WELCOME

Is Relational the New COBOL?

Mark Porter
Chief Technology Officer
MongoDB

Builder's Fest
June 9, 2022 | New York City
MongoDB World '22

Notes - not part of presentation

Is Relational The New COBOL? (~15 min talk) (2:30-2:50 pm)

Abstract: Relational databases were revolutionary when they were ideated and built. However, over time, various things have changed that make them no longer the best way to build modern Apps. I will trace those trends in data, computing and programming from the 1970s until today. And I will explain why those trends lead to needing a fundamentally different architecture underneath today's modern applications

Summary
15 - core content slides
4 - hidden optional slides, can use as substitutes or additions
8 - Navigation slides

Main to do's:
1) Markify
2) Do builds where needed
3) Add final quote
4) Finish speaker points

This notes slide is slide 1
2 - content - technology substitution this may be too busy.
If folks think so, it's easy to remove the corner pictures
3 - content - technology holdouts
4 - optional - zombie coal - this may be to direct for our comfort,
and my wish to omit to fit into the time budget in any case.
5 - nav - Technology Factors
6 - subNav - Relevance
7 - optional - full size disk growth - omit to save time?
8 - content
9 - content
10 - content
11 - subNav - Durable Technical Advantages
12 - content - KEY SLIDE, If Mark agrees, it needs builds
13 - content - KEY SLIDE, If Mark agrees, it needs builds
14 - optional - old Bookcase/warehouse differences slide
15 - optional - old Bookcase/warehouse differences slide
16 - content - horizontal scaling slide
17 - nav - Which is riskier?
18 - content - risk: slay dragon or maybe later?
19 - content - cost: be taxed or invest your way out
20 - nav - Is Relational The New COBOL
21 - subNav - The Foggy Future
22 - content - examples
23 - content - COBOL's niche
24 - subNav - Quitting is hard
25 - content - getting off relational
26 - subNav - Economics will decide
27 - content - concussion
28 - content - pithy quote

Technology Marches On Relentlessly

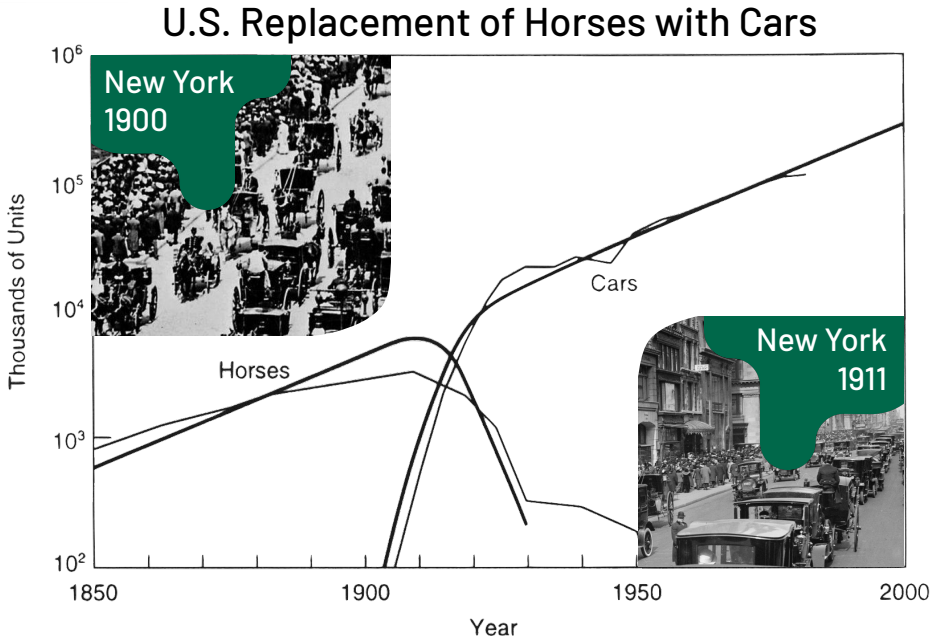
Cars replaced horses

Internal combustion replaced steam

Steam replaced water wheels

Water wheels replaced oxen

And It Can Happen Fast



After Nebojsa Nakicenovic (1997). Technological Change As A Learning Process. Induced Technology Workshop, IIASA. 26-27 June 1997.

Why Do Some Technologies Linger?

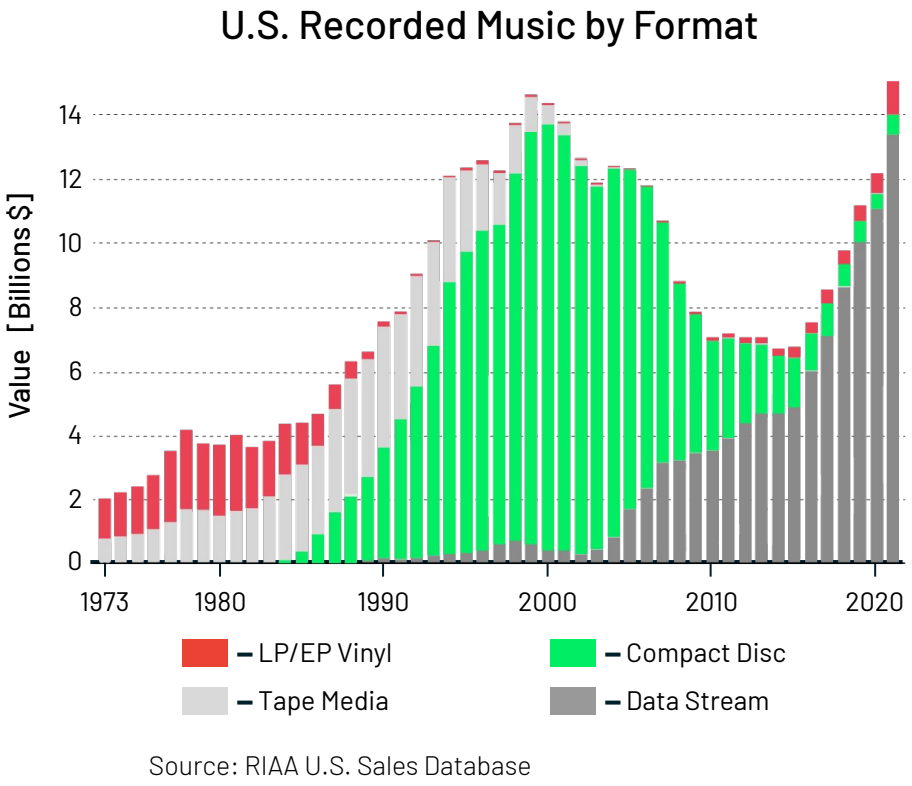
Some Technologies Survive in a Niche

CDs and Vinyl Records

FAX (and TELEX!)

CB Radio

Can You Spot The Zombies?



Source: RIAA U.S. Sales Database

Technology Longevity Factors

Relevance

Durable Technical Advantages

Risk and Switching Cost

Achieve What Matters

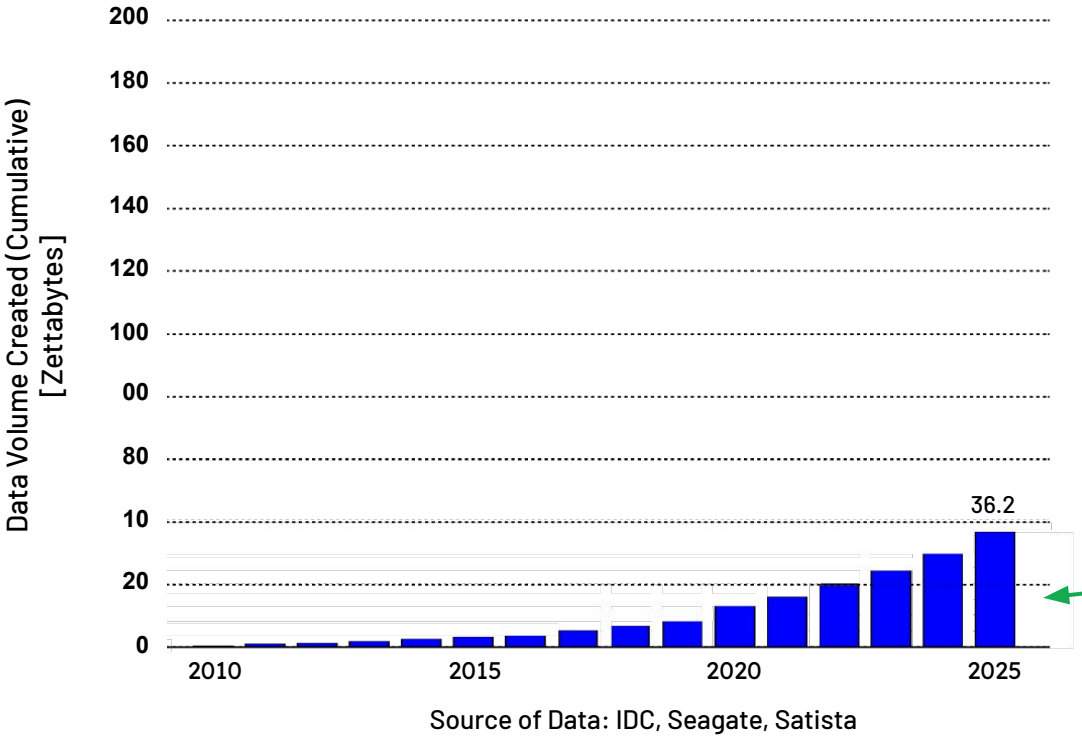
Relevance

Tomorrow's Challenges Are Different

Data Volumes Exploding

Innovation Required

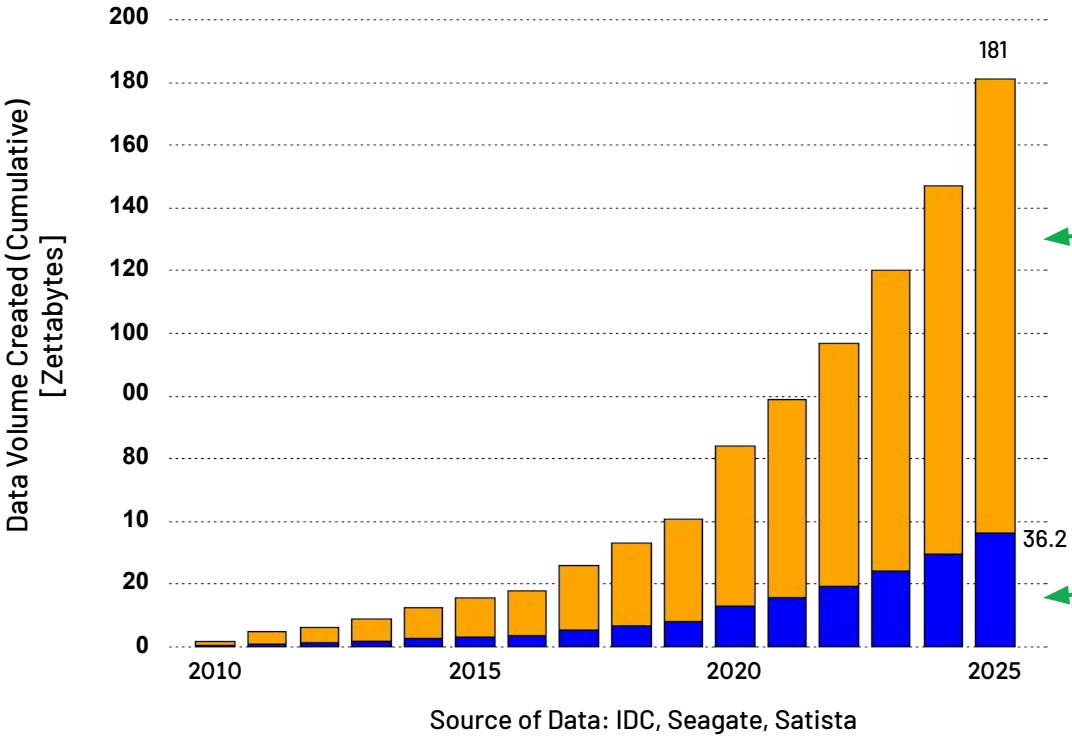
Structured data has been exploding for 50 years...



Source of Data: IDC, Seagate, Satista

Homogeneous Enterprise Data (Highly Structured)

Structured data has been exploding for 50 years...



Source of Data: IDC, Seagate, Satista

Heterogeneous Data
(semi-structured and unstructured)

IoT and Machine Data

Social Media

VoIP

Homogeneous Enterprise Data
(Highly Structured)

Relevant Database Technology Must Address

Exploding Data Volumes

Annual growth 23%/yr

Tomorrow's Challenges Are Different:

80% Of Data Heterogeneous

- Semi-structured/Unstructured
- Must look like scalar values in a Relational Schema

CPUs maxing out, Parallelism now imperative

- Can't scale up
- Must scale out

Innovation Required

Technology for Tomorrow

Durable Technical Advantages

Technical Contrasts

Data Model

Interaction model

Performance

Extension and Scaling

Relational and Document Models Compared

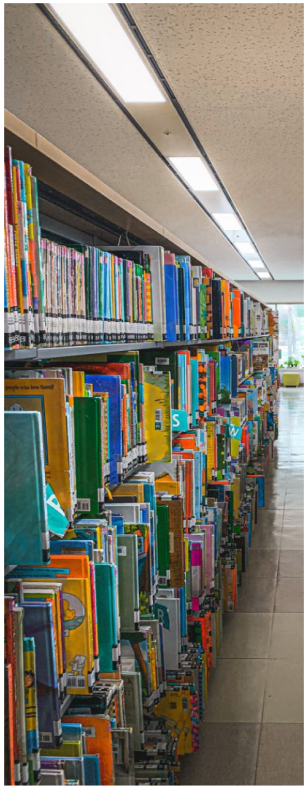
	Data Model	Interaction Model
Relational	Tables of typed scalars ● Predicate variables ● “Super Scalars” ○ JSON/JSONB ○ XML ○ Arrays ● Table entities often DO NOT correspond to real world entities	SQL + Extensions ● Two language abstractions ○ Declarative Relational Calculus ○ Imperative Extensions ● Example: SELECT JSON_VALUE(js, '$.floor[*] ? (@.level > 1).apt[*] ? (@.area > 40 && @.area < 90).no' RETURNING int) FROM house;
MongoDB	Containers of Documents ● Objects of Key/value pairs ● Scalar or non-scalar values ○ Nested objects ○ Arrays ○ BSON ● Document entities usually correspond to real world entities	MQL – Consistently Imperative ● Method Calls ● Aggregation Pipelines ○ “Pipes and filters” architecture pattern ● Example: db.house.find({{ "floor.apt.area": { $gt:40,$lt:80}}, {“floor.apt.no”: { $gt: 1} }},{ “floor.apt.no”: 1 }))

Relational and Document Model Features

	Performance Considerations	Extension & Scaling Considerations
Relational	ER Schema Design ● Denormalization ● Indexes DML Costs ● Normalize on write ● Joins on read ● SQL filters & conditions challenge optimization	Extension ● Must preserve global namespace & constraints ● Schema update often requires data lift and shift Scaling ● Distributed consistency hard w/ multiple servers ● CAP Theorem challenges RDBMS consistency
MongoDB	Document Schema Design ● Exploit Colcation (easy “tuple assembly”) ● Compound Indexes (avoid joins) DML Costs ● No normalization cost on insert ● Normalize only query results on read	Extension ● “Store and forget” easier than “normalize and store” ● Nested namespaces avoid global conflicts Scaling ● Adding servers doesn’t break schema constraints ● Developer controllable consistency

Schema Setup & Normalization Costs

Relational Storage

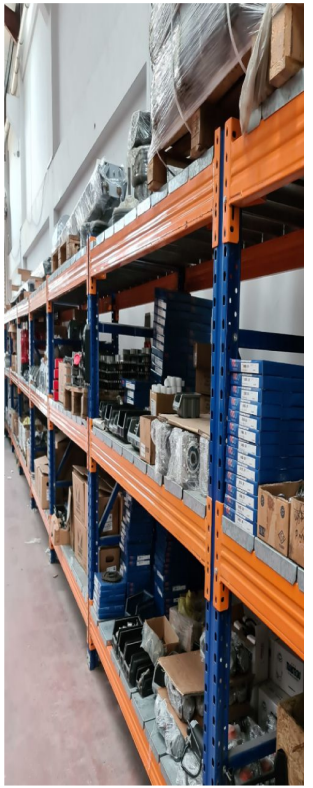


Schema Setup Cost

Must know entire data model beforehand to design schema

vs.

Only need to know enough to label for later retrieval & easier to change



Non-Relational Storage

Normalization Overhead

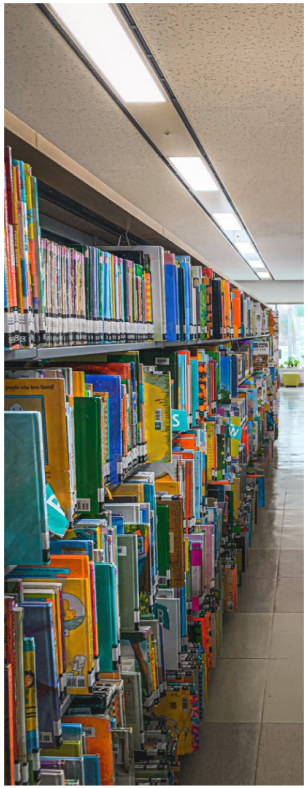
Unpack everything & put each in its proper place at insert/append time

vs.

Fetch only what's needed at query time, unpack only what's needed

Lower unpacking costs

Relational Storage



Gleaning Data Cost

Join together needed items from different sections (tables)

vs.

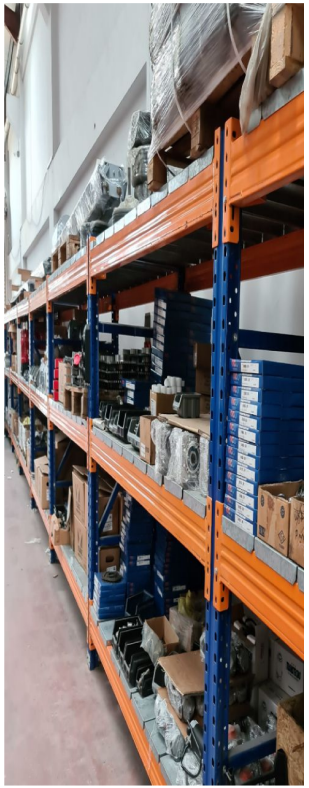
Items co-located so what we'll use is already persisted together

Normalize At Use Time

All data pre-normalized, so fields ready to assemble into tuple

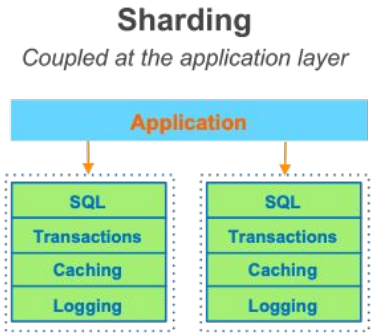
vs.

Once selected data are retrieved, normalization may be required

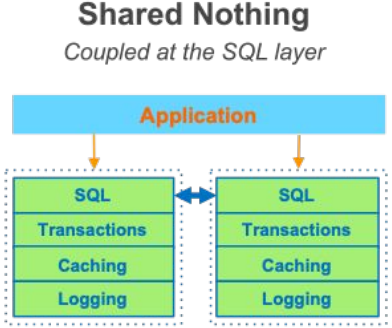


Non-Relational Storage

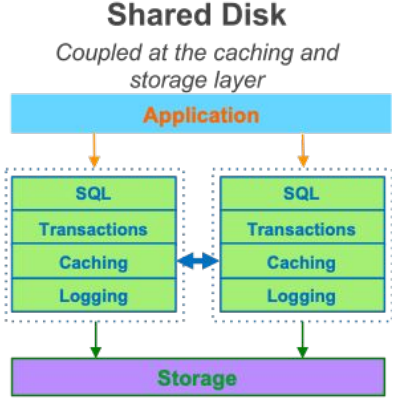
Sharing challenges horizontal scaling



Applications must understand topology



Queries easily partition and deploy



Orchestrated state management

Which is Riskier?

The Dragon:
Slay or Be Slain

Pay Now or Pay Later

Which is Riskier? Less Expensive Overall?

Don’t Wake The Dragon!

Legacy Application Risks

- Shared tables create Unknown Dependencies ➔ Brittleness
- Apps Undocumented ➔ Delays, Bugs

Applications Become Frozen In A Glacier of Dependencies

The Costs?

- Expensive to do, but can be done
- Late to Market
- Fragility and Down Time
- Missed Opportunities

Slay The Dragon & Migrate!

Migration Risk

- Unknown Dependency Risk Same
- Use Scaffolding and System Sync To Do *Safe Software Deployments*
  - Migrate incrementally and reversibly (usually possible)

The Costs?

- Migration is one time cost for ongoing savings
- Pay learning cost, reuse forever
- New tools reduce risk and cost

Enduring Non-Relational Cost Savings

Pay An Ongoing Legacy Relational Technology Tax?

Pay A One-Time Migration Cost For Ongoing Savings?

Easier schema design ➔ Lower setup (capital) costs

On-demand normalization ➔ Lower insert and query costs

Colocated data ➔ Lower “tuple assembly” costs

Simplified scaling ➔ Lower scaling costs

Simpler, non-declarative API ➔ More productive developers ➔ Lower Dev. costs

Is Relational the New COBOL?

The Future is Foggy

Quitting is HARD

Economics will Decide

No Crystal Ball

The Future Is Foggy

Forecasting is Hard

Innovator's Dilemma?

Why do some technologies endure while others do not?

Technologies Here but Fading

- Passenger Railroads (in US)
- Lotus Notes
- IBM Mainframes
- Coal
- FORTRAN, COBOL, PHP

Technologies Now Gone

- Horses
- 8-Track Tapes
- Lotus 123
- Blimps and Zeppelins
- Slide Rulers

COBOL has endured in its niche

COBOL Still Used Today

- Because it “just works”
- Can’t justify the switching cost
- Avoids perceived Migration Risk

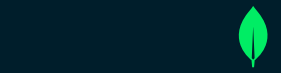
A Relational Resemblance?

- Will it “just work”?
- What will the cost difference be?
- Will it be less than switching costs?

An Innovator’s Dilemma…

Breaking Up
Is Hard To Do

Change Is Hard



Where to Start?

Deconstructing Old Code

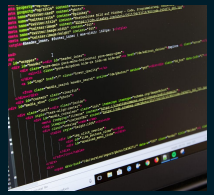
Designing For The Future

Moving The Data

Getting off Relational is Hard

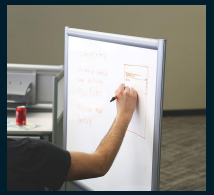
How do I get started?

A typical enterprise has hundreds of apps. Which ones are the best candidates for modernization, and which should be done first?



What happens in my old code?

How can we move our applications forward, when the legacy code may no longer be well documented or understood?



What does a modern schema look like?

Relational schema design is well-understood, but doesn't offer the best agility or performance. How should enterprise data be modelled in the modern age?

How do I get my data into the new schema?

Data needs to be transformed and migrated, taking into account any performance, security and integrity requirements.

The Dysmal Science

Economics Will Decide

Pay me now or pay me later?

Relational longevity possible but not assured

The Homogeneous/Heterogeneous data mix likely to will remain 20%/80%

- 20% of a lot of data is still a lot of relational-friendly data
- Non-relational is a preferred choice for new workloads

Possible futures for Relational:

- All data coerced into relational schemas, non relational was just a fad…
- Continue using relational technology for structured data; all/most new heterogeneous workloads are deployed onto non-relational platforms
- Migrate relational apps where the savings exceeds the switching costs
- Non-relational data platforms offering SQL compatibility become the norm

Economics and technical flexibility will matter more and more as data volumes grow

Thank you for your time.

Q&A

Mark Porter
Chief Technology Officer
MongoDB

linkedin.com/marklovestech
Twitter: @marklovestech
marklovestech.com

Thank You

“Whosoever desires constant success must change with the times.”

Niccollo Machiavelli

“They always say time changes things, but you actually have to change them yourself.”

Andy Warhol