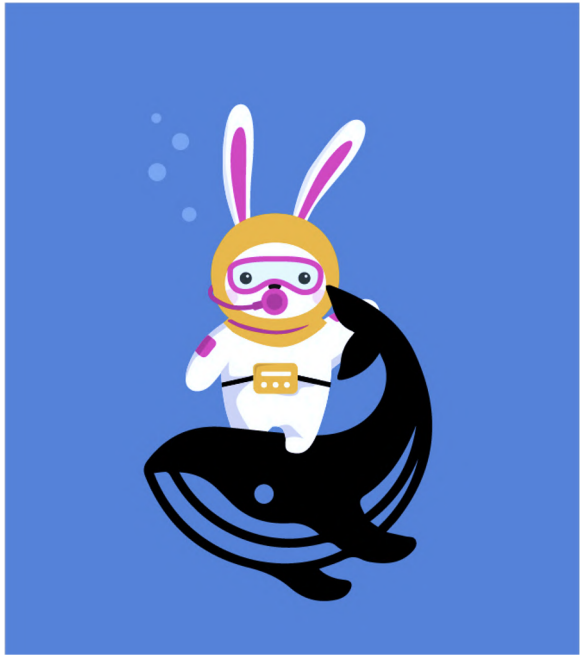
PERCONA LIVE ONLINE MAY 12 - 13th 2021

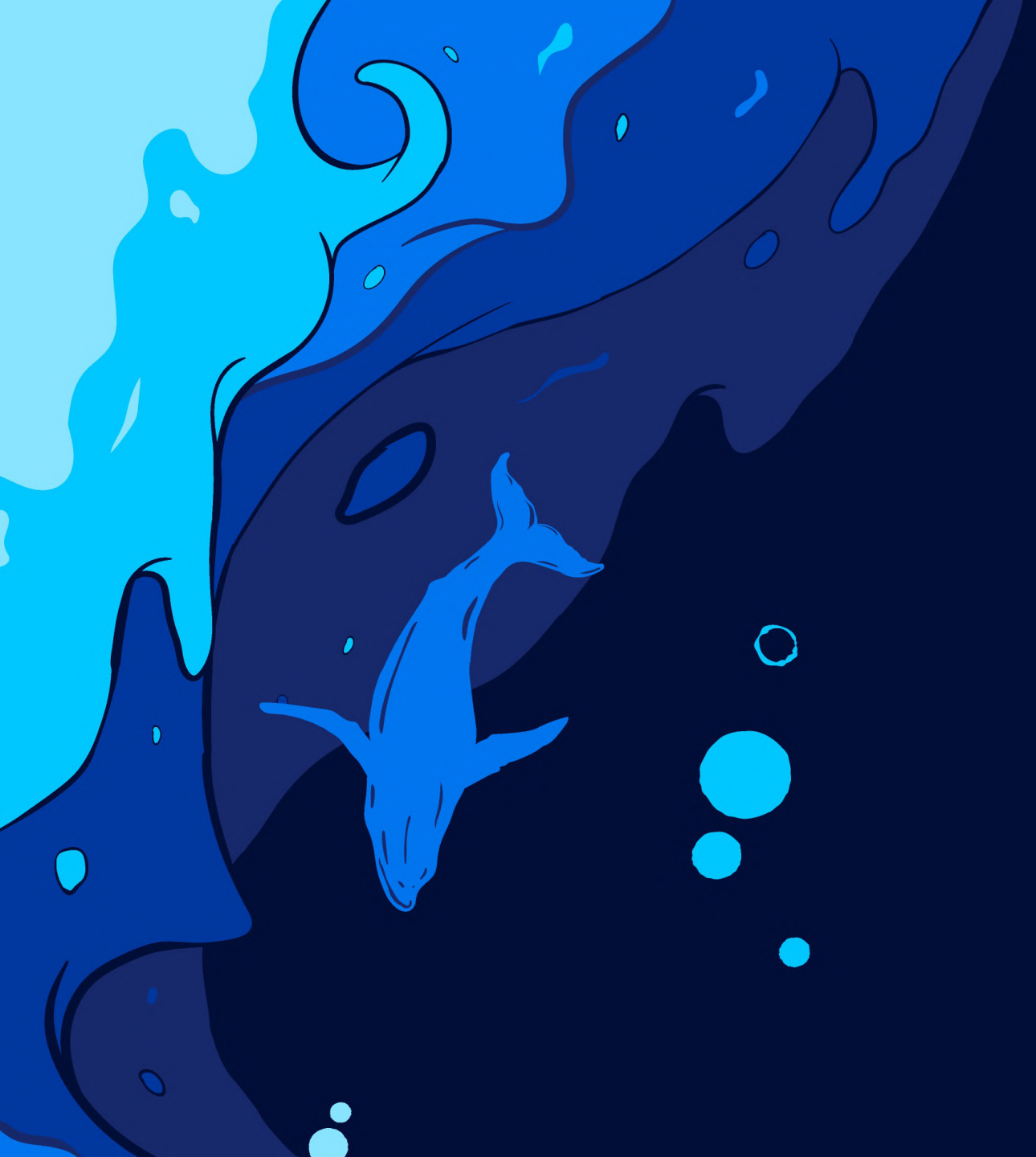
Shifting from Capture-First to Query-First Database Architectures

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 🥳

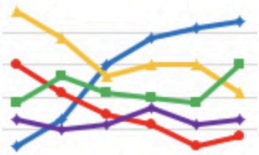
Let’s talk databases…

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.



trend chart

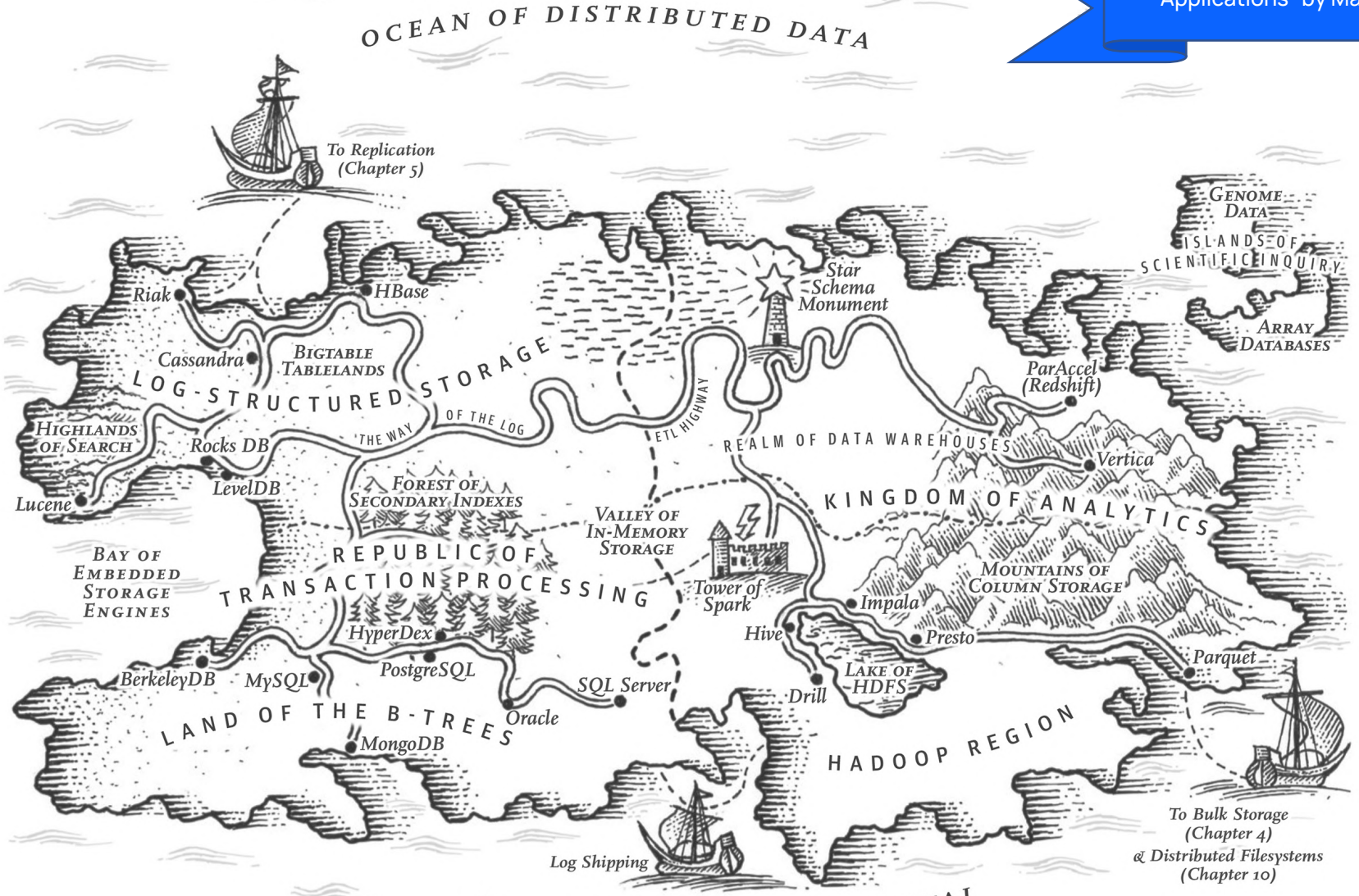
😱

370 systems in ranking, April 2021

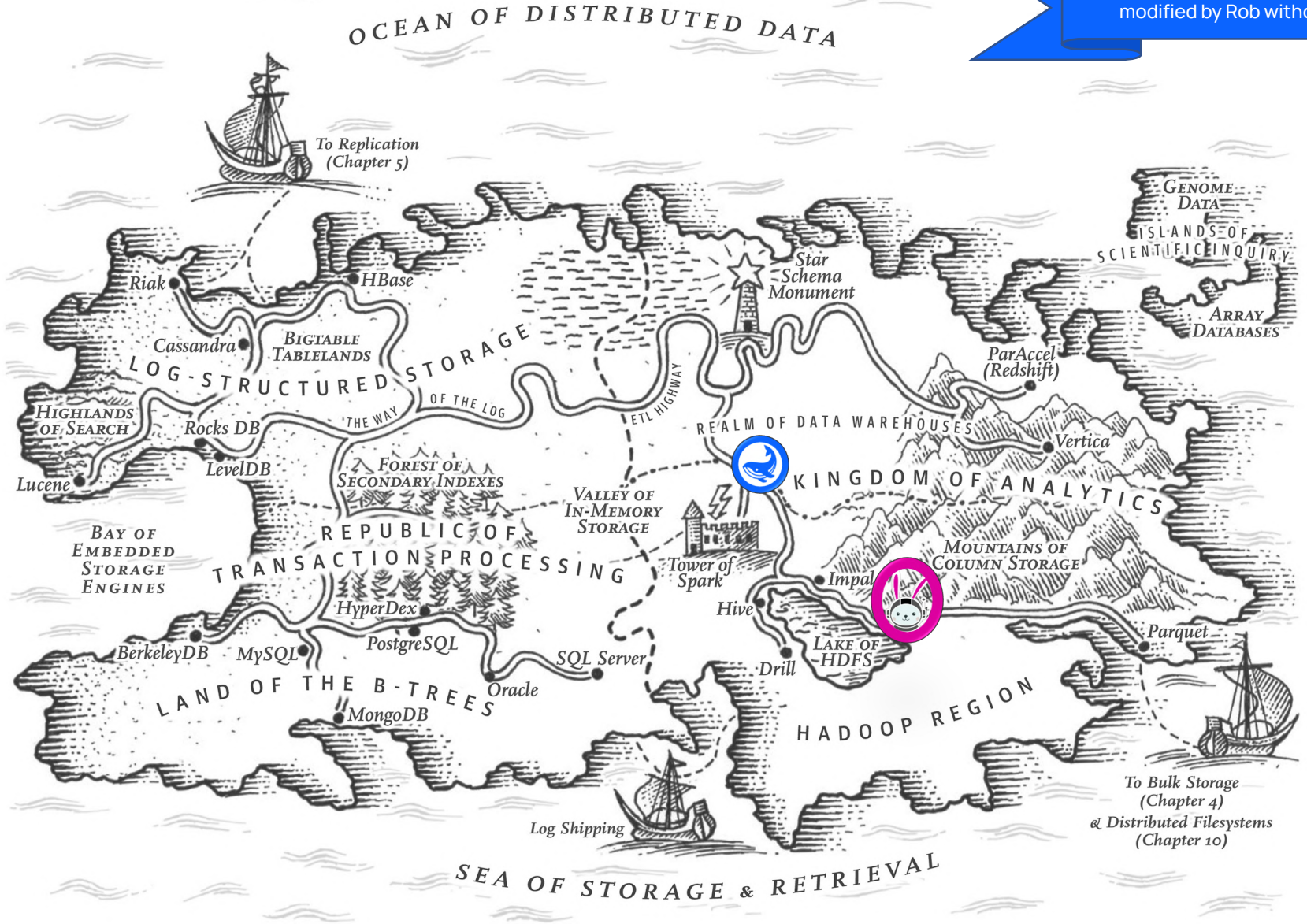
Rank			DBMS	Database Model	Score		
Apr 2021	Mar 2021	Apr 2020			Apr 2021	Mar 2021	Apr 2020
1.	1.	1.	Oracle	Relational, Multi-model	1274.92	-46.82	-70.51
2.	2.	2.	MySQL	Relational, Multi-model	1220.69	-34.14	-47.66
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1007.97	-7.33	-75.46
4.	4.	4.	PostgreSQL	Relational, Multi-model	553.52	+4.23	+43.66
5.	5.	5.	MongoDB	Document, Multi-model	469.97	+7.58	+31.54
6.	6.	6.	IBM Db2	Relational, Multi-model	157.78	+1.77	-7.85
7.	7.	↑ 8.	Redis	Key-value, Multi-model	155.89	+1.74	+11.08
8.	8.	↓ 7.	Elasticsearch	Search engine, Multi-model	152.18	-0.16	+3.27
9.	9.	9.	SQLite	Relational	125.06	+2.42	+2.87

(from db-engines.com)

From “Designing Data-Intensive Applications” by Martin Kleppmann

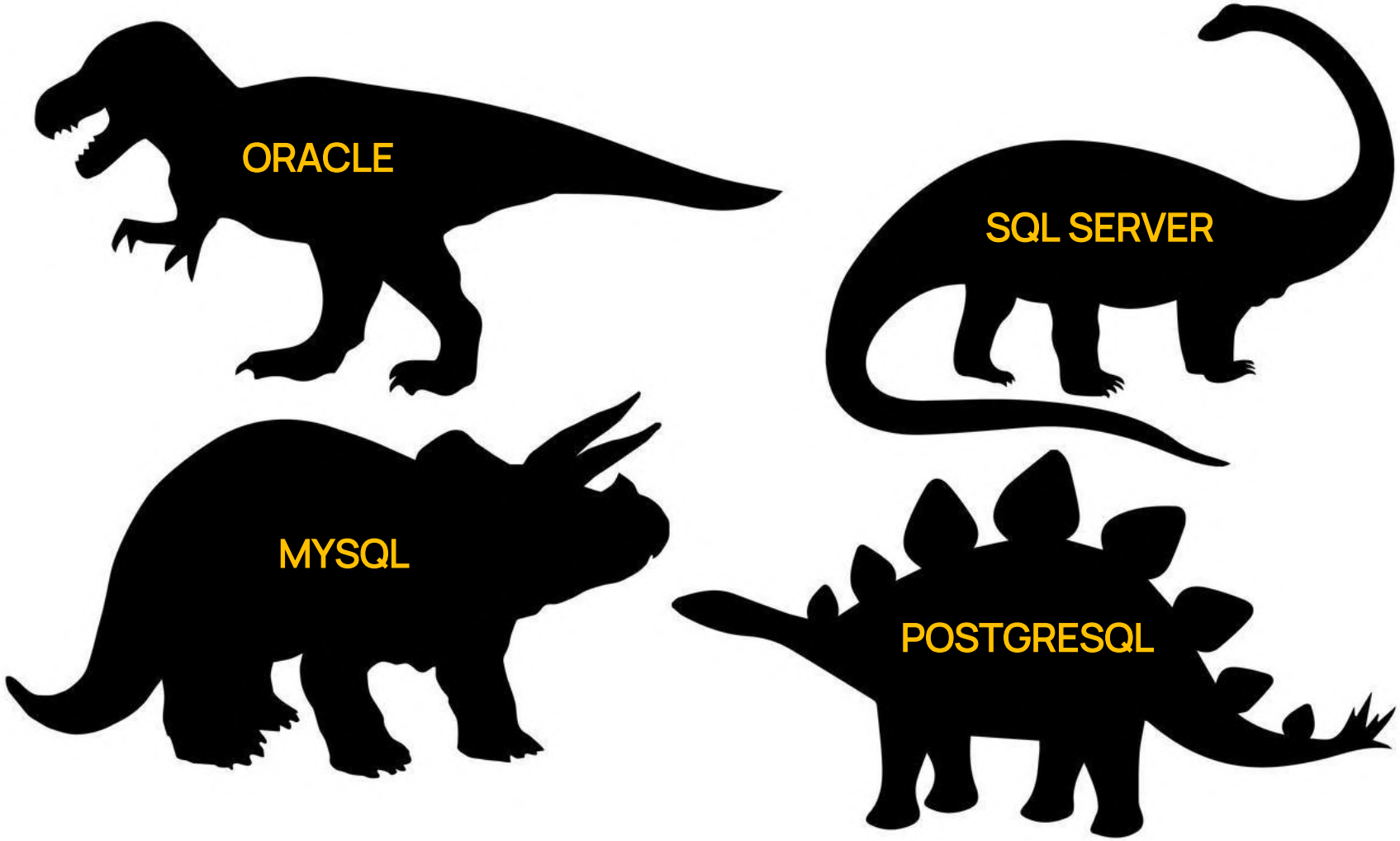


From “Designing Data-Intensive Applications” by Martin Kleppmann, modified by Rob without endorsement



How did we get here?

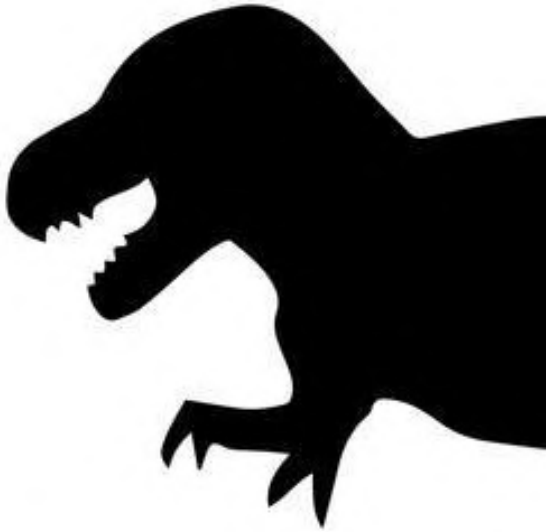
Long long ago…



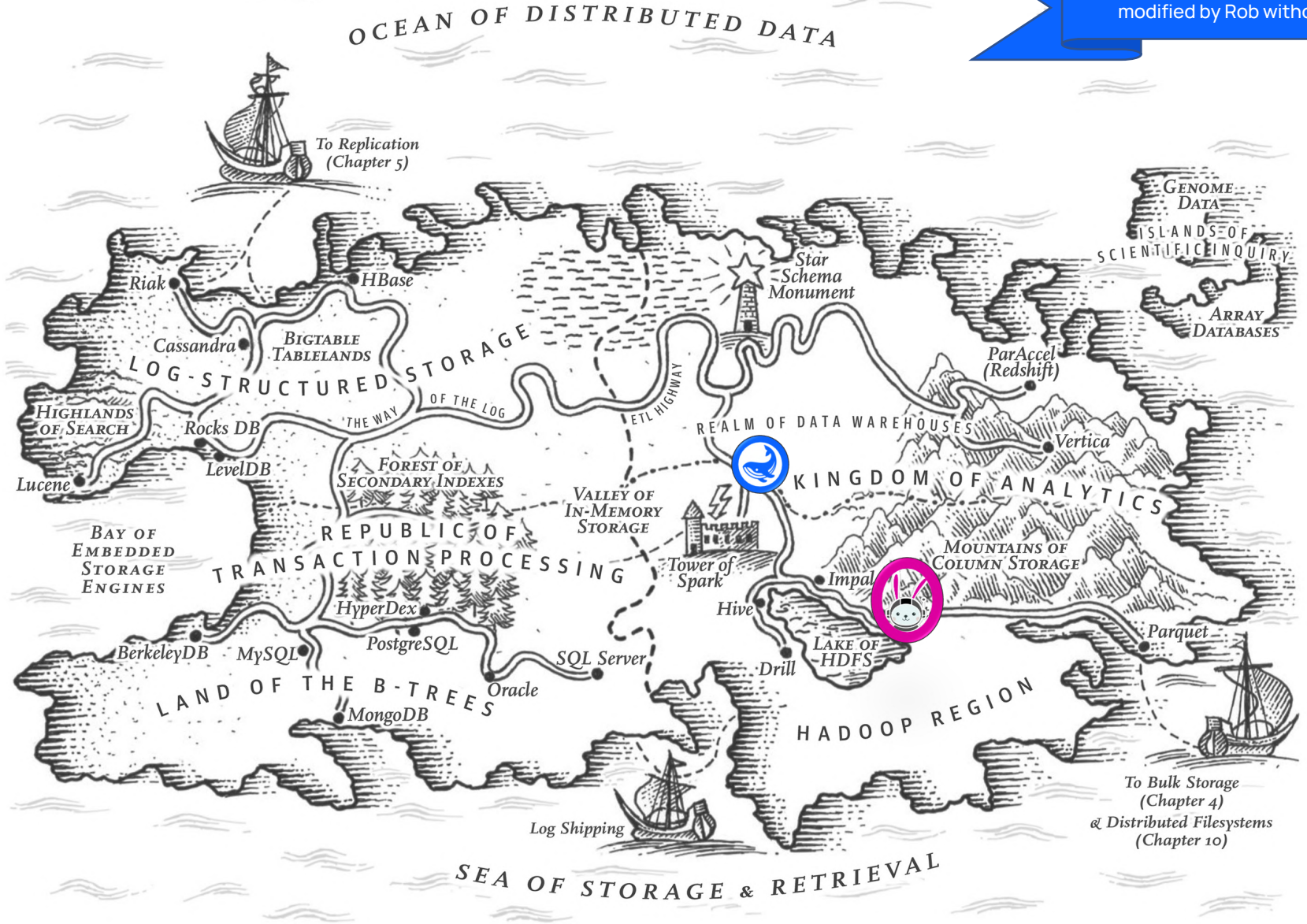
Database development for dinosaurs

- 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

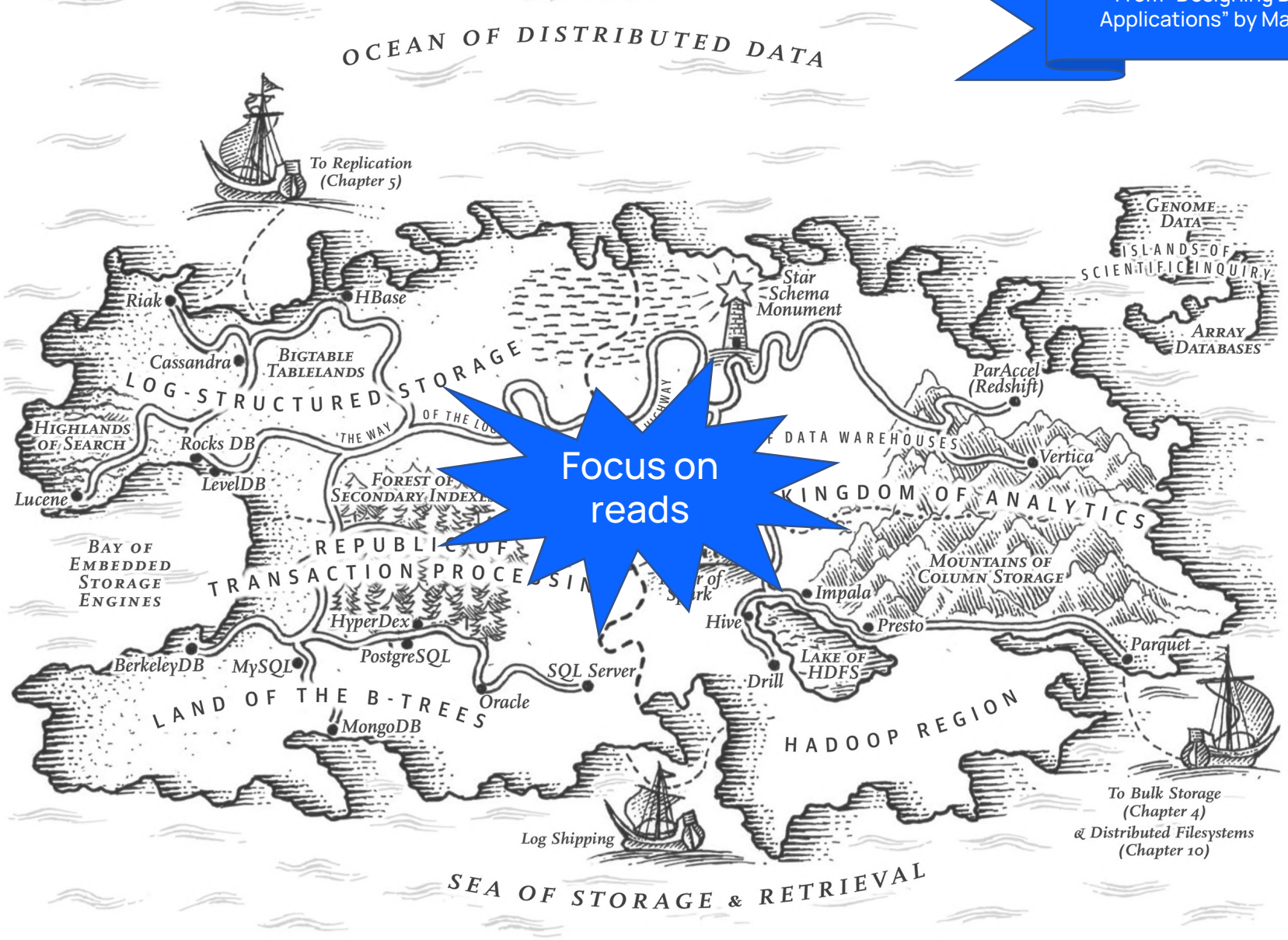
Query-first method for DB selection

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 🤩

From “Designing Data-Intensive Applications” by Martin Kleppmann



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

normal reaction to big-O notation

Types of read workloads
(read I/Os on log scale)

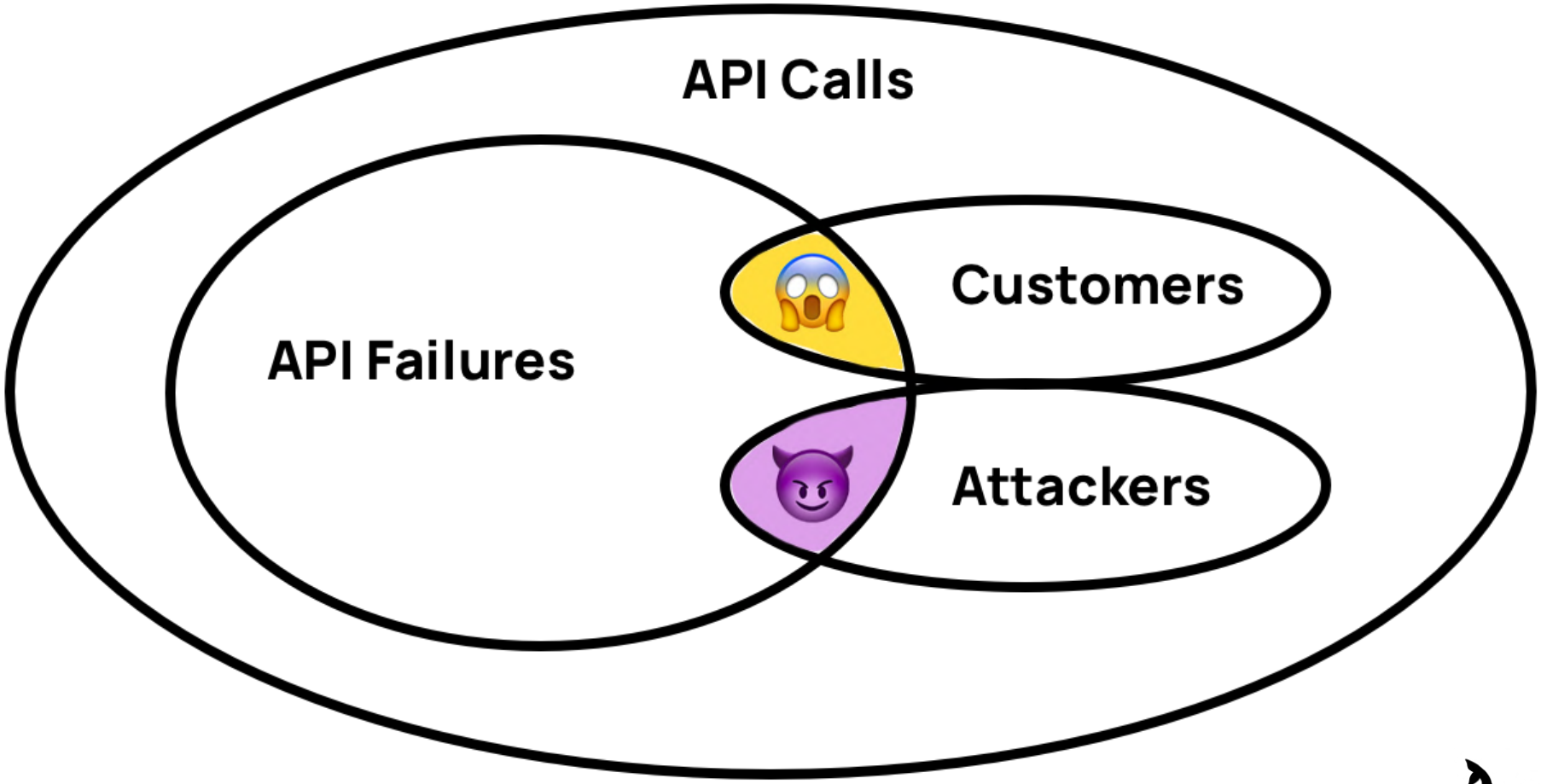
Read Workload	Category	Description
Fetch value for single key	Key/Value	Returns unstructured value
Fetch values for related keys	Key/Value	Returns collection of values
Find single row with criteria	OLTP	Returns tuple (row of named columns) using column indexes
Find group of rows with criteria	OLTP	Returns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

Read workloads by database

With ❤️ from Resurface

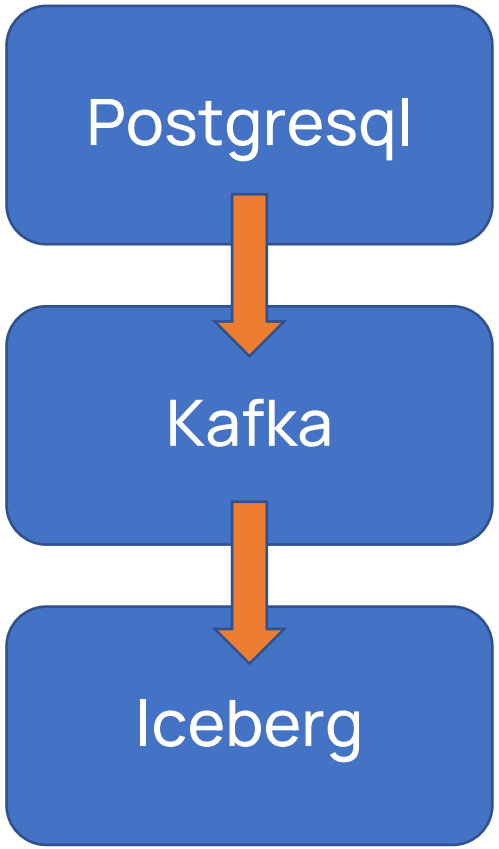
READ WORKLOAD	CACHE Redis, Memcached	LSM Cassandra, HBase, RocksDB, LevelDB	BTREE MySQL, Postgresql, SQLite, SQL Server	COLUMNAR Druid, Iceberg, Parquet, Orc	M/R Hadoop, Resurface
Fetch value for single key	🥳	😀	🤔		
Fetch values for related keys	😀	🥳	😀		
Find single row with criteria	😀	😀	🥳		
Find group of rows with criteria		😀	🥳		
Read row within transaction		🤔	🥳		
Join subset of rows & related rows			🥳	😀	
Join/summarize for few columns			😀	🥳	😀
Find/join/summarize for all columns			😱	😱	🥳

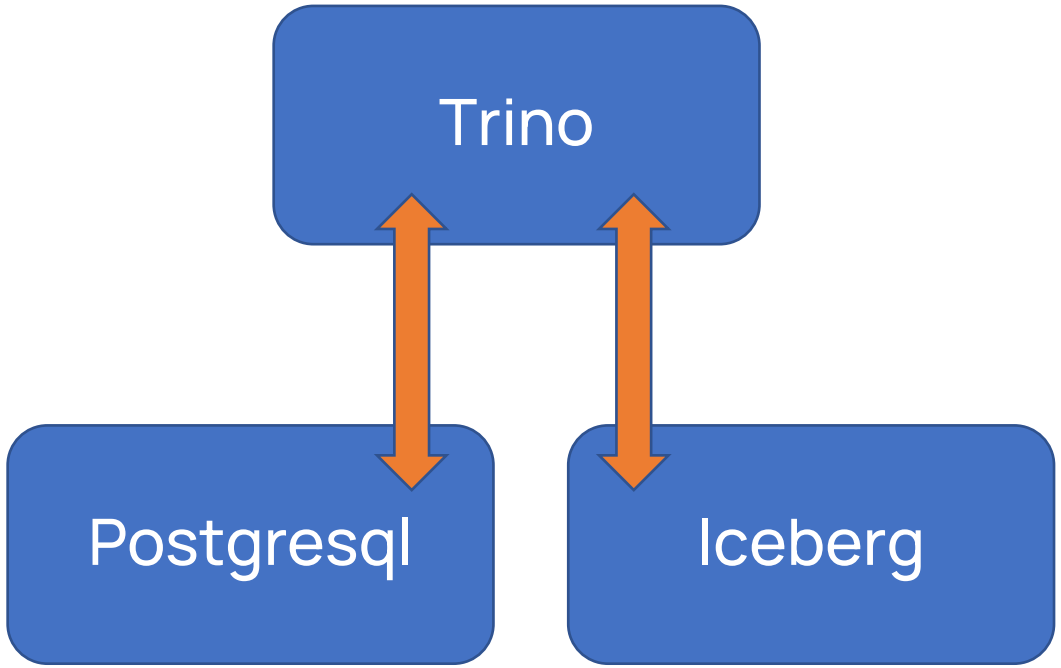
Super-columnar queries (all columns)



Resurface

Queues vs queries





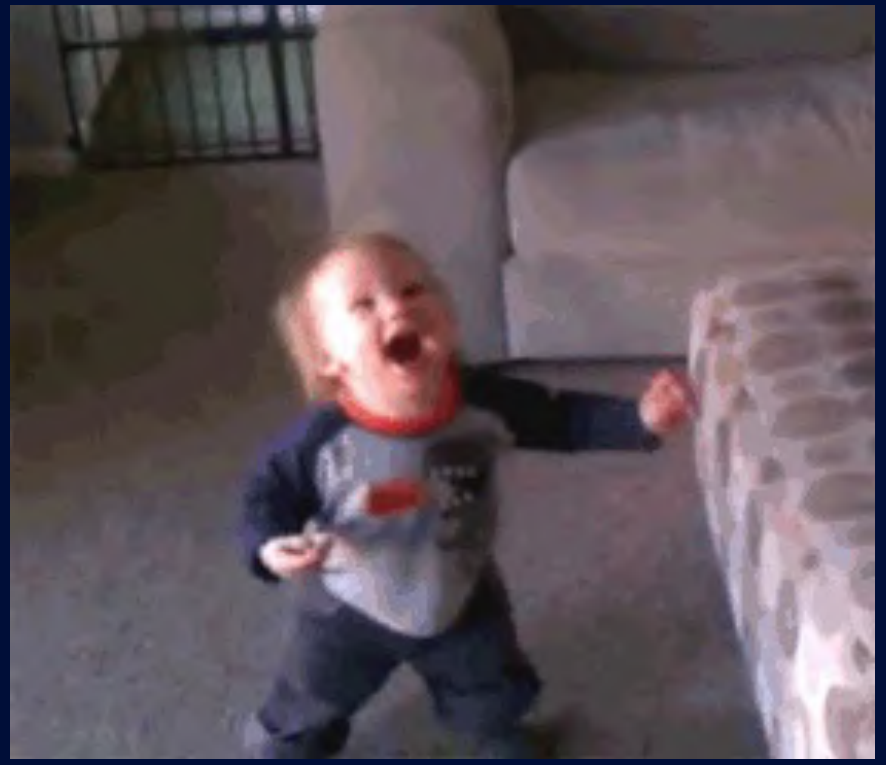
Advanced read optimizations

🧐 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

🤩 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
LIVEONLINE
MAY 12 - 13th
2021

Thank you!

Any questions?

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