SQL in the Hybrid World

Tanel Poder

a long time computer performance geek

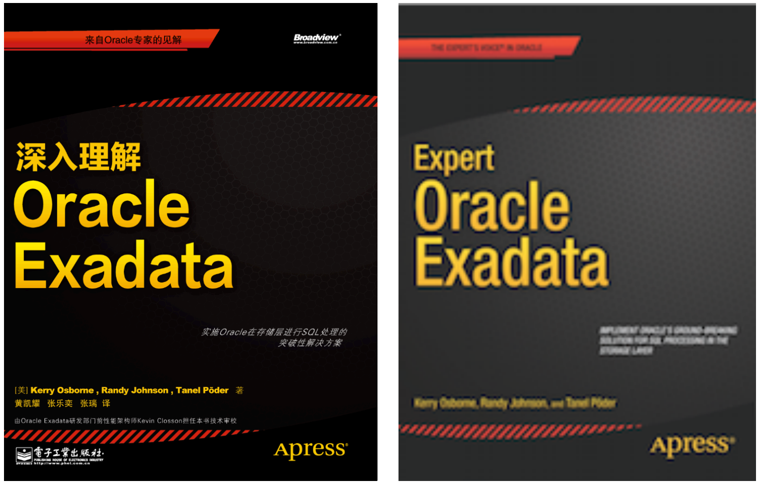
Intro: About me

- Tanel Põder
  - Oracle Database Performance geek (18+ years)
  - Exadata Performance geek
  - Linux Performance geek
  - Hadoop Performance geek

Instant promotion

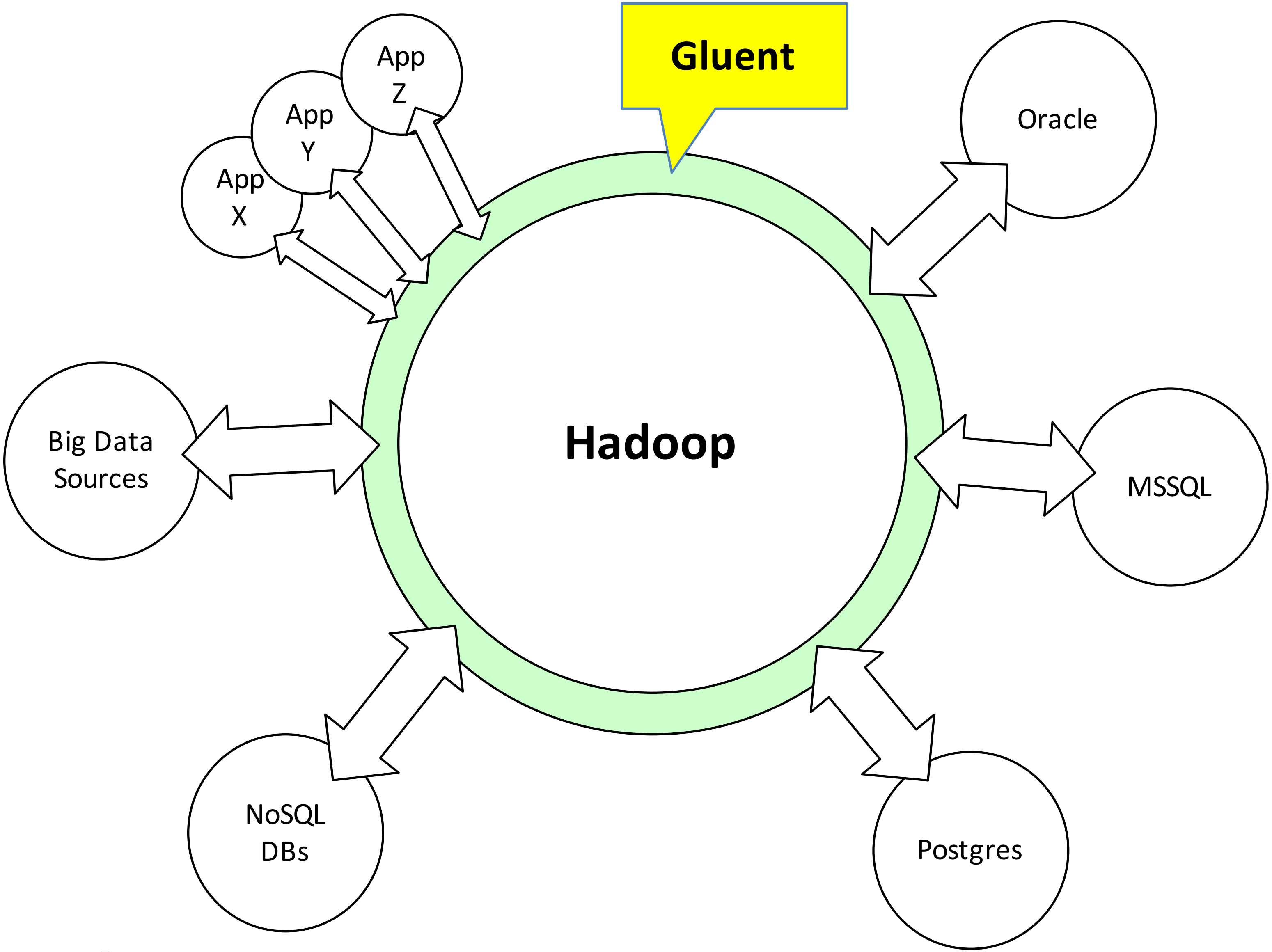
- CEO & co-founder:

gluent.



Expert Oracle Exadata book
(2nd edition is out now!)

The Hybrid World



Why this talk?

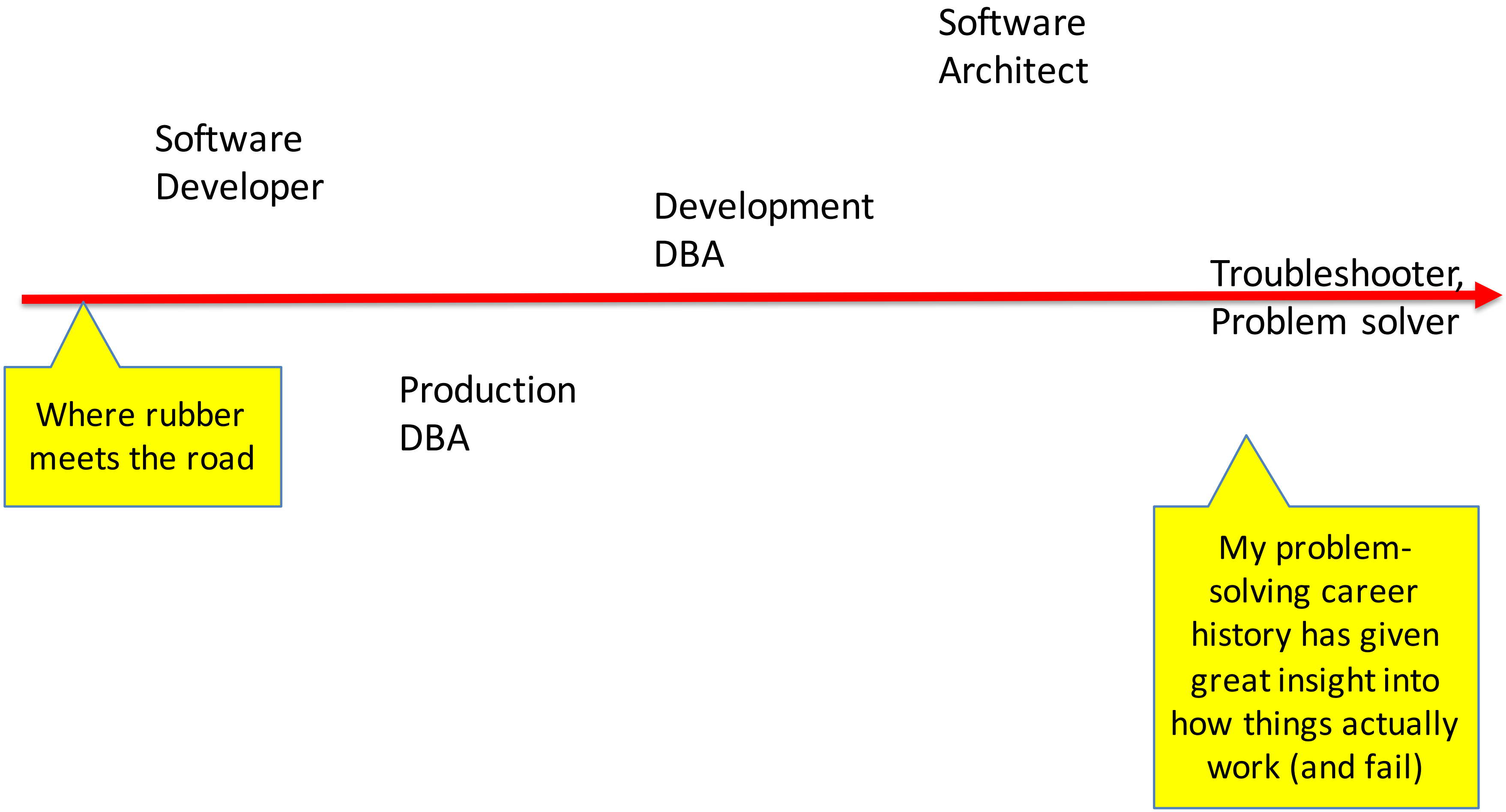
- Various NoSQL databases and Hadoop projects are very hot!
  - **Is this stuff for real?**
- I’m mainly talking about traditional enterprises
  - Not small startups
  - Not web/mobile giants

But hey, it’s 2016!

- This is not an academic analysis of data consistency theorems
  - A basic reasoning from a database professional’s point of view…

My data engineering career

(And why it is relevant to this talk)



Software Developer

Production DBA

Development DBA

Software Architect

Troubleshooter, Problem solver

Where rubber meets the road

My problem-solving career history has given great insight into how things actually work (and fail)

NoSQL

Typical properties associated with NoSQL

1. Schemaless / dynamic schema
   - Dump any fields into (and under) any records

Not all (NoSQL) databases are the same! No single definition of NoSQL!

2. Doesn't do joins
   - Everything related to an item should physically be stored with the item

3. Horizontally scalable distributed systems
   - Data in a "document" doesn't span cluster nodes
   - So, nodes don't coordinate with each other much

4. Not ACID compliant
   - Consistent or "eventually consistent"
   - No multi-row / multi-document transactions

As otherwise the cluster nodes would have to coordinate with each other (and two-phase commit)

Real Life: Early “NoSQL” drivers (2004)

1. Adding a column in a development **RDBMS** takes 2 days
   - Developer has to file a form or two for Dev DBA

2. Getting this column into production RDBMS takes 2 weeks
   - Structural table changes may break things, can’t release often

3. Not a technical issue, but a process limitation
   - App Dev not moving fast enough for business

Real Life: Early "NoSQL" drivers

4. Developers didn't like the organizational slowness, so:

   - **Reaction:** Using catch-all XML columns and generic entity-attribute-value designs for "dynamic" data models
   - **Result:** RDBMS Performance problems with real datasets
   - **Conclusion:** "RDBMS are too slow for modern applications"
   - **Actual reality:** You should use relational databases in a relational way

“Joins are slow!”

Compared to what?

Real Life: Joins are slow! (MySQL edition)

- Circa year 2008
  - **Claim:** All joins are slow!
  - **Look deeper:** Joining two small tables in MySQL is slow
  - **Look deeper:** MySQL didn't have hash joins back then
  - **Look deeper:** Joining unindexed tables with nested loops
  - **Actual Reality:** Optimize physical design, use a more powerful RDBMS

By now, MySQL has buffered (block) nested loop joins and MariaDB buffered hash joins

Joins are slow! (modern scale-out edition)

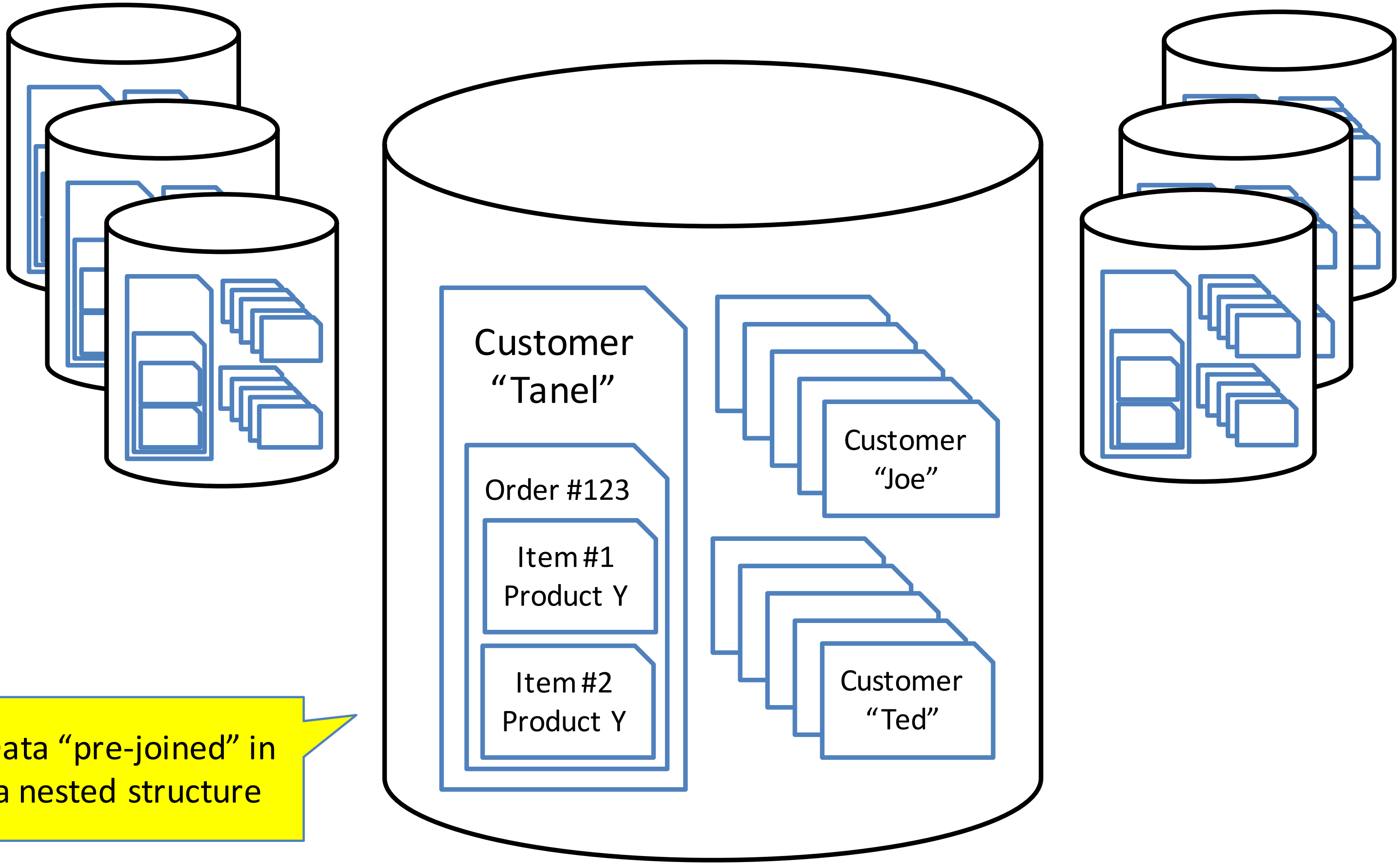
- Store all related records of a document in a single server node in a nested structure
  - *Orders* under a *Customer* document
  - *Order_items* stored under an *Order* document
  - Data “pre-joined” on write: De-normalization (embedding)
  - Physical co-location to avoid extra IO + talking to other cluster nodes
  - Document Schema design important for performance!

Plus mirroring to other server(s) of course

```
Customer (Name: Tanel, Address: Somestreet 12-34, SomeCity, ZZ)
  Order #123 (Order Date: 2015-12-01)
    Order Item #1 (Quantity: 1)
          Product (Name: Vacuum Cleaner)
    Order Item #2 (Quantity: 3)
          Product (Name: Air Filter)
```

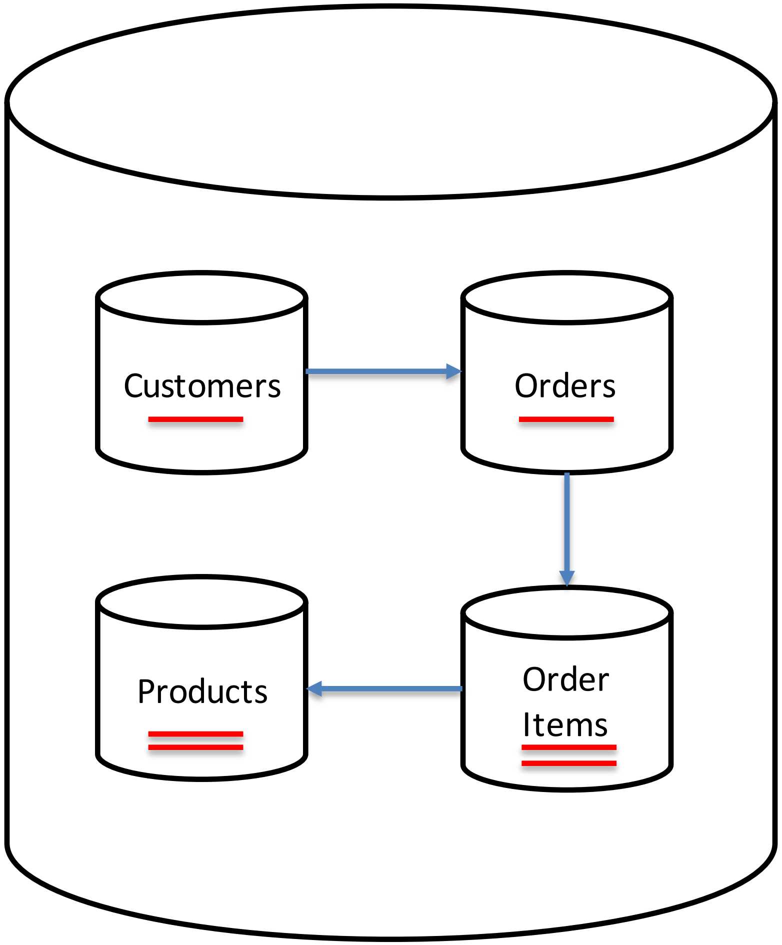
All the ”nested” data related to a customer contained in a single physical location & server

Document data retrieval (simplified)



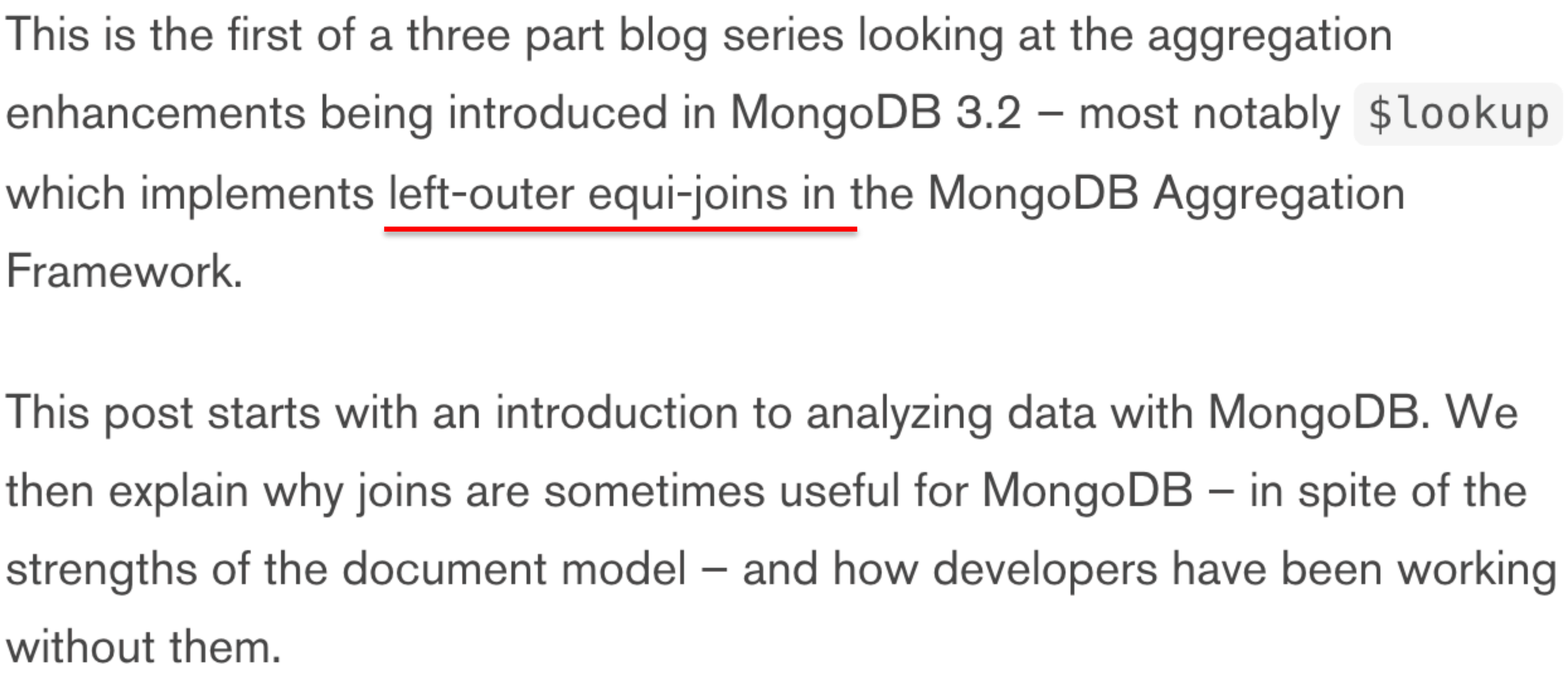
Data "pre-joined" in a nested structure

Relational data retrieval (simplified)



Retrieval must visit multiple separate (indexes) and tables

Joins in MongoDB 3.2



This is the first of a three part blog series looking at the aggregation enhancements being introduced in MongoDB 3.2 – most notably `$lookup` which implements left-outer equi-joins in the MongoDB Aggregation Framework.

This post starts with an introduction to analyzing data with MongoDB. We then explain why joins are sometimes useful for MongoDB – in spite of the strengths of the document model – and how developers have been working without them.

- Source: https://www.mongodb.com/blog/post/joins-and-other-aggregation-enhancements-coming-in-mongodb-3-2-part-1-of-3-introduction

Queries in MongoDB 3.2

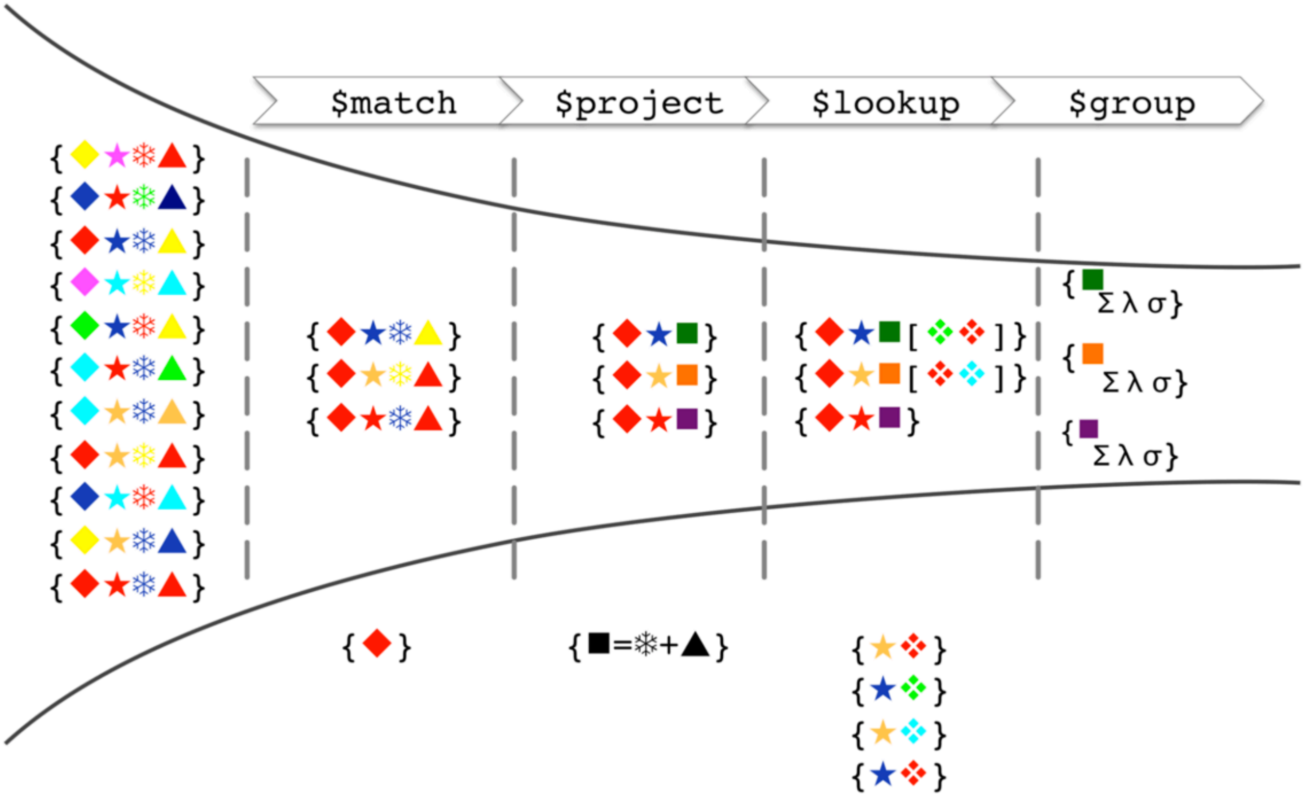
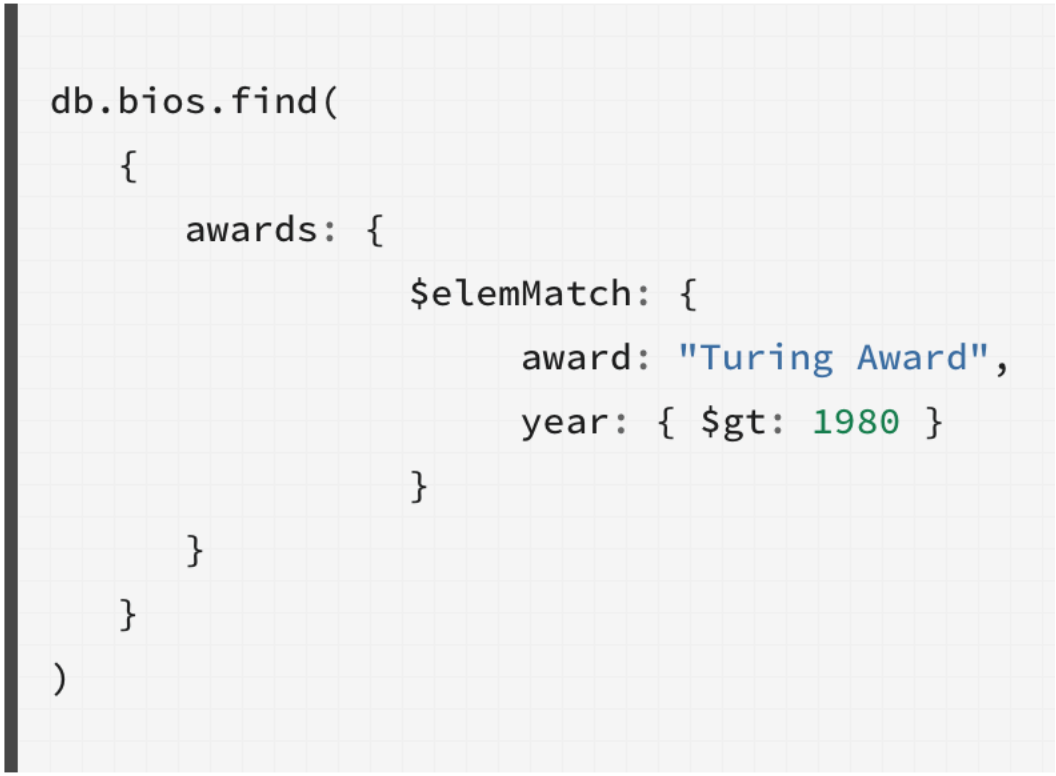


Figure 3: MongoDB Aggregation Framework pipeline

- Source: https://www.mongodb.com/blog/post/joins-and-other-aggregation-enhancements-coming-in-mongodb-3-2-part-1-of-3-introduction

Filtering in MongoDB 3.2

- Query & filter an array of elements:



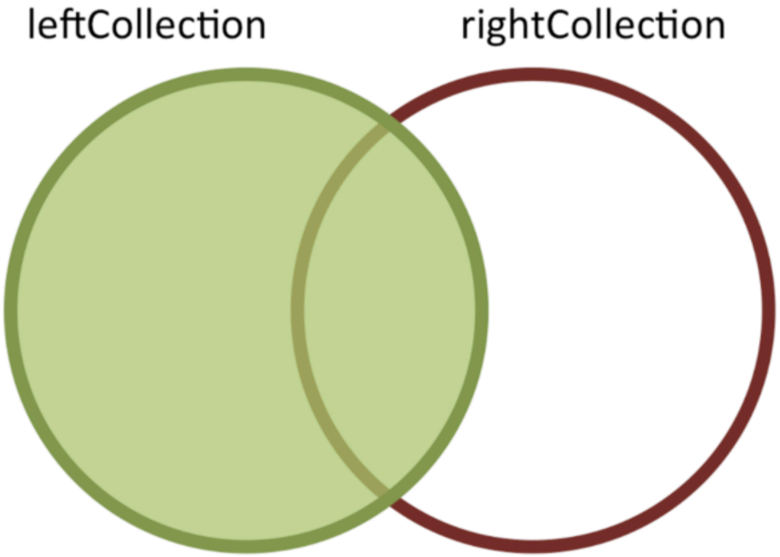
```
db.bios.find(
   {
      awards: {
                $elemMatch: {
                     award: "Turing Award",
                     year: { $gt: 1980 }
                }
      }
   }
)
```

SELECT
FROM bios
WHERE
    award = ‘Turing Award’
AND year > 1980 ?

- MongoDB Documentation: https://docs.mongodb.org/v2.6/reference/method/db.collection.find/#examples

Joins in MongoDB 3.2

$lookUp

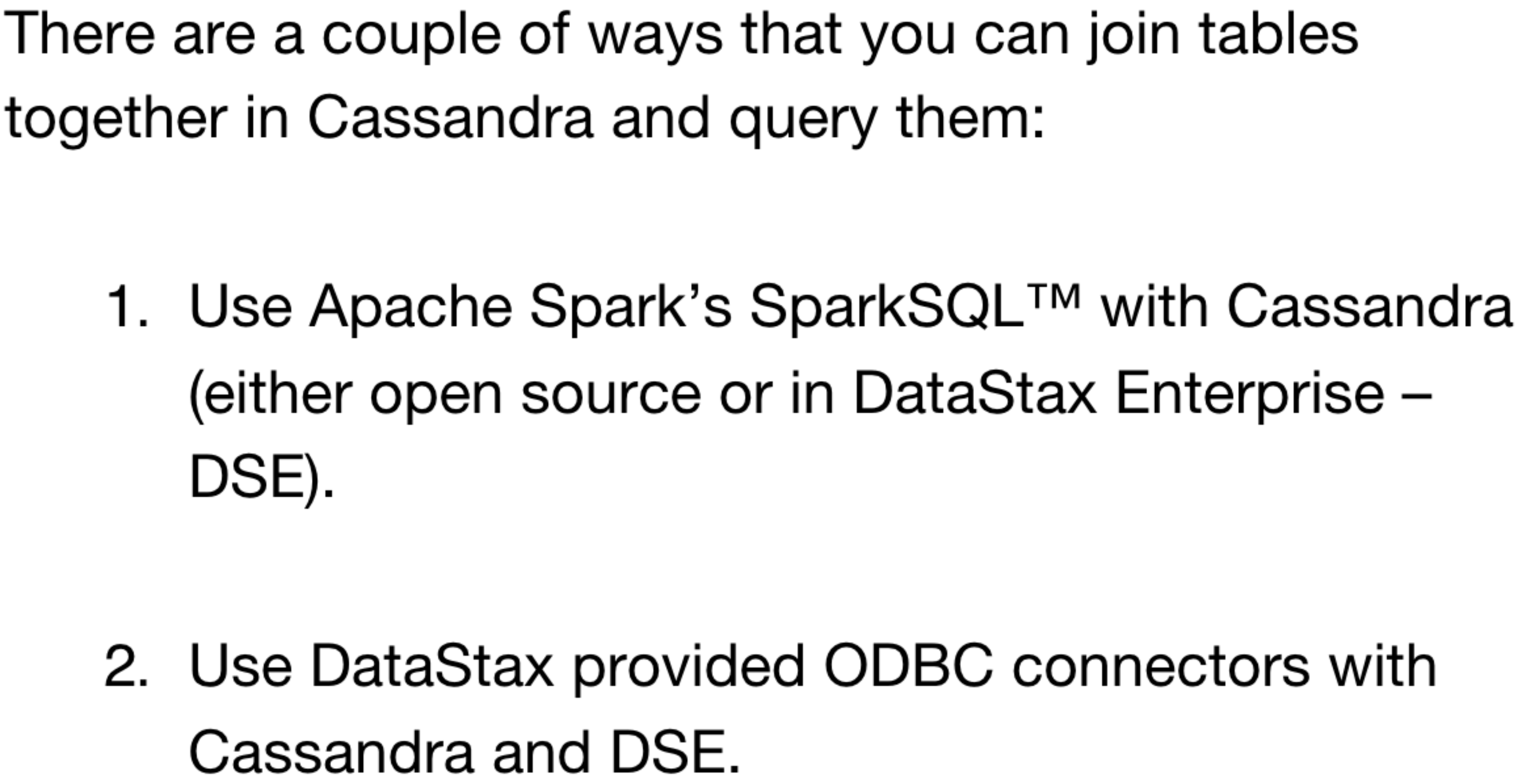


```
db.leftCollection.aggregate(
[{
  $lookUp:
    {
      from: “rightCollection”,
     localField: “leftVal”,
     foreignField: “rightVal”,
     as: “embeddedData”
    }
}])
```

Figure 4: $lookup – Left-Outer Joins for MongoDB

- Source: https://www.mongodb.com/blog/post/joins-and-other-aggregation-enhancements-coming-in-mongodb-3-2-part-1-of-3-introduction

Joins in Cassandra



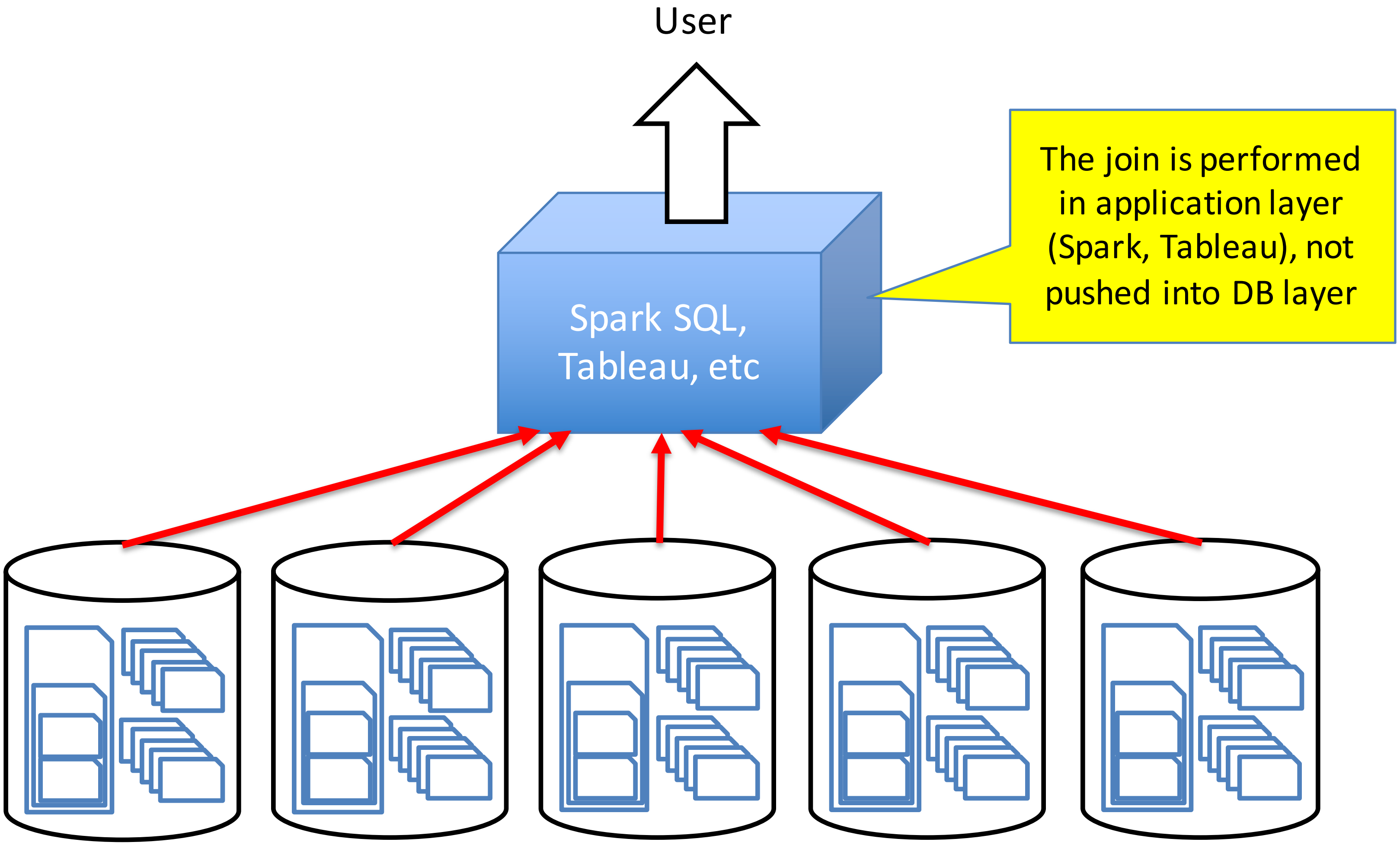
There are a couple of ways that you can join tables together in Cassandra and query them:

1. Use Apache Spark's SparkSQL™ with Cassandra (either open source or in DataStax Enterprise – DSE).

2. Use DataStax provided ODBC connectors with Cassandra and DSE.

- Source: http://www.datastax.com/2015/03/how-to-do-joins-in-apache-cassandra-and-datastax-enterprise

Joins in Cassandra



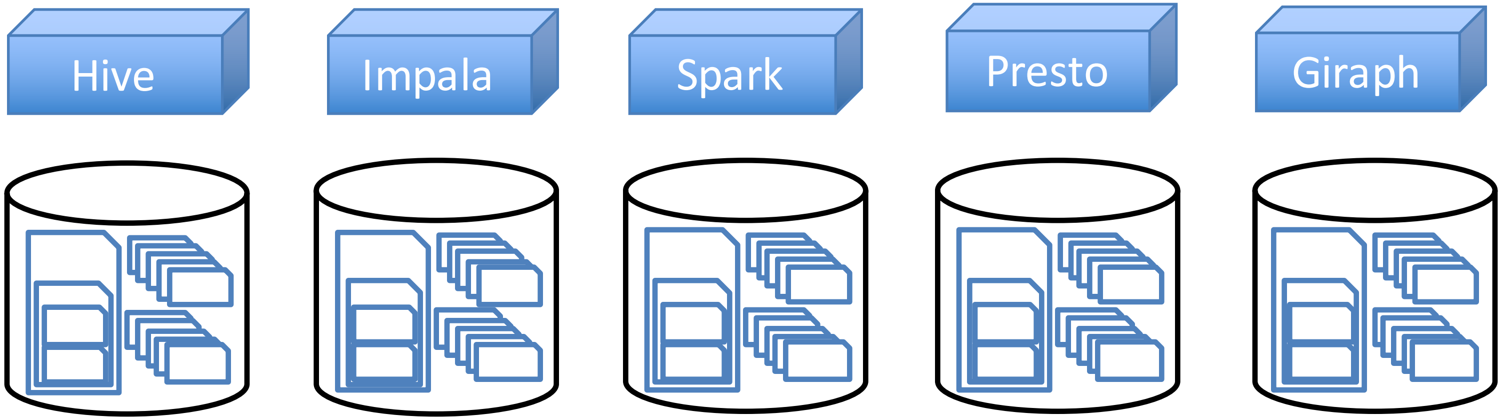
The join is performed in application layer (Spark, Tableau), not pushed into DB layer

SQL on

Hadoop!

Why Hadoop?

- Scalability in Software!
- Open Data Formats
  - Future-proof!
- One Data, Many Engines!



Scalability vs. Features (Hive example)

```
$ hive (version 1.2.1 HDP 2.3.1)

hive> SELECT SUM(duration)
    > FROM call_detail_records
    > WHERE
    >     type = 'INTERNET‘
    > OR  phone_number IN ( SELECT phone_number
    >                       FROM customer_details
    >                       WHERE region = 'R04' );

FAILED: SemanticException [Error 10249]: Line 5:17
  Unsupported SubQuery Expression 'phone_number':
  Only SubQuery expressions that are top level
  conjuncts are allowed
```

We Oracle users have been spoiled with very sophisticated SQL engine for years :)

Scalability vs. Features (Impala example)

Cloudera: CDH5.5
Impala 2.3.0

```
$ impala-shell
SELECT SUM(order_total)
FROM orders
WHERE order_mode='online'
OR customer_id IN (SELECT customer_id FROM customers
                   WHERE customer_class = 'Prime');

Query: select SUM(order_total) FROM orders WHERE
   order_mode='online' OR customer_id IN (SELECT
   customer_id FROM customers WHERE customer_class =
   'Prime')

ERROR: AnalysisException: Subqueries in OR predicates are
   not supported: order_mode = 'online' OR customer_id IN
   (SELECT customer_id FROM soe.customers WHERE
   customer_class = 'Prime')
```

Scalability vs. Features (Hive example)

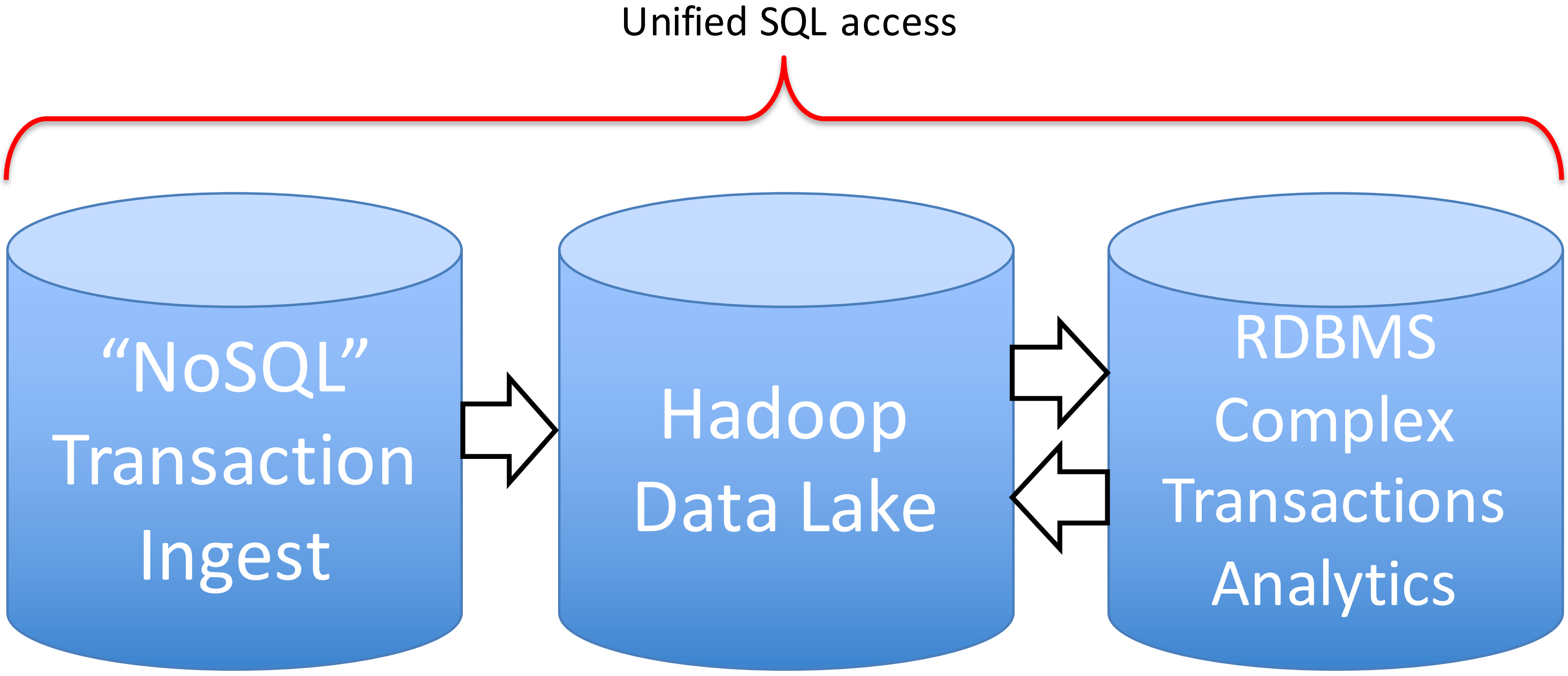
```
$ hive (version 1.2.1 HDP 2.3.1)

hive> SELECT COUNT(*) FROM sales10m
    > WHERE
    >     cust_id IN (SELECT cust_id FROM c)
    > AND prod_id IN (SELECT prod_id FROM p);

FAILED: SemanticException [Error 10249]: Line 1:85
  Unsupported SubQuery Expression 'prod_id': Only 1
  SubQuery expression is supported.
```

These are deliberately very simple examples. Not even talking about complex window functions etc

Engineering Tradeoffs

- Speed of Application Delivery
- Application Speed & Efficiency
- Workload Scalability
- Application Reliability
- Cost!

Hadoop, NoSQL, RDBMS One way to look at it



Unified SQL access
“NoSQL” Transaction Ingest
Hadoop Data Lake
RDBMS Complex Transactions Analytics

Scalable Simple Transactions

All Data! Scalable, Flexible

Sophisticated, Out-of-the-box, Mature

Thanks!

http://blog.tanelpoder.com

tanel@tanelpoder.com

@tanelpoder

We are hiring developers & data engineers!!!

http://gluent.com