Shifting from Capture-First to Query-First Database Architectures
Hello!

I am Rob Dickinson
CTO at Resurface Labs

You can find me at @robfromboulder
Agenda

- Review database landscape
- Review capture-first method
- Contrast with query-first thinking
- Method for query-first projects
- Cheat sheet for DB selection
Let’s talk databases...
How many databases are out there?

DB-Engines Ranking

The DB-Engines Ranking ranks database management systems according to their popularity. The ranking is updated monthly.

Read more about the method of calculating the scores.

<table>
<thead>
<tr>
<th>Rank</th>
<th>DBMS</th>
<th>Database Model</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Oracle</td>
<td>Relational, Multi-model</td>
<td>1274.92</td>
</tr>
<tr>
<td>2.</td>
<td>MySQL</td>
<td>Relational, Multi-model</td>
<td>1220.69</td>
</tr>
<tr>
<td>3.</td>
<td>Microsoft SQL Server</td>
<td>Relational, Multi-model</td>
<td>1007.97</td>
</tr>
<tr>
<td>4.</td>
<td>PostgreSQL</td>
<td>Relational, Multi-model</td>
<td>553.52</td>
</tr>
<tr>
<td>5.</td>
<td>MongoDB</td>
<td>Document, Multi-model</td>
<td>469.97</td>
</tr>
<tr>
<td>6.</td>
<td>IBM Db2</td>
<td>Relational, Multi-model</td>
<td>157.78</td>
</tr>
<tr>
<td>7.</td>
<td>Redis</td>
<td>Key-value, Multi-model</td>
<td>155.89</td>
</tr>
<tr>
<td>8.</td>
<td>Elasticsearch</td>
<td>Search engine, Multi-model</td>
<td>152.18</td>
</tr>
<tr>
<td>9.</td>
<td>SQLite</td>
<td>Relational</td>
<td>125.06</td>
</tr>
</tbody>
</table>

370 systems in ranking, April 2021

(from db-engage.com)
From "Designing Data-Intensive Applications" by Martin Kleppmann
How did we get here?
Long long ago…
Database development for dinosaurs

• Select a database platform ← this was easy! 🤪
• Define schema
• Start loading & integrating
• Tune normalization & queries
• Add materialized views & query caching
• Switch platforms if all else fails

🤔 this is “capture-first” thinking! queries arrive too late to influence DB choice
From “Designing Data-Intensive Applications” by Martin Kleppmann, modified by Rob without endorsement
How to tackle this “paradox of choice”?

• Bribe a trusted data architect or DBA 🍻
• Look at relevant benchmarks: TPC-XX
• What’s missing from your CV/resume?
• Stick to what you know?
• Dart board? Magic 8 ball?
The best way to pick a database is...

Work backwards from target read workloads

🔥 this is a “query-first” approach

🤔 kinda like TDD for database architecture
Not saying that write performance can be ignored or write benchmarks are bad
Why focus on read workloads?

🔥 For most systems, reads are the locus of value

• Zen koan: what’s the value of a write that can’t be read?
• Writes are just a cost of expected reads

Different databases have different tricks for reads:
• Indexes are extra writes to accelerate reads
• Replication is extra writes to ensure reads
Query-first method for DB selection

1. Define seed data that approximates a working system
2. Run read workloads for seed data on multiple DBs
3. Select the database with best workload fit (ops/sec)
4. Then optimize for loading/maintaining data
5. If no single database platform is a match:
   Use a distributed query engine like Trino
   Replicate data through queues like Kafka
   Or consider other tricks
A query-first example

• Resurface is a purpose-built database for API traffic

• We built our v1 product around Presto+Pulsar
  • We obsessed over ingest/indexing performance
  • Performance for actual customer queries was terrible 😱

• We started v2 with 1 year of high-quality data
  • Defined queries for identifying failures, slowdowns, and threats
  • Prototyped on Trino memory connector, 🚀 but not shippable
  • Tried on Trino+Redis, too much network time 😞
  • Tried on Trino+CSV, better but not awesome 😞
  • Built custom Trino connector & in-memory storage 😩
Focus on reads
Read algorithms

**Cache**: keys/values in hash table  
O(1) for a value

**B-tree**: rows in primary tree, indexes in other trees  
O(log n) for a row

**Columnar**: one tree per column, rows are links across trees  
O(log n) for a column – but fewer I/Os than b-tree  
O(log n) * k for a row with k columns

**LSM**: keys/values with leveled storage, background merging  
O(n) for a value

**M/R**: distributed table scan, partition elimination  
O(n) for any transformation – highest I/Os of any option
# Types of read workloads
(read I/Os on log scale)

<table>
<thead>
<tr>
<th>Read Workload</th>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fetch value for single key</td>
<td>Key/Value</td>
<td>Returns unstructured value</td>
</tr>
<tr>
<td>Fetch values for related keys</td>
<td>Key/Value</td>
<td>Returns collection of values</td>
</tr>
<tr>
<td>Find single row with criteria</td>
<td>OLTP</td>
<td>Returns tuple (row of named columns) using column indexes</td>
</tr>
<tr>
<td>Find group of rows with criteria</td>
<td>OLTP</td>
<td>Returns collection of tuples using column indexes</td>
</tr>
<tr>
<td>Read rows within transaction</td>
<td>OLTP</td>
<td>Returns value based on transaction isolation level</td>
</tr>
<tr>
<td>Join subset of rows &amp; related rows</td>
<td>OLAP</td>
<td>Returns collection of tuples joined across multiple tables</td>
</tr>
<tr>
<td>Join/summarize for few columns</td>
<td>OLAP</td>
<td>Returns count/histogram on a limited set of columns</td>
</tr>
<tr>
<td>Find/join/summarize for all columns</td>
<td>DSS</td>
<td>Returns data transformation computed against all available columns</td>
</tr>
</tbody>
</table>
## Read workloads by database

<table>
<thead>
<tr>
<th>READ WORKLOAD</th>
<th>CACHE</th>
<th>LSM</th>
<th>BTREE</th>
<th>COLUMNAR</th>
<th>M/R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fetch value for single key</td>
<td>😞 Redis, Memcached</td>
<td>😞 Cassandra, HBase, RocksDB, LevelDB</td>
<td>😞 MySQL, Postgresql, SQLite, SQL Server</td>
<td>😞 Druid, Iceberg, Parquet, Orc</td>
<td>😞 Hadoop, Resurface</td>
</tr>
<tr>
<td>Fetch values for related keys</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
</tr>
<tr>
<td>Find single row with criteria</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
</tr>
<tr>
<td>Find group of rows with criteria</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
</tr>
<tr>
<td>Read row within transaction</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
</tr>
<tr>
<td>Join subset of rows &amp; related rows</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
</tr>
<tr>
<td>Join/summarize for few columns</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
</tr>
<tr>
<td>Find/join/summarize for all columns</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
<td>😞</td>
</tr>
</tbody>
</table>

With 🎉 from Resurface
Super-columnar queries (all columns)
Queues vs queries

Postgresql → Kafka → Iceberg

Trino → Postgresql → Iceberg
Advanced read optimizations

😊 Move data in-memory to eliminate device I/O
  Use local/embedded store to eliminate network

😍 Use computed (virtual) columns
  Use optimized storage when table scanning

’util’ Immutable writes as safe transactions
  In-memory storage via page cache
  Push queries closer to data
With query-first methods, the possibilities are endless
THANK YOU!

PERCONA
LIVE ONLINE
MAY 12 - 13th
2021
Thank you!

Any questions?

You can find me at @robfromboulder or rob@resurface.io