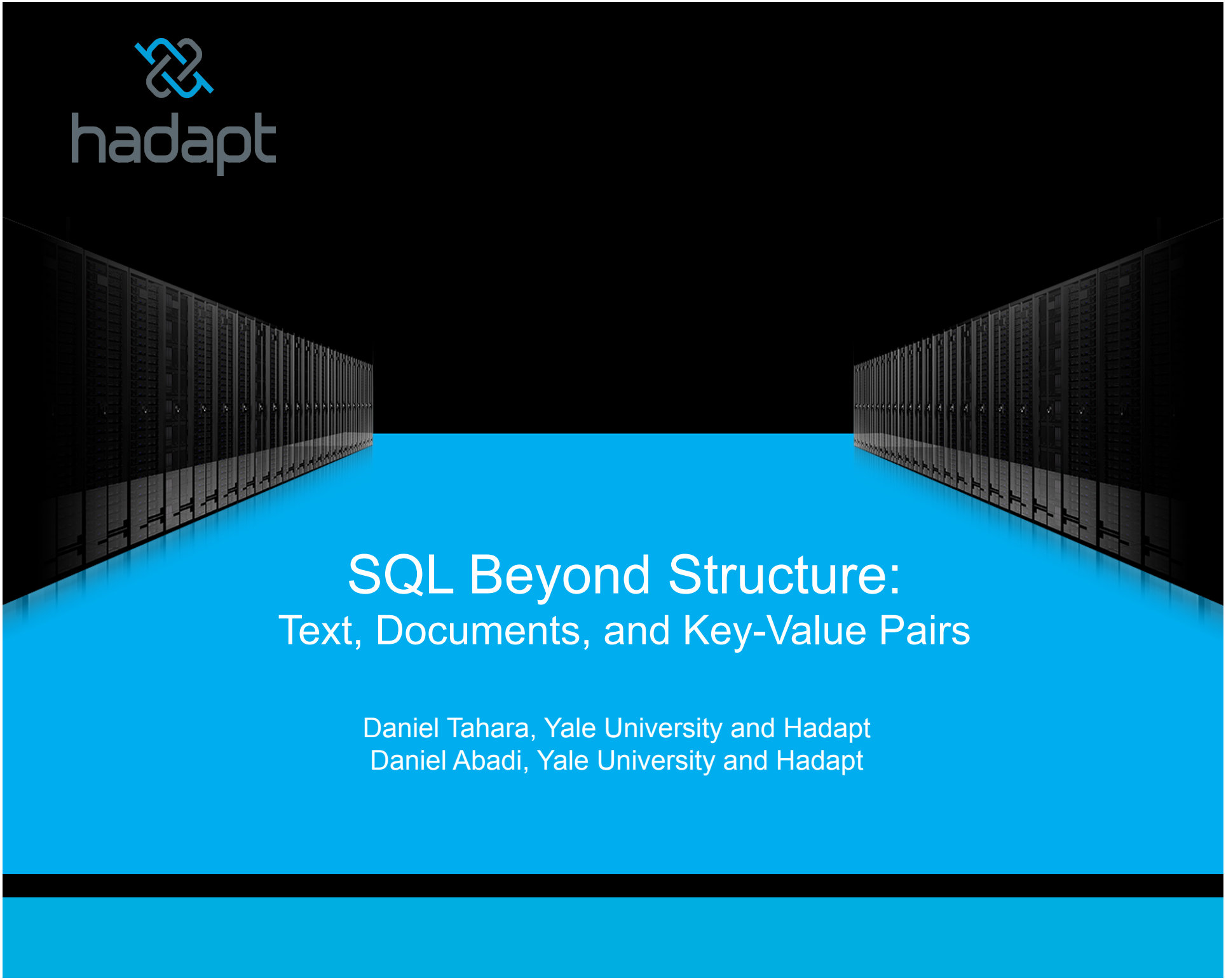


SQL Beyond Structure:
Text, Documents, and Key-Value Pairs

Daniel Tahara, Yale University and Hadapt
Daniel Abadi, Yale University and Hadapt

Important Trends

- Dramatic increase in utilization of “NoSQL” databases such as MongoDB, CouchDB, Riak, Cassandra, Hbase, etc.
- Key value and JSON are increasingly popular ways to represent data
  - See MongoDB’s $150 million “largest database funding ever”
- Performing analysis inside NoSQL databases not a major use-case
  - Poor analytics primitives
  - Poor integration with other data sources (data silos)
  - Poor scan performance
  - Poor integration with analytical tools

A Motivating Example

Relational Datastore

userID	name	email	preferred
1	Walter White	heisenberg@bb.com	false
2	Hank Schrader	mineral@rock.com	true
3	Jesse Pinkman	pink@thenewblack.ca	true
4	Gustavo Fring	gus@lospollos.com	true

Text Datastore

userID	Email
1	What is your return policy?
2	Do you have bulk discounts?
4	I can't figure out your Website. You should fire your designer!

What if you want to send an email to all customers who indicate a potential desire to buy in bulk, with a deal on items they looked at?

JSON Datastore

JSON Documents
{"userID": 1, "action":"view", "productName": "door", "price": 12.50,"color":"green","tags": ["home","green"]}
{"userID": 3, "action":"purchase","price": 57.12}
{"userID": 2, "action":"view", "productName": "window", "color":"orange, "price": 52.50,"tags": ["home","orange"]}

Option 1: User Code

Relational Datastore

	name	email	preferred
1	Walter White	heisenberg@bb.com	false
2	Hank Schrader	mineral@rock.com	true
3	Jesse Pinkman	pink@thenewblack.ca	true
4	Gustavo Fring	gus@lospollos.com	true

Text Datastore

userID	Email
1	What is your return policy?
2	Do you have bulk discounts?
4	I can't figure out your Website. You should fire your designer!

Extract data from each relevant store, perform join and further query analysis in user code.

```
#!/bin/sh

if ['some
  condition']
then
  'do
  something'
fi

'more
complicated
code'
```

JSON Datastore

JSON Documents
{"userID": 1, "action":"view", "productName": "door", "price": 12.50,"color":"green","tags": ["home","green"]}
{"userID": 3, "action":"purchase","price": 57.12}
{"userID": 2, "action":"view", "productName": "window", "color":"orange, "price": 52.50,"tags": ["home","orange"]}

Option 2: ETL

Relational Datastore

userID	name	email	preferred
1	Walter White	heisenberg@bb.com	false
2	Hank Schrader	mineral@rock.com	true
3	Jesse Pinkman	pink@thenewblack.ca	true
4	Gustavo Fring	gus@lospollos.com	true

Use ETL to combine data from separate backends into one unified datastore.

JSON Datastore

JSON Documents
{“userID”: 1, “action”:”view”, ”productName”: “door”, “price”: 12.50,”color”:”green”,”tags”: [“home”,”green”]}
{“userID”: 3, “action”:”purchase”,”price”: 57.12}
{“userID”: 2, “action”:”view”, ”productName”: “window”, “color”:”orange, “price”: 52.50,”tags”: [“home”,”orange”]}

Text Datastore

userID	Email
1	What is your return policy?
2	Do you have bulk discounts?
4	I can’t figure out your Website. You should fire your designer!

ETL

ETL

ETL

A “Unified” Datastore

userID	name	emailAddr	preferred	json	emails
1	Walter White	heisenberg@bb.com	false	{“eventID”: 1, “action”:”view”, ”productName”: “door”, “price”: 12.50,”color”:”green”,”tags”:[“home”,”green”]}	What is your return policy?
2	Hank Schrader	mineral@rock.com	true	{“eventID”: 2, “action”:”view”, ”productName”: “window”, “color”:”orange, “price”:52.50,”tags”: [“home”,”orange”]}	Do you have bulk discounts?
3	Jesse Pinkman	pink@thenewblack.ca	true	{“eventID”: 3, “action”:”purchase”,”price”: 57.12}	
4	Gustavo Fring	gus@lospollos.com	true		I can’t figure out your Website. You should fire your designer!

```
SELECT name,
    emailAddr,
    regex_value(json, ‘“productName”:\s*“([^”]*?)”’, 1, ‘i’)
FROM T
WHERE regex_match(json, ‘“action”:\s*”view”’,’i’)
    AND emails like ‘%bulk%’;
```

A Better Approach?

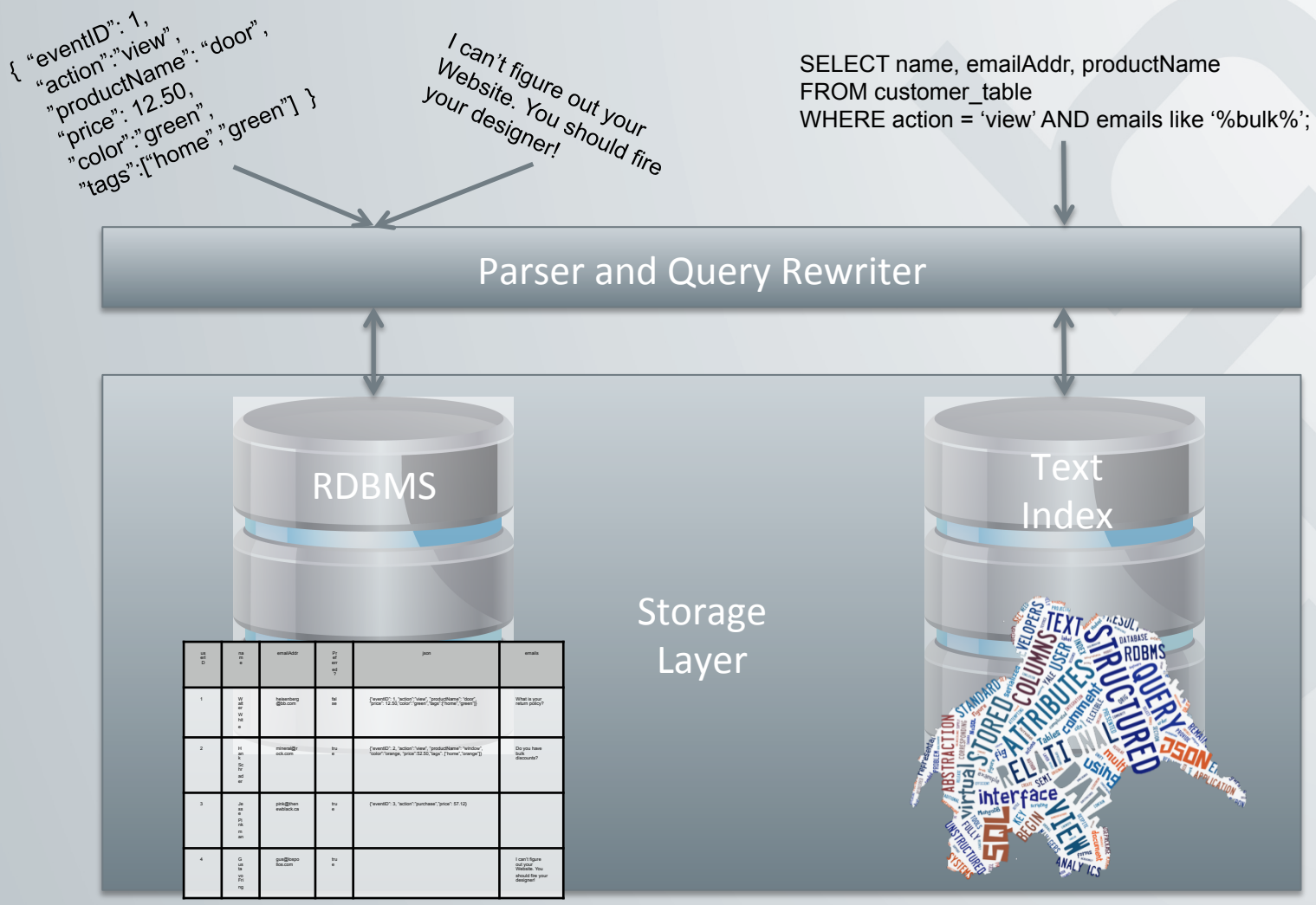
userID	name	emailAddr	preferred	product Name	price	tags	color	action	emails
1	Walter White	heisenberg@bb.com	no	door	12.50	home green	green	view	What is your return policy?
2	Hank Schrader	mineral@rock.com	yes	window	52.50	home orange	orange	view	Do you have bulk discounts?
3	Jesse Pinkman	pink@thenewblack.ca	yes		57.12			purchase	
4	Gustavo Fring	gus@lospollos.com	yes						I can't figure out your Website. You should fire your designer!

```
SELECT name,
    emailAddr,
    productName
FROM T
WHERE action = 'view'
    AND emails like '%bulk%';
```

Goal: Multistructured Tables

- Store data with various levels of structure side-by-side in an RDBMS
- Remove the ETL step in the processing pipeline
- Maintain a standard SQL interface to the data - allow elements to be manipulated as relational attributes irrespective of original format
- Bonus: Support for non-relational operations such as full text search

Implementation: Overview

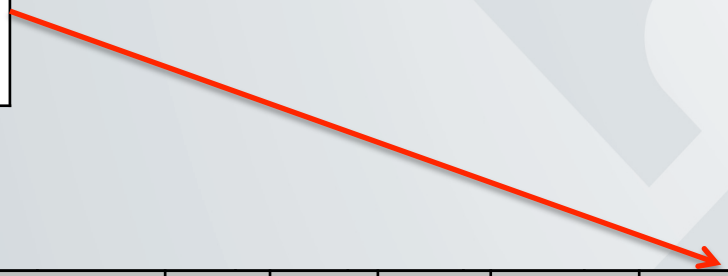


Implementation: Details

- System does not store data according to the virtual, “universal” view
- Store key-value data in a serialized, key-value format in order balance competing needs:
  - Limit table width
  - Speed up attribute extraction
  - Identify which attributes exist
- Inverted indexes over all keys and text columns
- Query rewriter transforms:
  - references to serialized, virtual columns into key extractions
  - predicates over virtual columns into queries to the text index, applied as a filter set on the attribute ids

Flexible Schema

json
{“userID”: 4, “action”:”view”, ”productName”: “door”, “price”: 12.50,”color”:”green”,”tags”: [“home”,”green”],”size”:”standard”}



userID	name	emailAddr	Preferred?	product Name	price	tags	color	action	size	emails
1	Walter White	heisenberg@bb.com	no	door	12.50	home green	green	view		What is your return policy?
2	Hank Schrader	mineral@rock.com	yes	window	52.50	home orange	orange	view		Do you have bulk discounts?
3	Jesse Pinkman	pink@thenewblack.ca	yes		57.12			purchase		
4	Gustavo Fring	gus@lospollos.com	yes						standard	I can’t figure out your Website. You should fire your designer!

Challenges with the Current Design

- Relying on a text index can be slow if the result set is large
- Can’t utilize B-tree indexes
- RDBMS optimizer often produces suboptimal query plans

Dynamic Column Creation/Deletion

- Specifications:
  - Create physical columns for most commonly used or frequently appearing keys.
  - Ongoing process. Columns can be created or deleted at any point.
- Benefits:
  - can create indices on physical columns
  - optimizer has better statistics on frequently queried data
  - dynamic physical schema allows the system to adapt to the evolving nature of datasets

“Smart” Query Rewrite and UDFs

- Specification:
  - Define an extraction function and a text index query function for use within the RDBMS
  - *Selectively* execute text predicates in the RDBMS or push them out to the text index
- Benefits:
  - More code running within the RDBMS
  - Take into account system knowledge of the data to produce globally optimal execution

Alternatives

- Entity-Attribute-Value
  - “flatten” key-value data types
  - stored in 3-column tables
- PostgreSQL 9.3
  - use JSON text data type
  - extraction operators
- Column Store
  - store in the form of universal relation

Conclusions

- Bevy of options for formatting data
- As long as SQL remains a standard of data analytics, there will be a need for unified solution
- Tighter RDBMS integration correlates with improved performance

Questions?

- Email: daniel.tahara@yale.edu
- Further Resources:
  - Hadapt Schemaless SQL:
    - http://hadapt.com/schemaless-sql-overview/
  - Yale Database Group:
    - http://db.cs.yale.edu/
  - My Website:
    - http://danieltahara.com/