Big Data, Bad Analogies

GOTO Copenhagen
September, 2014

Mark Madsen
www.ThirdNature.net
@markmadsen

Third Nature

The problem with bad framing

Leads to bad assumptions about use, inappropriate features, poor understanding of substitutability and the impacts it will have.

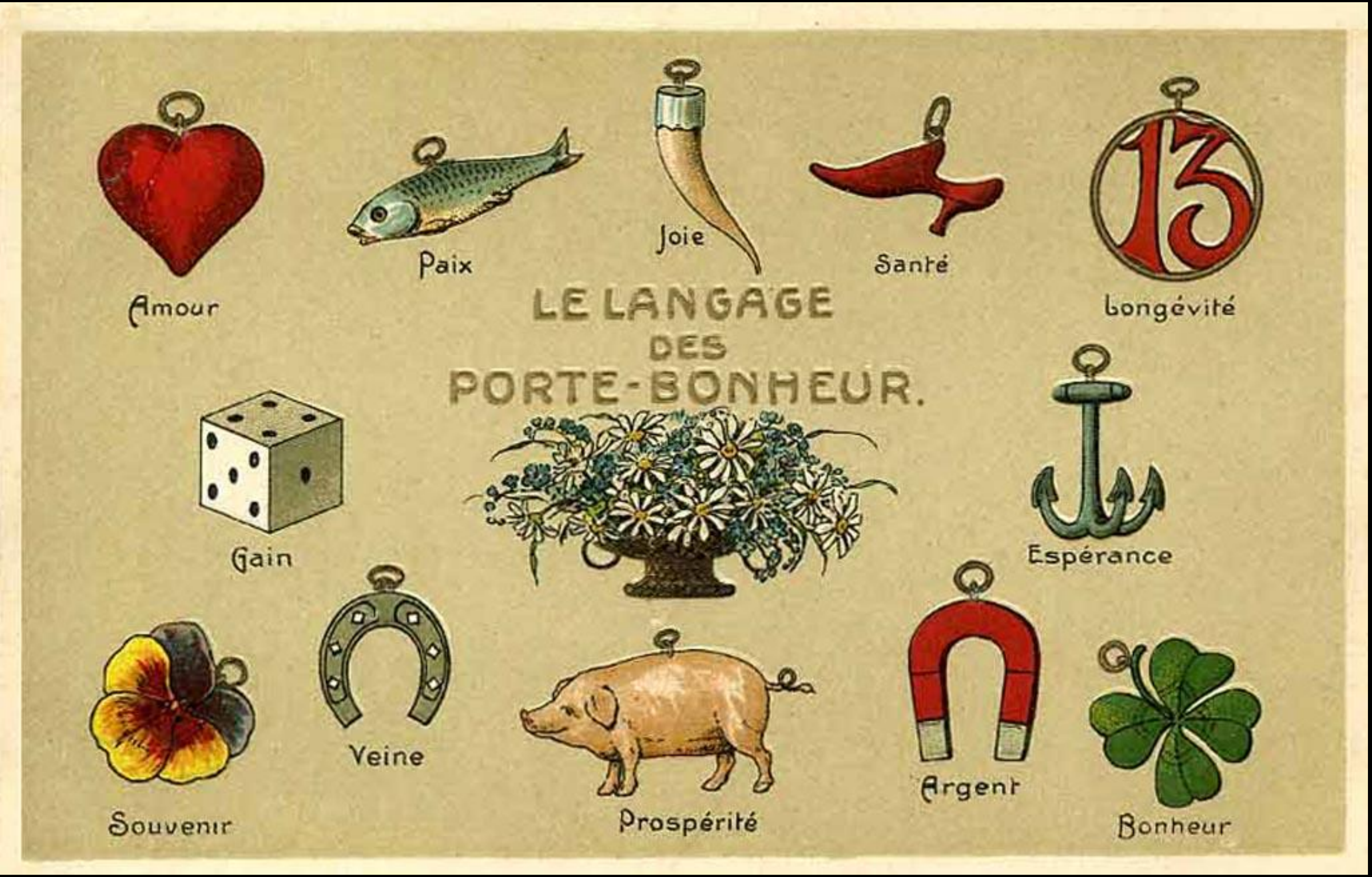
The data lake

The data lake after a little while

Data Exhaust

Data is the new oil

Reality: data is a choice



Technical Debt: the gist of this talk

tek-ni-kuh l det: the cost that accrues due to decisions made in software design and coding.

Look at the choices and mistakes in development:

Intentional	Unintentional
Purposeful choices to optimize schedule, budget, satisfaction	**Missed requirements, poor code quality, poor design**

Technical Debt

The cost of some choices can be dealt with in the short term (e.g. the next sprint) and some only in the long term (redesign, start over)

Short term	**Mostly about the application code**
Long term	**Mostly about architecture, design, and infrastructure**

If you enter into decisions knowing the true nature of your coding alternatives, you will be better off

Green: these are deliberate, the tradeoffs known

Yellow : these are minor defects

Red: these are the things that kill a system

	Intentional	Unintentional
Short term	**Code choices**	**Code flaws (i.e. bugs)**
Long term	**Design choices**	**Design flaws**

Third Nature

Technical Debt can't be avoided

(but it can be managed)

Sometimes you think it's intentional: incremental design

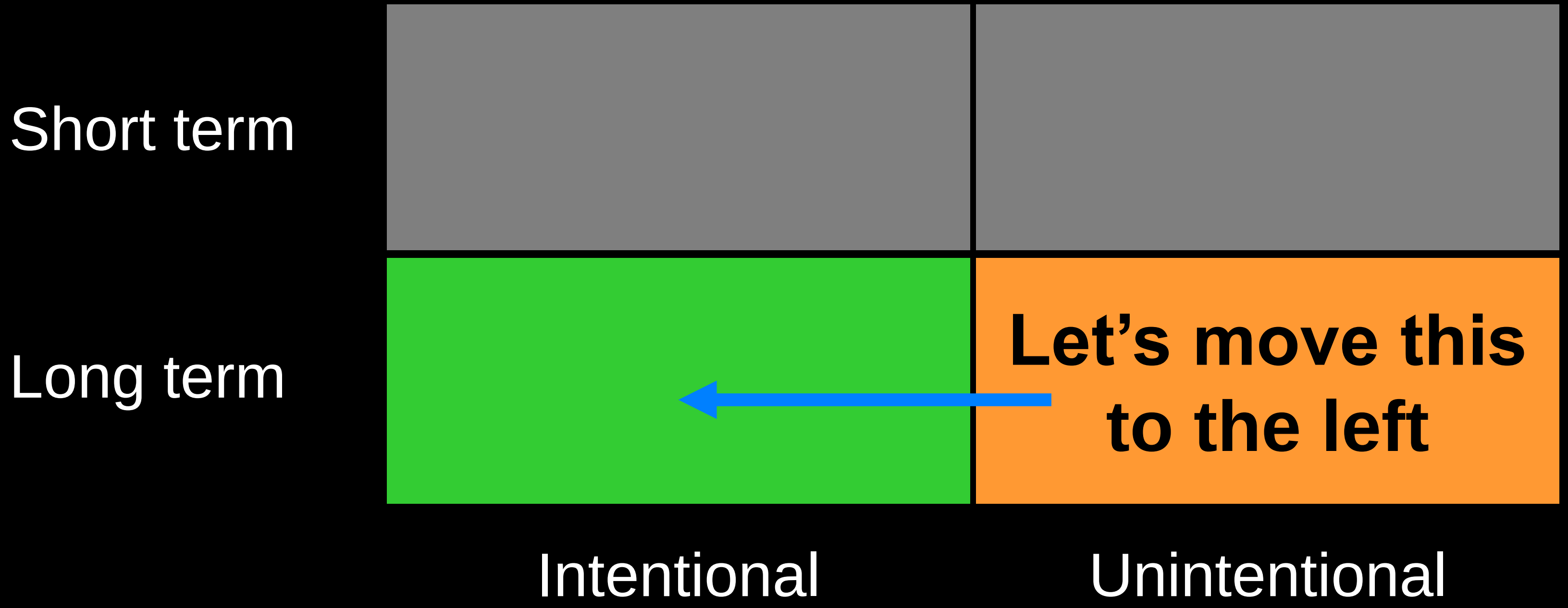
Long term debts can only be dealt with through planning

Short term	Agile methods	Development methods
Long term	Redesign	Experience, education
	Intentional	Unintentional

Third Nature

Technical Debt can't be avoided

What you believe about the technology underlying your system has a big influence on design choices, so the focus of this talk is on architecture and design with the hope it will help reduce or avoid long term debt.



Third Nature

“There is nothing new under the sun but there are lots of old things we don't know.”

Ambrose Bierce

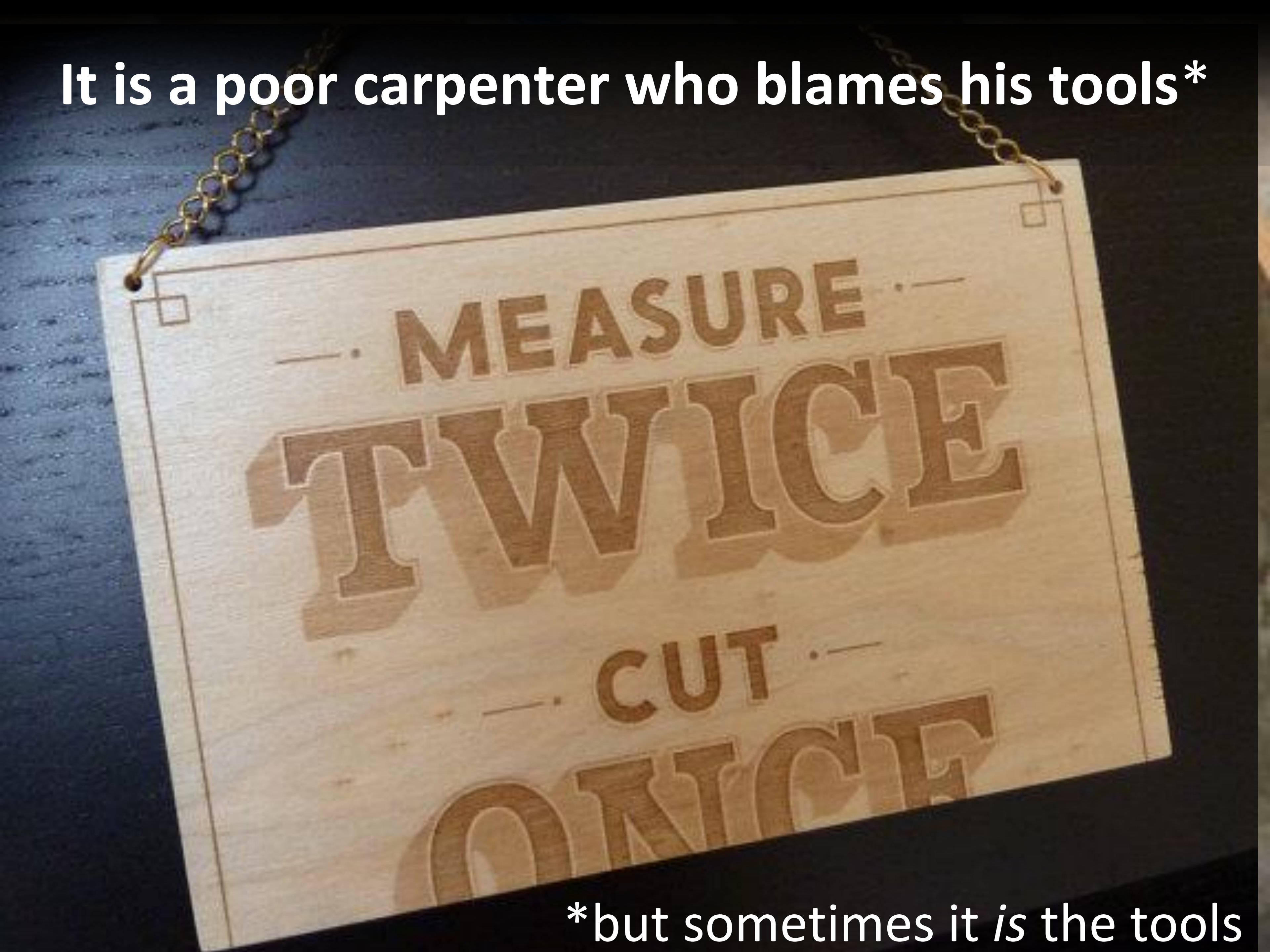
Things I hear

Why is the system slow? *The database*

Why doesn't the system scale? *The database*

Why is the system so expensive? *The database*

It is a poor carpenter who blames his tools*



*but sometimes it *is* the tools

Why do we need an entirely new model for the data management layer?

Because scalability aka “big”

Because “unstructured”

Because flexibility

Third Nature

What is best in life?

What is best in life?

Third Nature

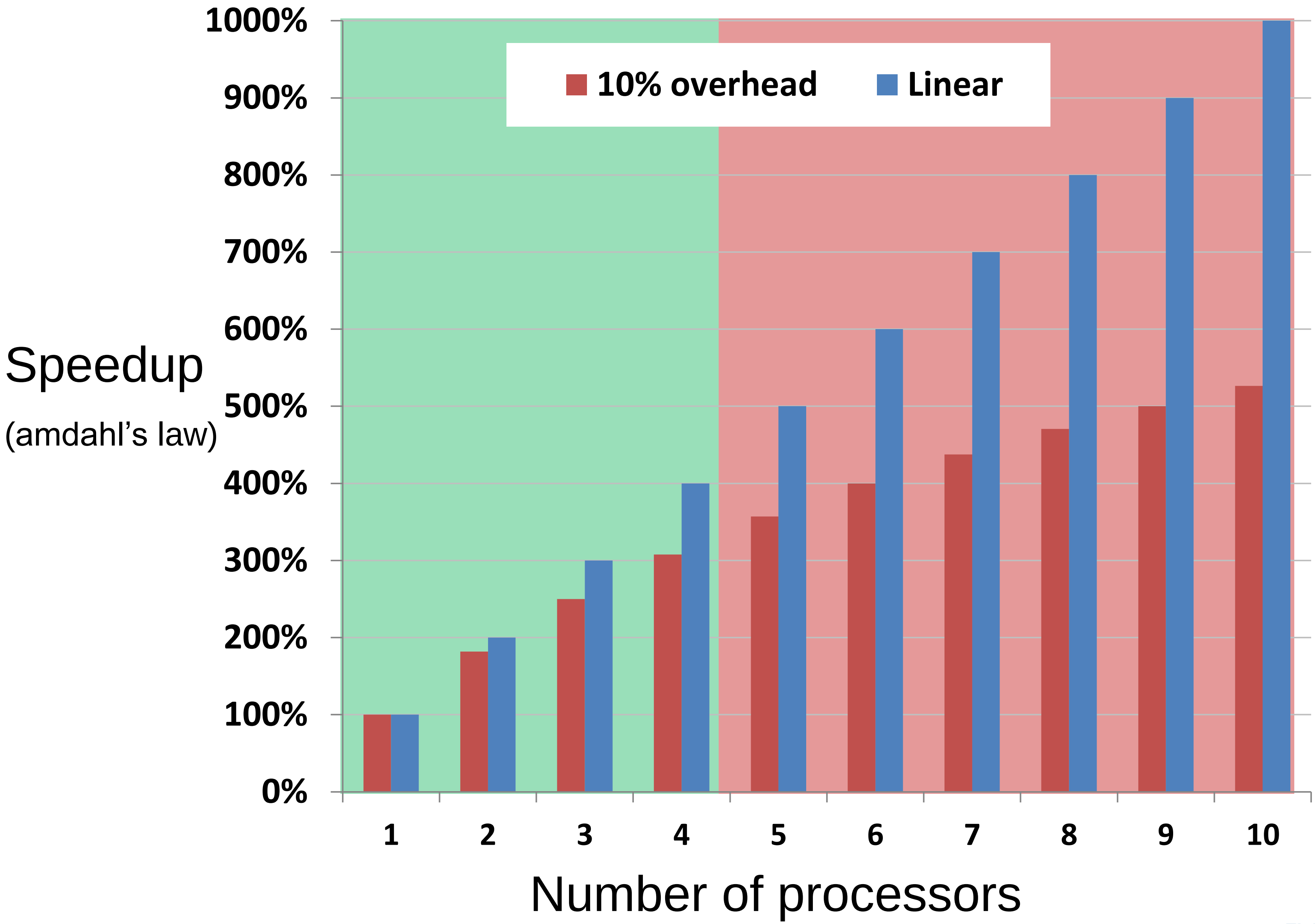
What is best in life?

Scalability? Just add hardware

"The most amazing achievement of the computer software industry is its continuing cancellation of the steady and staggering gains made by the computer hardware industry." *– Henry Peteroski*

After all, your database is web scale, isn't it?

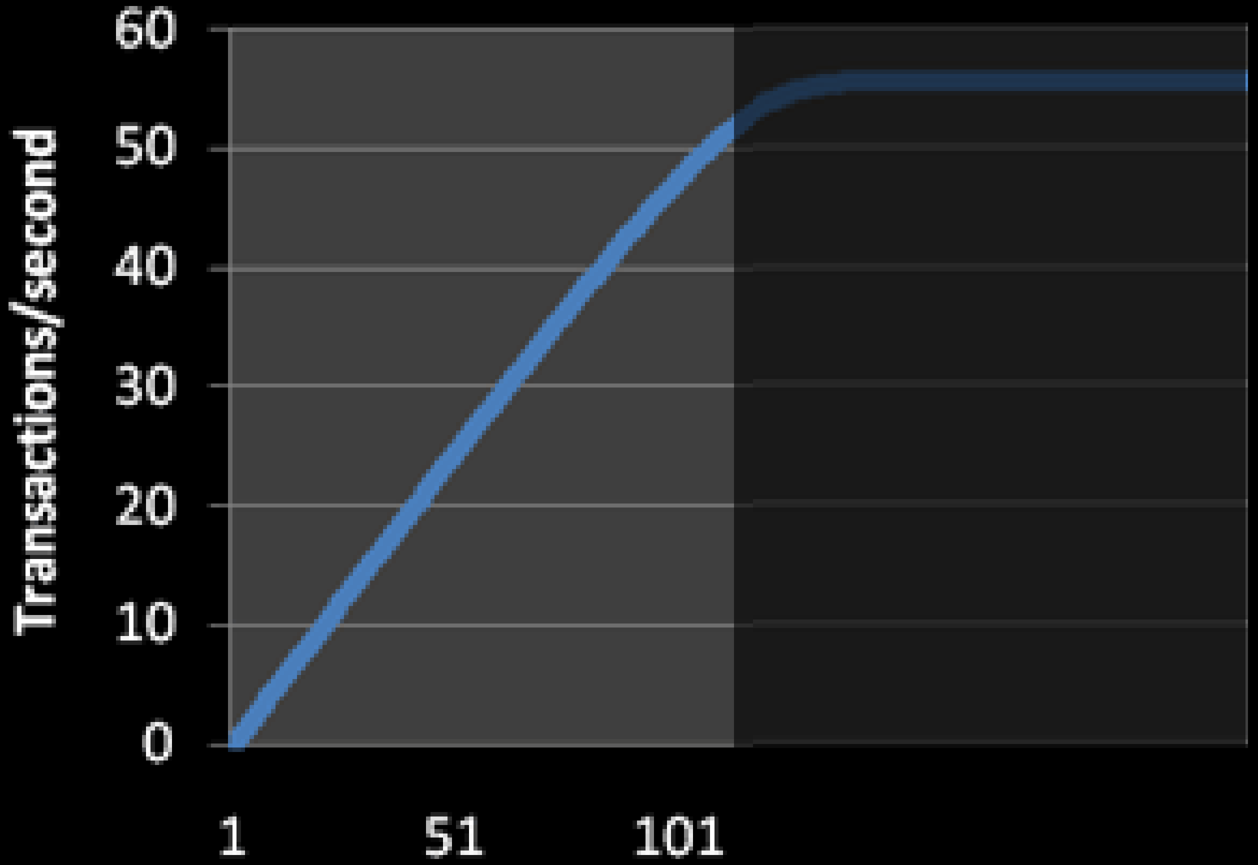
Parallelism solves everything?



Third Nature

How vendors demonstrate "Linear Scalability"

This is the part of the chart most vendors show.



If you're lucky they leave the bottom axis on so you know where their system flatlines.

Third Nature

Parallelism

You need to coordinate transactions in a distributed environment. Coordination is the enemy of scale. Here's math.

What needs coordination?

Updates, inserts and deletes.

Hence ACID compliance.

But there are other ways.

Amdahl's Law

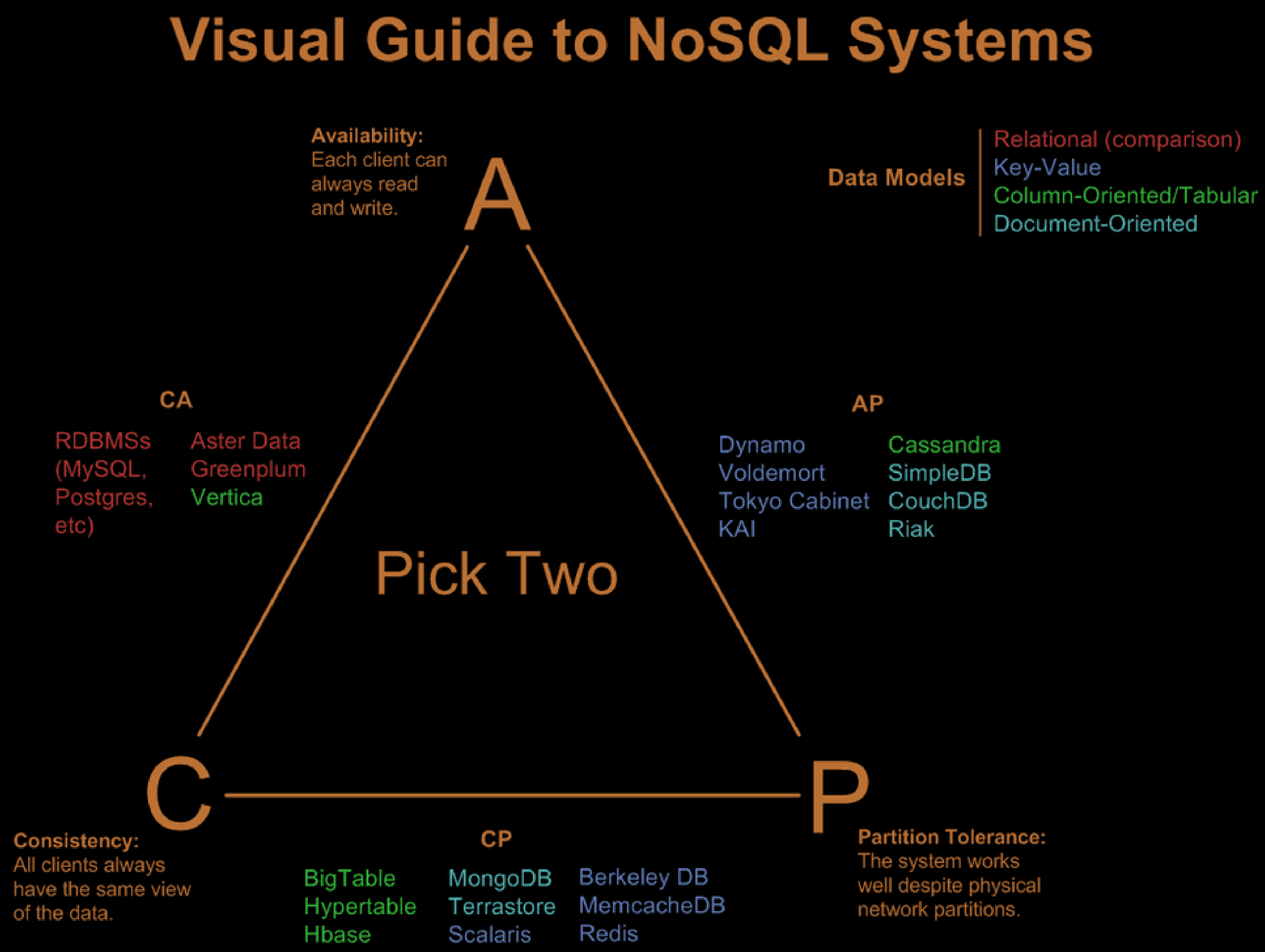
$$T_p = \left(\%S + \frac{1 - \%S}{N} \right) * T_s$$

$$Speedup = \frac{T_s}{T_p}$$

T p	Parallel runtime
Ts	Serial runtime
%S	Percentage of time spent in serial code
N	Number of processors

Third Nature

The CAP theorem and ACID vs BASE



http://blog.nahurst.com/visual-guide-to-nosql-systems

Simplifying ACID vs BASE

MT.GOX

Eventually consistent is a nice way of saying “not correct”

Trade with confidence on the world's largest Bitcoin exchange!

Mt.Gox is the world's most established Bitcoin exchange. You can quickly and securely trade bitcoins with other people around the world with your local currency!

“transaction malleability” is a nice way of saying “broken”

SIGN UP NOW

Remember: it’s a poor carpenter who blames his tools.

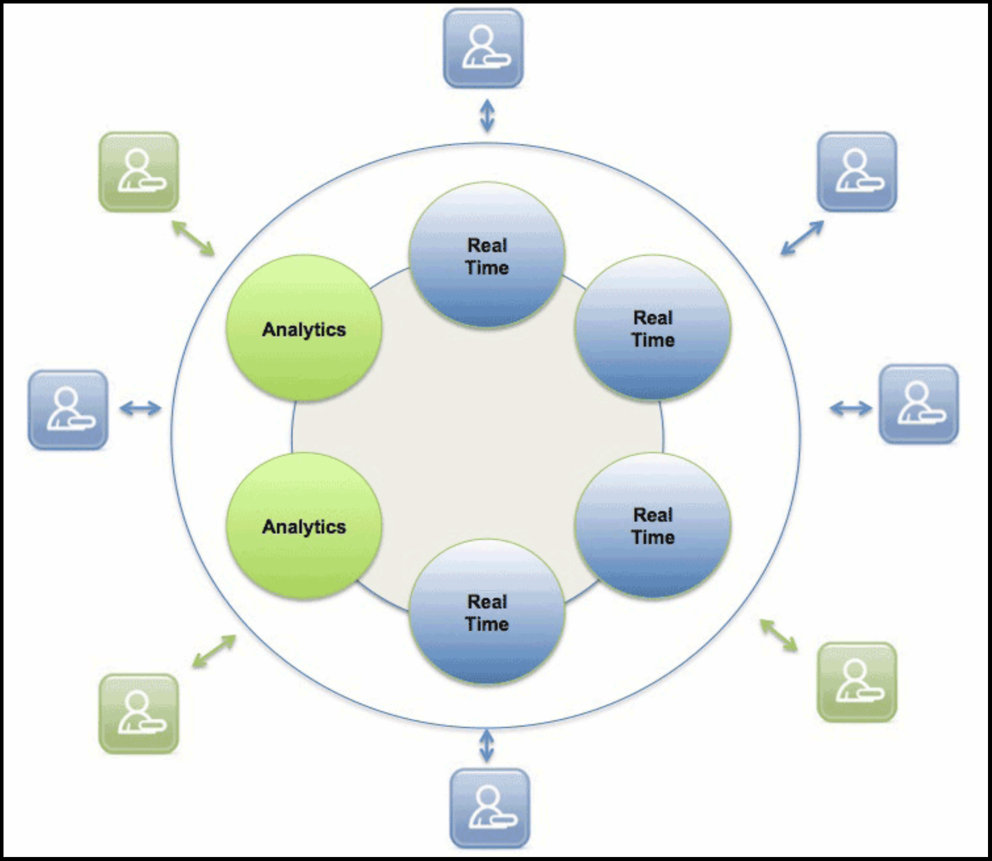
90% of

EVERYTHING

is **crap**

Eventual consistency: a little bird told me…

Let’s analyze some data with long-running queries



Remember: it’s a poor carpenter who blames his tools.

Third Nature

Why doesn't your database scale?

Just add hardware?

No amount of hardware will make incorrectly coded software run in parallel.

Declarative languages make this easier by turning the problem over to the computer to resolve.

Guess which runs in parallel:

```
Open cursor
Loop
  Fetch row
  Do-things
  Join table 2 row
  Insert result
End loop
```

```
INSERT INTO table (
  SELECT do-things(cols)
  FROM table2, table
  WHERE x=y )
```


Third Nature

90% of

EVERYTHING

is **crap**

Third Nature

Hipster bullshit

I can't get MySQL to scale

therefore

Relational databases don't scale

therefore

We must use NoSQL* for everything

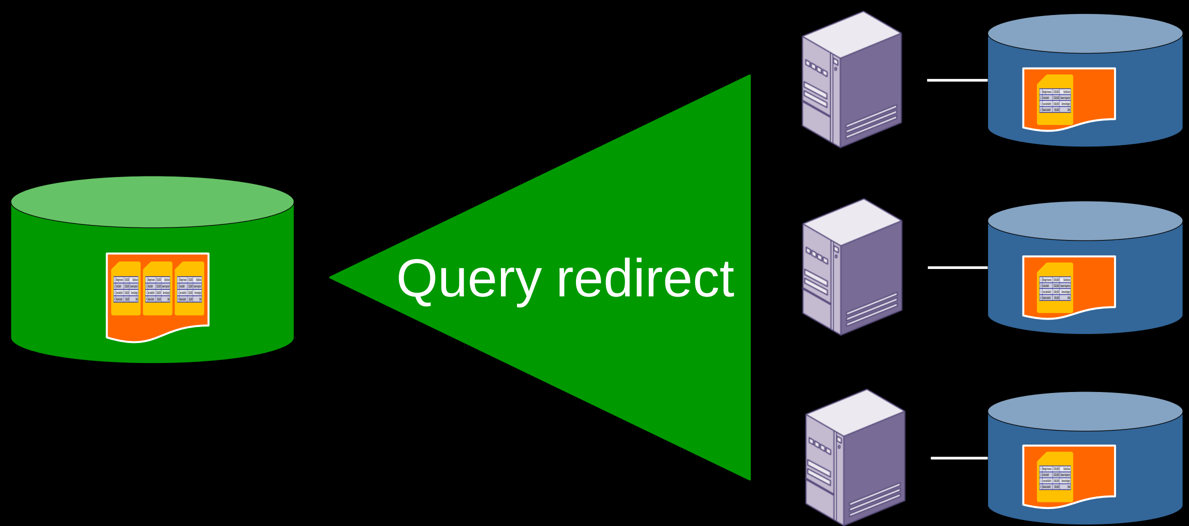
**including Hadoop and related*

Sharding, Making Mess in One Easy Step

Sharding is basically partitioning applied across multiple database servers, faking a distributed DB.

Each node holds an independent and (hopefully) self-consistent portion of the database.

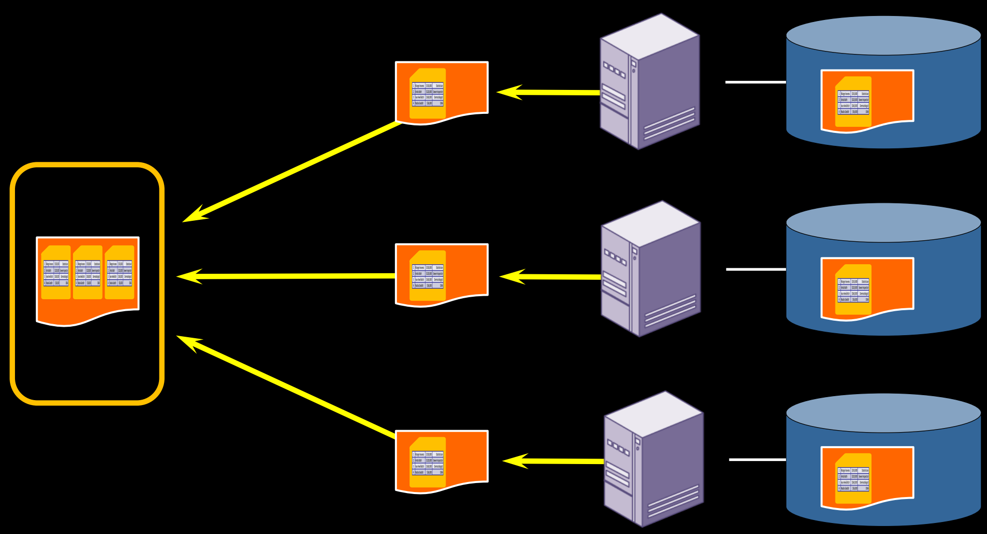
Good as long as queried data lives on a single node.



One large database is carved into several smaller databases

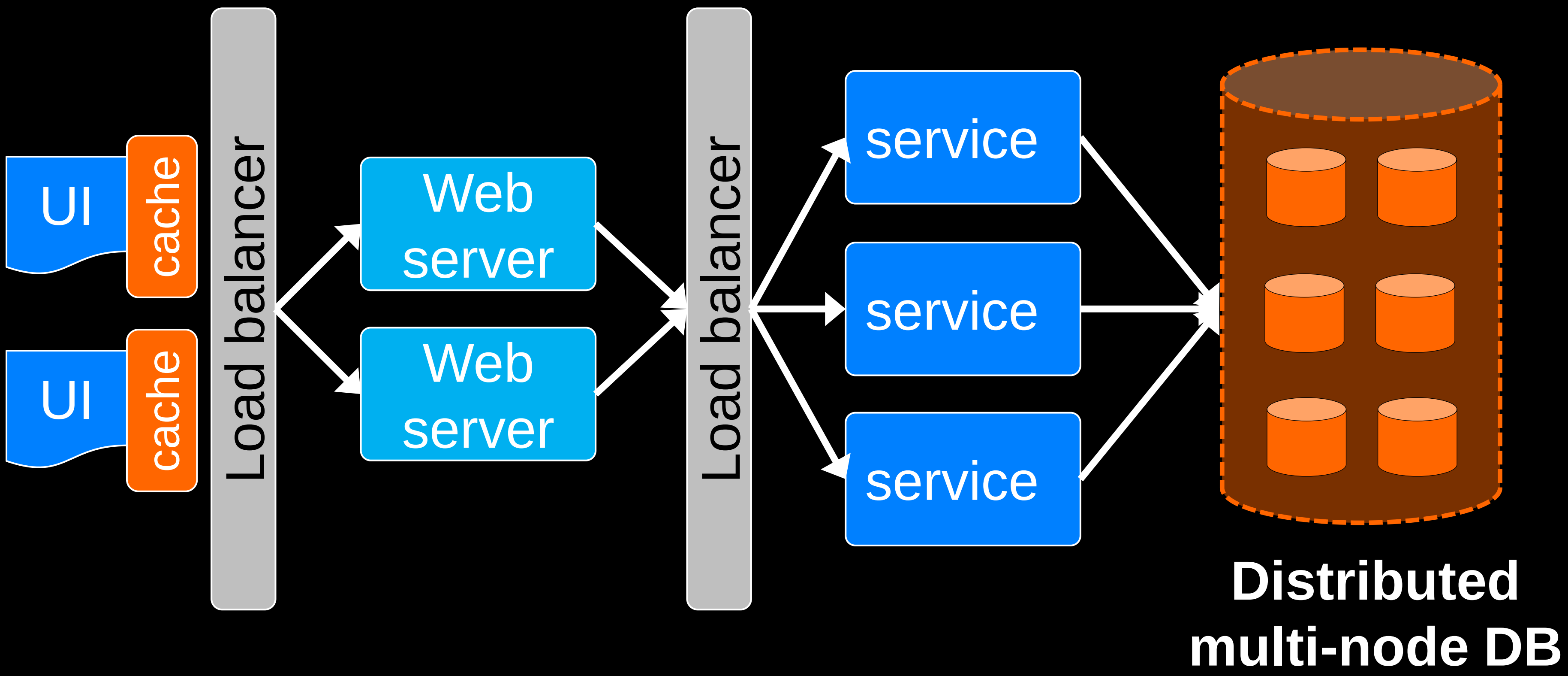
Sharding, Databases and Queries

What happens when you need to scan a full table or join tables across nodes? Multiple queries and stitching at the application level.



Sharding works well for fixed access paths, uniform query plans, and data sets that can be isolated. Mainly this describes an OLTP-style workload.

What application is this hardware and database topology designed for?



“In pioneer days they used oxen for heavy pulling, and when one ox couldn't budge a log, they didn't try to grow a larger ox. We shouldn't be trying for bigger computers, but for more systems of computers.”

Grace Hopper

What is the nature of the workloads?

Two workloads, two not dissimilar architectures:

- Load-balanced front ends
- Distributed caching layers
- Scalable distributed parallel databases

The nature of the OLTP and BI workloads is very different above the hardware and below the application. This I where the moving parts are.

Forcing them into one platform is almost impossible at scale*

Third Nature

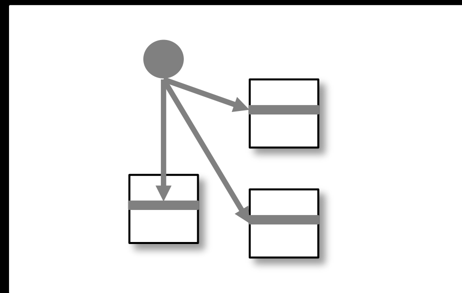
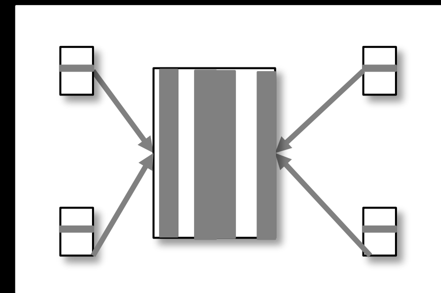
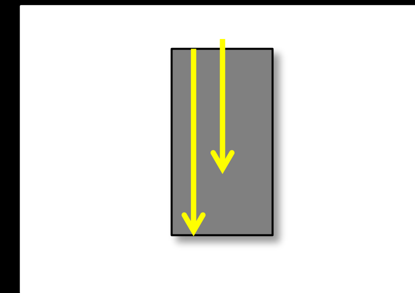
A key point worth remembering:

Performance over size <> performance over complexity

OLTP performance is mostly related to transaction coordination challenges under high concurrency.

BI performance is mostly related to data volume and query complexity.

Analytics performance is about the intersection of these with computational complexity.

Workloads	OLTP	BI	Analytics
			
Access	Read-Write	Read-only	Read-mostly
Predictability	Fixed path	Unpredictable	All data
Selectivity	High	Low	Low
Retrieval	Low	Low	High
Latency	Milliseconds	<seconds	msecs to days
Concurrency	Huge	Moderate	1 to huge
Model	3NF, nested object	Dim, denorm	BWT
Task sizw	Small	Large	Small to huge


The big change in the IT market isn’t technology, it’s architecture.

We are in a transitional phase in IT architecture

	Then	State of Practice	Now, forward
Architecture	Timeshare	Client/server	Cloud
Data	Core TXs	All TXs, some events	All data
Rate of change	Slow	Rapid	Continuous
Uses	Few	Many	Everything
Latency	Daily+++	< daily to minutes	Immediate
Data platform	Uniprocessor	SMP, cluster	Shared nothing

How did we get here?

There's a difference between having no past and actively rejecting it.

Third Nature

1960s

MultiValue, Hierarchical
PICK
IMS
IDS
ADABAS

1970s

Relational
CODASYL
System R (SEQUEL)
SQL/DS
INGRES (QUEL)
Mimer
Oracle

1980s

RDBMS, SQL standard
DB2
Teradata
Informix
Sybase
Postgres

1990s

OODBMS, ORDBMS
Versant
Objectivity
Gemstone
Informix*
Oracle*

2000s

MPP Query, NoSQL
Netezza
Paraccel
Vertica
MongoDB
CouchBase
Riak
Cassandra

2010s

NewSQL
SciDB
MonetDB
NuoDB
CitusDB

Spanner
F1

In the beginning: RMSs and pre-relational DBs

At first, common code libraries so there was reusability for file ops.

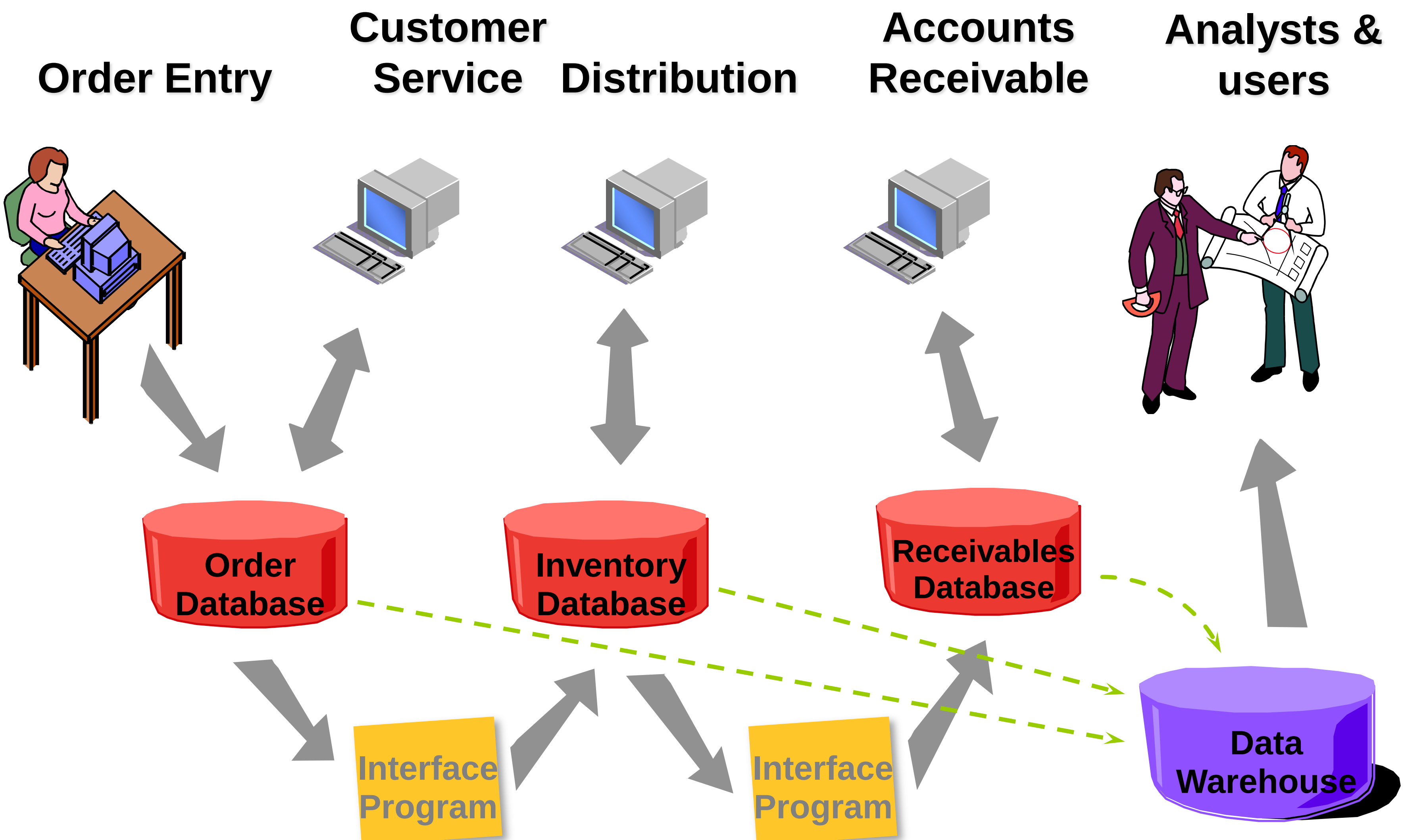
Problems:

- Portability across languages, OSs
- Queries of more than one file
- Concurrency
- No metadata, what’s in there? Who wrote it?

The databases brought things like recoverability, durability, ACID transactions. But tey were rigid, prone to breakage.

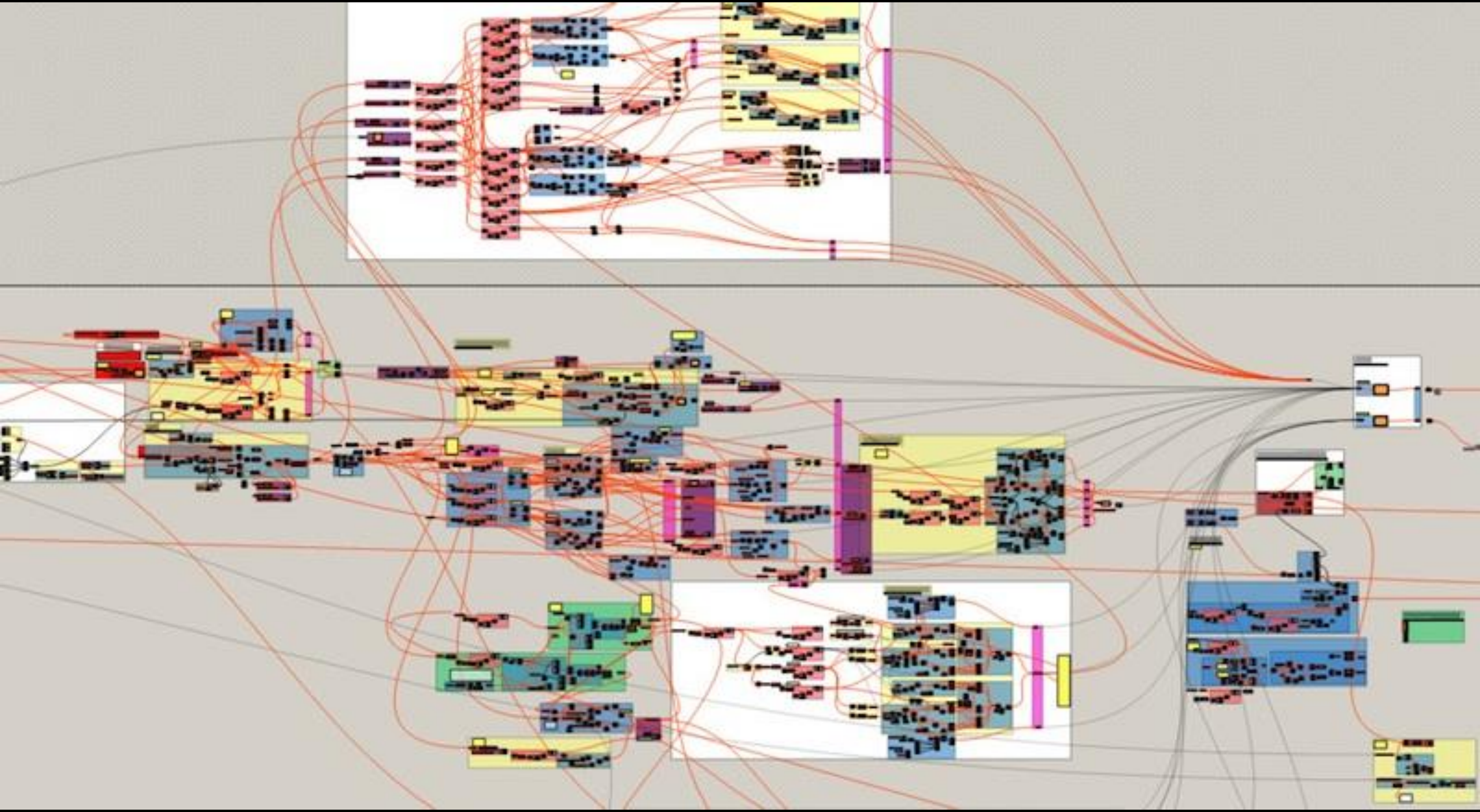
Third Nature

Someone else always wants to use your data



Third Nature

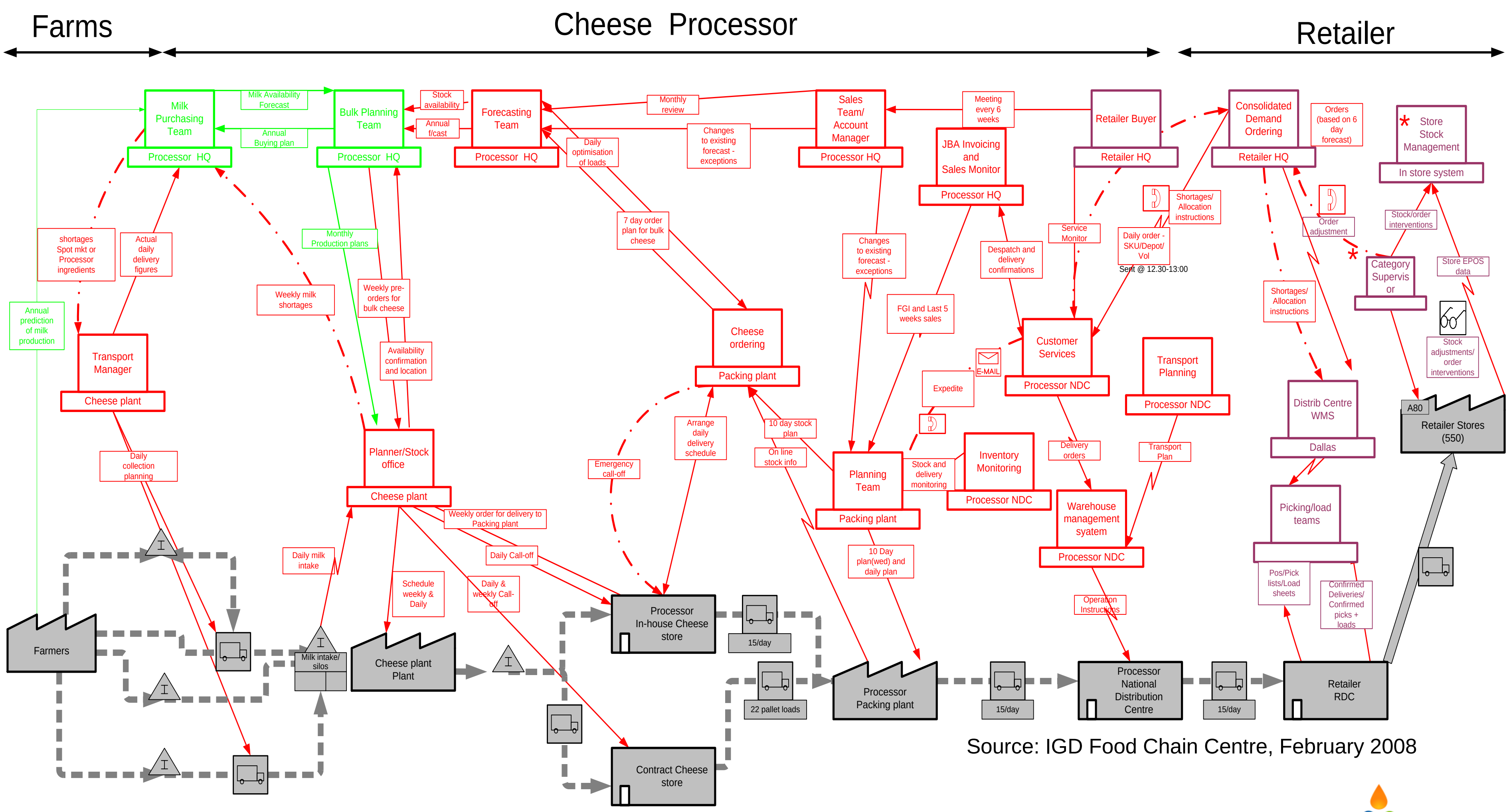
Context (one company)



"In an infinite universe, the one thing sentient life cannot afford to have is a sense of proportion." – *Douglas Adams*

Context (multiple company supply chain)

A value chain diagram, showing the data supply chain for cheese. The side effects of a single bug can be massive.



Source: IGD Food Chain Centre, February 2008

Third Nature

Centralizing ERP & vendors didn't solve the context problem; only increased cohesion. Now data lakes?

The miracle of pre-relational DB: schema

Loose coupling – the physical model of data structures and physical placement are no longer a program's responsibility; data portability ensues.

Reusability – More than one program can access the same data, and no more custom coding for each application or OS

Scalability – Constraints of schema and typing reduce resource usage, have finer granularity for concurrent access, multiple online users.

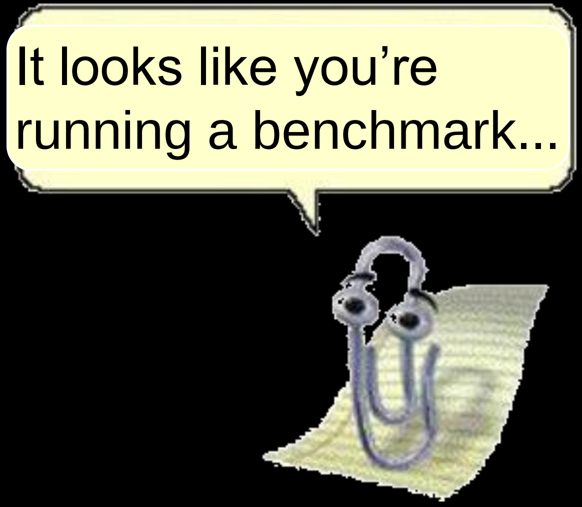
Third Nature

Party like it's 1985

Fastest TPC-B benchmark in 1985 was IMS running on an IBM 370, 100 TPS, 400 iops, 30 iops/disk

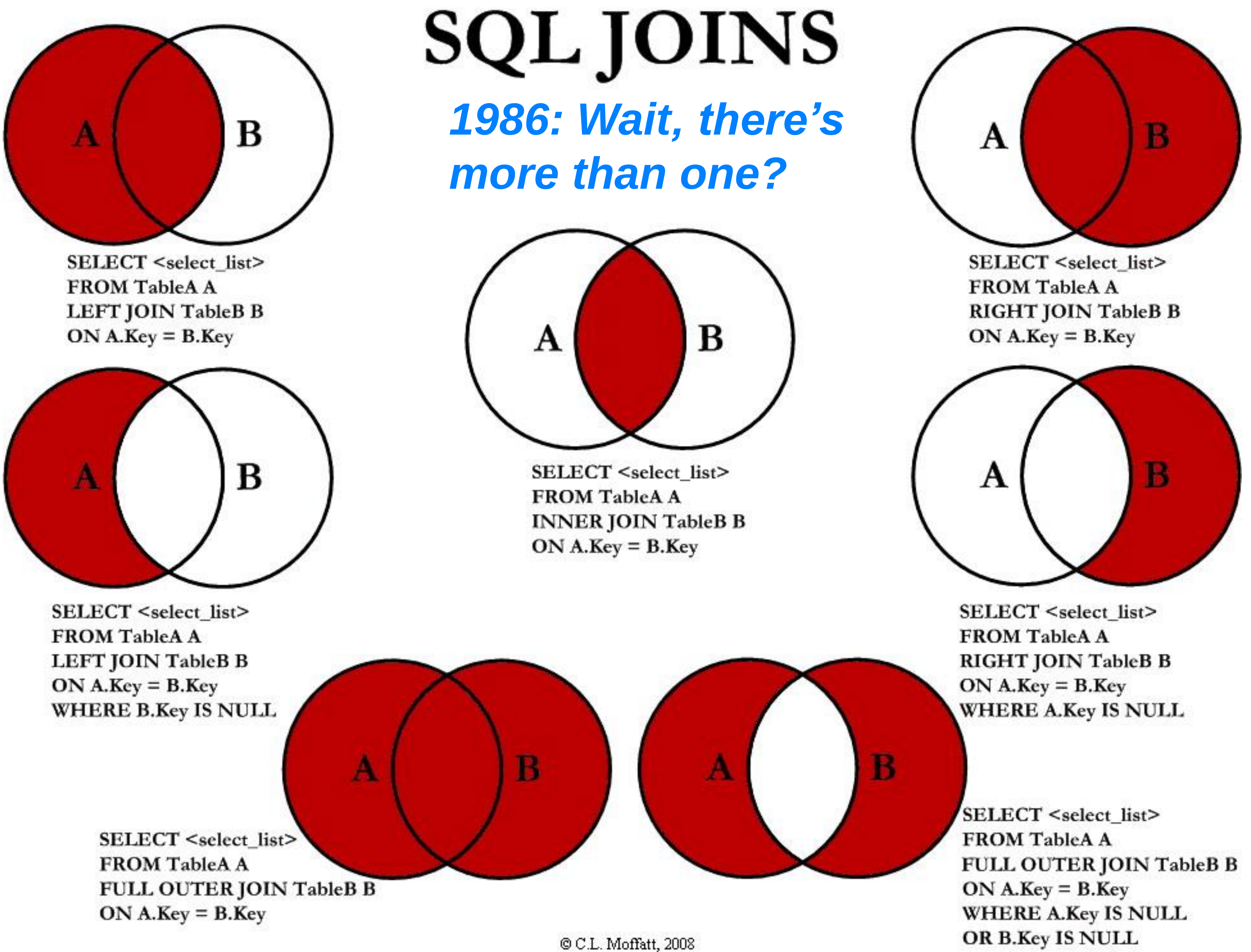
The best relational vendors could muster was 10 TPS

25 years later, SQLServer on an Intel box ran the TPC-B at 25,000 TPS, 100,000 iops, 300 iops/disk



Third Nature

1986: Wait, there’s more than one?



What the optimizer does

It turns a SQL query into an optimal* execution plan for a parallel pipelined dataflow engine

1. Enumerate logically equivalent plans by applying equivalence rules
2. For each logically equivalent plan, enumerate all alternative physical query plans
3. Estimate the cost of each of the alternative physical query plans
4. Run the plan with lowest estimated overall cost

Diagram: David J. DeWitt

A simple 3 table join

```
SELECT C.name, O.num
FROM Orders O, Lines L, Customers C
WHERE C.City = “Copenhagen” AND L.status = “X”
      AND O.num = L.num AND C.cid = O.cid
```

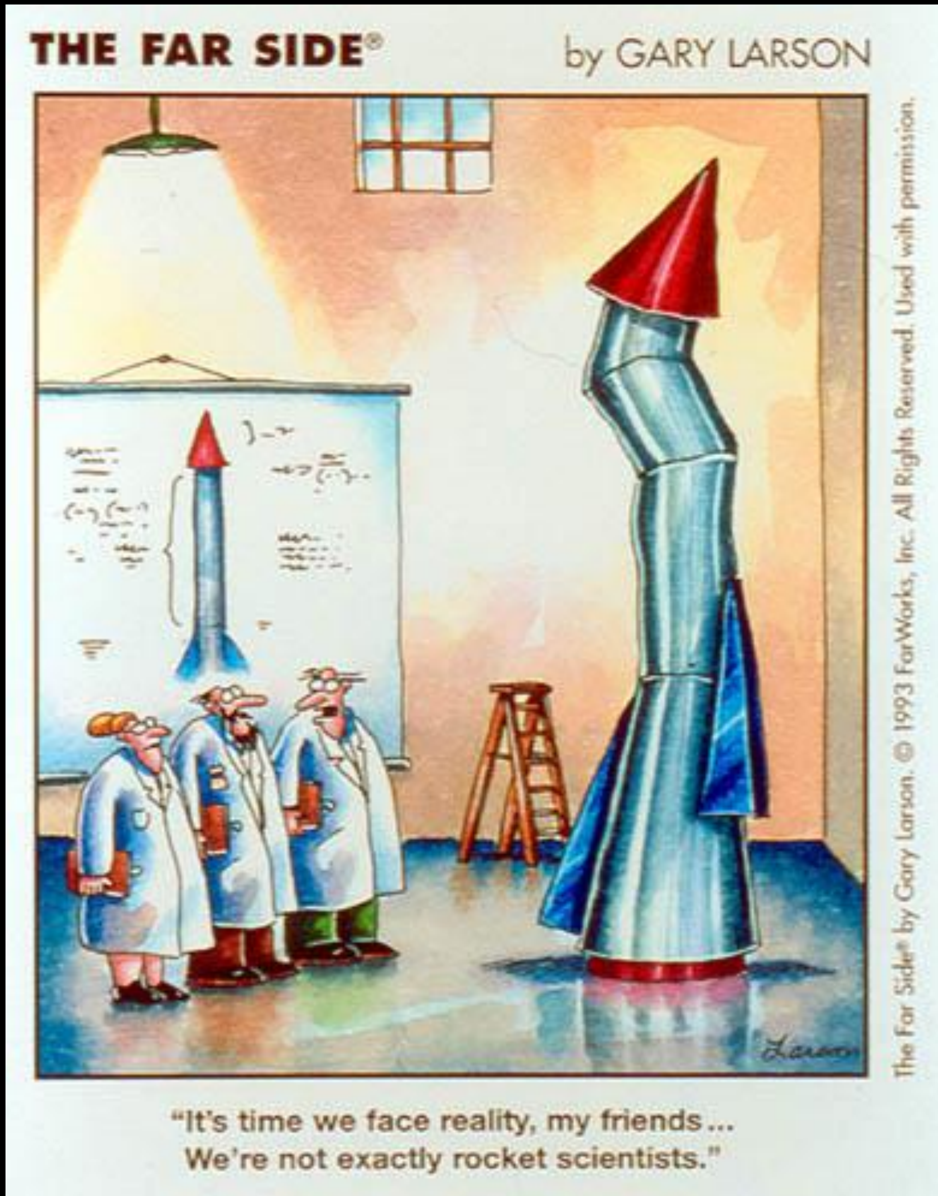
Number of logical plans: 9

Ways to join (hash, merge, nested): 3

For each plan, there are multiple physical plans: 36

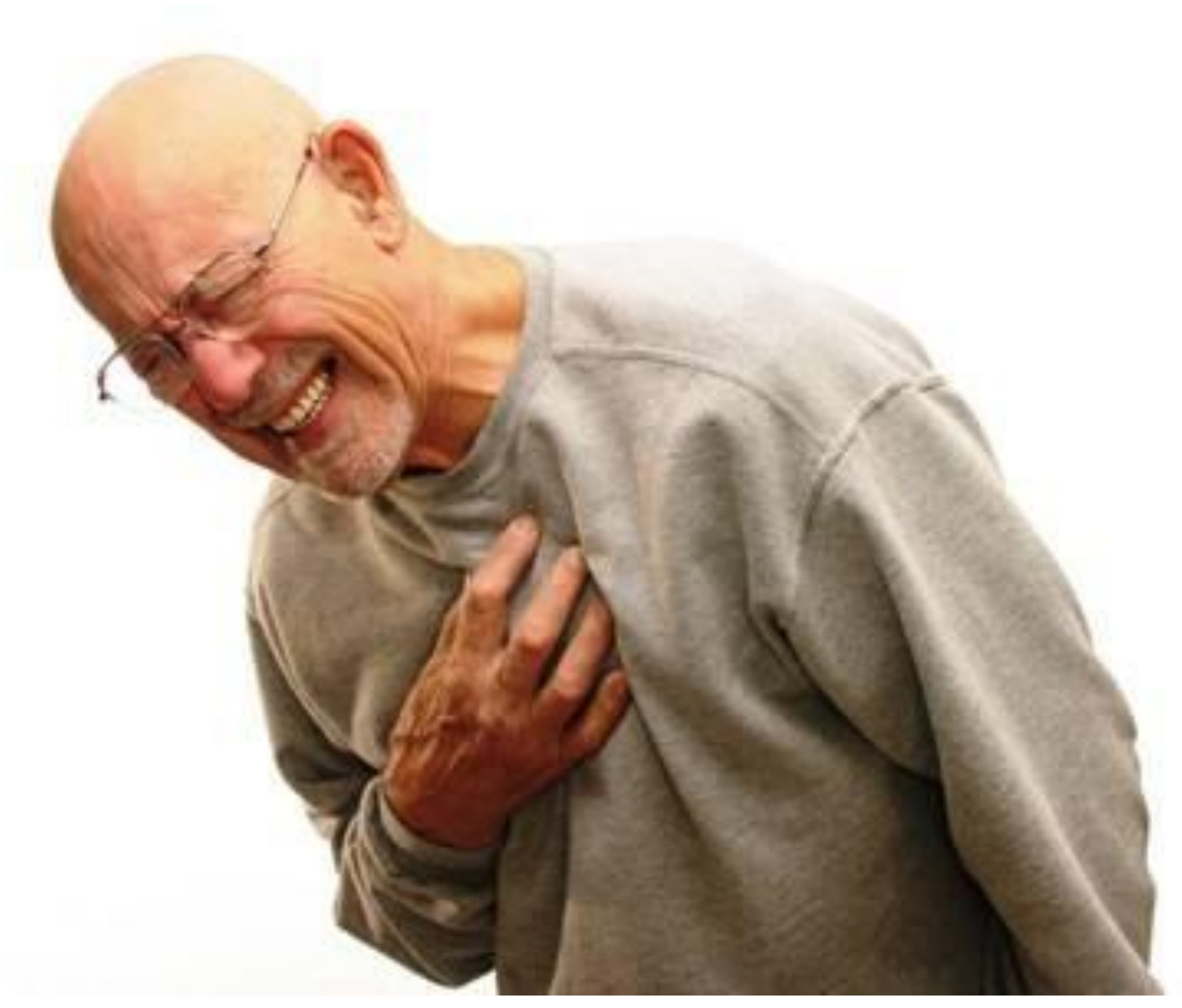
That makes a total of 324 physical plans, the efficiency of which changes based on cardinality.

NoSQL tradeoffs?



"Query optimization is not rocket science. When you flunk out of query optimization, we make you go build rockets."

In NoSQL Land, Optimizer is You!



You did review each plan in your MapReduce job, right?.

Third Nature

Tradeoffs: In NoSQL the DBMS is in your code

SQL database

NoSQL database

Application

Database

Services provided

- Standard API/query layer*
- Transaction / consistency
- Query optimization
- Data navigation, joins
- Data access
- Storage management

Application

Database

Anything not done by the DB becomes a developer's task.

Copyright Third Nature, Inc.

Relational: a good conceptual model, but a prematurely standardized implementation

The relational database is the franchise technology for storing and retrieving data, but…

1. Global, static schema model
2. No rich typing system
3. No management of natural ordering in data
4. Pretends to separate logical and physical schema, <u>but it's partial</u>
5. Limited API in atomic SQL statement syntax & simple result set return
6. Poor developer support (in languages, in IDEs, in processes)

Relational: a good conceptual model, but a prematurely standardized implementation

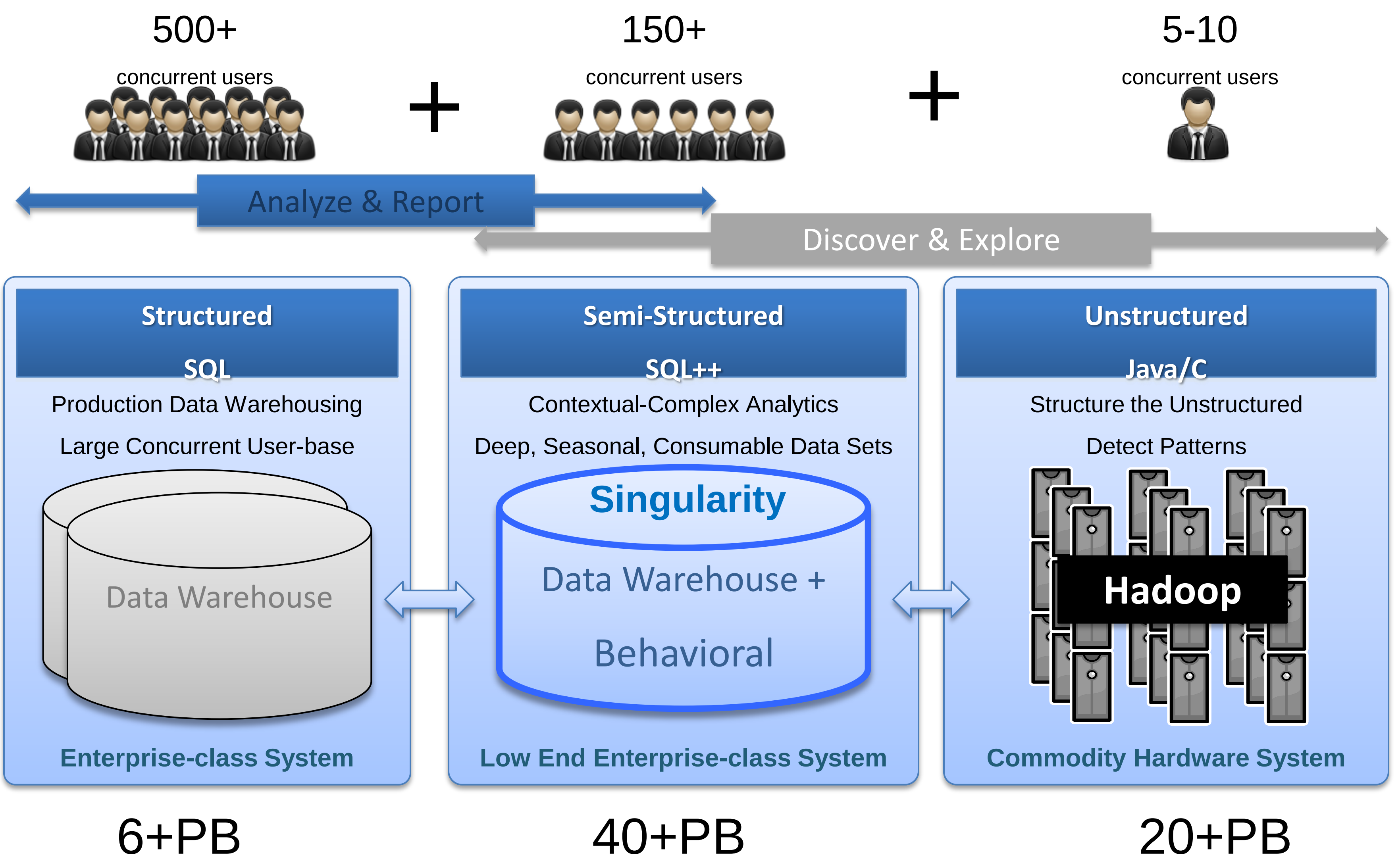
What I did not list:
Scalability and performance

The relational database is the franchise technology for storing and retrieving data, but…

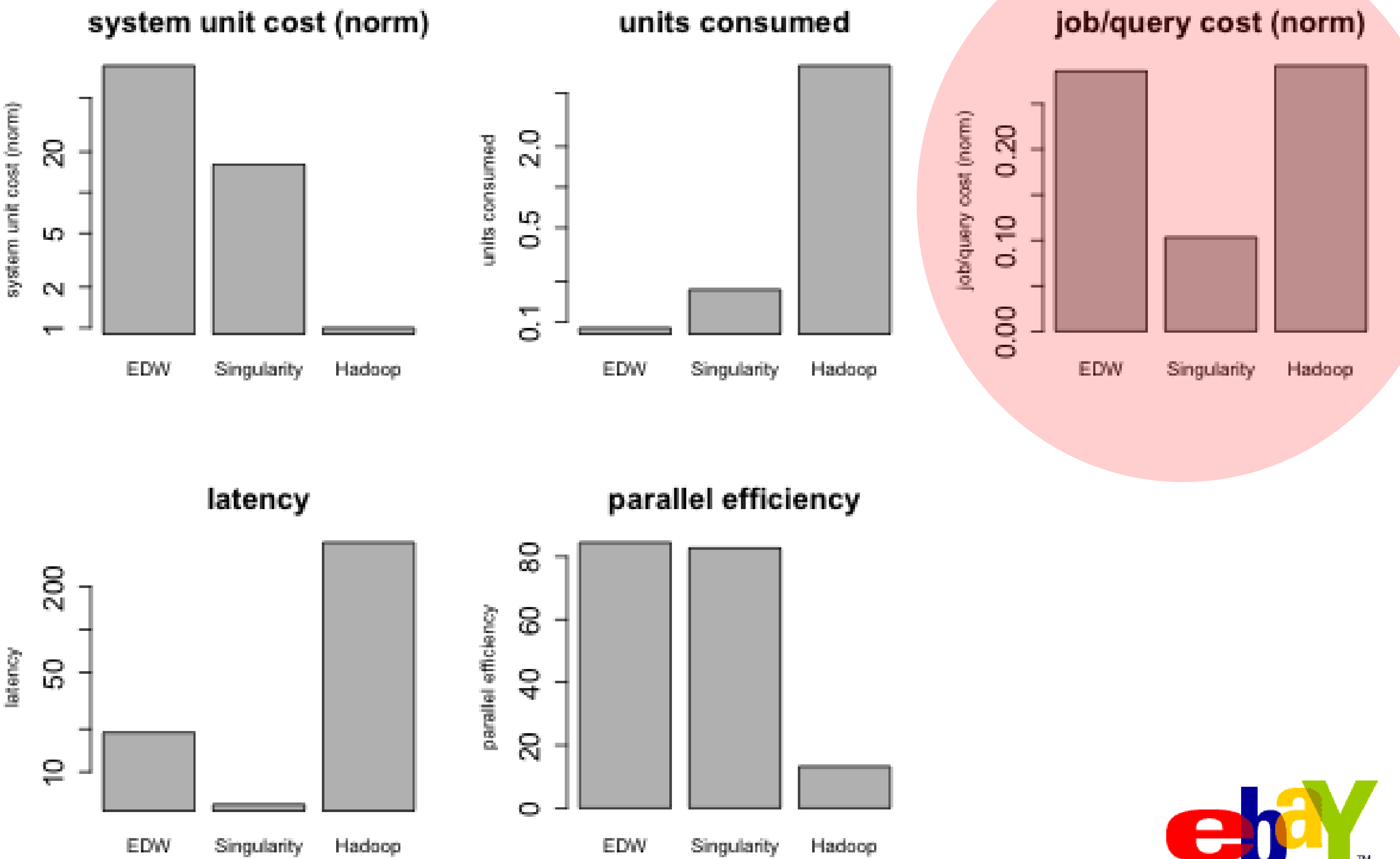
1. Global, static schema model
2. No rich typing system
3. No management of natural ordering in data
4. Many are not a good fit for network parallel computing, aka cloud
5. Limited API in atomic SQL statement syntax & simple result set return
6. Poor developer support

Third Nature

Parallel Efficiency and Platform Costs



Platform Metrics for Table Scan and Sum, Hadoop vs Teradata



Relational schema inflexibility

Change my-crappy-code-with-select-* everywhere in it:

ALTER TABLE really_big

ADD COLUMN omg_wtf

I meant…

create new table, back up old table, wait, load new table, wait 4ever, drop new table, recreate new table with new column default, reload new table, wait 4ever

On each shard...

The Developer View of DBAs

The DBA view of developers

Third Nature

Flexibility – an experience in query

The problem with many of these databases is tight coupling between a program and data structures.

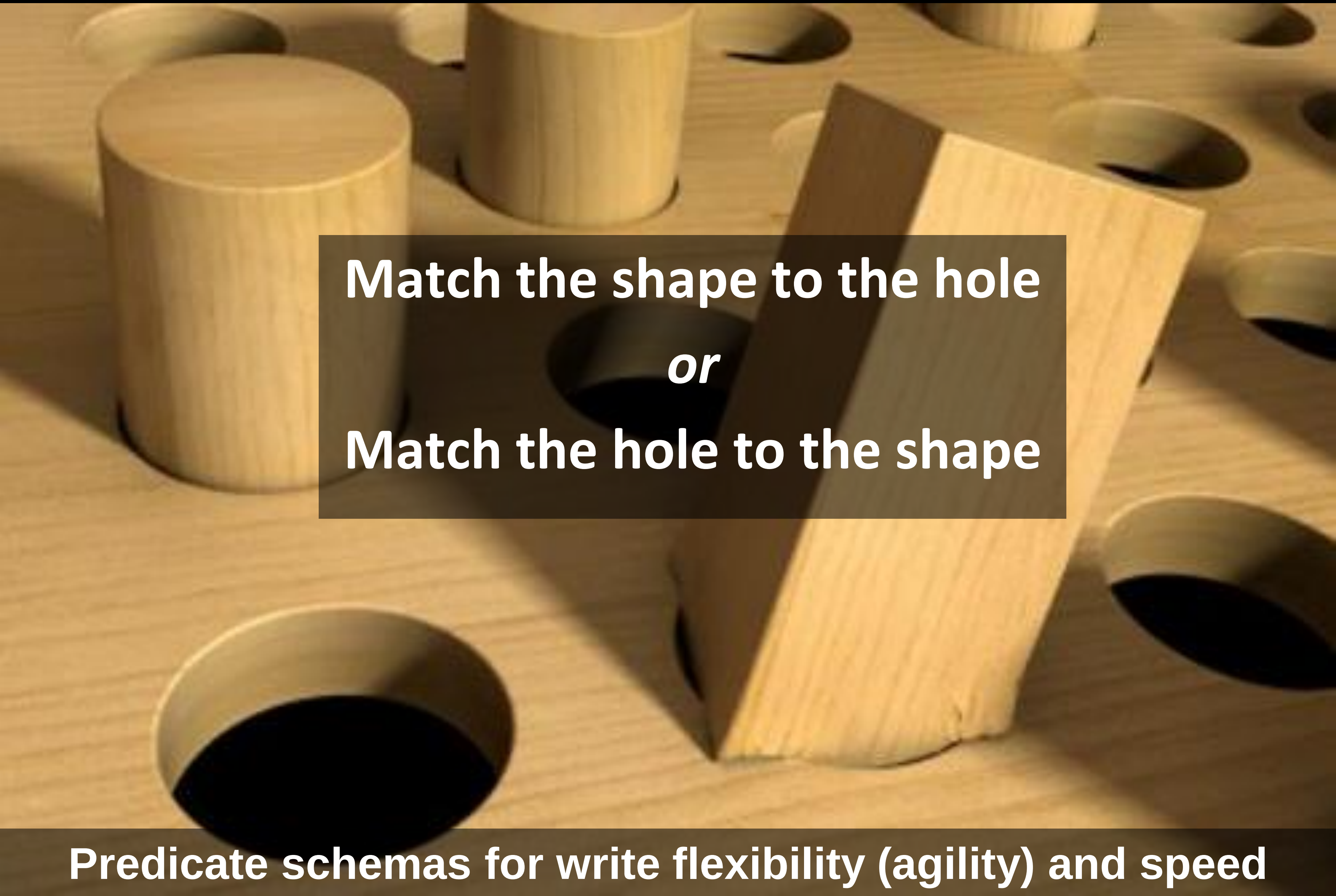
The physical model leaks into the logical with potentially career-ending effects if the DB is used for the wrong thing.

mongoDB

It's a poor carpenter who blames his tools. Or the users.

Third Nature

Schema on write vs schema on read



Match the shape to the hole

or

Match the hole to the shape

Predicate schemas for write flexibility (agility) and speed

Key schema flexibility tradeoff for data management

Global validation
vs
contextual
validation
=

Strict rules
vs
lenient rules
=

Write rules
vs
read rules

When to use implicit schema?

Use **implicit** when:

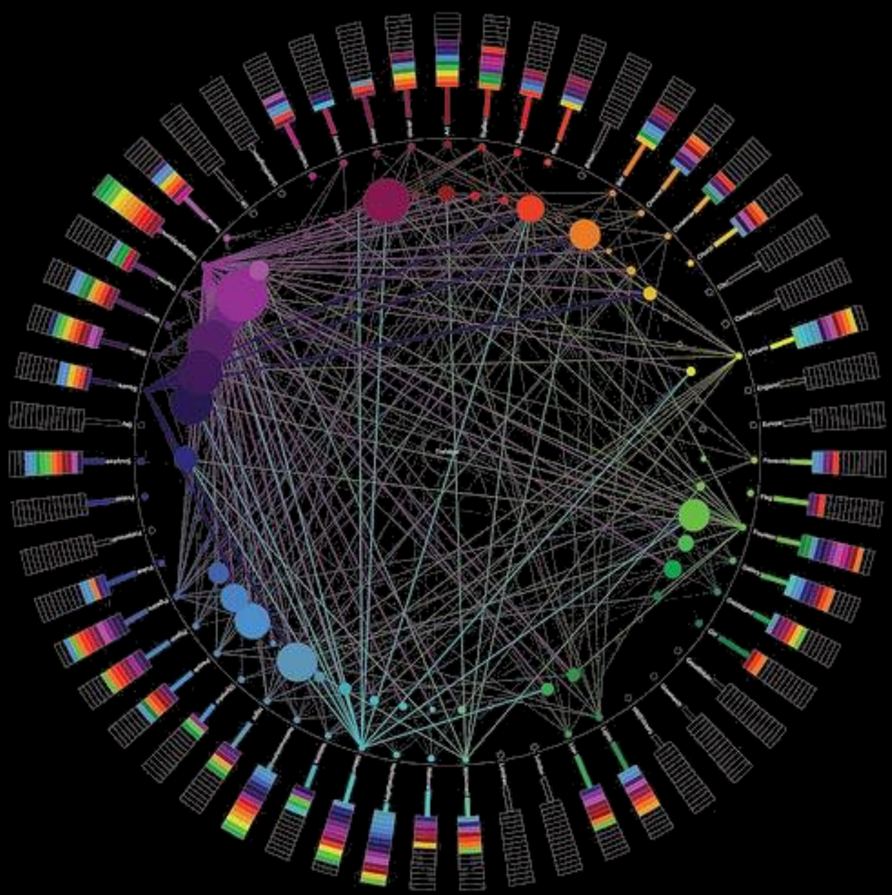
- You can hide the persistence of your data behind a service
- Nobody will ever want access to that data except you
- When data dies with the code
- You need to write data at a very high rate
- Your data sources change or are variable

Use **explicit** when:

- you need to send data to another application
- when more than one application (or person) needs to use data
- when data lives longer than your code
- When the data is regular
- When the sources and structure do not change
- When querying is more important than writing

Third Nature

Unstructured is Not Really Unstructured



Unstructured data isn't really unstructured: language has structure. So do images, audio, video. They can contain traditional structured data elements. The problem is that the content is unmodeled.

Conclusion: a database must cope with more complex data structures, storage and processing.

Third Nature

TANSTAAFL

Technologies are not perfect replacements for one another. Often not better, only different.

When replacing the old with the new (or ignoring the new over the old) you make *tradeoffs*, and usually you won't see them for a long time.

There is no silver bulllet.

Third Nature

Unintended consequences

Away from “one throat to choke”, back to best of breed

Tight coupling leads to slow change. The market is not in the tight coupling phase

In a rapidly evolving market, componentized architectures, modularity and loose coupling are favorable over monolithic stacks, single-vendor architectures and tight coupling.

Think like an architect, not like a consumer

The technology providers are selling you what *they have*, not necessarily what *you need*.

Follow the goals of the business.

Translate the goals into capabilities needed and match those to the architecture required.

How we develop best practices: survival bias

We don't need best practices today, we need worst failures.

Third Nature

The big data revolution, more of an evolution

References (things worth reading on the way home)

A relational model for large shared data banks, Communications of the ACM, June, 1970, http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf

Column-Oriented Database Systems, Stavros Harizopoulos, Daniel Abadi, Peter Boncz, VLDB 2009 Tutorial http://cs-www.cs.yale.edu/homes/dna/talks/Column_Store_Tutorial_VLDB09.pdf

Nobody ever got fired for using Hadoop on a cluster, 1st International Workshop on Hot Topics in Cloud Data ProcessingApril 10, 2012, Bern, Switzerland.

A co-Relational Model of Data for Large Shared Data Banks, ACM Queue, 2012, http://queue.acm.org/detail.cfm?id=1961297

A query language for multidimensional arrays: design, implementation and optimization techniques, SIGMOD, 1996

Probabilistically Bounded Staleness for Practical Partial Quorums, Proceedings of the VLDB Endowment, Vol. 5, No. 8, http://vldb.org/pvldb/vol5/p776_peterbailis_vldb2012.pdf

“Amorphous Data-parallelism in Irregular Algorithms”, Keshav Pingali et al

MapReduce: Simplified Data Processing on Large Clusters, http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/en//archive/mapreduce-osdi04.pdf

Dremel: Interactive Analysis of Web-Scale Datasets, Proceedings of the VLDB Endowment, Vol. 3, No. 1, 2010 http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/en//pubs/archive/36632.pdf

Spanner: Google’s Globally-Distributed Database, SIGMOD, May, 2012, http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/es//archive/spanner-osdi2012.pdf

F1: A Distributed SQL Database That Scales, Proceedings of the VLDB Endowment, Vol. 6, No. 11, 2013, http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/en/us/pubs/archive/41344.pdf

Third Nature

About Third Nature

Third Nature is a research and consulting firm focused on new and emerging technology and practices in business intelligence, analytics and performance management. If your question is related to BI, analytics, information strategy and data then you‘re at the right place.

Our goal is to help companies take advantage of information-driven management practices and applications. We offer education, consulting and research services to support business and IT organizations as well as technology vendors.

We fill the gap between what the industry analyst firms cover and what IT needs. We specialize in product and technology analysis, so we look at emerging technologies and markets, evaluating technology and hw it is applied rather than vendor market positions.

CC Image Attributions

Thanks to the people who supplied the creative commons licensed images used in this presentation:

text composition - http://flickr.com/photos/candiedwomanire/60224567/
twitter_network_bw.jpg - http://www.flickr.com/photos/dr/2048034334/
round hole square peg - https://www.flickr.com/photos/epublicist/3546059144
glass_buildings.jpg - http://www.flickr.com/photos/erikvanhannen/547701721
CAP diagram - http://blog.nahurst.com/visual-guide-to-nosql-systems
refinery-hdr.jpg - http://www.flickr.com/photos/vermininc/2477872191/
refinery-night.jpg - http://www.flickr.com/photos/vermininc/2485448766/

Third Nature