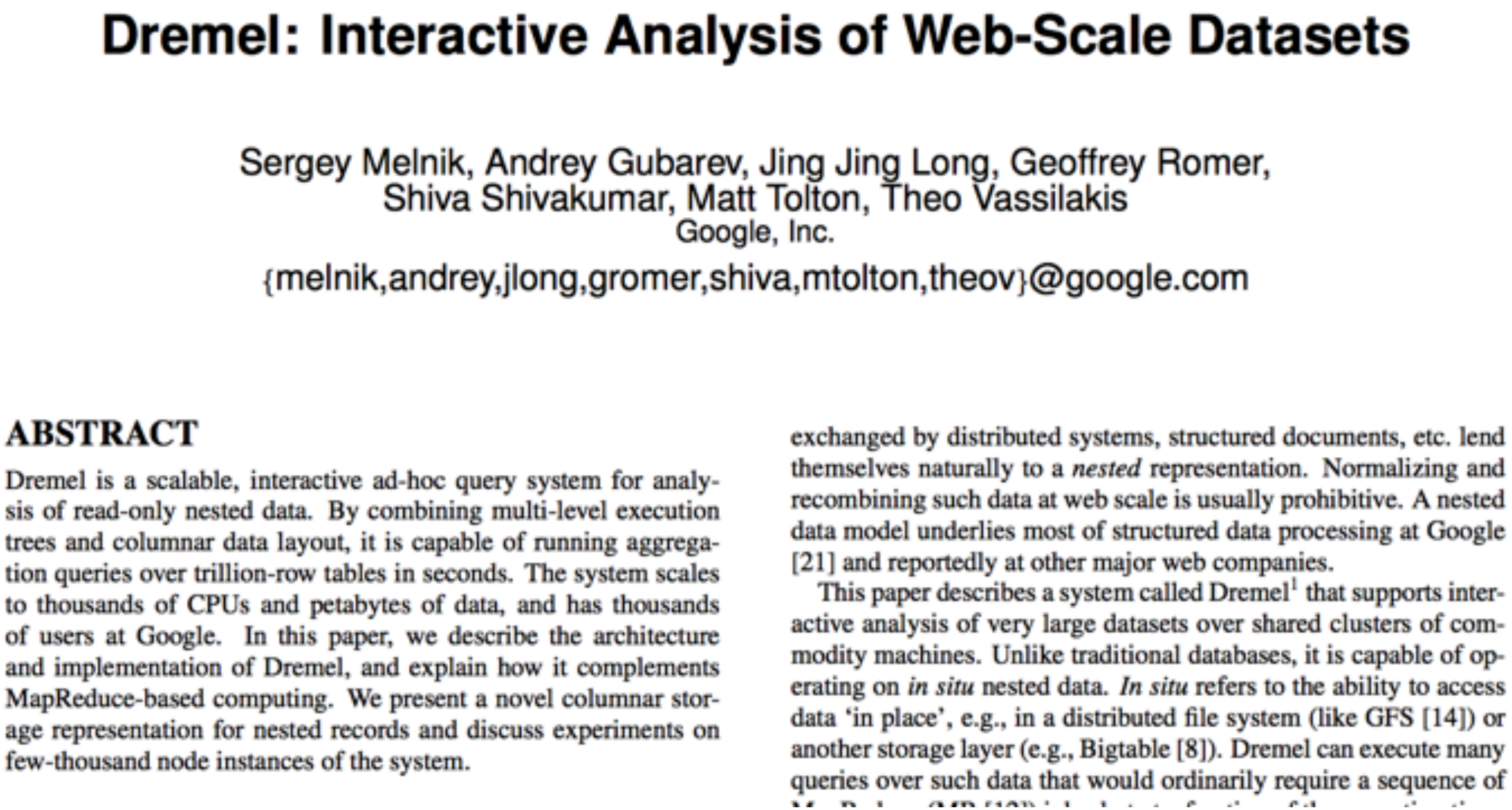
Actively Supported by the community

# The Future

SQL for $NoSQL $RDBMS

- Full Beta this month
- Production grade 1.0 before end of 2014
- Monthly releases
- Support for other NoSQL databases & data sources
- One SQL to rule them all

# The Paper



## Dremel: Interactive Analysis of Web-Scale Datasets

Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer,
Shiva Shivakumar, Matt Tolton, Theo Vassilakis
Google, Inc.
{melnik,andrey,jlong,gromer,shiva,mtolton,theov}@google.com

## ABSTRACT

Dremel is a scalable, interactive ad-hoc query system for analysis of read-only nested data. By combining multi-level execution trees and columnar data layout, it is capable of running aggregation queries over trillion-row tables in seconds. The system scales to thousands of CPUs and petabytes of data, and has thousands of users at Google. In this paper, we describe the architecture and implementation of Dremel, and explain how it complements MapReduce-based computing. We present a novel columnar storage representation for nested records and discuss experiments on few-thousand node instances of the system.

exchanged by distributed systems, structured documents, etc. lend themselves naturally to a *nested* representation. Normalizing and recombining such data at web scale is usually prohibitive. A nested data model underlies most of structured data processing at Google [21] and reportedly at other major web companies.

This paper describes a system called Dremel[1] that supports interactive analysis of very large datasets over shared clusters of commodity machines. Unlike traditional databases, it is capable of operating on *in situ* nested data. *In situ* refers to the ability to access data 'in place', e.g., in a distributed file system (like GFS [14]) or another storage layer (e.g., Bigtable [8]). Dremel can execute many queries over such data that would ordinarily require a sequence of

# The Summary

### AGILITY
INSTANT INSIGHTS TO BIG DATA

- Direct queries on self describing data
- No schemas or ETL required

### FLEXIBILITY
ONE INTERFACE
FOR HADOOP & NOSQL

- Query HBase and other NoSQL stores
- Use SQL to natively operate on complex data types (such as JSON)

### FAMILIARITY
EXISTING SKILLS &
TECHNOLOGIES

- Leverage ANSI SQL skills and BI tools
- Plug-n-play with Hive schema, file formats, UDFs

hadoop

APACHE DRILL

APACHE HBASE

# The End,
# Thank You

| | | | |
|---|---|---|---|
| @mapr | t | f | maprtech |
| mapr-technologies | in | g+ | MapRTechnologies |
| rshaw@mapr.com | ✉ | You Tube | maprtech |