

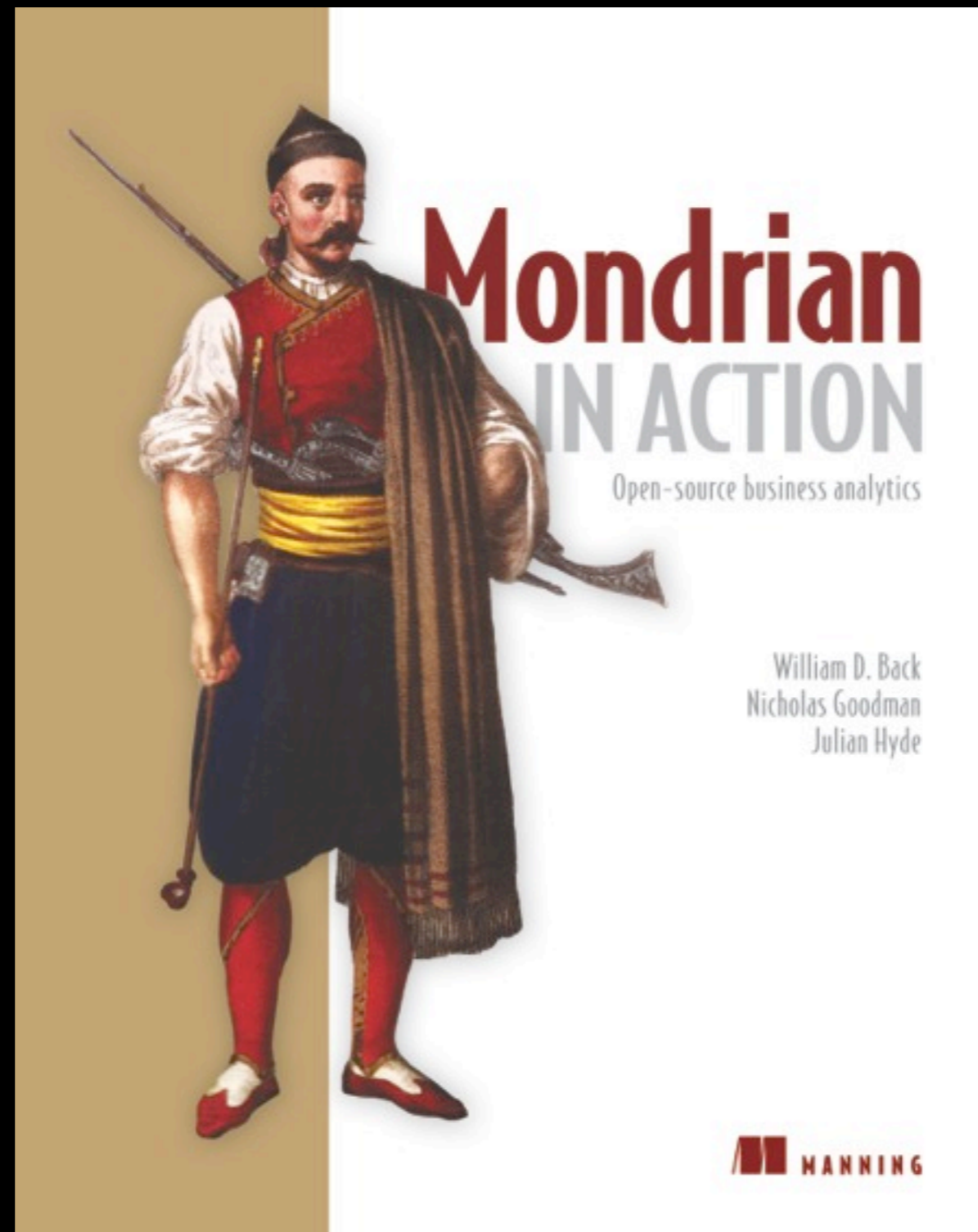
# SQL Now!

NoSQL Now!  
San Jose, California  
August, 2013

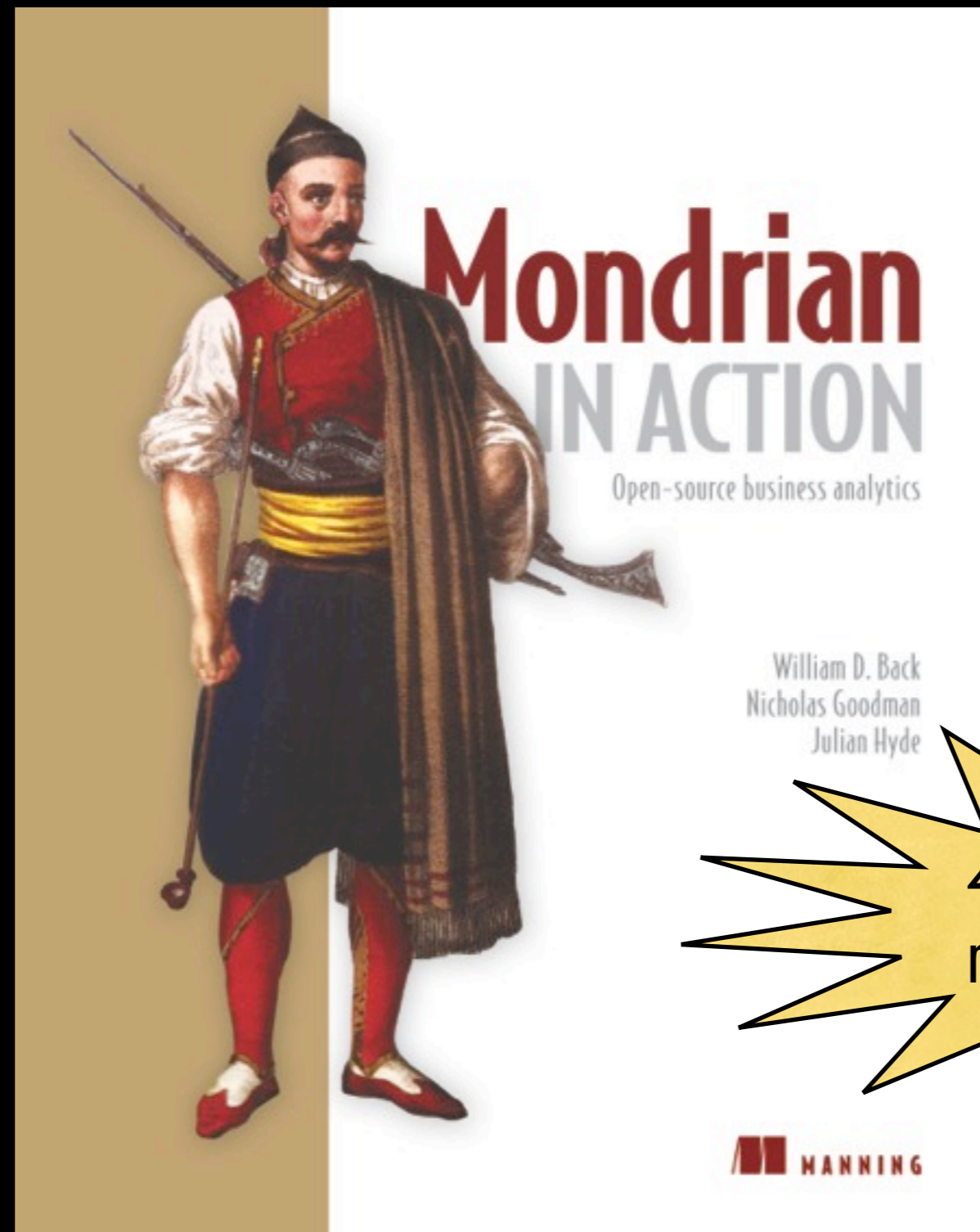
Julian Hyde @julianhyde

# About me

- Database: Oracle, Broadbase
- Streaming query: SQLstream
- Open source BI: Mondrian / Pentaho
- Open source SQL: LucidDB, Optiq
- Contributor to Apache Drill, Cascading  
Lingual



<http://manning.com/back/>



45% off code  
mlnosql13

<http://manning.com/back/>

# 3 things

1. Modern data challenges need both NoSQL and SQL.
2. Optiq is not a database (and this is a Good Thing).
3. How to use Optiq with your data.



**A disturbance in the Force**



SQL has been around a  
very, very long time

# NoSQL: trade-offs vs SQL

Gain	Lose
Scale out	Optimization
Flexibility	Data independence
Productivity	Central control
Purchase cost	Tool integration

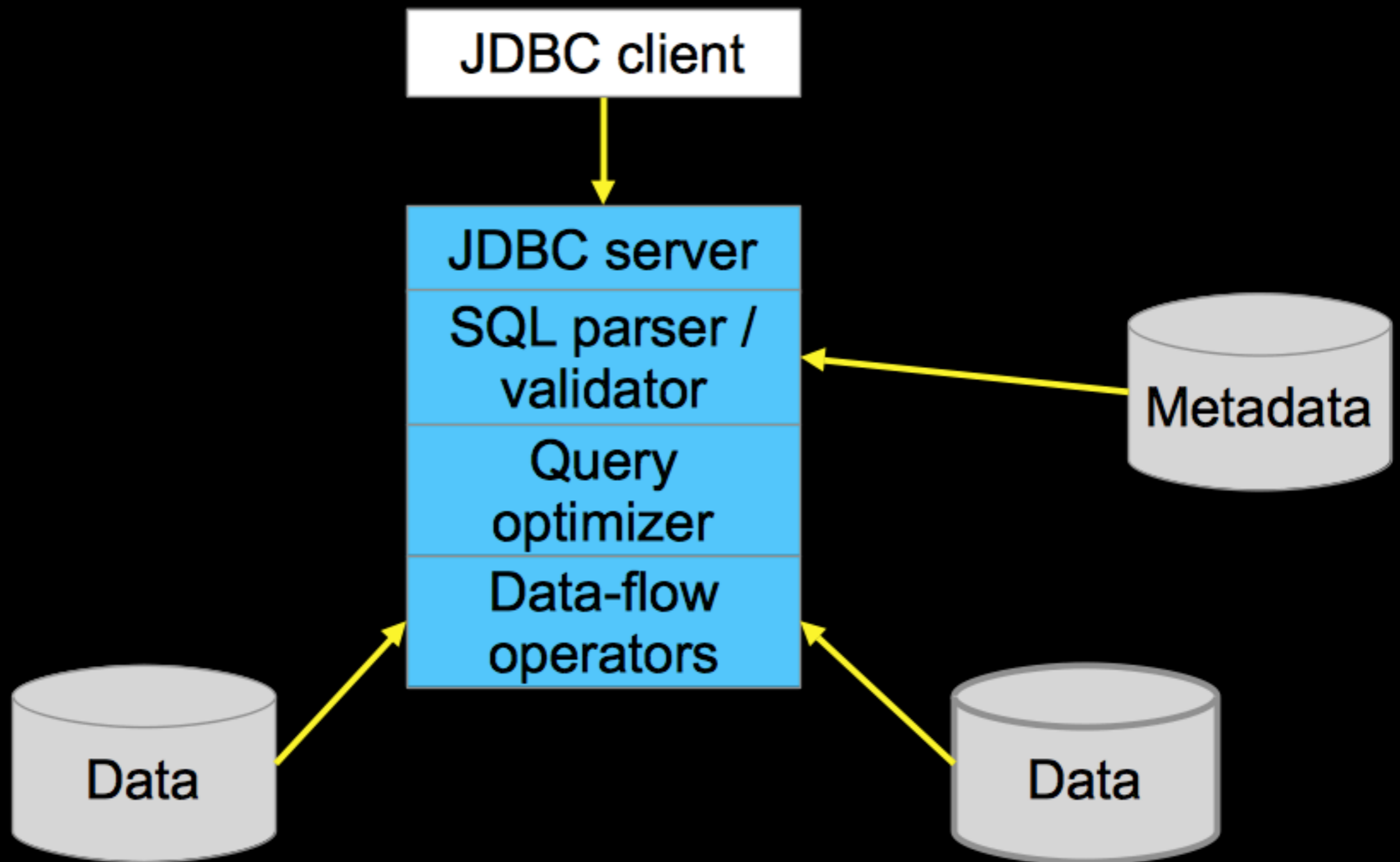
# SQL: key features

- Inspired by Codd's 1970 "relational database" paper
- Semantics based on relational operators (scan, filter, project, join, agg, union, sort)
- All implementations transform queries
- Optimizer rewrites queries onto available data structures and operators

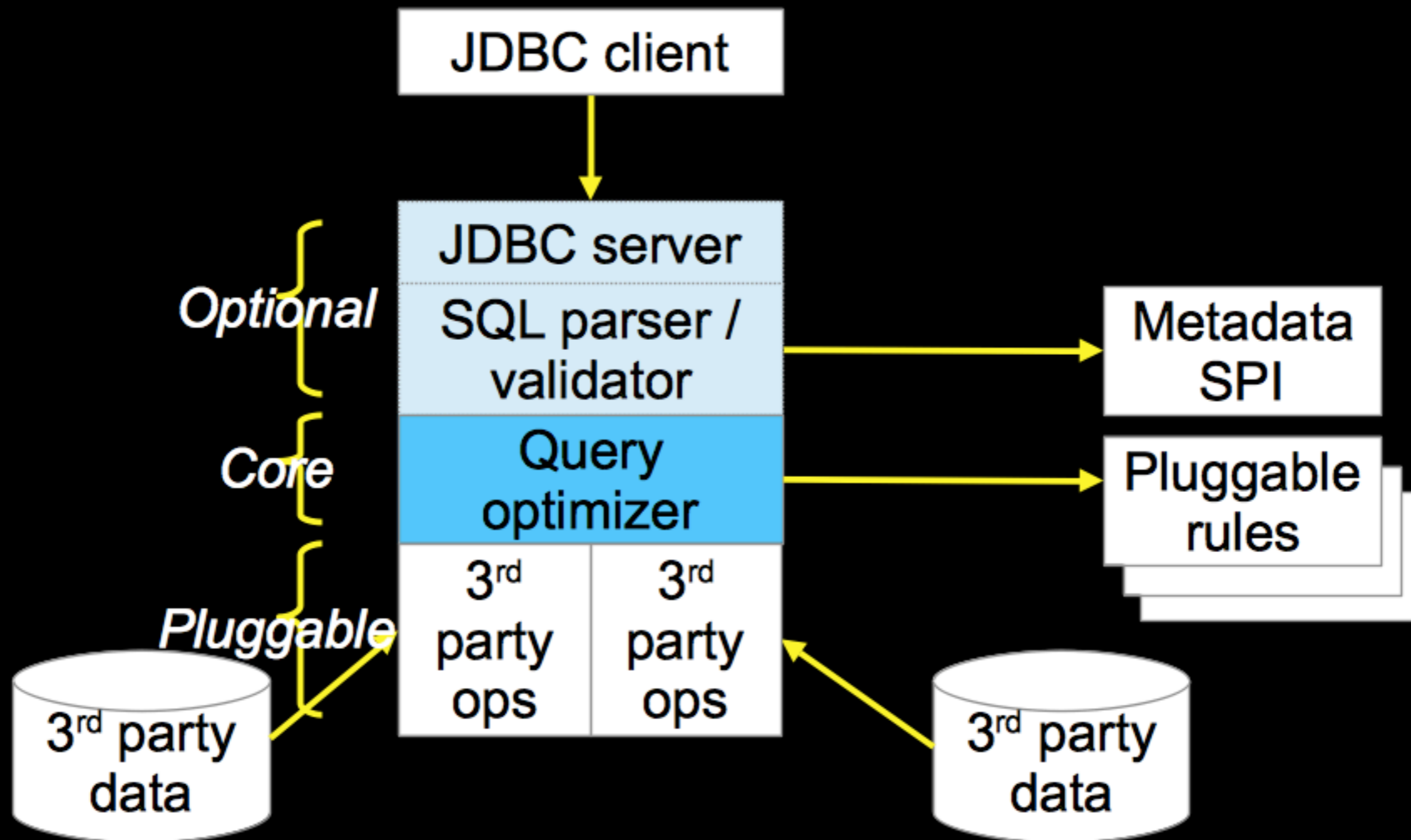
Optiq

# Optiq design principles

1. Do the stuff that SQL does well
2. Let other systems do what they do well



# Conventional DB architecture



# Optiq architecture

# Examples

# Example #1: CSV

- Uses CSV adapter (optiq-csv)
- Demo using sqlline
- Easy to run this for yourself:

```
$ git clone https://github.com/julianhyde/optiq-csv  
$ cd optiq-csv  
$ mvn install  
$ ./sqlline
```

```
!connect jdbc:optiq:model=target/test-classes/model.json admin admin
```

```
!tables
```

```
!describe emps
```

```
SELECT * FROM emps;
```

```
EXPLAIN PLAN FOR SELECT * FROM emps;
```

```
SELECT depts.name, count(*)  
FROM emps JOIN depts USING (deptno)  
GROUP BY depts.name;
```

```
model.json:
```

```
{  
  version: '1.0',  
  defaultSchema: 'SALES',  
  schemas: [  
    {  
      name: 'SALES',  
      type: 'custom',  
      factory: 'net.hydromatic.optiq.impl.csv.CsvSchemaFactory',  
      operand: {  
        directory: 'target/test-classes/sales'  
      }  
    }  
  ]  
}
```

# More adapters

Adapters	Embedded	Planned
CSV	Cascading (Lingual)	HBase (Phoenix)
JDBC	Apache Drill	Spark
MongoDB		Cassandra
Splunk		Mondrian
linq4j		

# SQL on NoSQL

- No schema or schema-on-read
  - Optiq MAP columns, dynamic schema
- Nested data (Drill, MongoDB)
  - Optiq ARRAY columns
- Source system optimized for transactions & focused reads, not analytic queries
  - Optiq cache and materializations

# Example #2: MongoDB

- Mongo's standard "zips" data set with all US zip codes
- Raw ZIPS table with `_MAP` column
- ZIPS view extracts named fields, including nested latitude and longitude, and converts them to SQL types

# Splunk

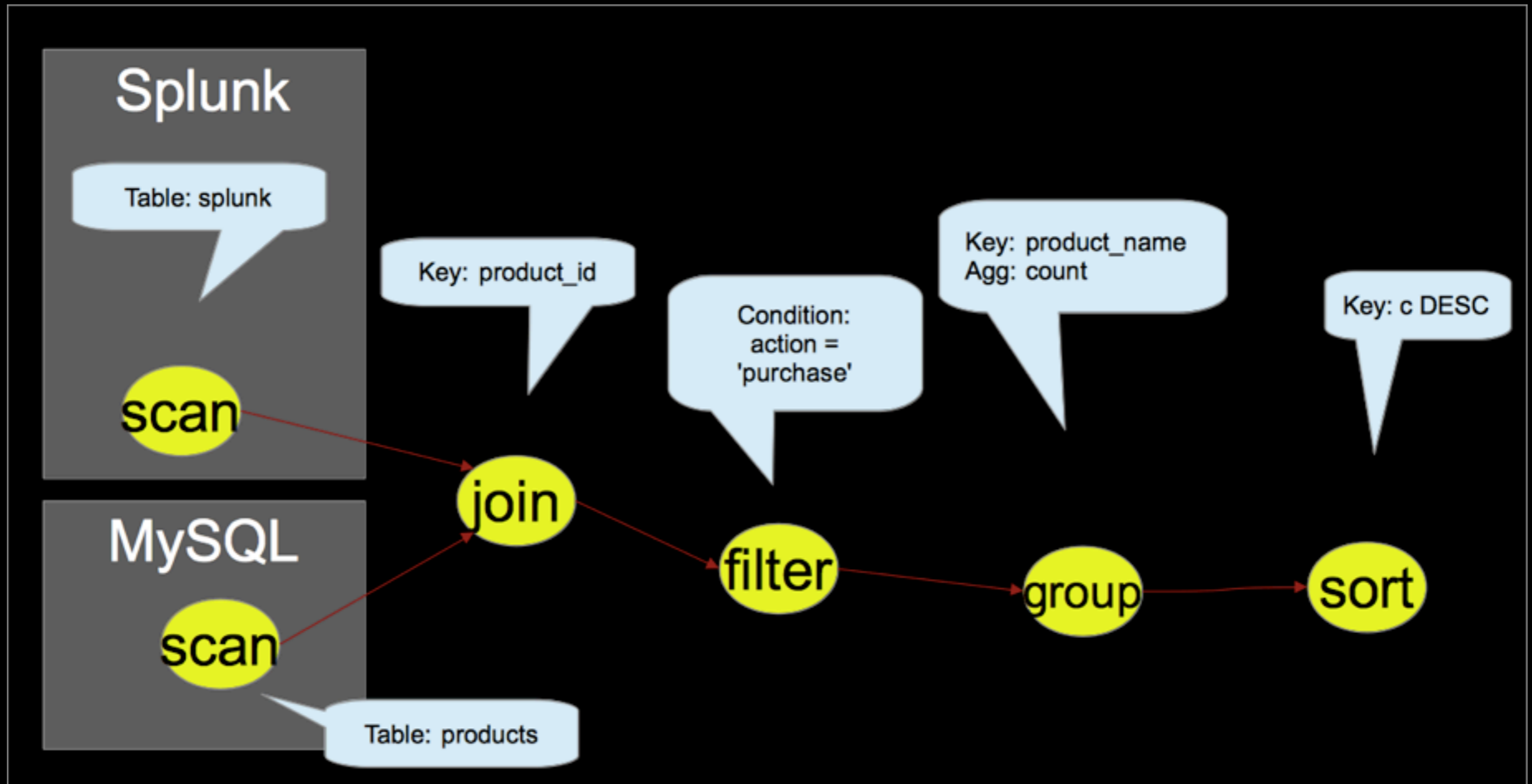
- NoSQL database
- Every log file in the enterprise
- A single “table”
- A record for every line in every log file
- A column for every field that exists in any log file
- No schema

# Example #3:

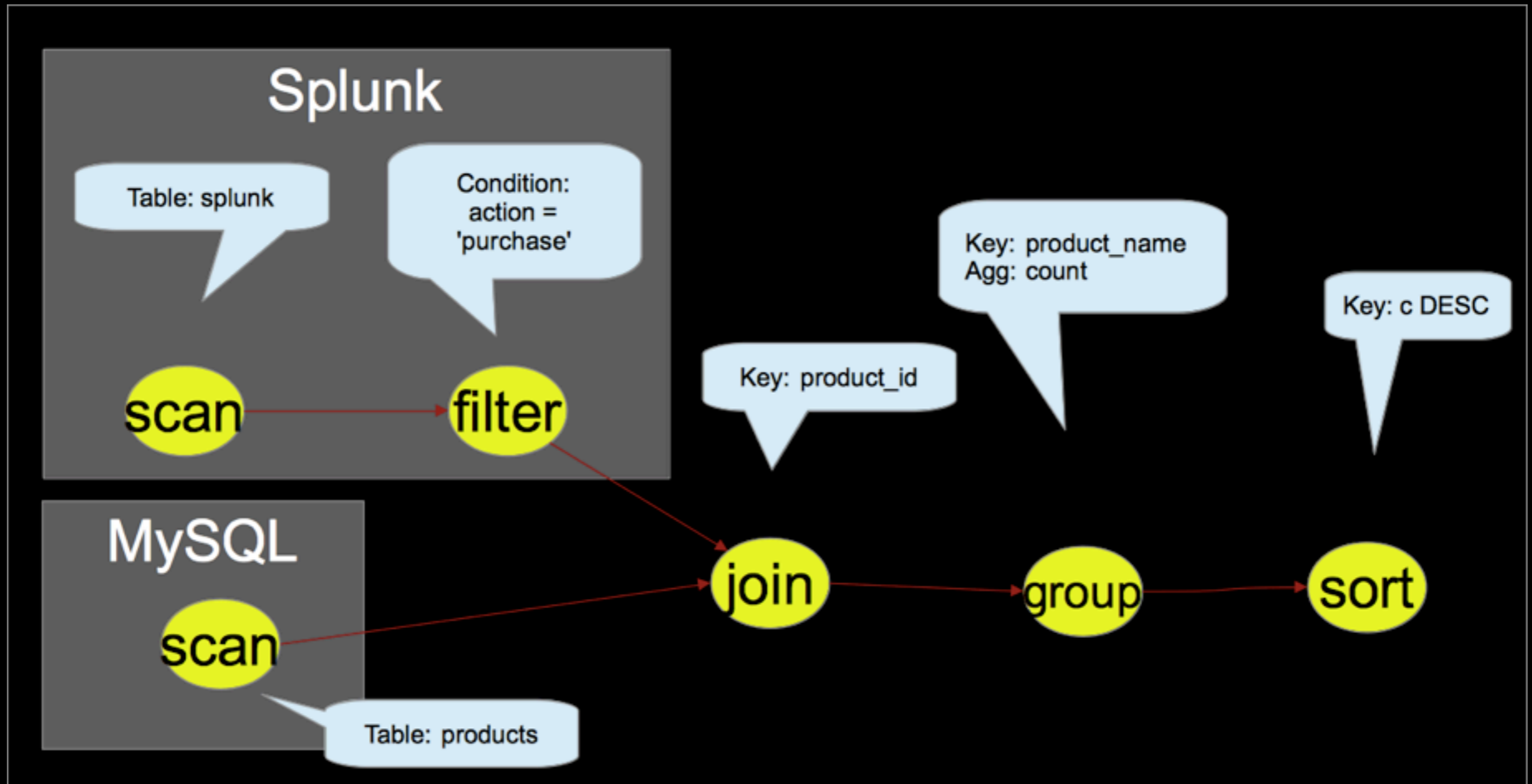
## Splunk + MySQL

```
SELECT p.product_name,  
       COUNT(*) AS c  
FROM splunk.splunk AS s  
     JOIN mysql.products AS p  
     ON s.product_id = p.product_id  
WHERE s.action = 'purchase'  
GROUP BY p.product_name  
ORDER BY c DESC
```

# Expression tree



# Optimized tree



# Analytics

# Interactive analytics on NoSQL?

- NoSQL operational DB (e.g. HBase, MongoDB, Cassandra)
- Analytic queries aggregate over full scan
- Speed-of-thought response (< 5 seconds)
- Data freshness (< 10 minutes)

# Simple analytics problem

- 100M U.S. census records
- 1KB each record, 100GB total
- 4 SATA3 disks, total 1.2GB/s
- How to count all records in under 5s?

# Simple analytics problem

- 100M U.S. census records
- 1KB each record, 100GB total
- 4 SATA3 disks, total 1.2GB/s
- How to count all records in under 5s?

# Simple analytics problem

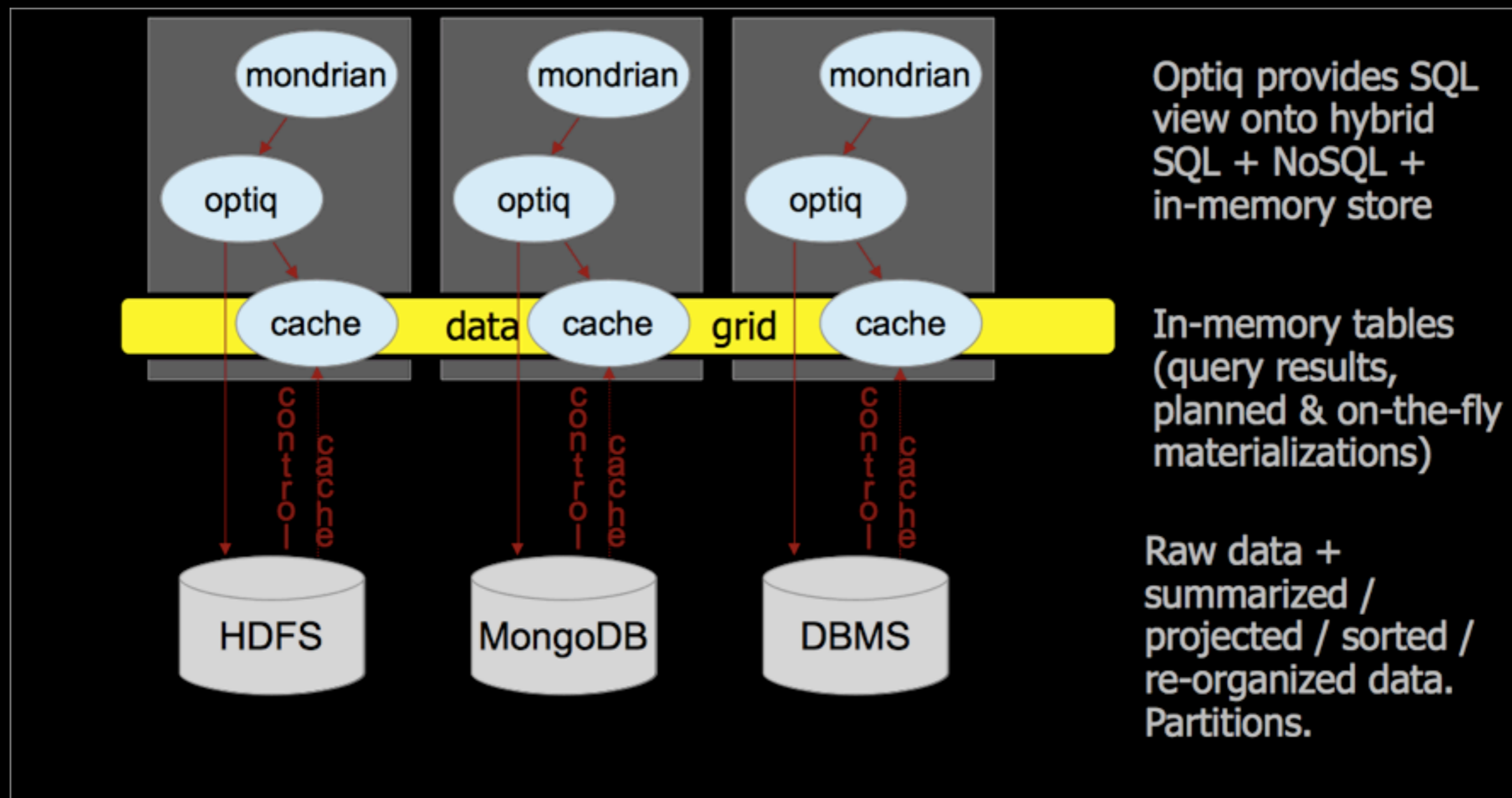
- 100M U.S. census records
- 1KB each record, 100GB total
- 4 SATA3 disks, total 1.2GB/s
- How to count all records in under 5s?
- Not possible?! It takes 80s just to read the data.

# Solution: Cheat!

# Solution: Cheat!

- Compress data
- Column-oriented storage
- Store data in sorted order
- Put data in memory
- Cache previous query results
- Pre-compute (materialize) aggregates

		Years								
		2003			2004			2005		
Territory	Line	Sales	Quantity	Unit Sales	Sales	Quantity	Unit Sales	Sales	Quantity	Unit Sales
APAC	Classic Cars	\$115,011	1,052	\$109	\$199,372	1,785	\$112	\$97,574	1,015	\$96
	Vintage Cars	\$111,639	1,243	\$90	\$147,212	1,587	\$93	\$105,688	1,067	\$99
	Motorcycles	\$60,789	654	\$93	\$63,159	540	\$117	\$65,870	658	\$100
	Trucks and Buses	\$11,298	91	\$124	\$80,634	801	\$101	\$53,735	488	\$110
	Planes	\$42,663	456	\$94	\$67,681	723	\$94	\$11,082	151	\$73
APAC Total		\$341,400	3,496	\$98	\$558,057	5,436	\$103	\$333,948	3,379	\$99
EMEA	Classic Cars	\$691,273	5,853	\$118	\$1,015,790	8,976	\$113	\$384,538	3,463	\$111
	Vintage Cars	\$263,695	3,094	\$85	\$504,062	5,472	\$92	\$83,324	1,094	\$76
	Motorcycles	\$141,836	1,428	\$99	\$204,042	2,177	\$94	\$161,260	1,501	\$107
	Trucks and Buses	\$228,699	2,261	\$101	\$185,421	1,558	\$119	\$86,859	836	\$104
	Planes	\$154,519	1,723	\$90	\$209,128	2,326	\$90	\$128,008	1,464	\$87
EMEA Total		\$1,480,021	14,359	\$103	\$2,118,443	20,509	\$103	\$843,989	8,358	\$101
Japan	Classic Cars	\$120,696	898	\$134	\$42,071	307	\$137	\$18,835	122	\$154
	Planes	\$60,556	677	\$89	\$49,177	547	\$90	-	-	-
	Trucks and Buses	\$44,498	415	\$107	\$13,349	102	\$131	-	-	-
	Motorcycles	\$16,485	205	\$80	\$31,959	380	\$84	\$4,176	44	\$95
	Vintage Cars	\$22,888	308	\$74	\$21,470	229	\$94	\$7,979	84	\$95
Japan Total		\$265,123	2,503	\$106	\$158,026	1,565	\$101	\$30,990	250	\$124
NA	Classic Cars	\$587,428	4,959	\$118	\$581,043	5,017	\$116	\$237,791	2,105	\$113
	Vintage Cars	\$281,727	3,268	\$86	\$324,815	3,576	\$91	\$191,727	1,871	\$102
	Motorcycles	\$178,109	1,744	\$102	\$291,421	2,809	\$104	\$55,020	568	\$97
	Trucks and Buses	\$135,936	1,289	\$105	\$252,572	2,563	\$99	\$61,281	597	\$103
	Planes	\$90,016	977	\$92	\$202,942	2,224	\$91	\$60,985	592	\$103
NA Total		\$1,273,216	12,237	\$104	\$1,652,792	16,189	\$102	\$606,803	5,733	\$106
Grand Total		\$3,359,761	32,595	\$103	\$4,487,319	43,699	\$103	\$1,815,730	17,720	\$102



# Mondrian on Optiq on NoSQL (under construction)

# Hybrid analytic architecture

1. NoSQL source system
2. Access via Optiq SQL
3. Optiq rewrites queries to use materialized data (e.g. aggregate tables)
4. Cached results are treated as “in-memory tables”
5. Materializations offline dynamically as underlying data changes, and go online as they are refreshed

# Summary

# 3 things (reprise)

1. SQL allows you to reorganize your data and optimize your queries—while still using your NoSQL database for what it does best.
2. Optiq is not a database. It lets you create very powerful federated data architectures.
3. Access your data using Optiq adapters.  
Write schemas in JSON or use schema SPI.  
Connect via JDBC.

Questions?

# Thanks!

@julianhyde

optiq <https://github.com/julianhyde/optiq>

optiq-csv <https://github.com/julianhyde/optiq-csv>

drill <http://incubator.apache.org/drill/>

lingual <http://www.cascading.org/lingual/>

mondrian <http://mondrian.pentaho.com>

blog <http://julianhyde.blogspot.com/>

book <http://manning.com/back/> 45% off code **m1nosql13**