Making Sense of NoSQL

Version 7.0

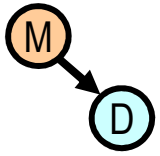
Dan McCreary

Tuesday November 19, 2013

8:30 AM to 11:30 AM

Description

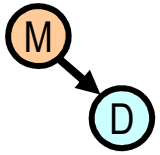
Many database architects are familiar with the "classic" database architecture patterns: the row-store used in SQL/RDBMS systems and star schemas used in analytical/OLAP systems. Now the NoSQL movement has introduced four new database architecture patterns to the enterprise solutions. This presentation will review these for new NoSQL database patterns: key-value stores, column-family stores, graph stores and document stores. We review the business drivers behind each pattern and the types of business problems they address. We then present some real-world case studies of how NoSQL database patterns are being used today. Finally we address the database selection process and the training challenges organizations face.



Session Goals

- Targeted at anyone that is new to the NoSQL movement
- Help you understand the business drivers behind NoSQL
- No program code

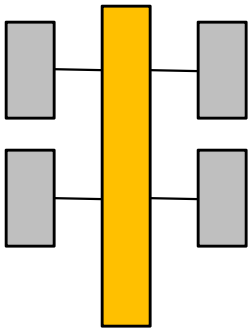
  ...but experience with databases helps
- Explain NoSQL "buzzwords" in context
  - NoSQL vs. "Big Data"
- Understand four new "NoSQL" Patterns
  - Understand what types of problems they solve
  - Key products and players
- Suggest an objective database selection process

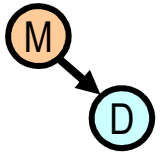


Before NoSQL

Relational

Analytical (OLAP)

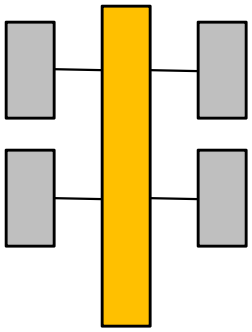




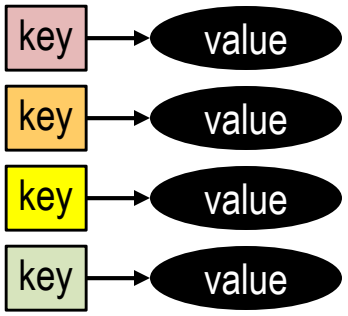
After NoSQL

Relational

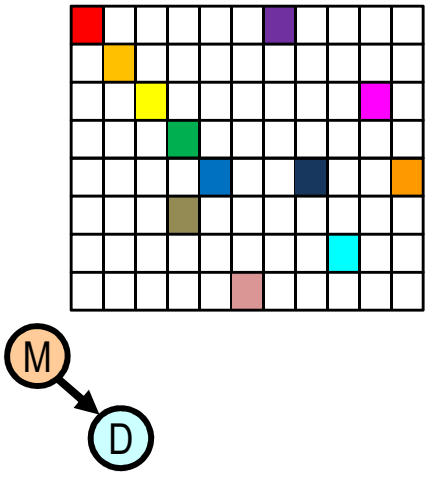
Analytical (OLAP)



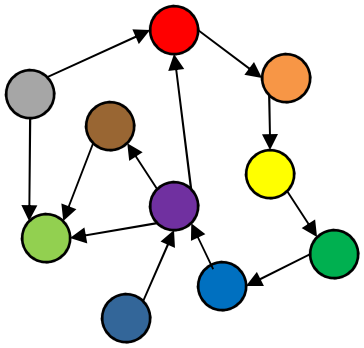
Key-Value



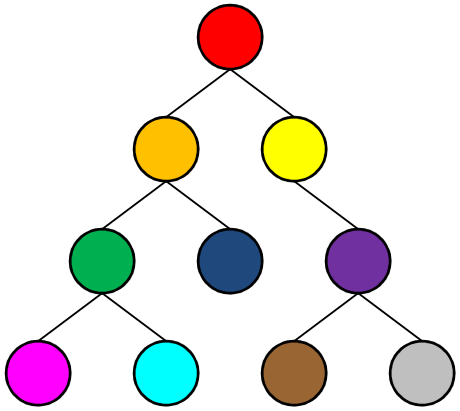
Column-Family



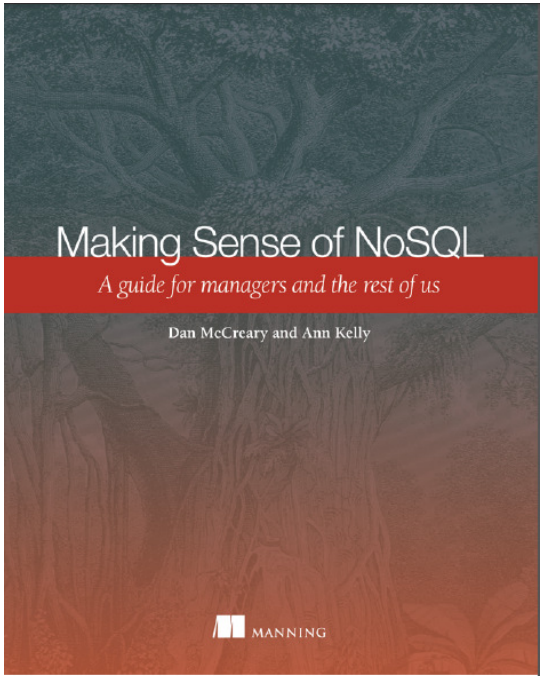
Graph



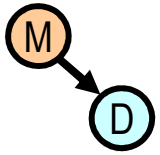
Document



Making Sense of NoSQL

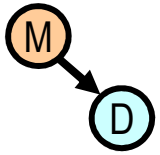


- Authors of Manning book on NoSQL (MEAP now, in print Aug 2013)
- Guide for managers and architects
- Focus on NoSQL architectural tradeoff analysis
- Basis for 40 hour course on database architectures
- http://manning.com/mccreary



Background for Dan McCreary

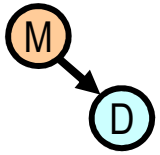
- Co-founder of the **NoSQL Now!** conference
- Background
  - Bell Labs
  - NeXT Computer (Steve Jobs)
  - Owner of 75-person software consulting firm
  - US Federal data integration (National Information Exchange Model NIEM.gov)
  - Native XML/XQuery for metadata management since 2006
  - Advocate of web standards, NoSQL and XRX systems



Food for thought…

- What percentage of database transactions run on **RDBMSs** in the following organizations?

  - What percentage of all transactions in **Iowa** run on RDBMSs?
  - Why is this number different?
  - Is our data fundamentally different?

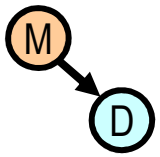


Sample of NoSQL Jargon

Document orientation
Schema free
MapReduce
Horizontal scaling
Sharding and auto-sharding
Brewer's CAP Theorem
Consistency
Reliability
Partition tolerance
Single-point-of-failure
Object-Relational mapping
Key-value stores
Column stores
Document-stores
Memcached

Indexing
B-Tree
Configurable durability
Documents for archives
Functional programming
Document Transformation
Document Indexing and Search
Alternate Query Languages
Aggregates
OLAP
XQuery
MDX
RDF
SPARQL
Architecture Tradeoff Modeling
ATAM
Erlang

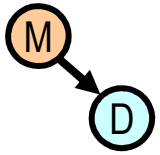
Note that within the context of NoSQL many of these terms have different meanings!



NoSQL – The Big Tent

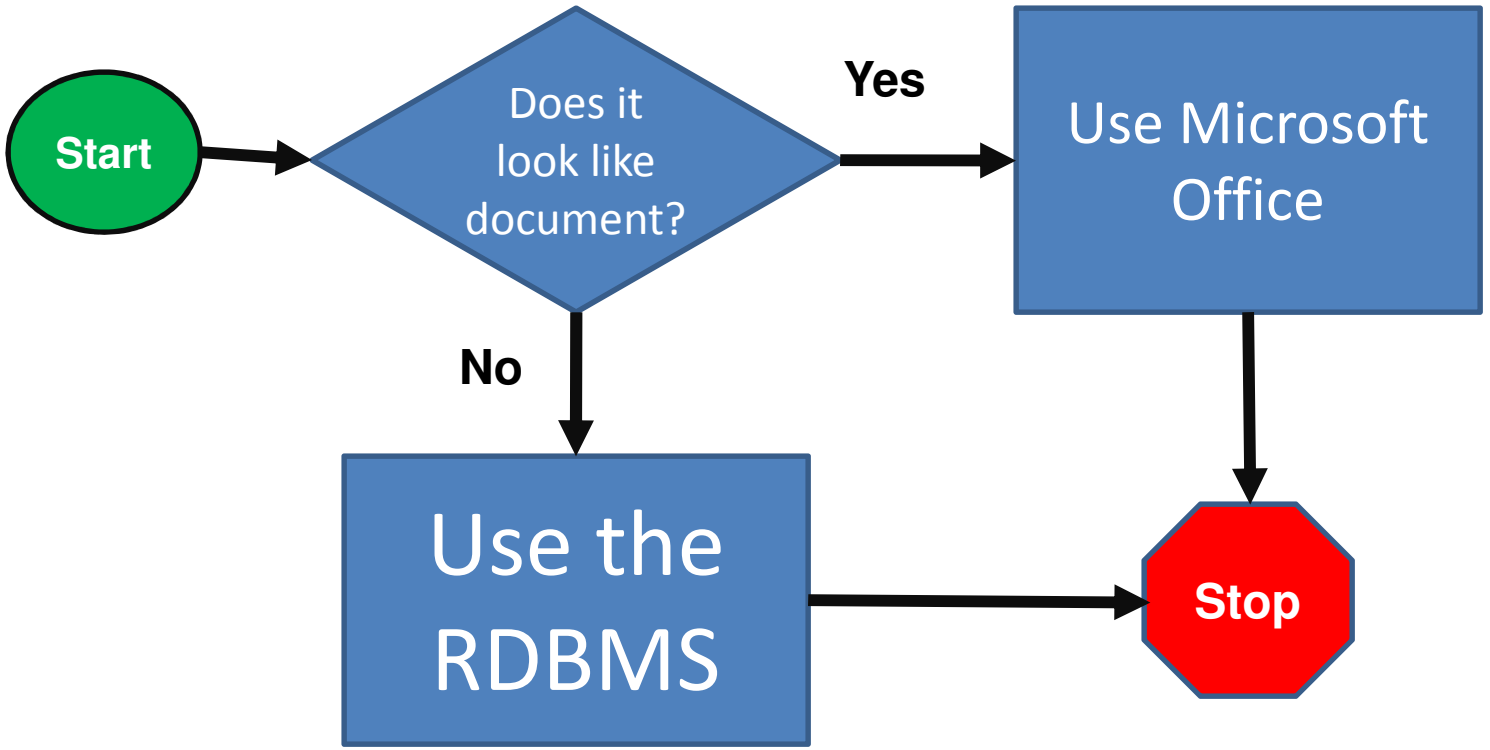
http://www.flickr.com/photos/morgennebel/2933723145/

- **"NoSQL"** – a label for a "meme" that now encompasses a large body of innovative ideas on data management
- "Not Only SQL"
- Focus on non-relational databases and hybrids
- A community where new ideas are quickly recombined to create innovative new business solutions

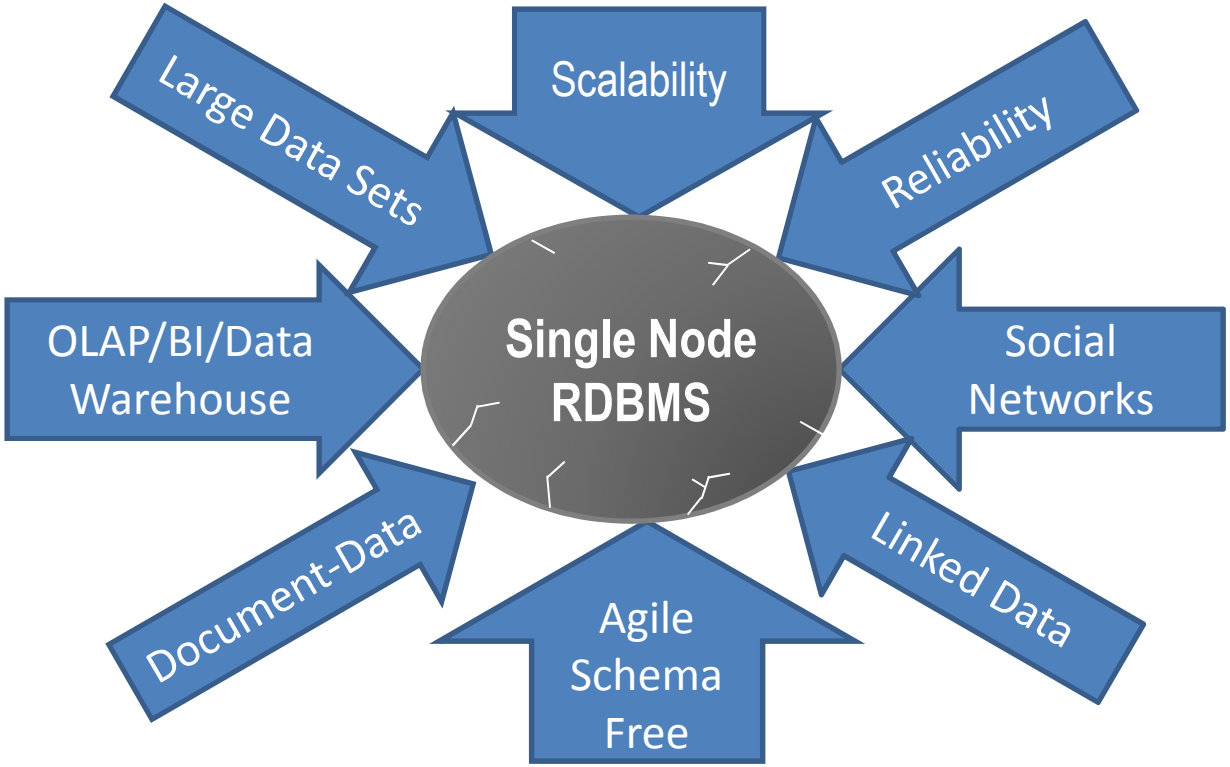


Database Selection in the Past…

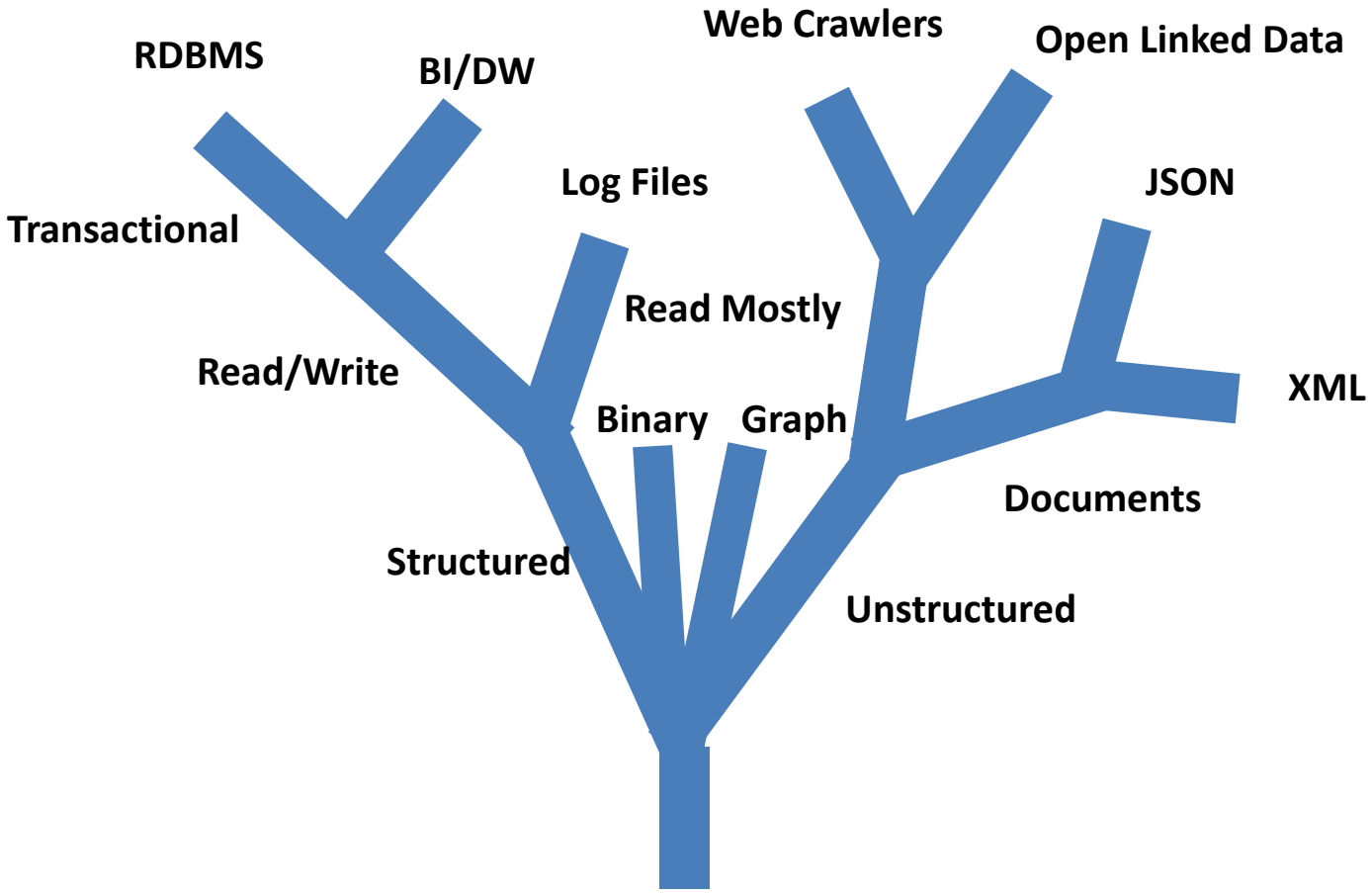
- *Selecting the right data management solution is no longer a trivial task*



Pressures on Single Node RDBMS Architectures

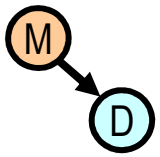


An evolving tree of data types

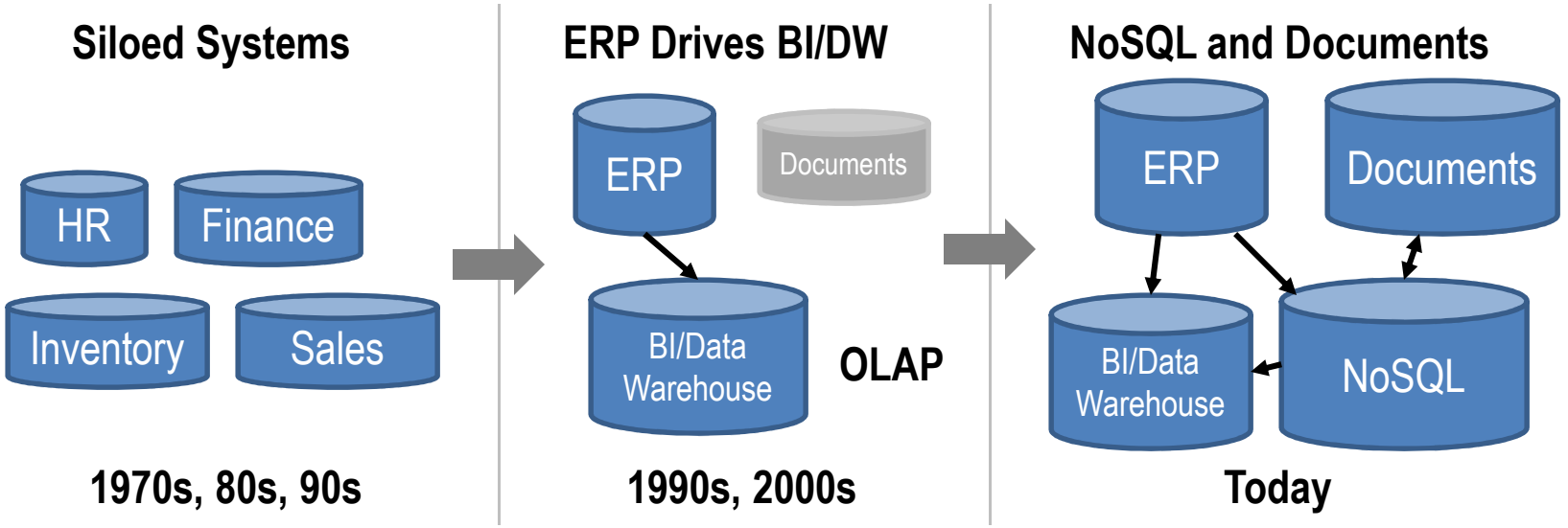


Many Uses of Data

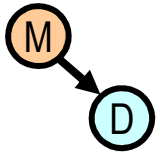
- Transactions (OLTP)
- Analysis (OLAP)
- Search and Findability
- Enterprise Agility
- Discovery and Insight
- Speed and Reliability
- Consistency and Availability



Three Eras of Enterprise Data



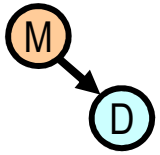
- NoSQL will not replace ERP or BI/DW systems – but they will complement them and also facilitate the integration of unstructured document data



Simplicity is a Virtue

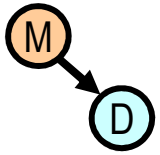
Photo from flickr by PSNZ Images

- Many modern systems derive their strength by dramatically limiting the features in their system and focus on a specific task
- Simplicity allows database designer to focus on the primary business drivers
- Simplicity promotes "separation of concerns"



Simplicity is a Design Style

- Focus only on simple systems that solve many problems in a flexible way
- Examples:
  - Touch screen interfaces
  - Key/Value data stores



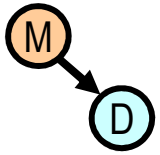
Historical Context

Mainframe Era

- 1 CPU
- COBOL and FORTRAN
- Punchcards and flat files
- $10,000 per CPU hour

Commodity Processors

- 10,000 CPUs
- Functional programming
- MapReduce "farms"
- Pennies per CPU hour



Two Approaches to Computation

1930s and 40s

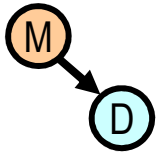
John von Neumann

Alonzo Church

*Manage **state** with a program counter.*

*Make computations act like math **functions**.*

Which is simpler? Which is cheaper? Which will scale to 10,000 CPUs?



Standard vs. MapReduce Prices

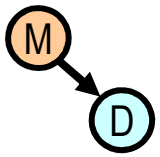
John's Way Alonzo's Way

Pricing for Amazon EC2 (On-Demand) and Amazon Elastic MapReduce

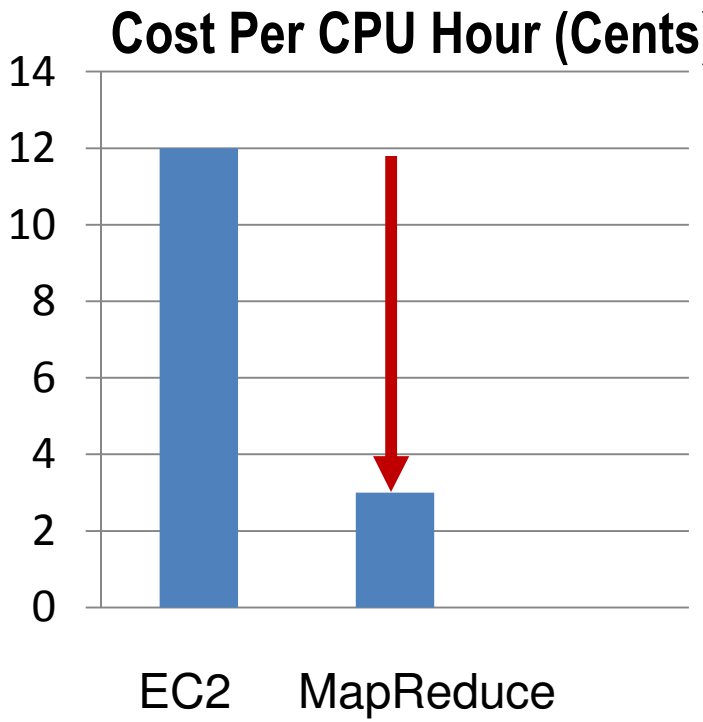
Region: US East (N. Virginia)

	Amazon EC2 Price	Amazon Elastic MapReduce Price
Standard On-Demand Instances		
Small (Default)	$0.06 per hour	$0.015 per hour
Medium	$0.12 per hour	$0.03 per hour
Large	$0.24 per hour	$0.06 per hour
Extra Large	$0.48 per hour	$0.12 per hour

http://aws.amazon.com/elasticmapreduce/#pricing

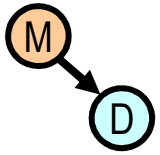


MapReduce CPUs Cost Less!



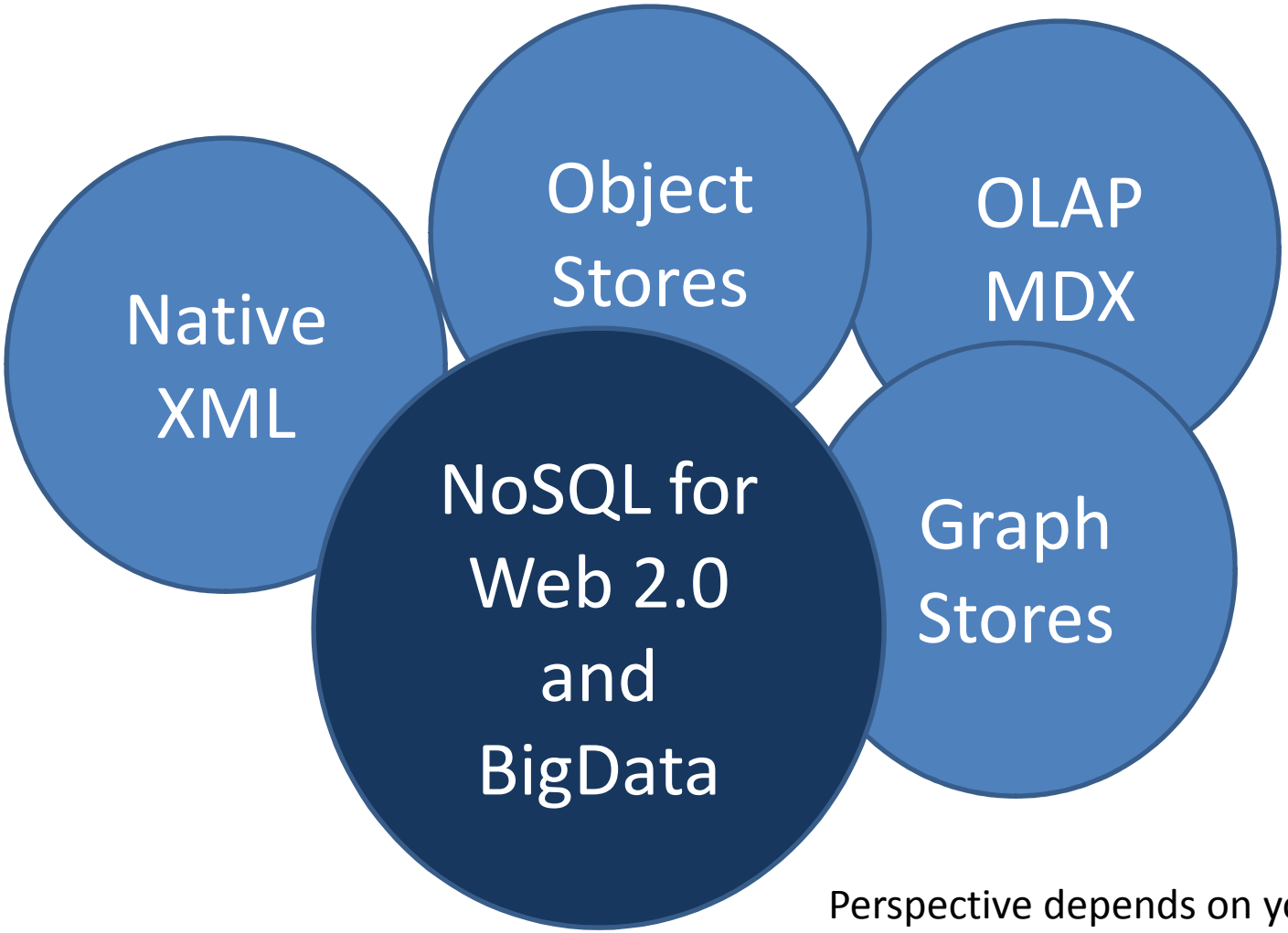
Cut cost from 12 to 3 cents per CPU hour!
Perhaps Alonzo was right!

Why? (hint: how "shareable" is this process)



http://aws.amazon.com/elasticmapreduce/#pricing

Perspectives

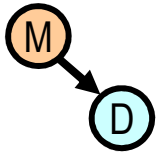


Perspective depends on your context

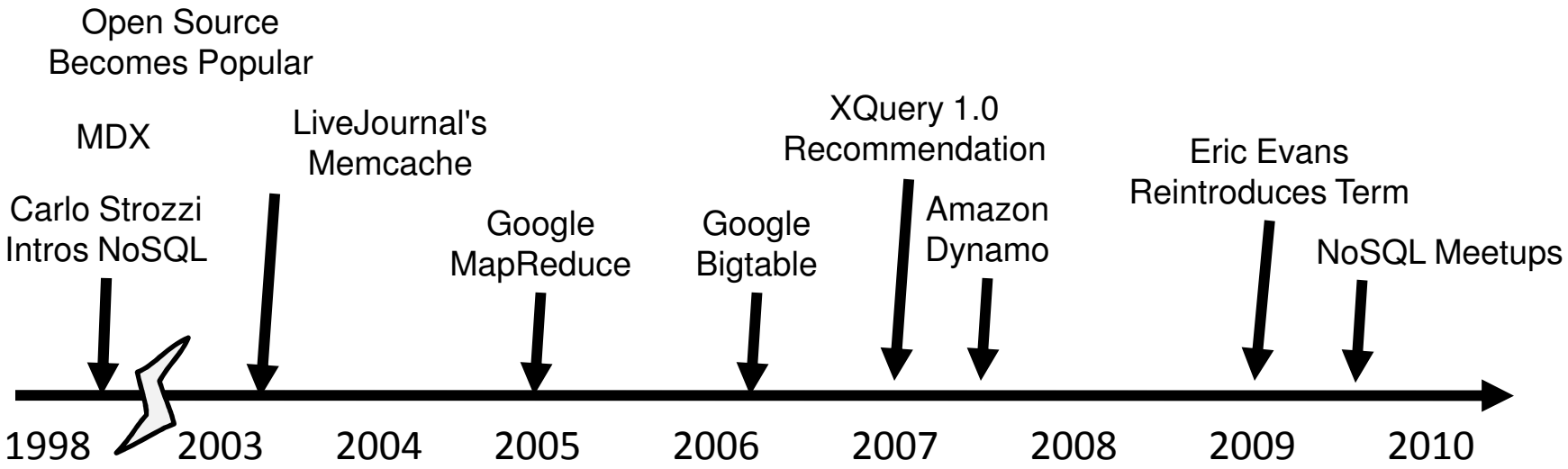
Architectural Tradeoffs

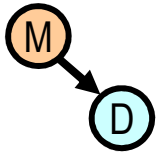
I want a fast car with good mileage.

I want a scaleable database with low cost that runs well on the 1000 CPUs in our data center.

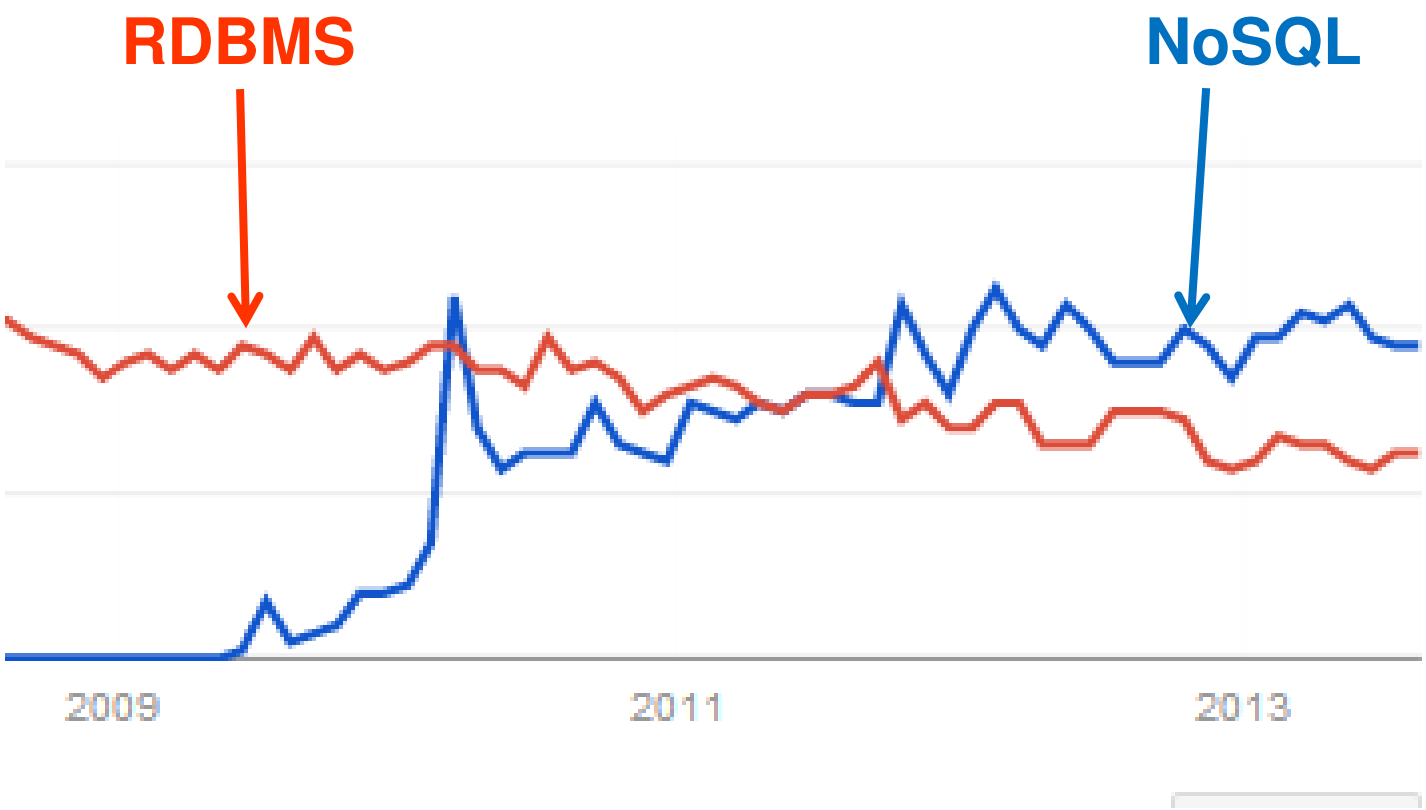


Key Events on NoSQL Timeline

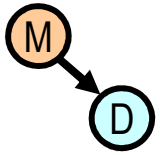




NoSQL on Google Trends



http://www.google.com/trends/explore?q=NoSQL%2C+RDBMS#q=NoSQL%2C%20RDBMS&cmpt=q



Google MapReduce

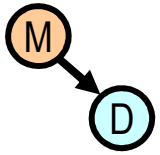
MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

- 2004 paper that had huge impact of functional programming on the entire community
- Copied by many organizations, including Yahoo

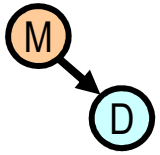


Google Bigtable Paper

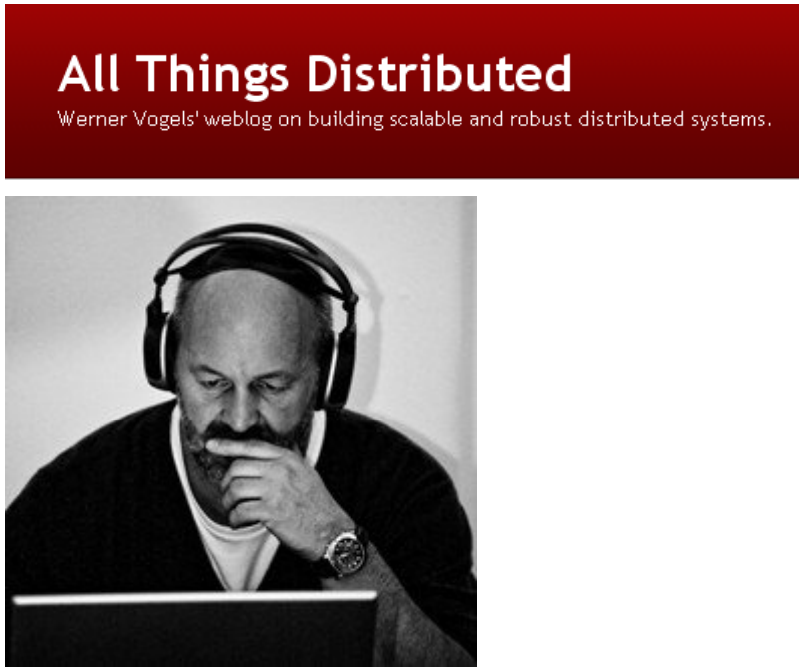
Bigtable: A Distributed Storage System for Structured Data

Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach
Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber
{fay,jeff,sanjay,wilsonh,kerr,m3b,tushar,fikes,gruber}@google.com
Google, Inc.

- 2006 paper that gave focus to scaleable databases
- designed to reliably scale to petabytes of data and thousands of machines

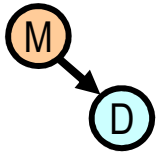


Amazon's Dynamo Paper



- Werner Vogels
- CTO - Amazon.com
- October 2, 2007
- Used to power Amazon's Web Sites
- One of the most influential papers in the NoSQL movement

Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swami Sivasubramanian, Peter Vosshall and Werner Vogels, “Dynamo: Amazon's Highly Available Key-Value Store”, in the *Proceedings of the 21st ACM Symposium on Operating Systems Principles*, Stevenson, WA, October 2007.



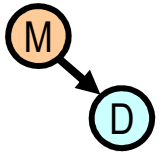
NoSQL "Meetups"

"NoSQLers came to share how they had overthrown the tyranny of slow, expensive relational databases in favor of more efficient and cheaper ways of managing data."

Jul 1, 2009

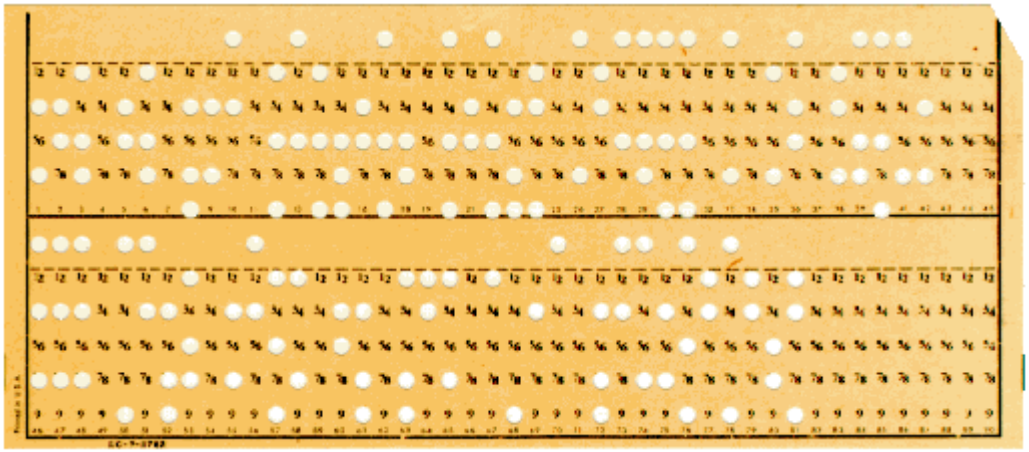
Computerworld magazine

http://www.computerworld.com/s/article/9135086/No_to_SQL_Anti_database_movement_gains_steam_



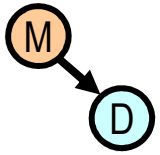
Many Processes Today Are Driven By…

The constraints of yesterday…



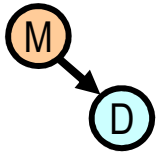
Challenge:

- Ask ourselves the question…
- Do our current method of solving problems with tabular data…
- Reflect the storage of the 1950s…
- Or our actual business requirements?
- What structures best solve the actual business problem?

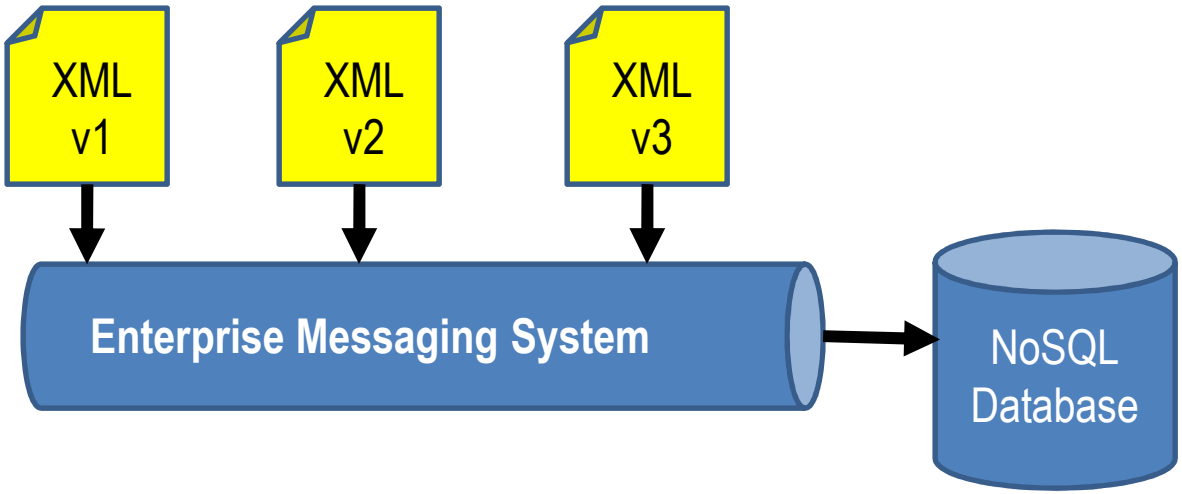


"Schema Free"

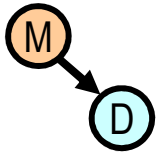
- Systems that automatically determine how to index data **as the data is loaded** into the database
- No ***a priori*** knowledge of data structure
- No need for up-front logical data modeling
  - …but some modeling is still critical
- Adding new data elements or changing data elements is not disruptive
- Searching millions of records still has sub-second response time



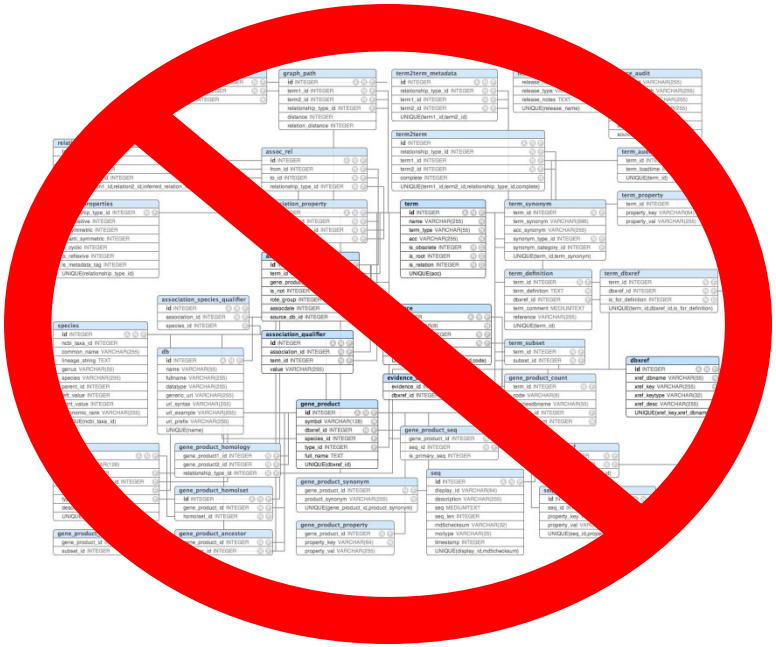
Schema-Free Integration



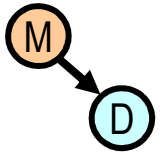
"We can easily store the data that we **actually** get, not the data we **thought** we would get."



Upfront ER Modeling is Not Required



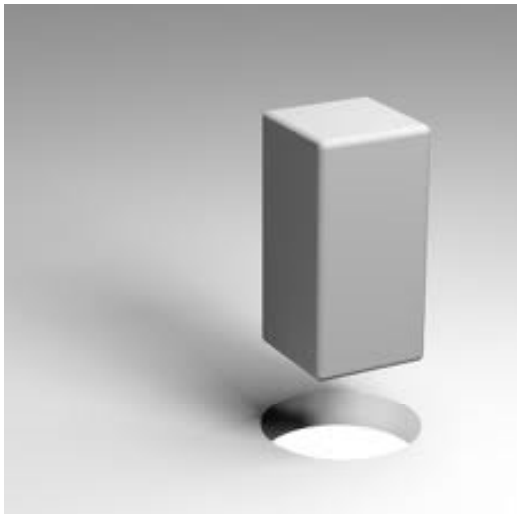
- You do not have to **finish** modeling your data before you insert your first record
- No Data Definition Language "DDL" is needed
- Metadata is used to create indexes as data arrives
- Modeling becomes a statistical process – write queries to find exceptions and normalize data
- Exceptions make the rules but can still be used
- Data validation can still be done on documents using tools such as XML Schema and business rules systems like Schematron



Monoculture and Mono-architecture

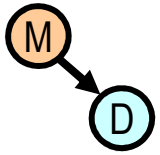
Image Source: Wikipedia

Eric Evans

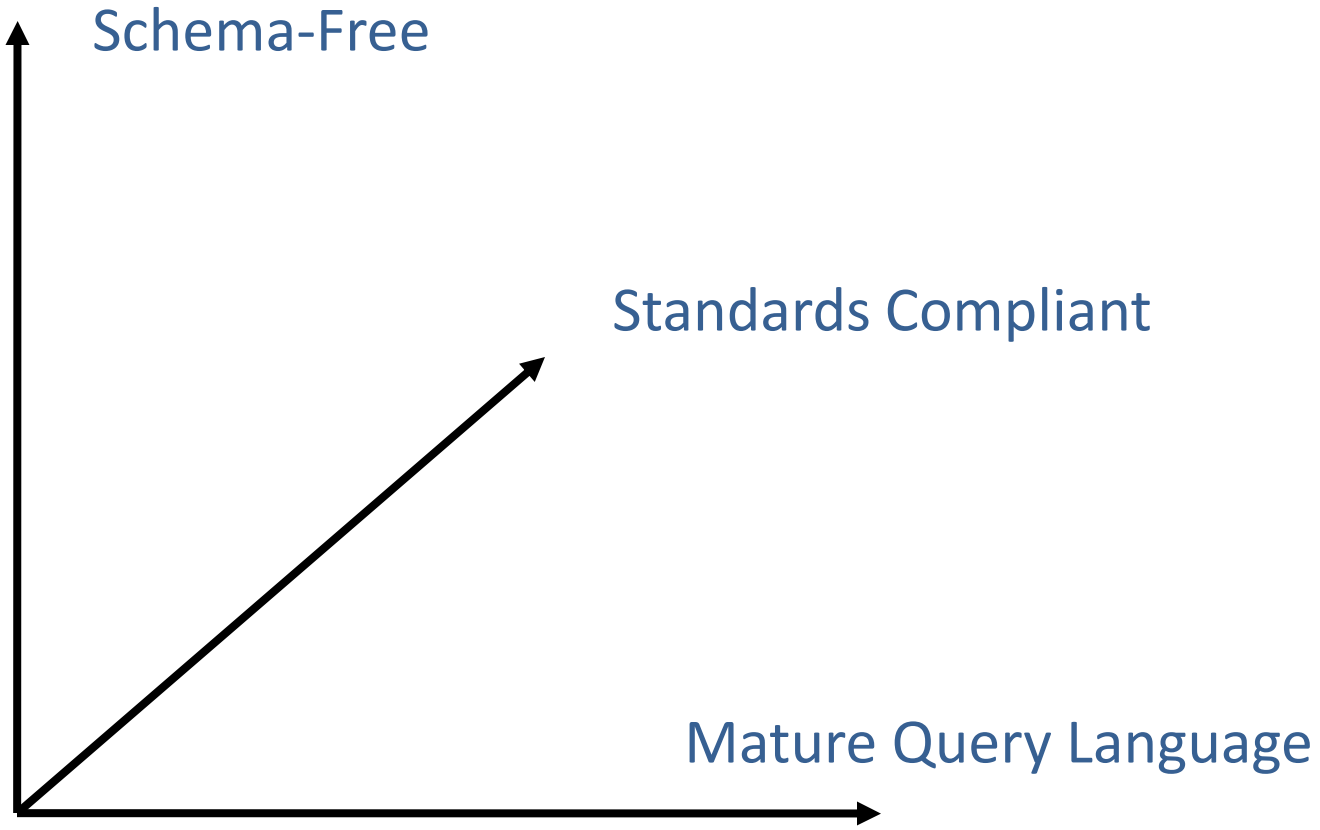


"The whole point of seeking alternatives [to RDBMS systems] is that you need to solve a problem that relational databases are a bad fit for."

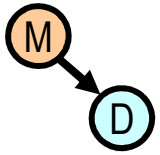
Eric Evans
Rackspace



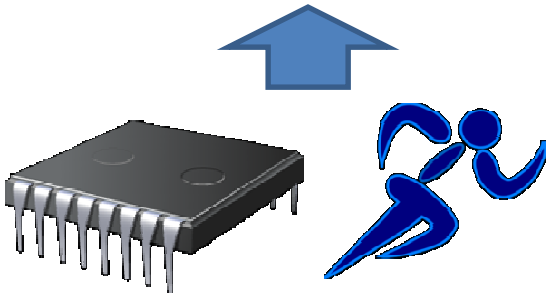
Finding the Right Match

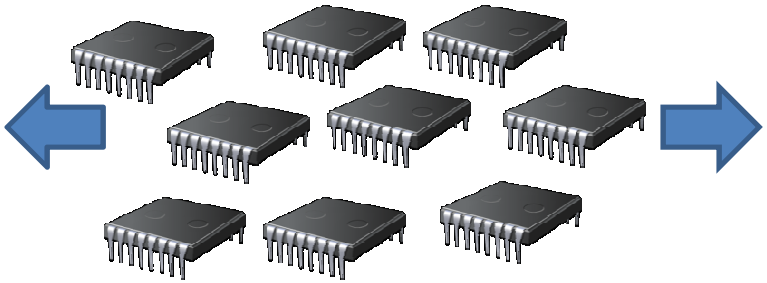


Use CMU's Architectural Tradeoff and Modeling (ATAM) Process



Scale Up vs. Scale Out



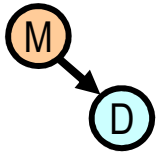


Scale Up

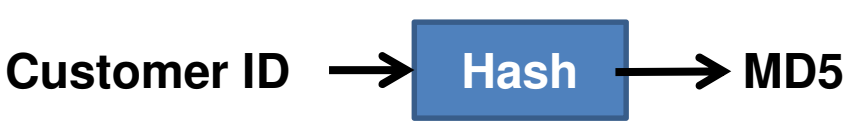
- Make a single CPU as fast as possible
- Increase clock speed
- Add RAM
- Make disk I/O go faster

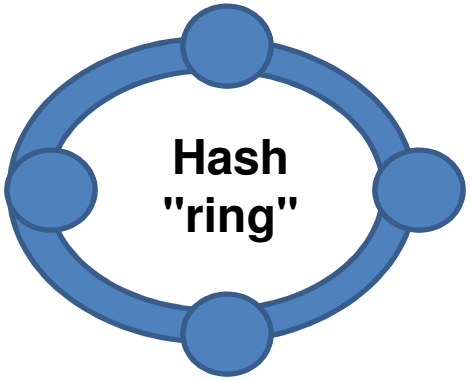
Scale Out

- Make Many CPUs work together
- Learn how to divide your problems into independent threads

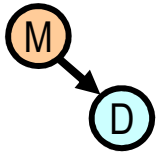


The "Hash Mod"

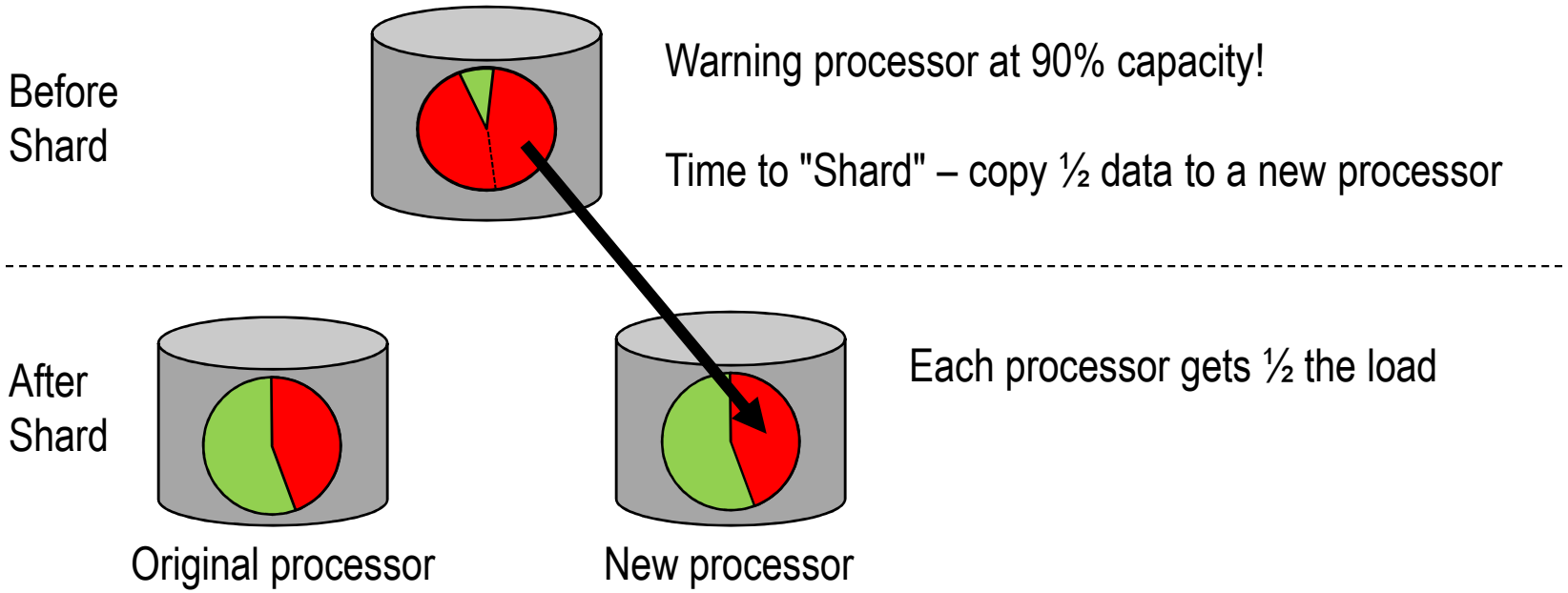




- Create a hash of the "key"
- Use the initial bits or the "modulo" of the hashed value
- Use this to distribute the item to the cluster
- Allows even distribution of data over a cluster



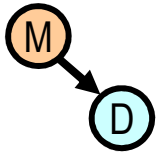
Automatic Sharding



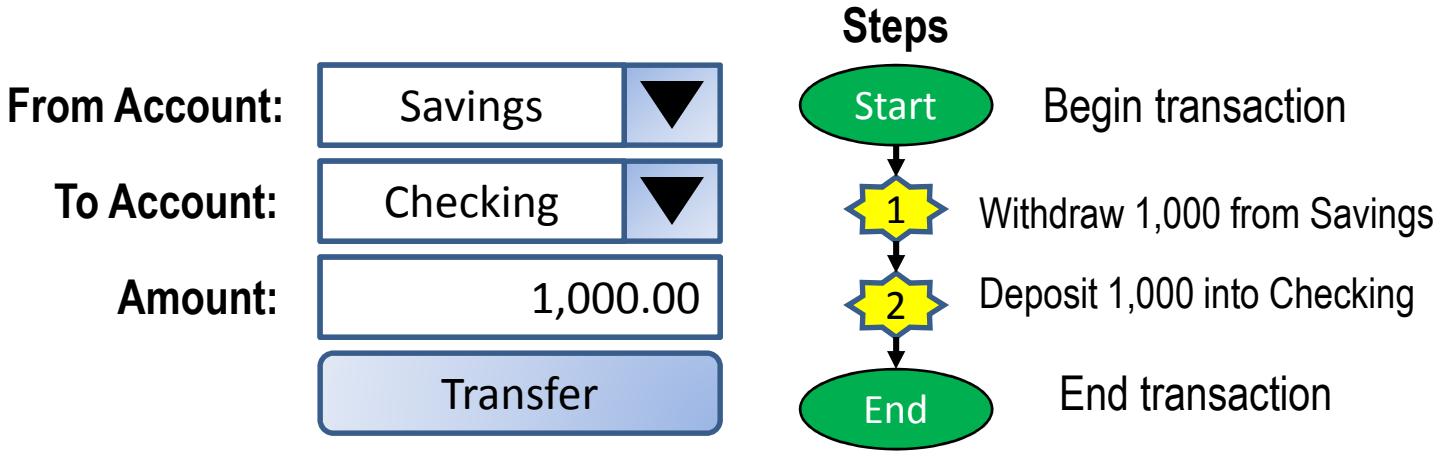
- When one node in a cluster has too much of a load

ACID Transactions

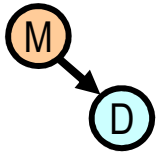
- **Atomic** – transaction either complete or fail – no in-between
- **Consistent** – no reports can ever catch data between states
- **Isolated** – never allow users to touch data that is incomplete
- **Durable** – recover committed transactions from partial system failures



ACID Definition of Consistency

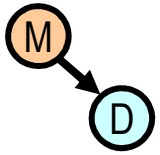


- Reports will **never** show a total balance after withdrawal and before deposit



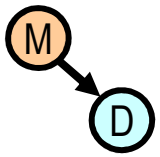
BASE

- **B**ase Availability of Soft state **E**ventually Consistent
- "Eventually Consistent" is not appropriate for many financial reporting systems

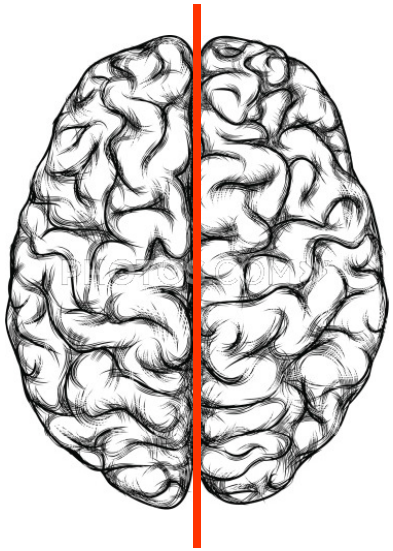


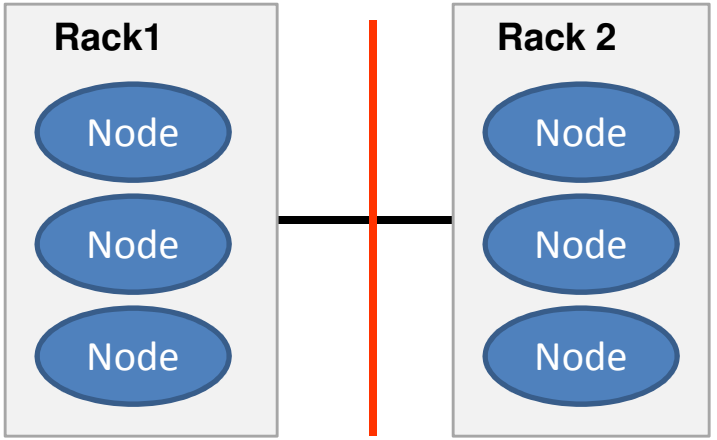
Consistency

- In RDBMS Systems
  - reports will always get a consistent value
  - never results from first part of a transaction
- In Distributed Computing
  - rules that determine if a read from a cluster has verified that multiple nodes all have the same value

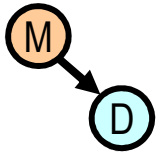


Partition Tolerance

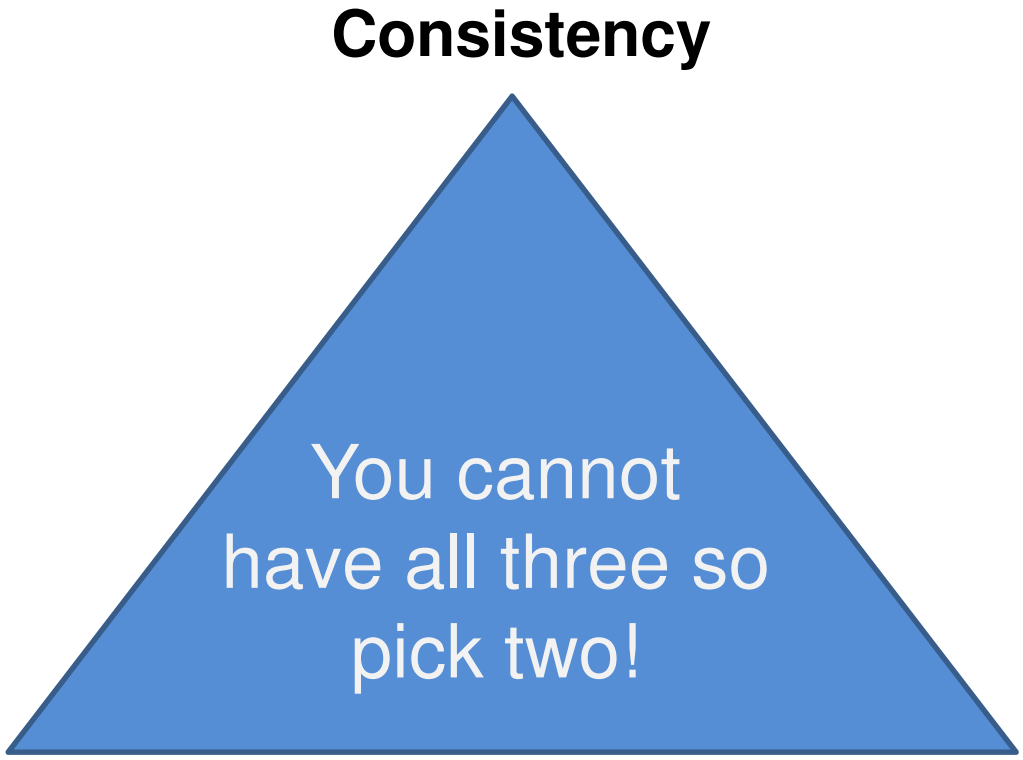




- What happens when you get a "split brain" in your data center?
- One part of your cluster can not talk to the other part of the cluster?
- High availability systems want to write to multiple nodes in a cluster if one node fails

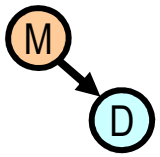


Brewer's CAP Theorem

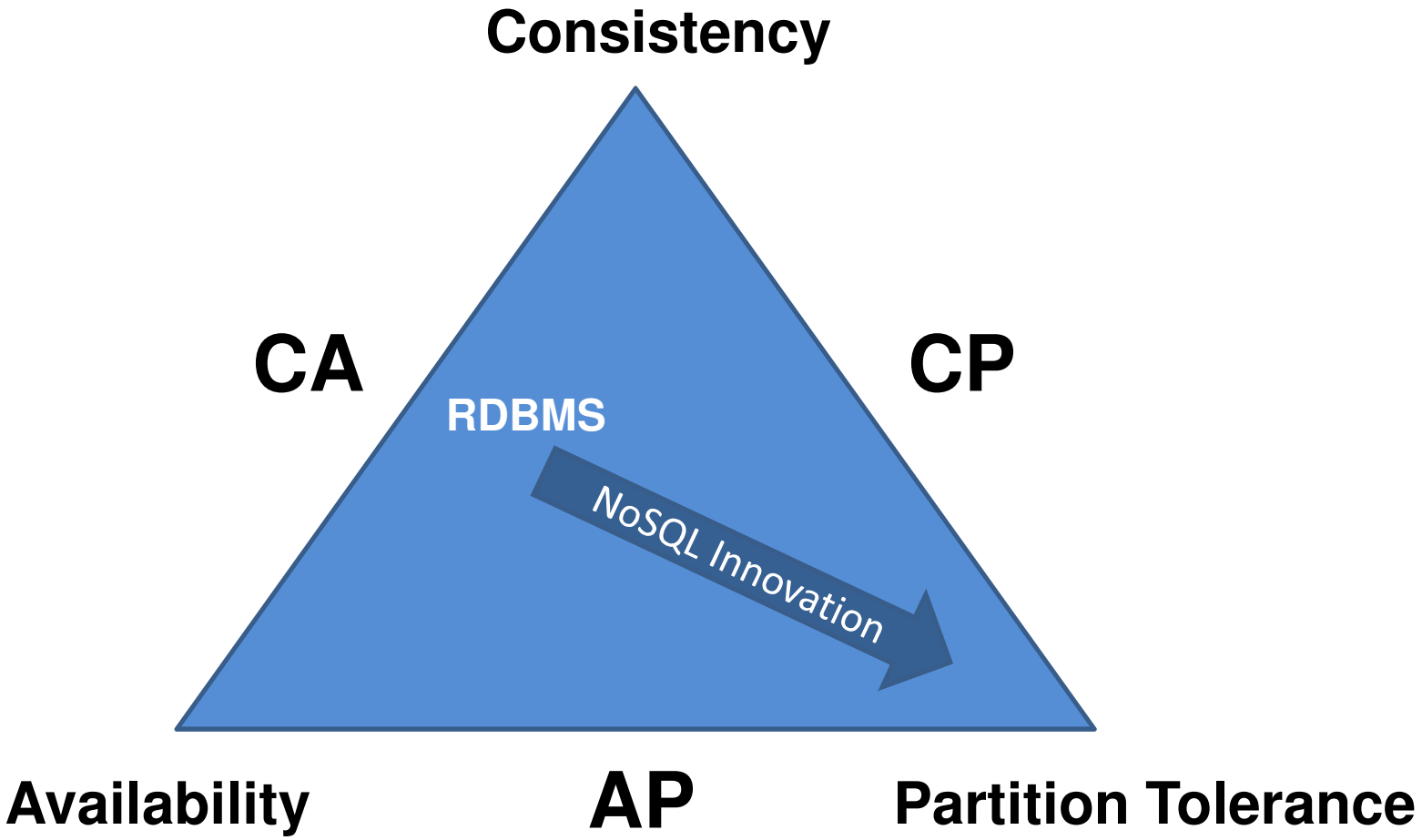


CAP Systems

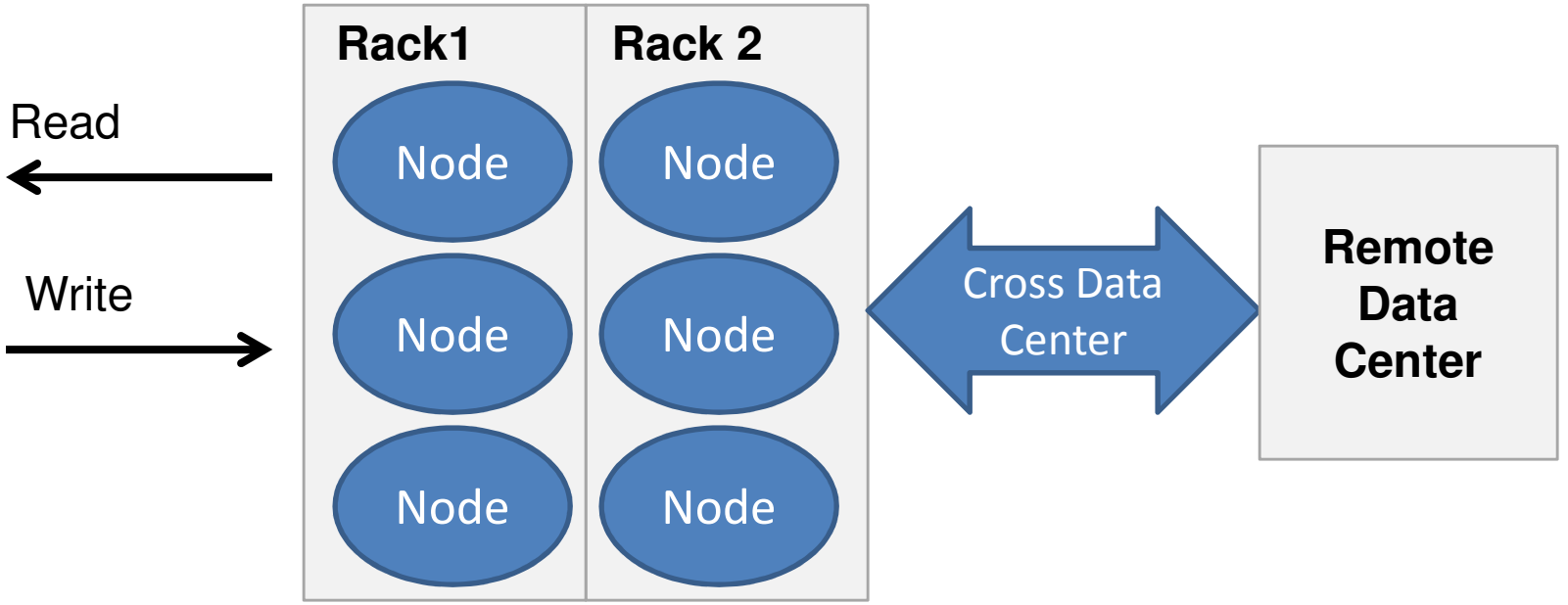
- **CA** – Consistent and Available
  - Example – a single site, single database, no auto-sharding
  - If you have to partition, stop, partition, restart
- **CP** – Consistent with Partition Tolerance
  - Some data not available when partitioning
  - What data you have is always consistent
- **AP** – Available with Partition Tolerance
  - Always on, but during partitioning, some data may be inconsistent
  - After sharding eventual consistency



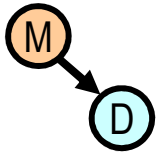
NoSQL Movement



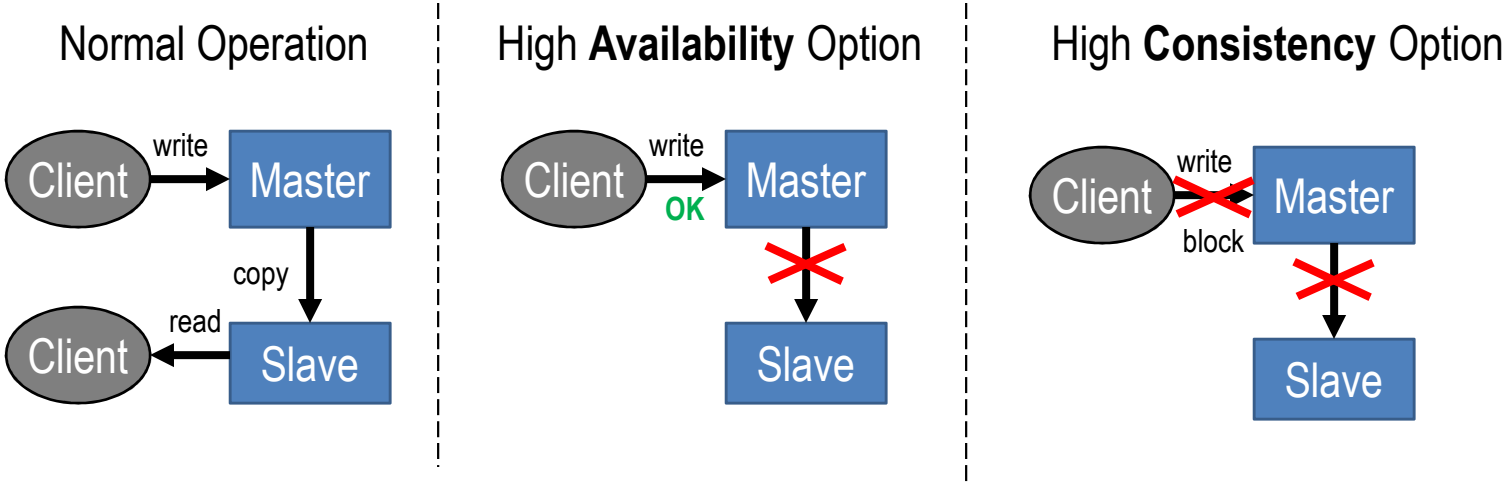
NoSQL "Clusters"



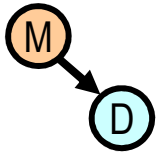
- How many nodes should we check to guarantee a consistent read?
- How many nodes should we write to in order to make sure our write is "durable"



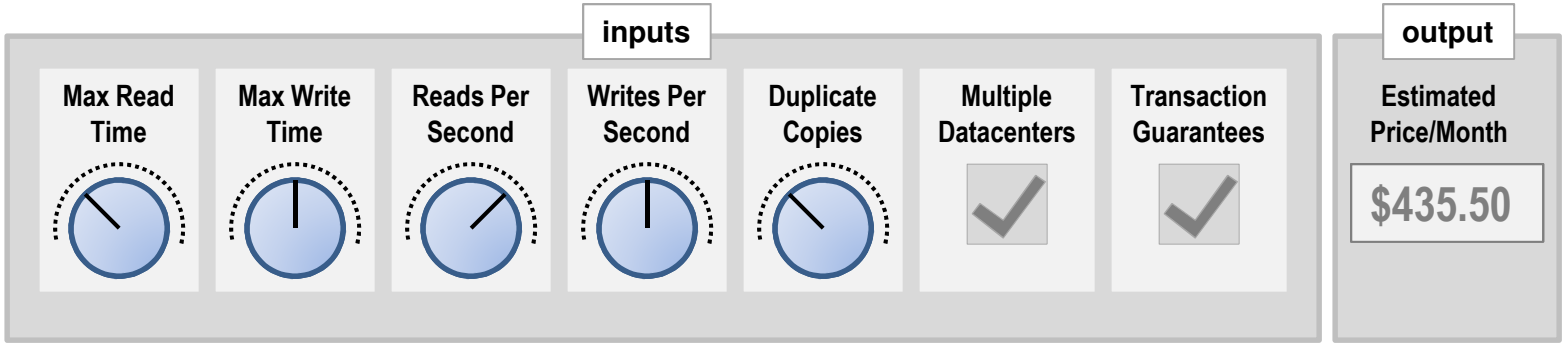
Availability Consistency Option



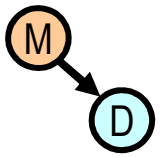
- The CAP theorem gives designers **choices** on how to behave when a remote system is down – during a "Network Partition"
- Application designers can select different options for each **transaction**



The Tunable SLA



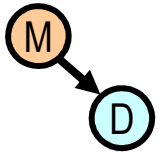
- NoSQL systems that use many commodity processors can be precisely tuned to meet an organizations service level agreements



Example: Amazon DynamoDB

Tuning Knobs

- Tune your service to the business requirements (challenging to do with a RDBMS)

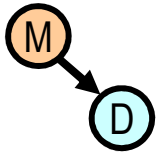


"One Size Fits…"

"One Size Does Not Fit All"

James Hamilton Nov. 3rd, 2009

http://perspectives.mvdirona.com/CommentView,guid,afe46691-a293-4f9a-8900-5688a597726a.aspx

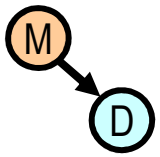


The Eight Fallacies of Distributed Computing

Essentially everyone, when they first build a distributed application, makes the following eight assumptions. All prove to be false in the long run and all cause *big* trouble and *painful* learning experiences.

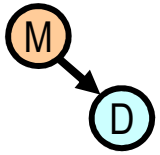
1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology doesn't change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

Peter Deutsch

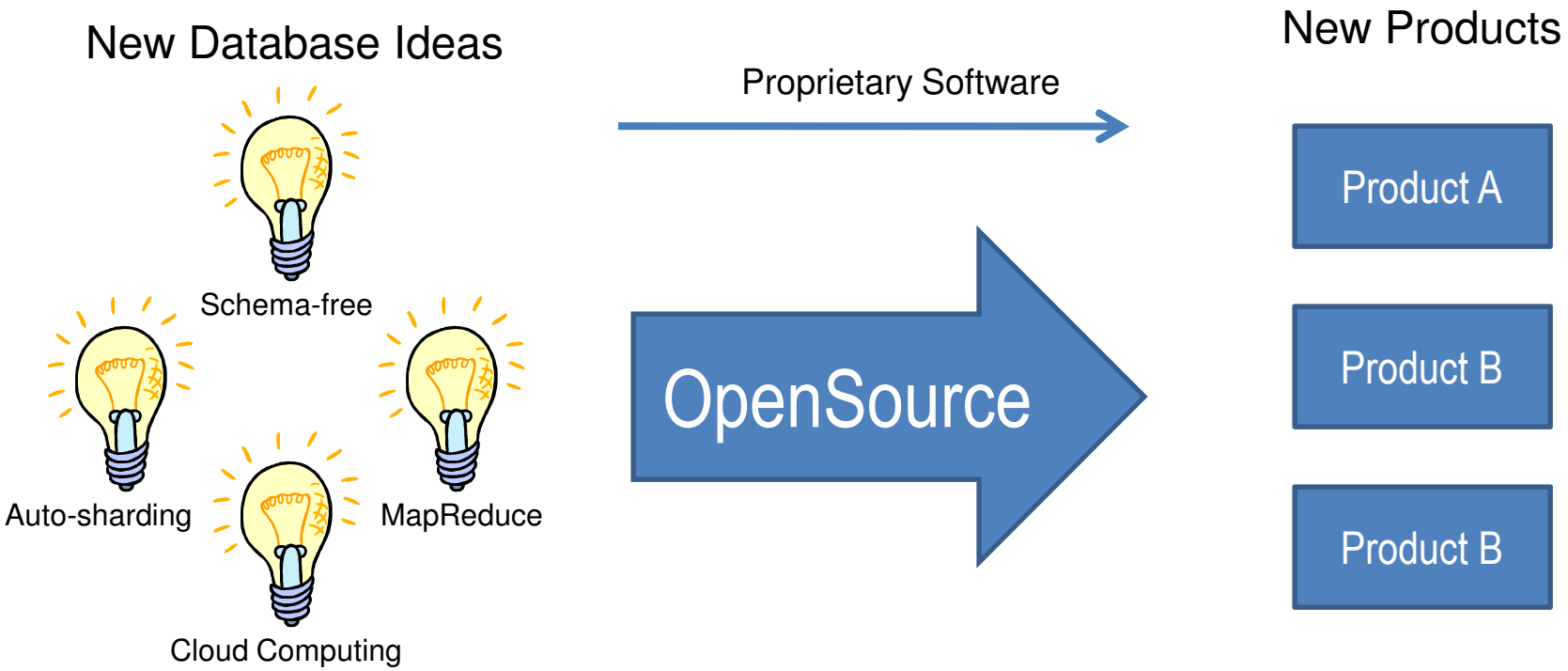


NoSQL and Cloud Computing

- High scalability
  - Especially in the horizontal direction (multi CPUs)
- Low administration overhead
  - Simple web page administration
- Lower fees for MapReduce transformations



Evolution of Ideas in OpenSource



- How quickly can new ideas be **recombined** into new database products?
- OpenSource software has proved to be the most **efficient** way to quickly recombine new ideas into new products

The Nature of Technology: What It Is and How It Evolves by W. Brian Arthur

The NO-SQL Universe

Key-Value Stores

riak

FOUNDATIONDB

redis

AEROSPIKE

Document Stores

Couchbase

mongoDB

apache CouchDB relax

RAVENDB

Graph/Triple Stores

neotechnology

InfiniteGraph

AllegroGraph® RDFStore

bigdata®

Column-Family Stores

HBASE

Cassandra

Object Stores

VERSANT

GEMSTONE

existdb

MarkLogic

Zorba The XQuery Processor

BASE

XML

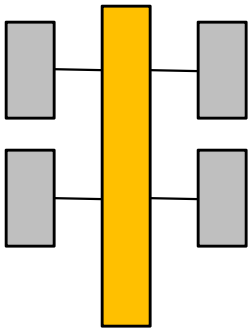
M

D

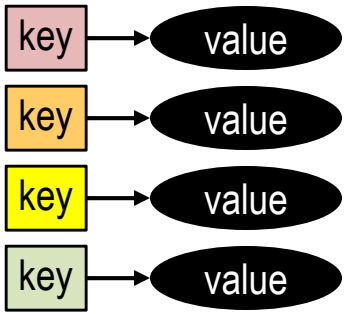
NoSQL Database Patterns

Relational

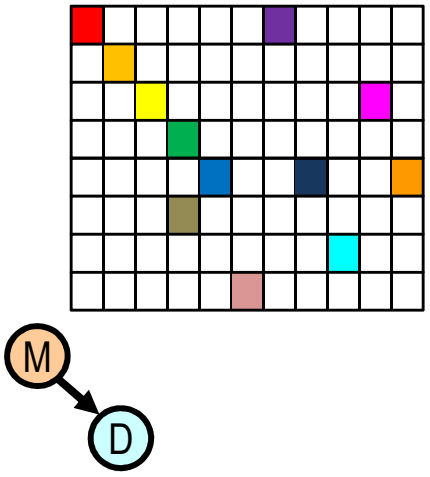
Analytical (OLAP)



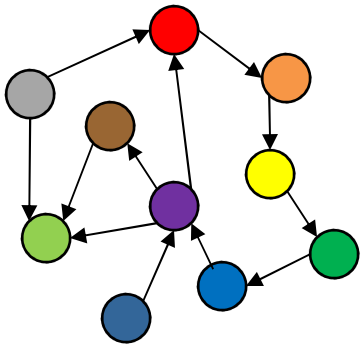
Key-Value



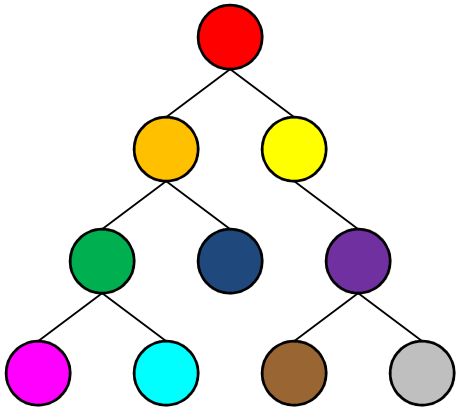
Column-Family



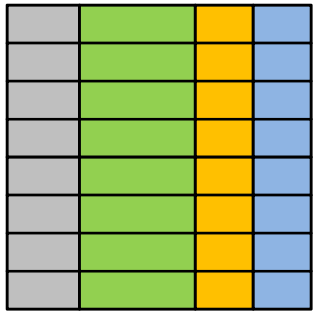
Graph



Document

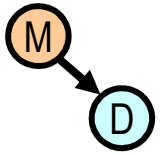


Relational

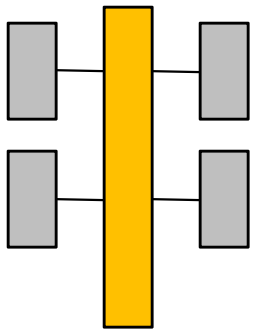


- Data is usually stored in row by row manner (row store)
- Standardized query language (SQL)
- Data model defined **before** you add data
- Joins merge data from multiple tables
- Results are tables
- **Pros:** mature ACID transactions with fine-grained security controls
- **Cons:** Requires up front data modeling, does not scale well

Examples:
Oracle, MySQL, PostgreSQL, Microsoft SQL Server, IBM DB/2

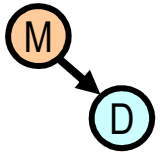


Analytical (OLAP)

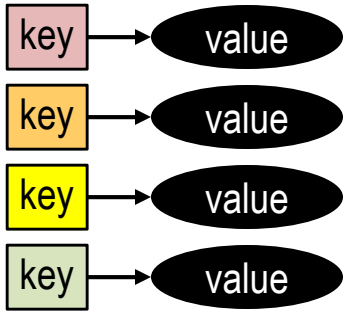


- Based on "Star" schema with central fact table for each event
- Optimized for analysis of read-analysis of historical data
- Use of MDX language to count query "measures" for "categories" of data
- **Pros:** fast queries for large data
- **Cons:** not optimized for transactions and updates

Examples:
Cognos, Hyperion, Microstrategy, Pentaho, Microsoft, Oracle, Business Objects



Key-Value Stores

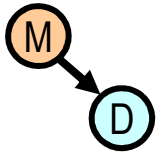


- Keys used to access opaque blobs of data
- Values can contain any type of data (images, video)

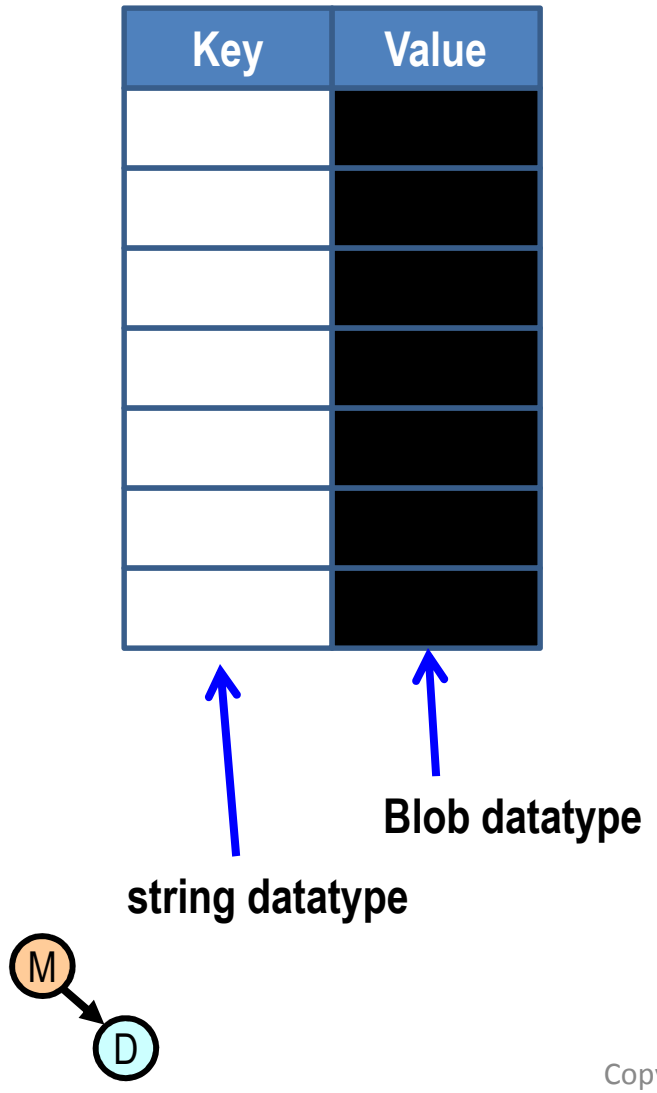
Pros: scalable, simple API (put, get, delete)

Cons: no way to query based on the content of the value

Examples:
Berkley DB, Memcache, DynamoDB, S3, Redis, Riak



Key Value Stores



- A table with two columns and a simple interface
  - Add a key-value
  - For this key, give me the value
  - Delete a key
- Blazingly fast and easy to scale (no joins)

Key-Values Stores are Like Dictionaries

Key: "amphora"

Value: Text of dictionary entry

am·pho·ra

noun

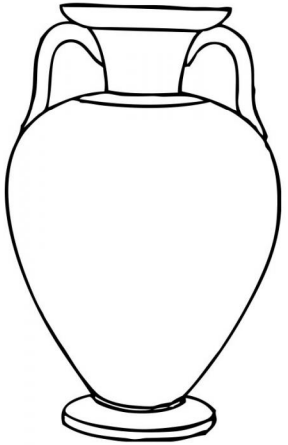
1. A large two handled jar with a narrow neck used in ancient times by Greek and Romans to store or carry wine or oils.
2. A vessel, usually made of clay, with two handles or ears for liquids.
3. A measure for liquids; quadrantal; the measure of a ship.

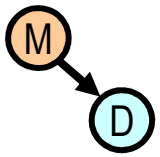
plural: amphorae or amphoras

Similar words: flagon, pitcher, flask, bottle, jar.

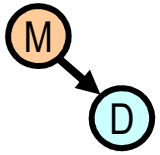
Etymology

From ancient Greek, a vase shaped ornament with a narrow neck



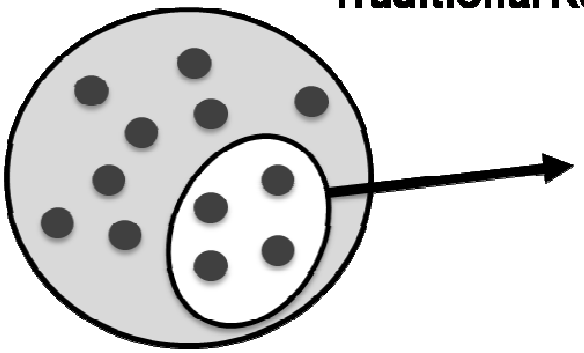


The Locker Metaphor



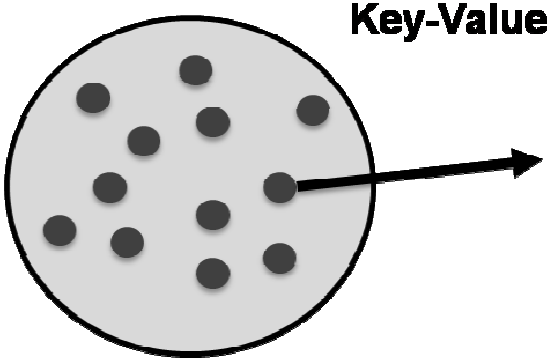
No Subset Queries in Key-Value Stores

Traditional Relational Model

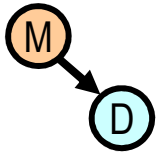


- Result set based on row values
- Value of rows for large data sets must be indexed
- Values of columns must all have the same data type

Key-Value Store Model

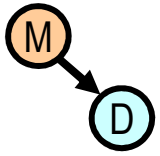


- All queries return a single item
- No indexes on values
- Values may contain any data type



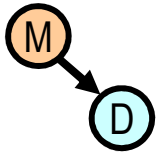
Types of Key-Value Stores

- Eventually-consistent Key-Value store
- Hierarchical Key-Value Stores
- Key-Value Stores In RAM
- Key-Value Stores on Disk
- Ordered Key-Value Stores

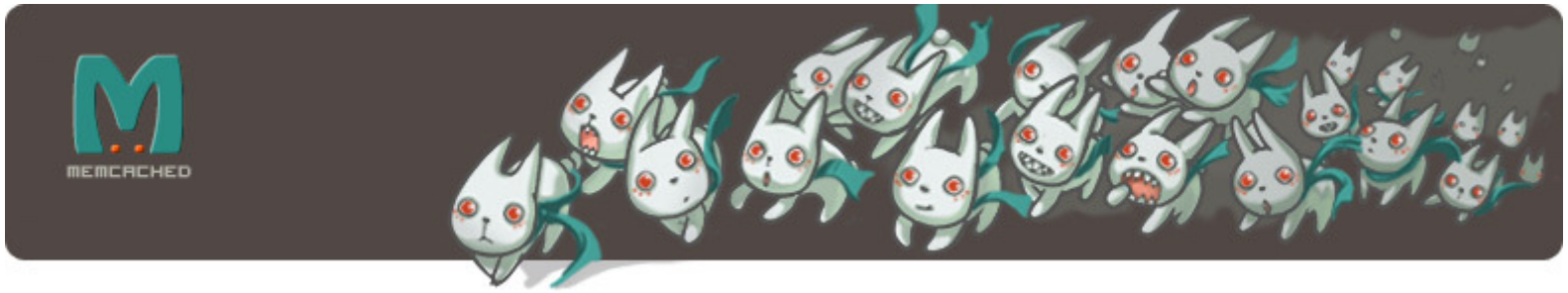


LiveJournal and Memcache

- A story about sharing information in a distributed computing environment
- A story about hashing, caching, APIs, open source …and **standards**

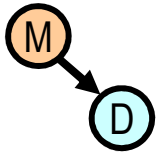


2003 - Memcache

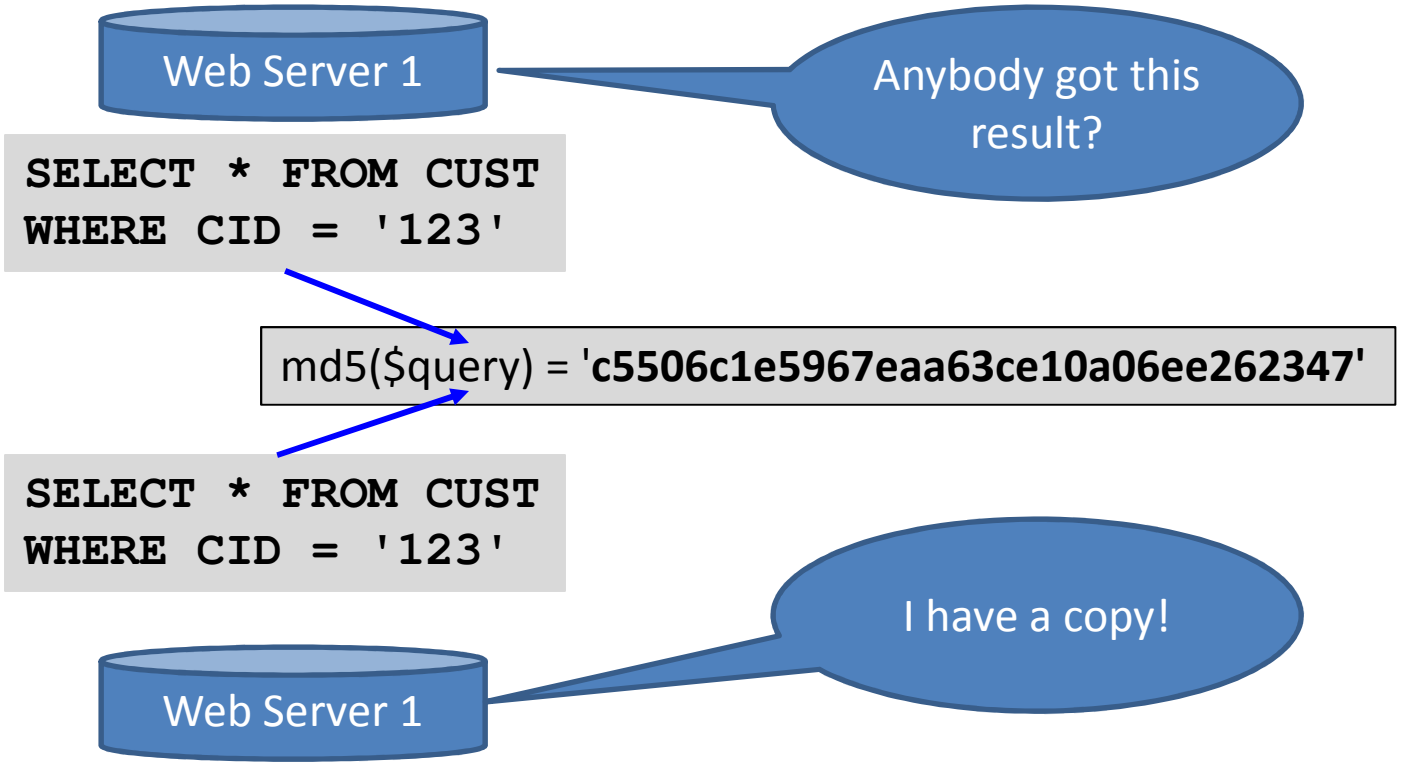


Brad Fitzpatrick

- Originally developed by Brad Fitzpatrick for LiveJournal as a "side-cache"
- Allowed databases to put data in a "RAM Cache" in front of a SQL server to speed up read-mostly operations
- Dramatically lowers load on backend SQL server
- Ideal "front end" to database system to increase performance



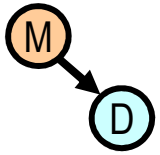
Same query -> Same hash value



Same query, same hash of query, same result!

Foundation DB

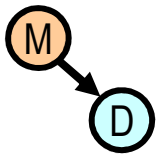
- Ordered key-value storage
- Focus on scalability **and** consistency
- Support for ACID transactions
- Replication and partitioning
- Use of solid state drives
- Vision: be the foundation layer for other database types



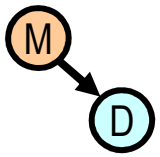
Oracle

- Oracle NoSQL database is a key-value
- Use hashed value of the primary key to select node
- Storage nodes are replicated to ensure high availability
- Rapid failover in the event of a node failure and optimal load balancing of queries
- Applications are written using either a Java or C API

http://www.oracle.com/technetwork/products/nosqldb/overview/index.html

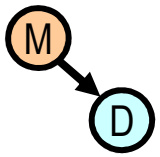


- Open source distributed key-value store with support and commercial versions by Basho
- A "Dynamo-inspired" database
- Focus on availability, fault-tolerance, operational simplicity and scalability
- Support for replication and auto-sharding and rebalancing on failures
- Support for MapReduce, fulltext search and secondary indexes of value tags
- Written in ERLANG

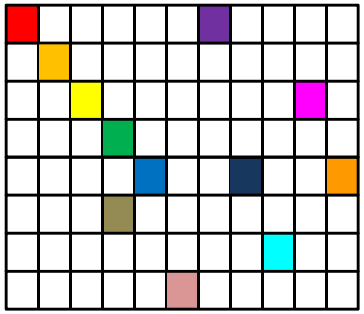


Aerospike

- Commercial distributed key-value store
- Focus on low-latency transactions
- Optimized for solid-state drives
- High availability – no single point of failure
- Support for cross data center replication
- APIs in C, C#, Java, Ruby, PHP and Python

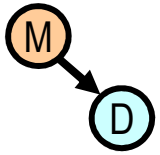


Column-Family

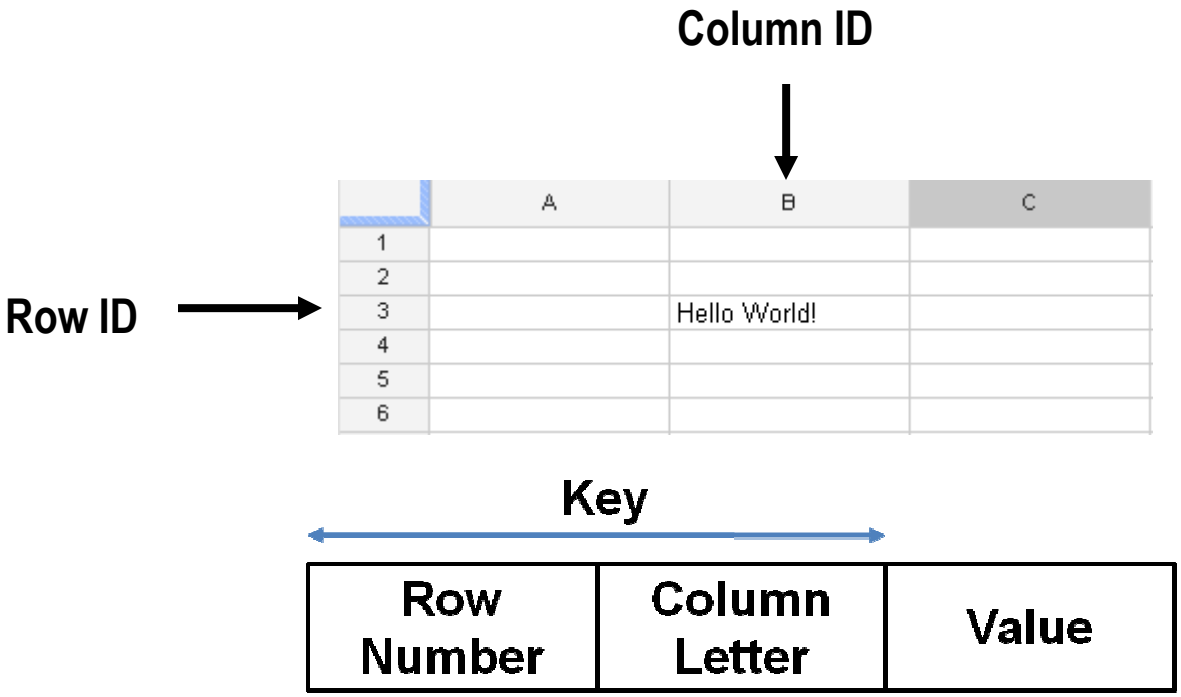


- Key includes a row, column family and column name
- Store versioned blobs in one large table
- Queries can be done on rows, column families and column names
- Pros: Good scale out, versioning
- Cons: Cannot query blob content, row and column designs are critical

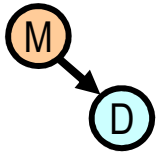
Examples:
Cassandra, HBase, Hypertable, Apache Accumulo, Bigtable



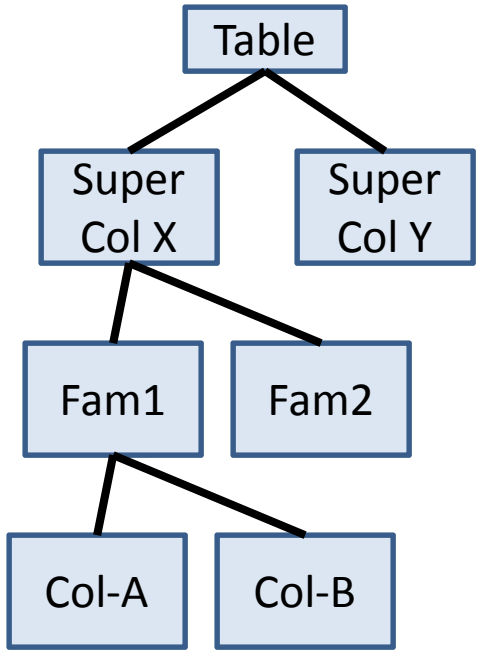
Spreadsheets Use a Row/Column as a Key



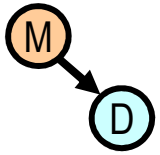
- Bigtable systems use a combination of row and column information as part of their key



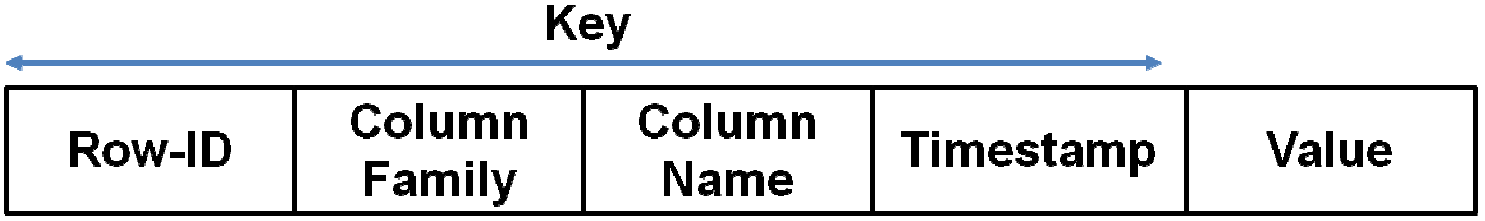
Column Families



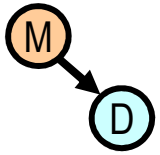
- Group columns into "Column families"
- Group column families into "Super-Columns"
- Can query all columns with a family or super family
- Similar data grouped together to improve speed



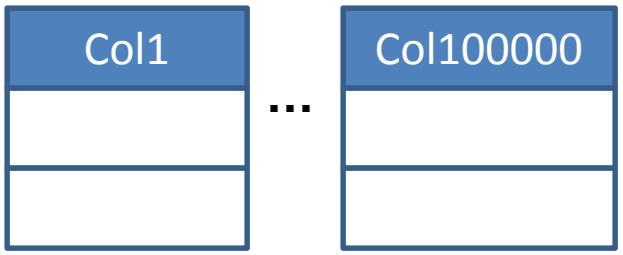
Keys Include Family and Timestamps



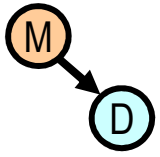
- Bigtable systems have keys that include not just row and column ID but other attributes
- Column Families are created when a table is created
- Timestamps allows multiple versions of values
- Values are ordered bytes and have no strongly typed data system



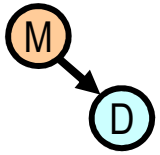
Column Store Concepts

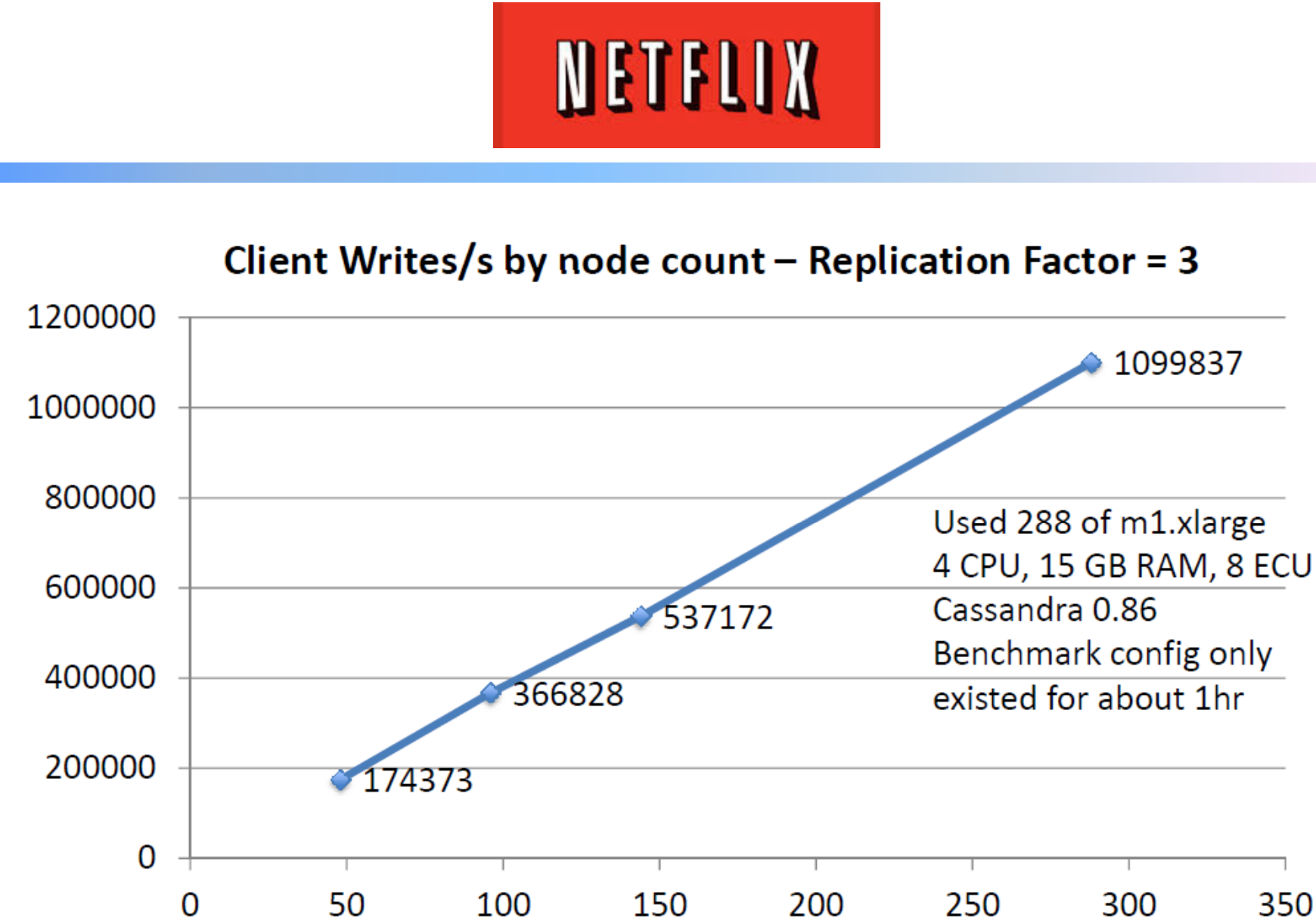


- Preserve the table-structure familiar to RDBMS systems
- Not optimized for "joins"
- One row could have millions of columns but the data can be very "sparse"
- Ideal for high-variability data sets
- Column families allow you to query all columns that have a specific property or properties
- Allow new columns to be inserted without doing an "alter table"
- Trigger new columns on inserts

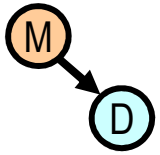


- Apache open source project
- Originally developed by Facebook
- Used as a key-value store and a bigtable store
- Now used by Twitter and many others
- Written in Java
- Similar to Bigtable and DynamoDB
- Designed for highly distributed systems
- Column-family data model



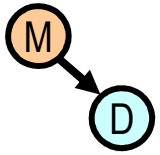


- Open source implementation of MapReduce algorithm written in Java
- Initially created by Yahoo
  - 300 person-years development
- Column-oriented data store
- Java interface
- HBase designed specifically to work with Hadoop
- High-level query language (Pig)
- Strong support by many vendors

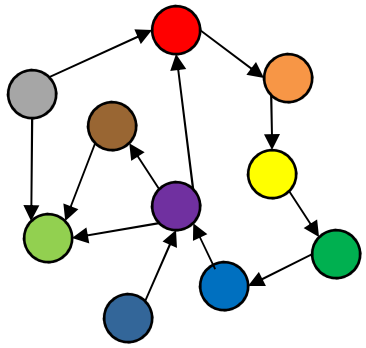


Apache Accumulo

- Apache open source column-family store
- Distributed key-value based on BigTable
- Focus on scalability, availability and security
- Strong support for in-database fine-grained security
- Support by sqrrl.com

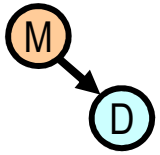


Graph Store



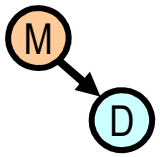
- Data is stored in a series of nodes, relationships and properties
- Queries are really graph traversals
- Ideal when relationships between data is key:
  - e.g. social networks
- **Pros:** fast network search, works with public linked data sets
- **Cons:** Poor scalability when graphs don't fit into RAM, specialized query languages (RDF uses SPARQL)

Examples:
Neo4j, AllegroGraph, Bigdata triple store, InfiniteGraph, StarDog

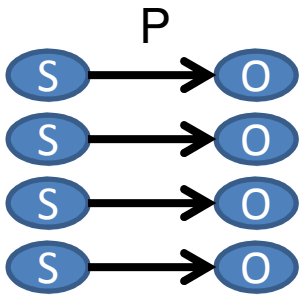


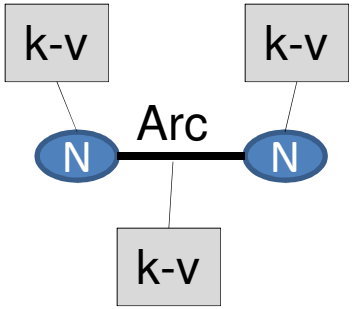
Graph Stores (Continued)

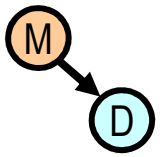
- Used when the relationship and relationships types between items are critical
- Used for
  - Social networking queries: "friends of my friends"
  - Inference and rules engines
  - Pattern recognition
  - Used for working with open-linked data
- Automate "joins" of public data



Two Major Graph Models

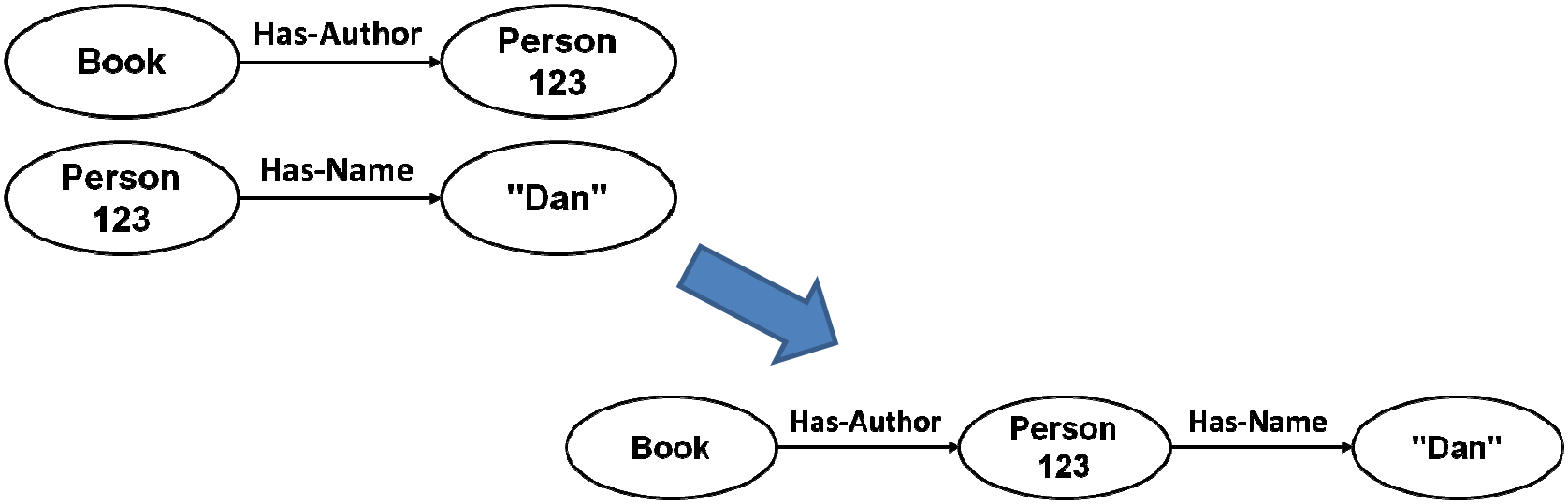




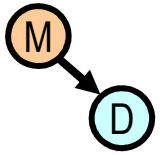


- RDF
  - Uniform triple stores (subject, predicate, object)
  - W3C Standards
    - Resource Description Format (RDF)
    - Standardized query language (SPARQL)
- Property Graphs
  - Nodes and arcs have key-value properties
  - No standardized query language
  - Example: Neo4J - Cypher

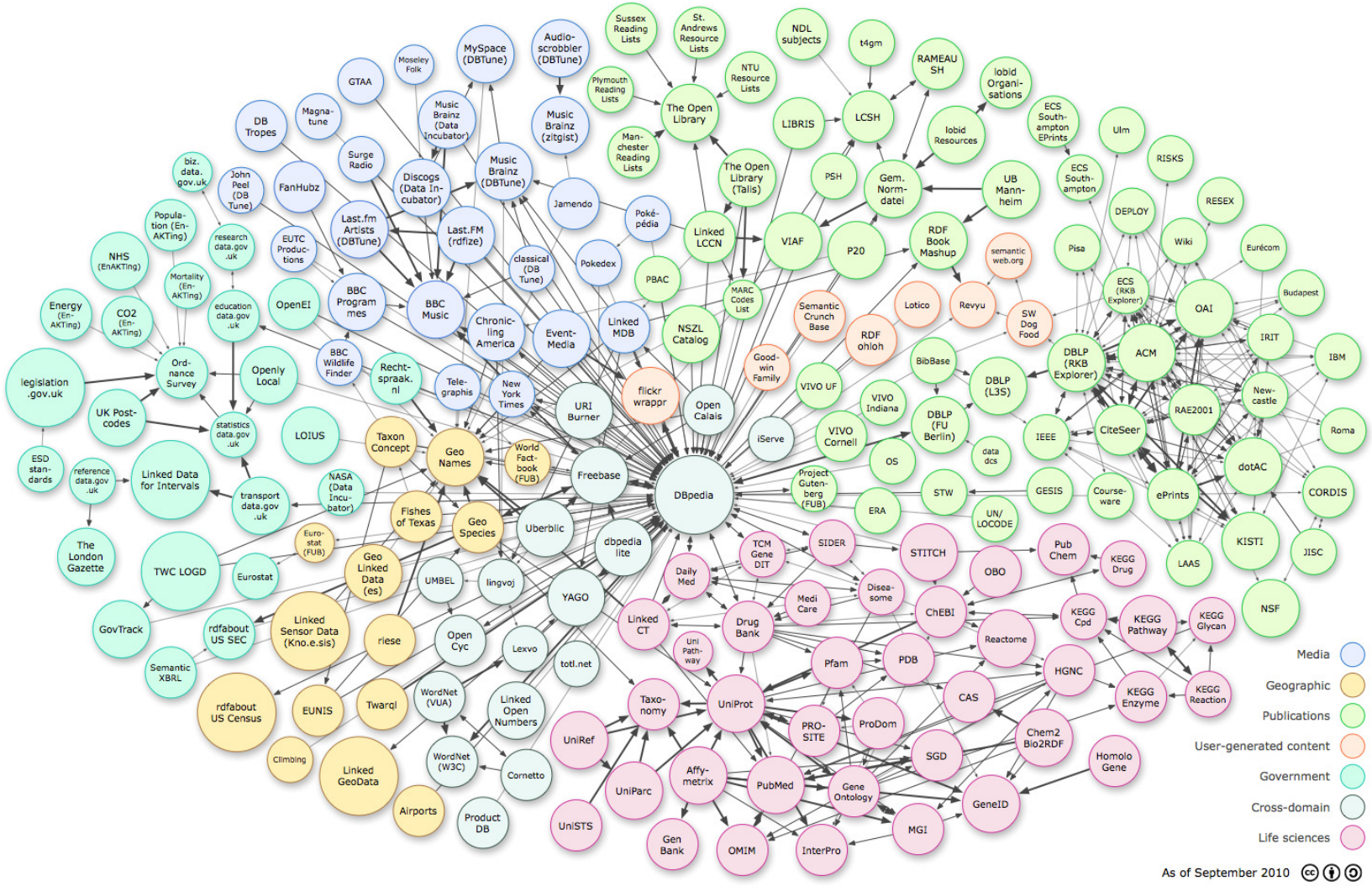
Nodes are "joined" to create graphs



- How do you know that two items reference the same object?
- Node identification – URI or similar structure

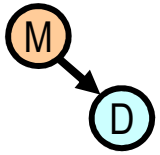


Open Linked Data



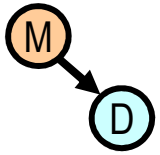
Neo4J

- Property Graph database designed to be easy to use by Java developers
- Dual license (community edition is GPL)
- Works as an embedded Java library in your application
- Disk-based (not just RAM)
- Full ACID
- See http://neo4j.org

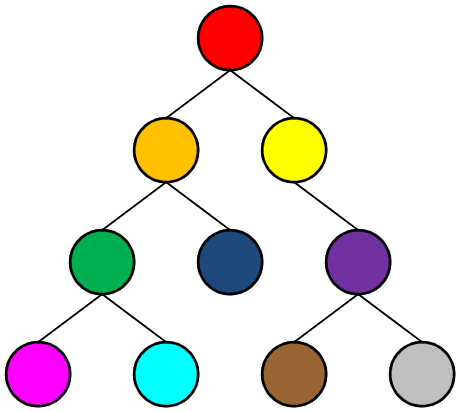


InfiniteGraph

- Commercial graph database
- Graph property model
- Extensive use in intelligence and federal systems
- Extensive graph analysis tools
- Modular "plug in" security architecture
- Used for persisting and traversing complex relationships requiring multiple hops across large distributed graph stores

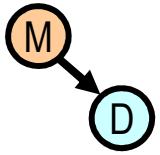


Document Store

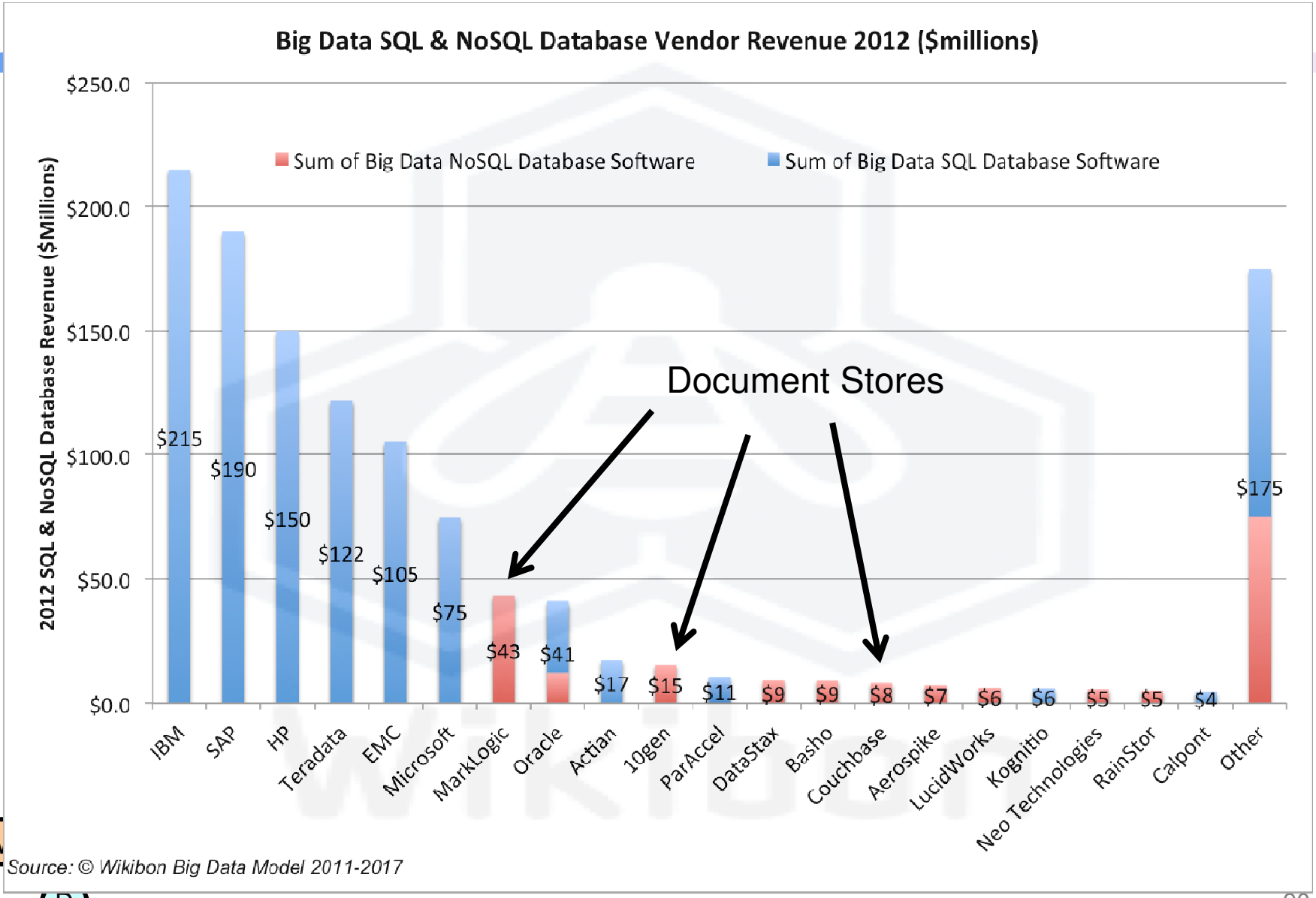


Examples:
MarkLogic, MongoDB, Couchbase, CouchDB, RavenDB, eXist-db

- Data stored in nested hierarchies
- Logical data remains stored together as a unit
- Any item in the document can be queried
- **Pros:** No object-relational mapping layer, ideal for search
- **Cons:** Complex to implement, incompatible with SQL

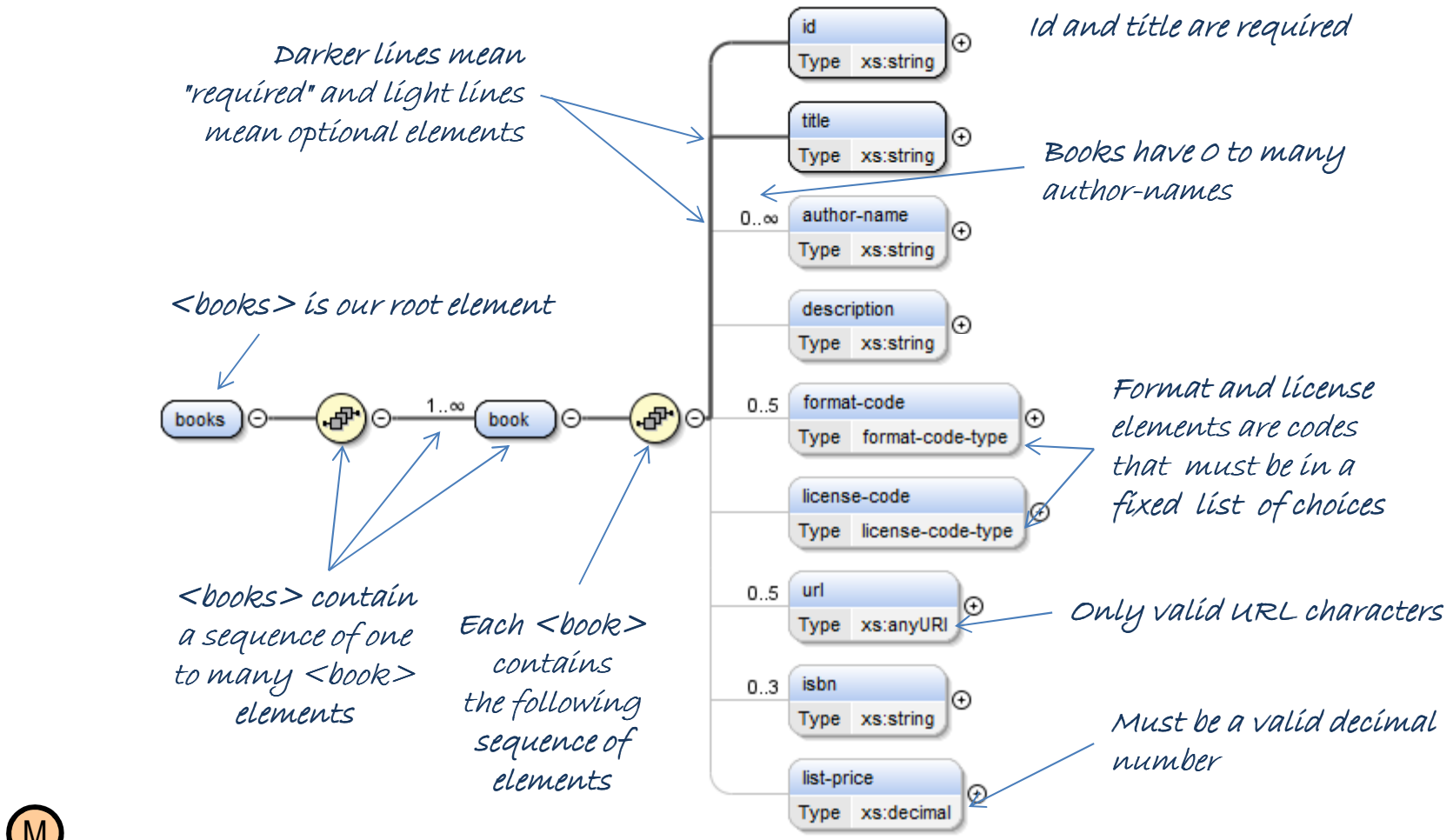


Estimated Big Data and NoSQL Sales

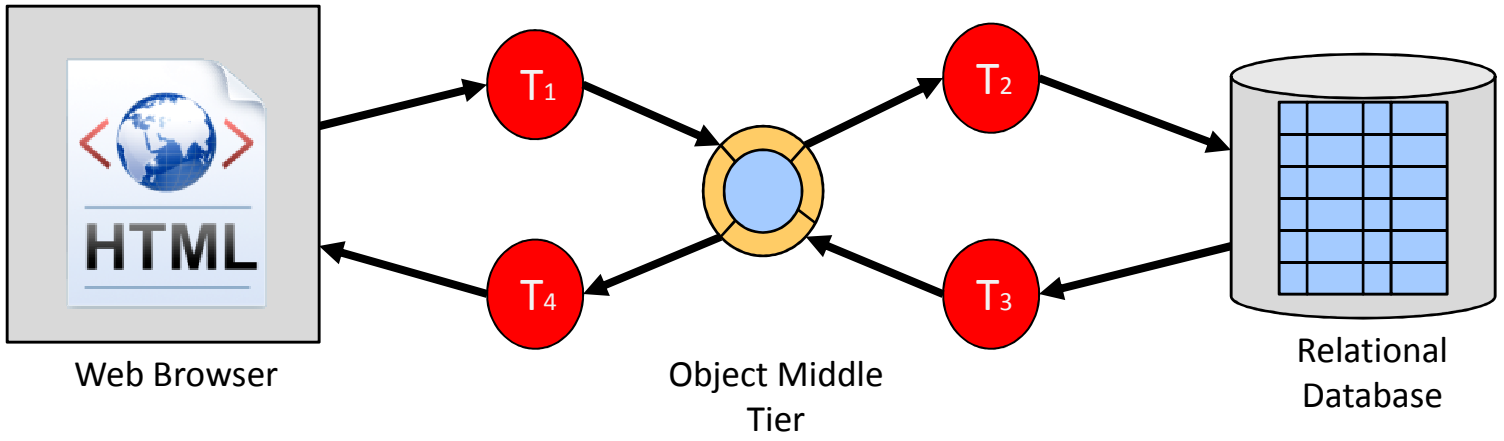


Source: © Wikibon Big Data Model 2011-2017

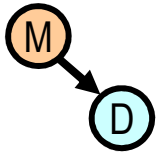
Document Structure



Object Relational Mapping

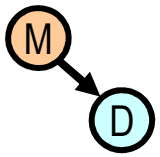


- T_1 – HTML into Objects
- T_2 –Objects into SQL Tables
- T_3 – Tables into Objects
- T_4 – Objects into HTML

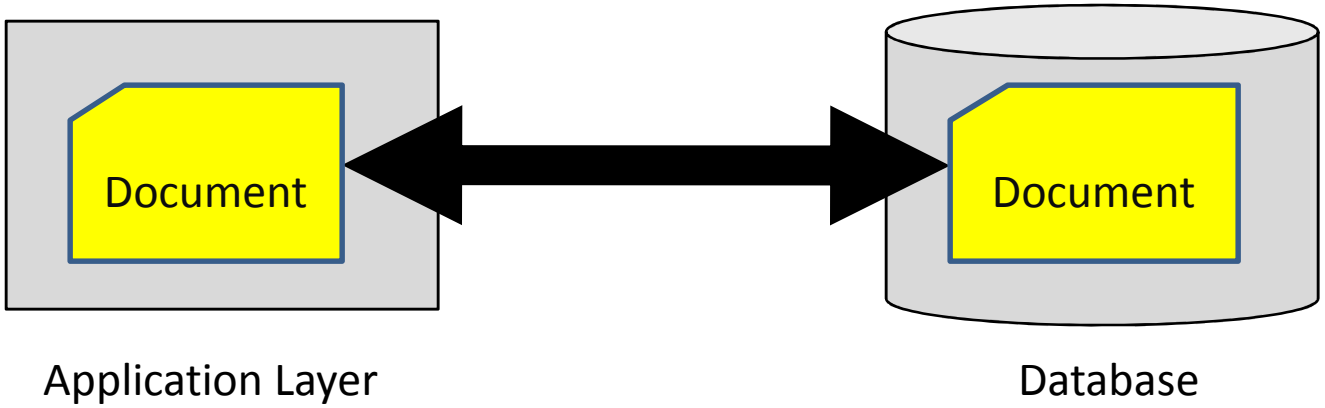


"The Vietnam of Applications"

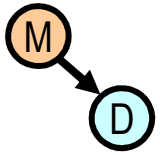
- Object-relational mapping has become one of the most complex components of building applications today
- A "Quagmire" where many projects get lost
- Many "heroic efforts" have been made to solve the problem
  - Java Hibernate Framework
  - Ruby on Rails
- But sometimes the **best** way to avoid complexity is to keep your architecture very simple



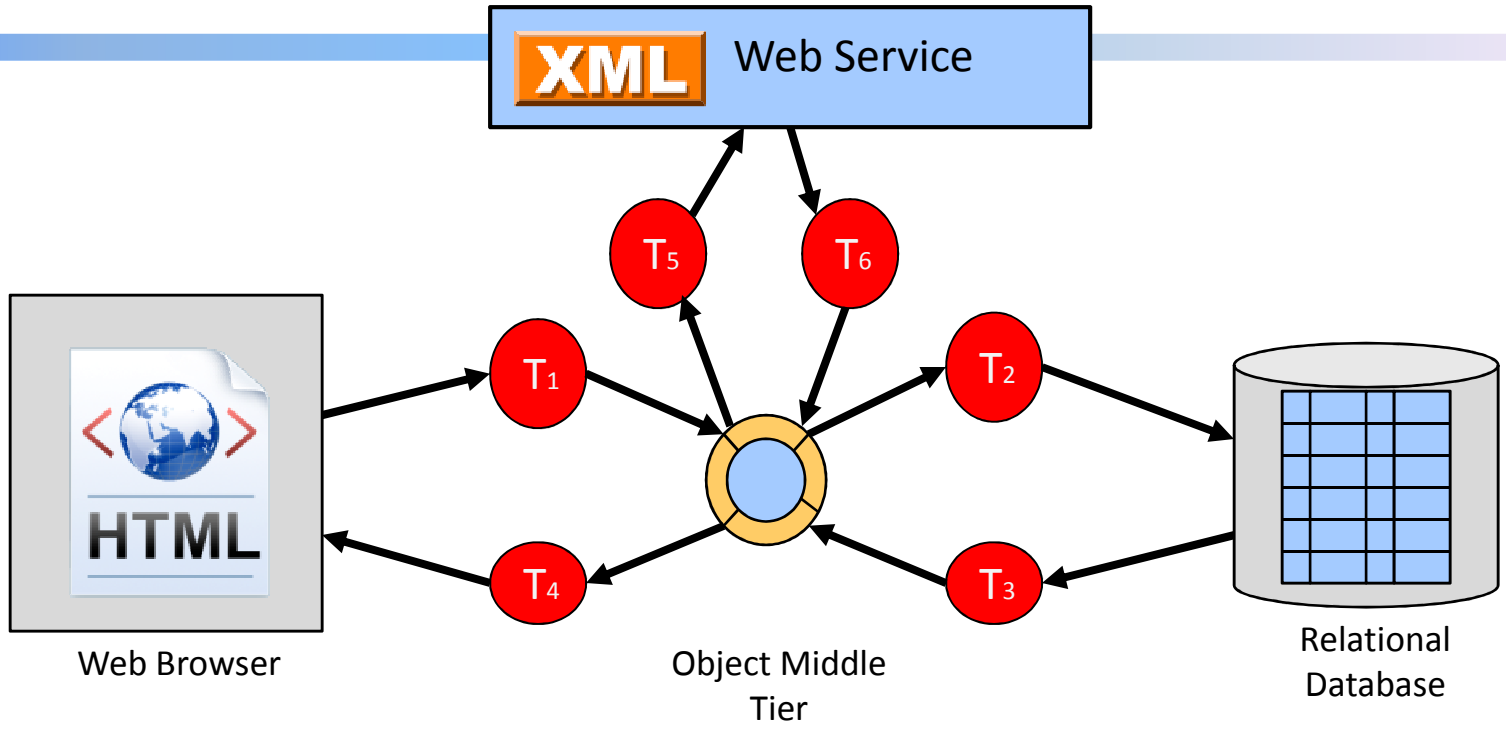
Document Stores Need No Translation



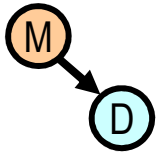
- Documents in the database
- Documents in the application
- No object middle tier
- No "shredding"
- No reassembly
- Simple!



The Addition of XML Web Services

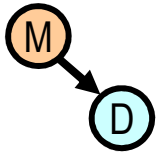


- T1 – HTML into Java Objects
- T2 – Java Objects into SQL Tables
- T3 – Tables into Objects
- T4 – Objects into HTML
- T5 – Objects to XML
- T6 – XML to Objects



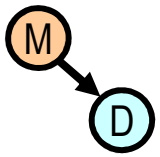
Zero Translation (XML)

- XML lives in the web browser (**X**Forms)
- **R**EST interfaces
- XML in the database (Native XML, **X**Query)
- **XRX** Web Application Architecture
- No translation!

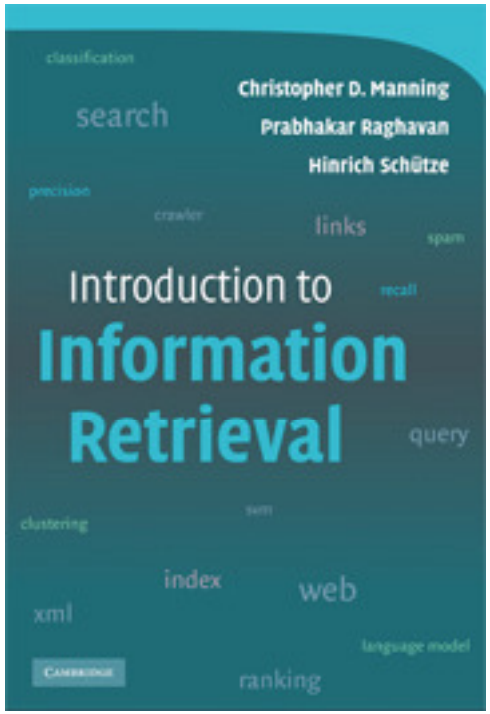


Structured Search

- What happens if you retain document structure when you index your documents?
- How can use document structure to increase your precision and relevancy?



Information Retrieval Textbook



Introduction to Information Retrieval

by Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze

Cambridge University Press, 2008

http://nlp.stanford.edu/IR-book/information-retrieval-book.html

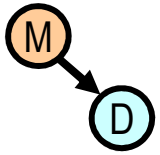
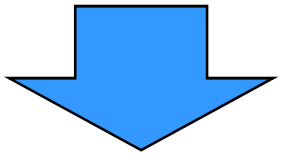


Table 10.1



	RDB search	unstructured retrieval	structured retrieval
objects	records	unstructured documents	trees with text at leaves
model	relational model	vector space & others	?
main data structure	table	inverted index	?
queries	SQL	free text queries	?

XML - Table 10.1 and structured information retrieval. SQLRDB (relational database) search, unstructured information retrieval

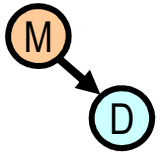
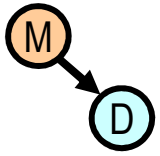


Table 10.1 - Revised

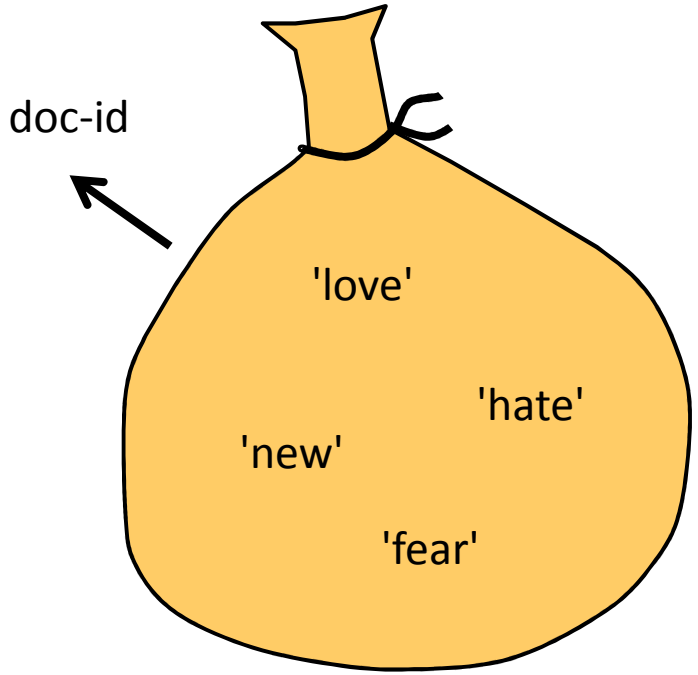
	RDB search	unstructured retrieval	structured retrieval
objects	records	unstructured documents	trees with text at leaves
model	relational model	vector space & others	**XML hierarchy**
main data structure	table	inverted index	**trees with node-ids for document ids**
queries	SQL	free text queries	**XQuery fulltext**

XML - Table 10.1 and structured information retrieval. SQLRDB (relational database) search, unstructured information retrieval



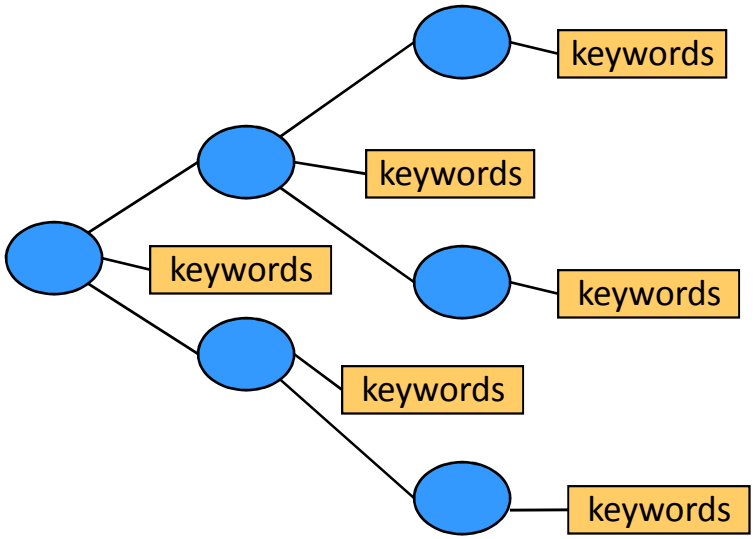
Two Models

"Bag of Words"

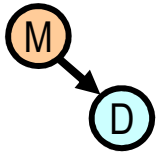


- All keywords in a single container
- Only count frequencies are stored with each word

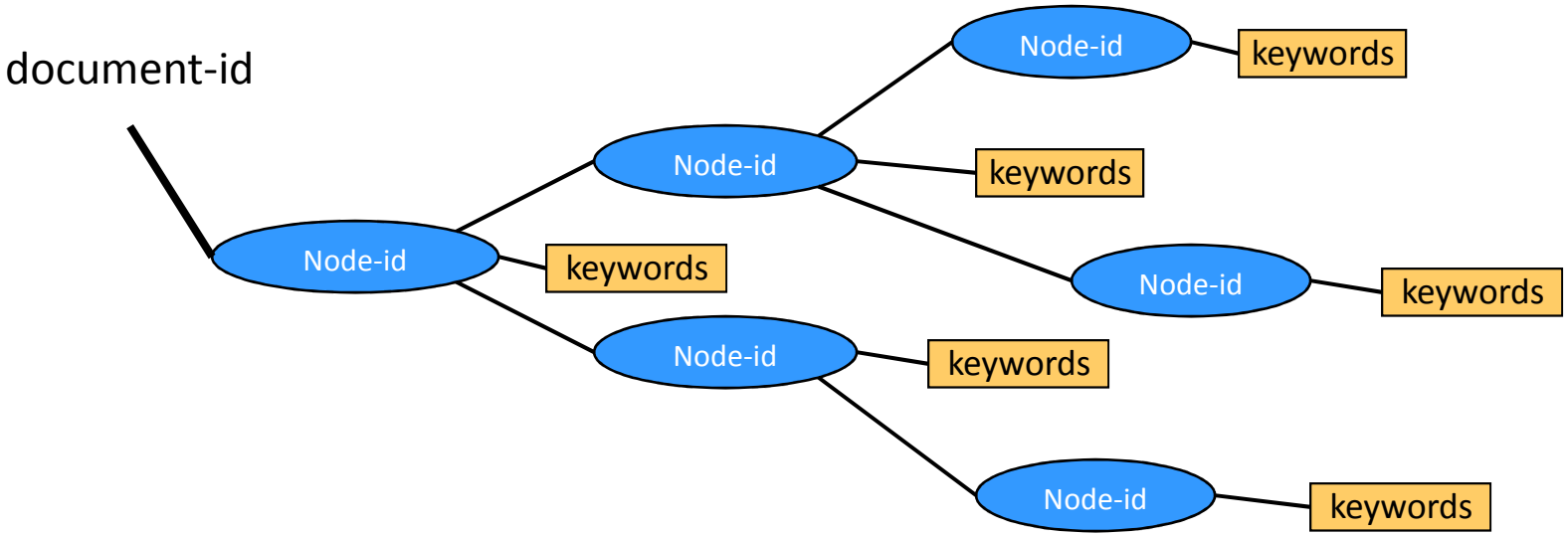
"Retained Structure"



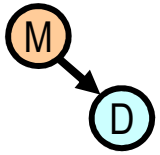
- Keywords associated with each sub-document component



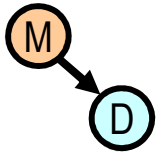
Keywords and Node IDs



- Keywords in the reverse index are now associated with the **node-id** in every document

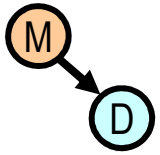


- Originally created by 10gen
  - Originally New York, now CA
- Open Source License (AGPL)
- Document/collection centric
- Sharding built-in, automatic
- Stores data in binary JSON format BSON
- Native Query language is JSON-like
- Implemented in C++ with many APIs (C++, JavaScript, Java, Perl, Python etc.)

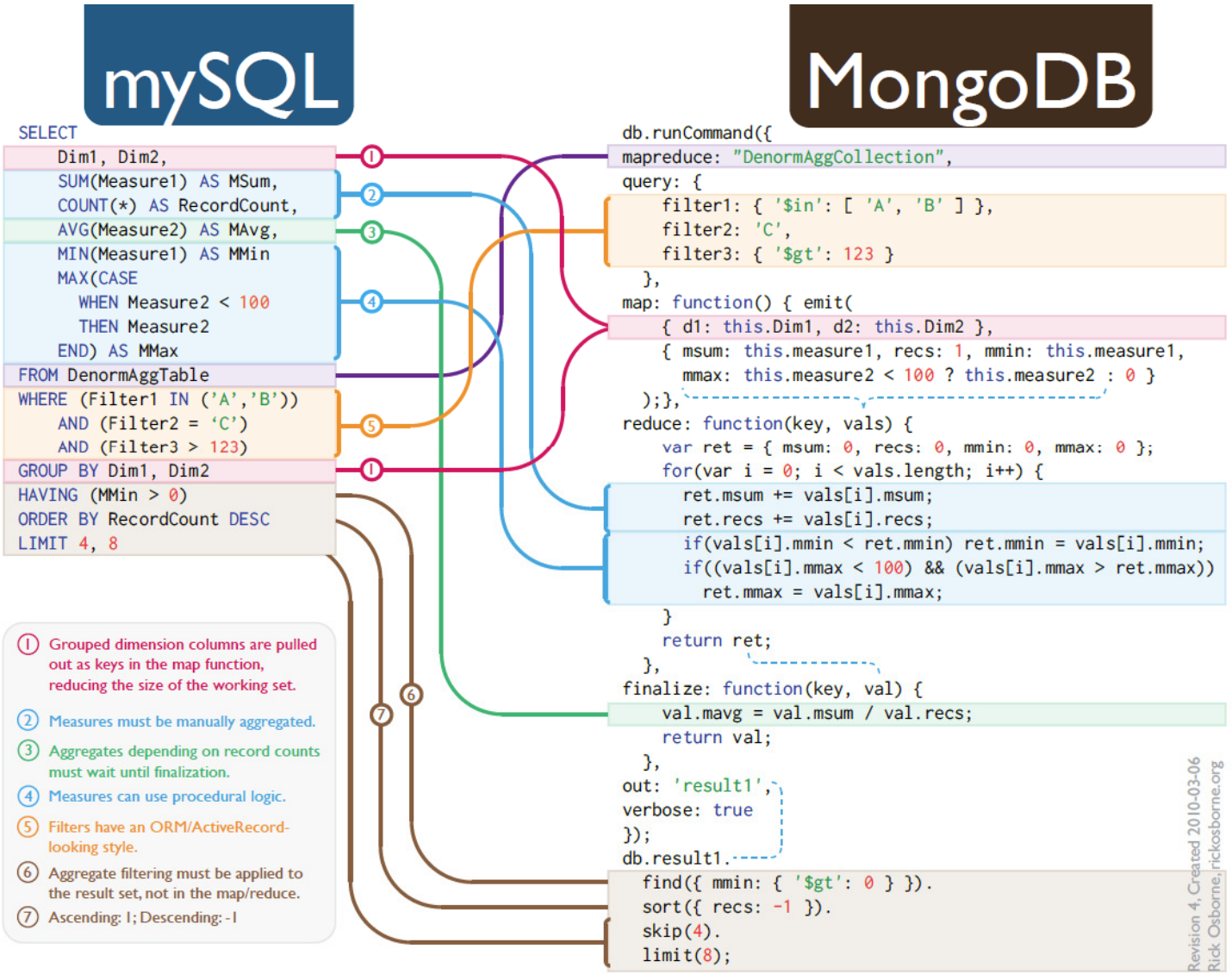


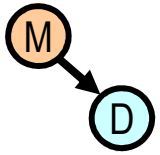
Why is Mongo so popular?

- Agility!
  - Document stores eliminate Object/Relation mapping
  - No more Hibernate/RAILs complexity
- Developers love JSON
  - fewer complexities of XML
  - familiar to JavaScript developers
- Open source license
- Excellent developer relationships

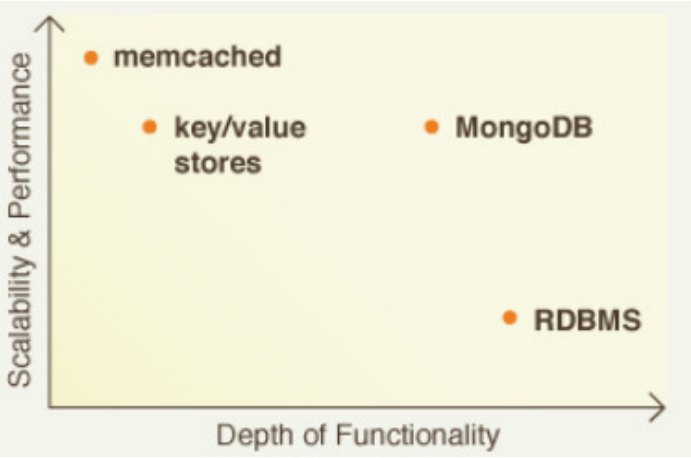


Mongo MapReduce

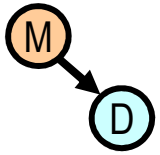




Mongo vs. key/value stores

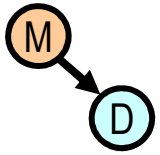


- More functionality than simple key value stores
- Functionally similar to a RDBMS
- Many features for big-data systems



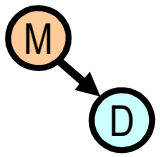
Couchbase

- Open source document store supported by Couchbase, Inc.
  - Code base no longer directly related to CouchDB!
  - Code base built on memcached
- Written in ERLANG/C++
- RESTful JSON API
- Distributed, featuring robust, incremental replication with bi-directional conflict detection and management
- Strong integration with ElasticSearch for document search

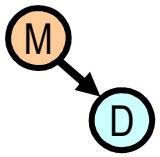


CouchDB

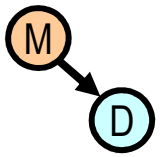
- Apache open source document store (JSON)
- Strong focus on building web applications (REST, JavaScript on server)
- Strong support for high-availability, replication and scalability
- Written in Erlang
- Support for mobile versions
- Strong developer community



- Native XML database designed to scale to Petabyte data stores
- Leverages commodity hardware
- ACID compliant, schema-free document store
- Heavy use by federal agencies, document publishers and "high-variability" data
- Arguably the most successful NoSQL company



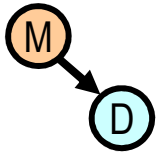
- Open source native (LGPL) XML database with strong support in Europe
- Strong support for XQuery and XQuery extensions
- Heavily used by the Text Encoding Initiative (TEI) community and XRX/XForms communities
- Integrated Lucene search
- Collection triggers and versioning
- Extensive XQuery libraries (EXPath)
- Version 2.0 also has replication



Hybrid architectures

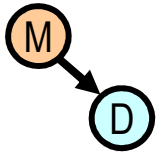
Image: scienceblogs

- Most real-world implementations use multiple NoSQL solutions together in an application
- Example:
  - Use document stores for data
  - Use key-value store for distributed cache
  - Use S3 for image/pdf/binary storage
  - Use Apache Lucene for document index stores
  - Use MapReduce for real-time index and aggregate creation and maintenance
  - Use OLAP for reporting sums and totals



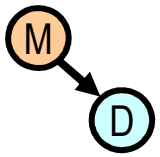
Other Terms

- **Merkle Trees** – a binary hash tree – where a hash function is used for each leaf and branches up to the root of a file system – fast ways to see if complex documents in tree structures have changed
- **B-Trees, B+ Trees** – binary trees that are used to create fast searches of large data sets
- **Vector Clocks** – an algorithm for detecting dependencies between remote computer systems – used to track what systems may have corrupted data
- **Bloom Filters** – hashing tricks to avoid unnecessary disk operations on large data sets
- **Consistent Hashing** – A way of converting strings of data into a unique key that can be used for fast comparisons

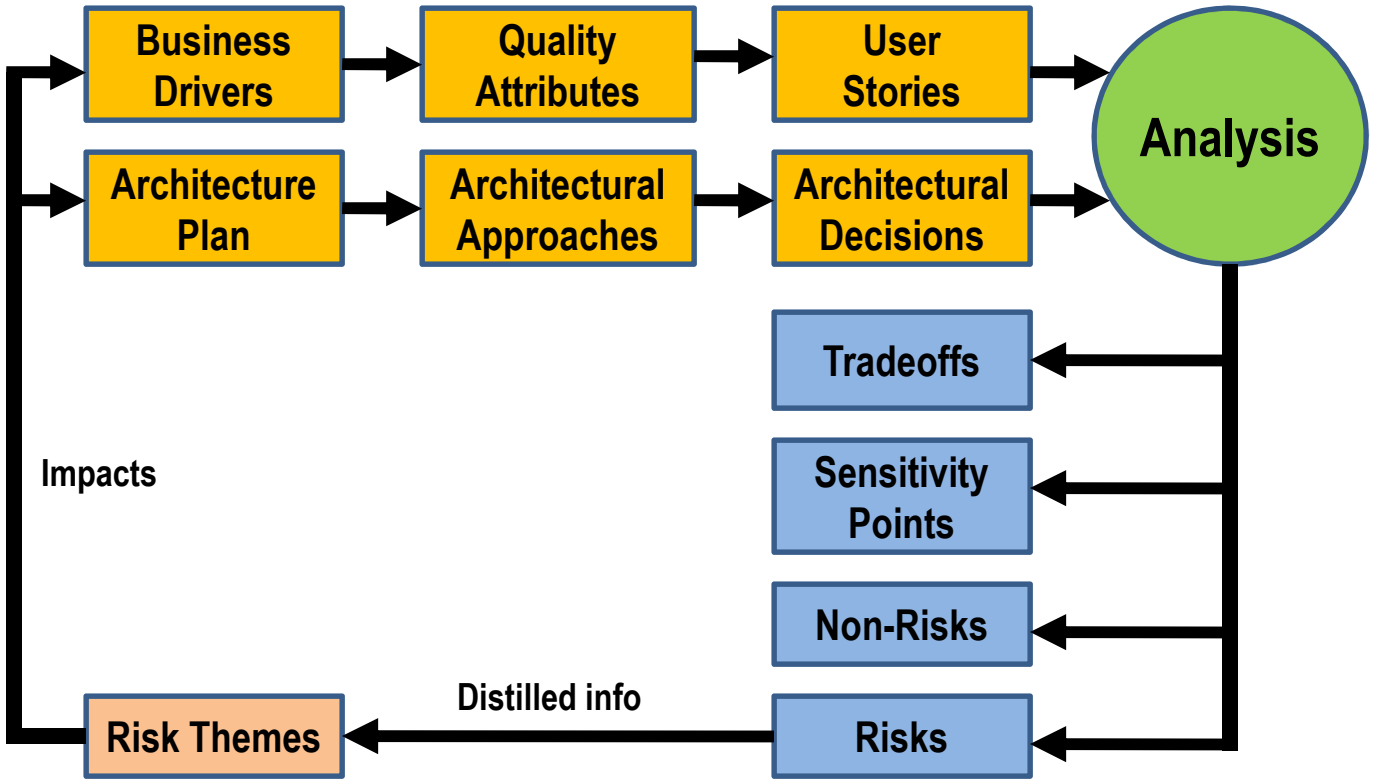


Tools to Help You Select A System

- **ATAM** – Architecture Tradeoff and Modeling
- CMU developed process to objectively select a system architecture based on business driven use-cases and quality metrics

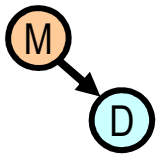


ATAM Process Flow

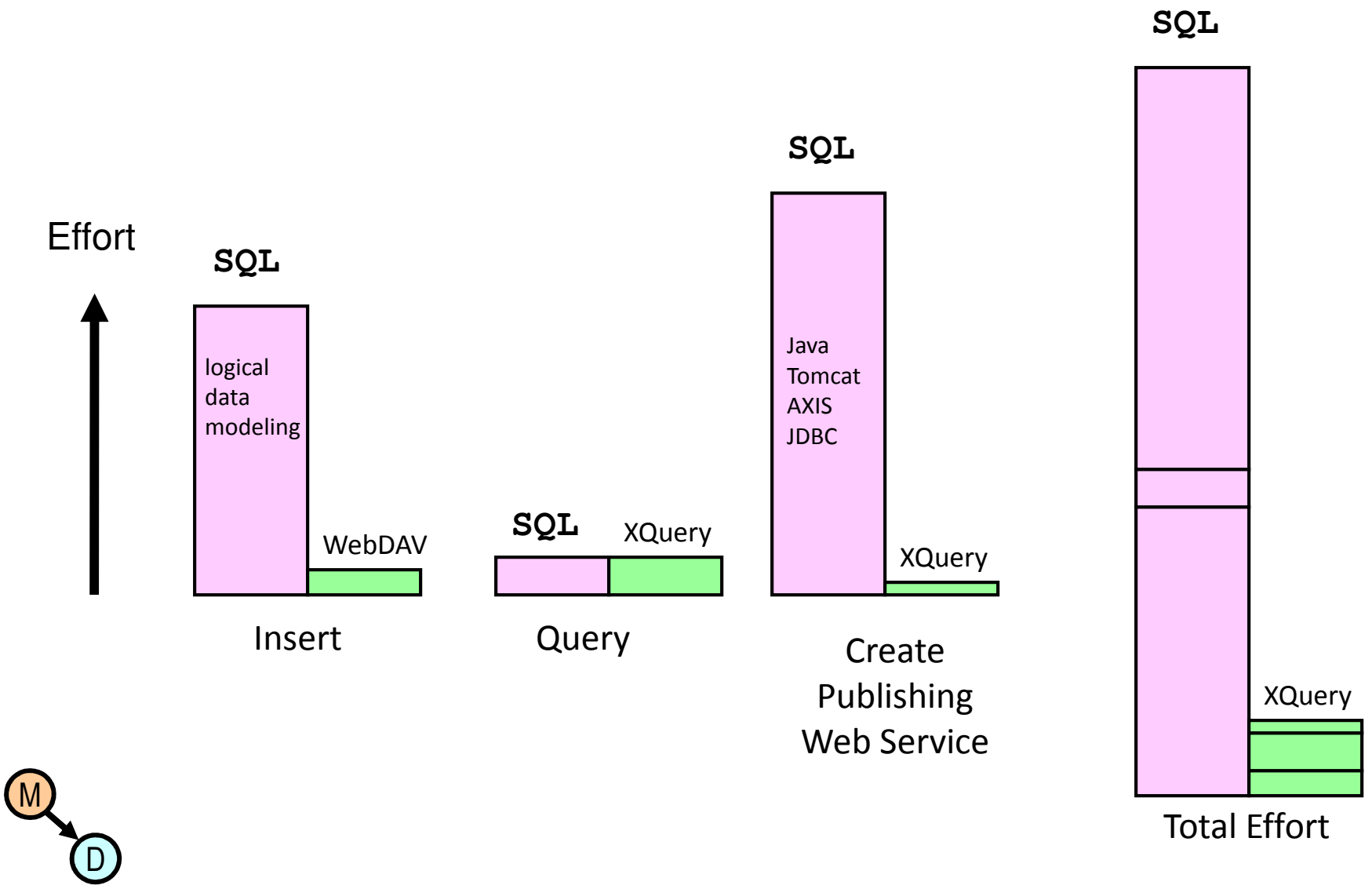


Sample Use Cases/User Stories

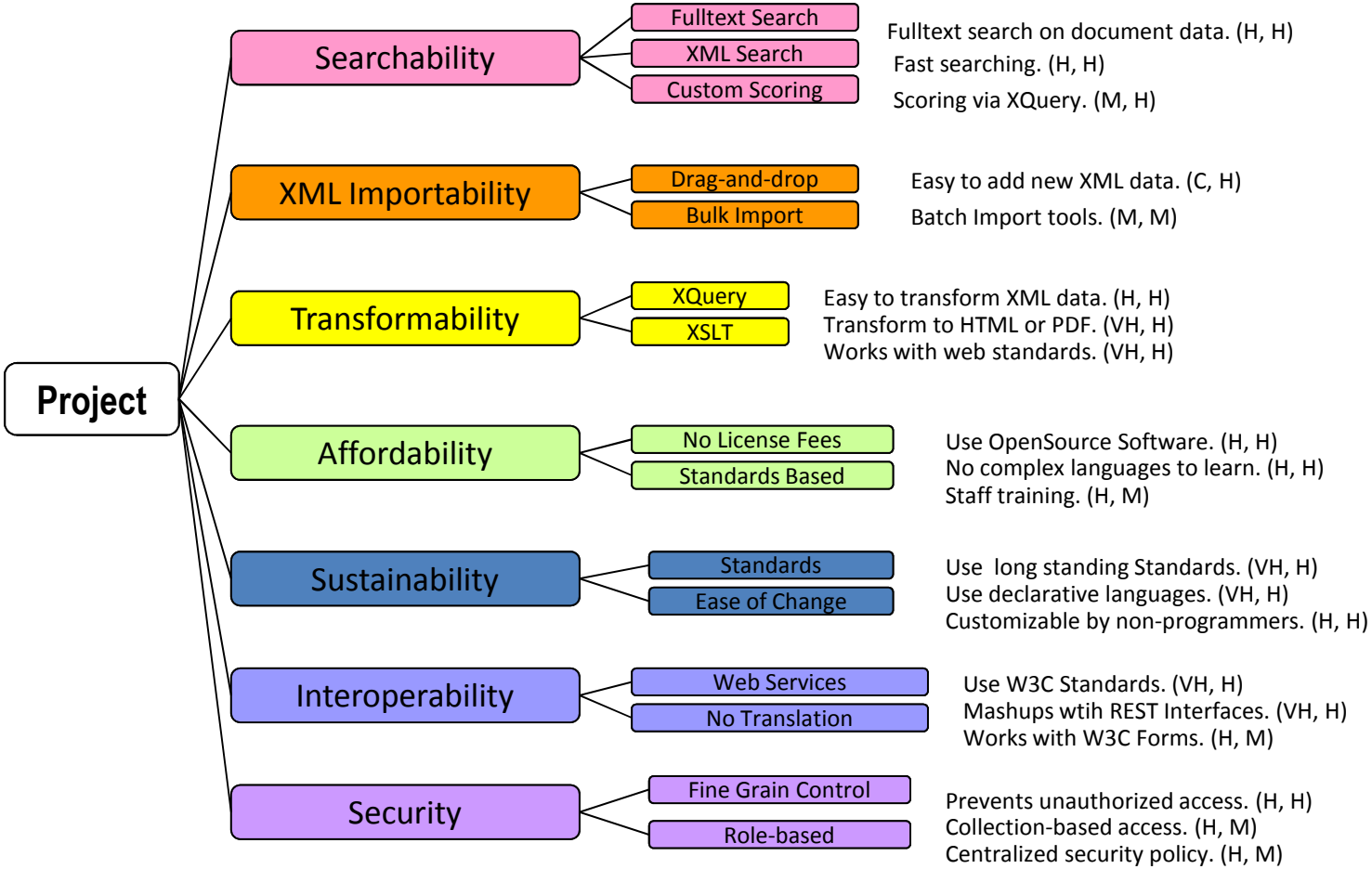
- Effort to Load New Data
- Effort to Query Data
- Effort to Transform Data
- Effort to Create RESTful Web Services



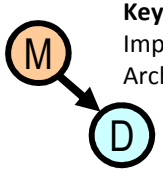
Insert/Select/Publish Comparison



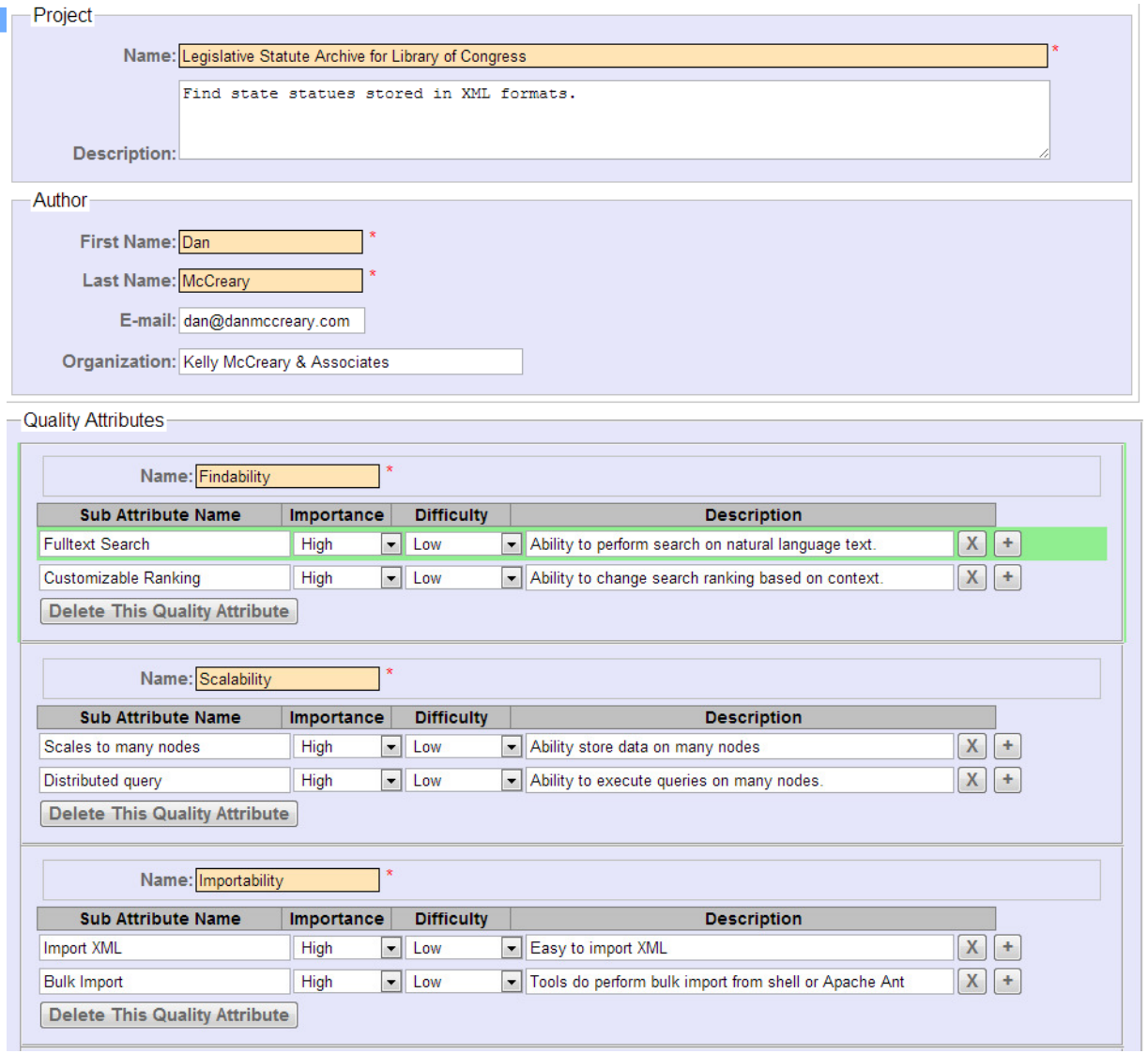
Sample Quality Tree

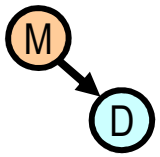


Key: (Importance, Score)
Important to the Project C=Critical, VH=Very High, H=High, M=Medium
Architectural Score H=High, M=Medium, L=Low



Quality Attribute Tree App

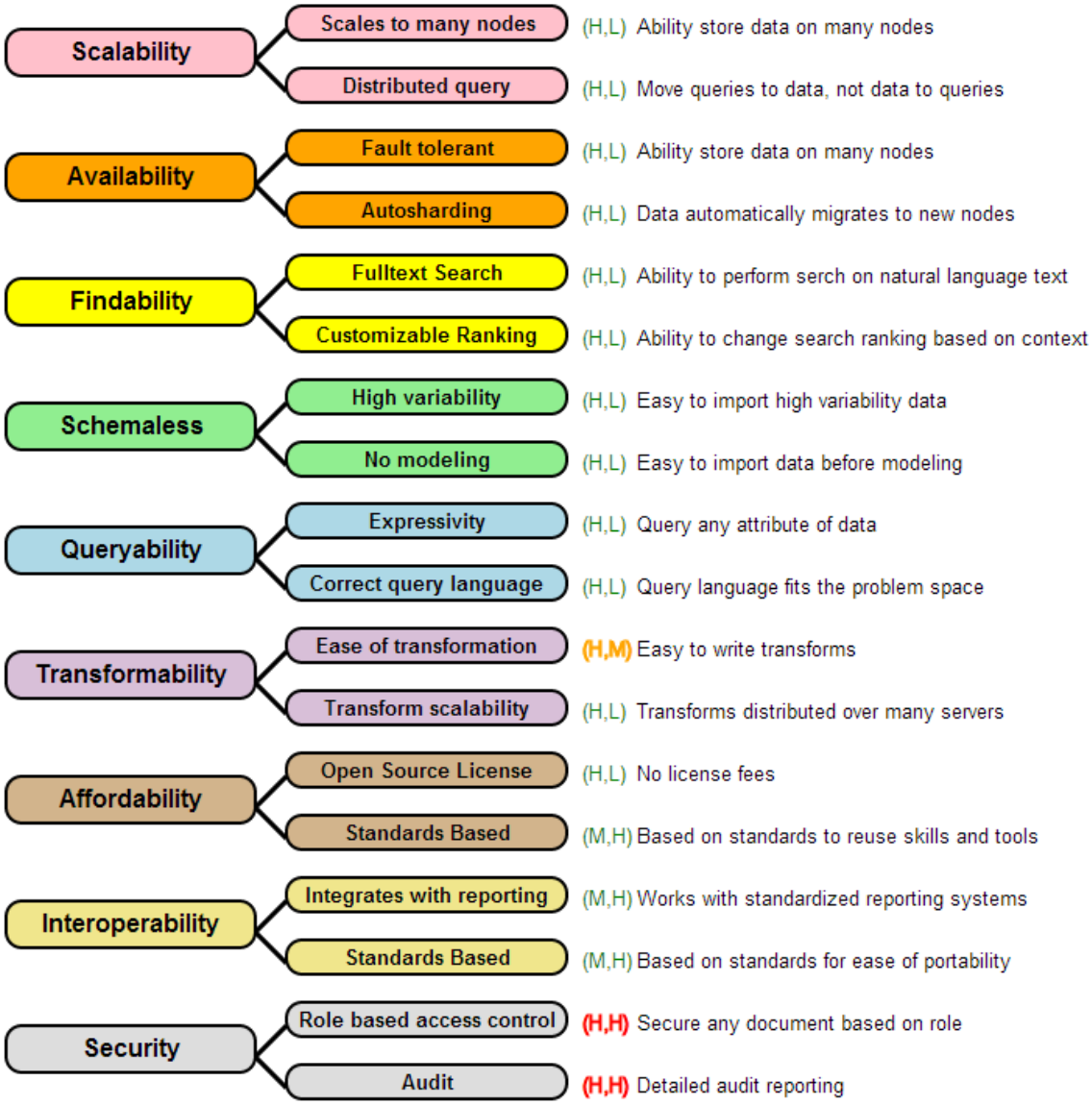




Quality Attribute Utility Tree
Project: Typical Big Data Project

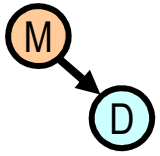
Author: Dan McCreary

Description: Requirements for a typical Big Data project that has a strong focus on scalability and availability

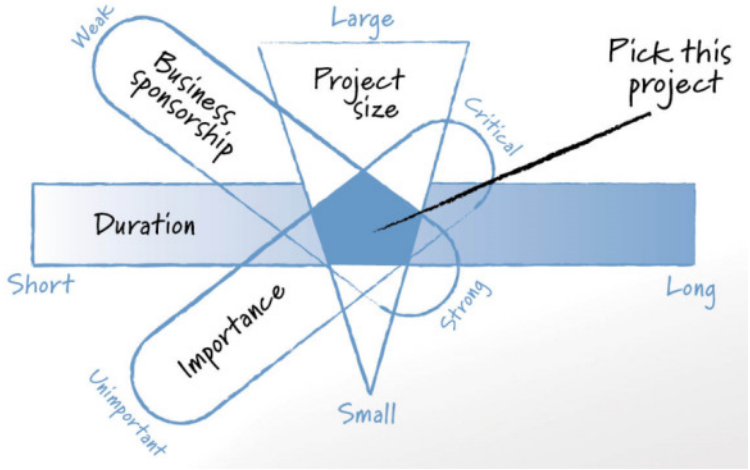


The Hard Part

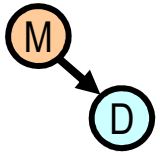
- Assigning organizational "value" to
  - Ease of use
  - Standards
  - Vendor viability
  - Open source project viability
  - Portability
- Example:
  - WebDAV support



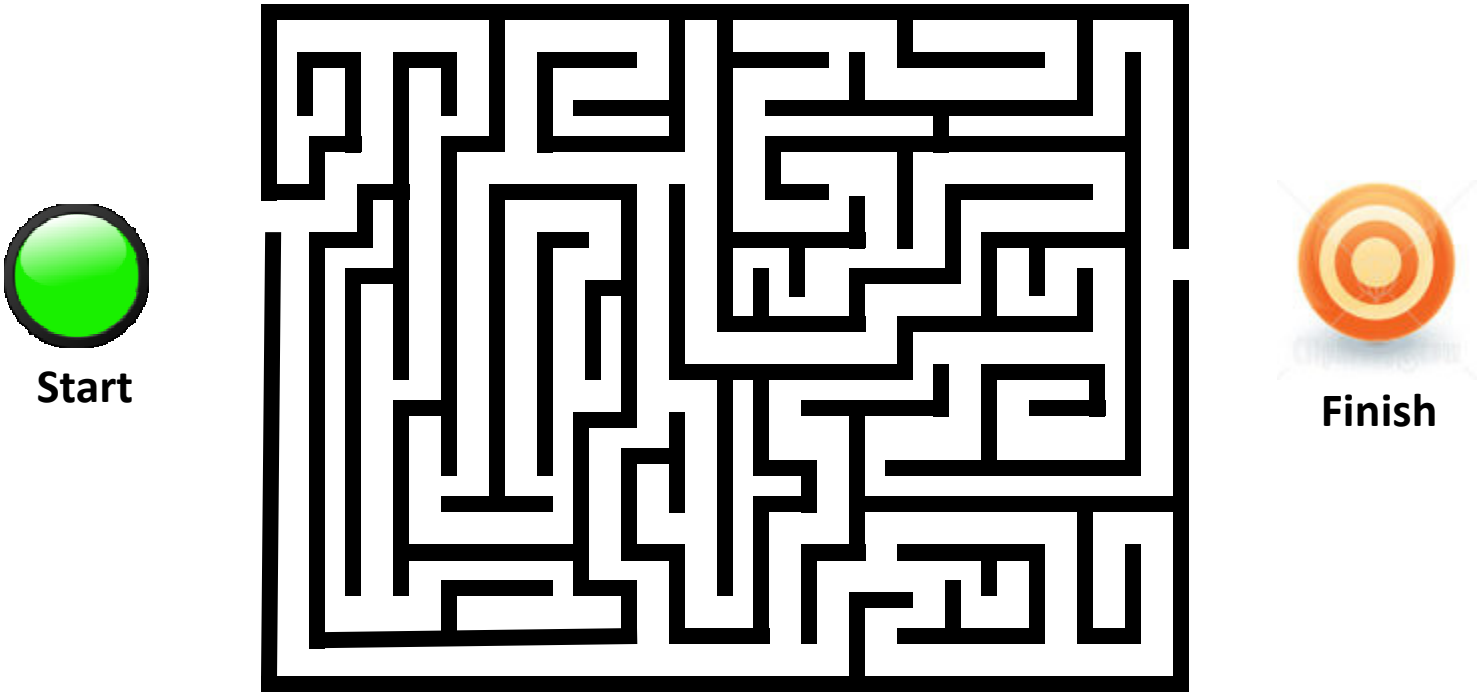
Selecting a NoSQL Pilot Project



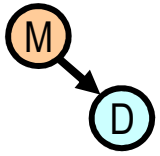
- The "Goldilocks Pilot Project Strategy"
- Not too big, not too small, just the right size
- Duration
- Sponsorship
- Importance
- Skills
- Mentorship



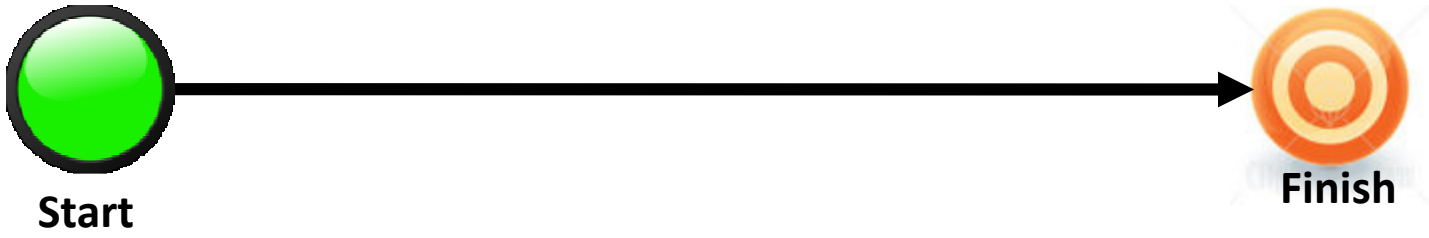
Using the Wrong Architecture



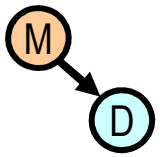
Credit: Isaac Homelund – MN Office of the Revisor



Using the Right Architecture



Find ways to remove barriers and empower the non programmers on your team.

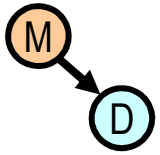


If You Give a Kid a Hammer…

…the whole world becomes a nail

- People solve problems using familiar tools
- People develop specific *Cognitive Styles** based on training and experience
- What are we teaching the next generation of developers?

* Source: Shoshana Zuboff: In the Age of the Smart Machine (1988)

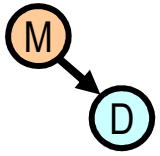


The Future of the NoSQL Movement

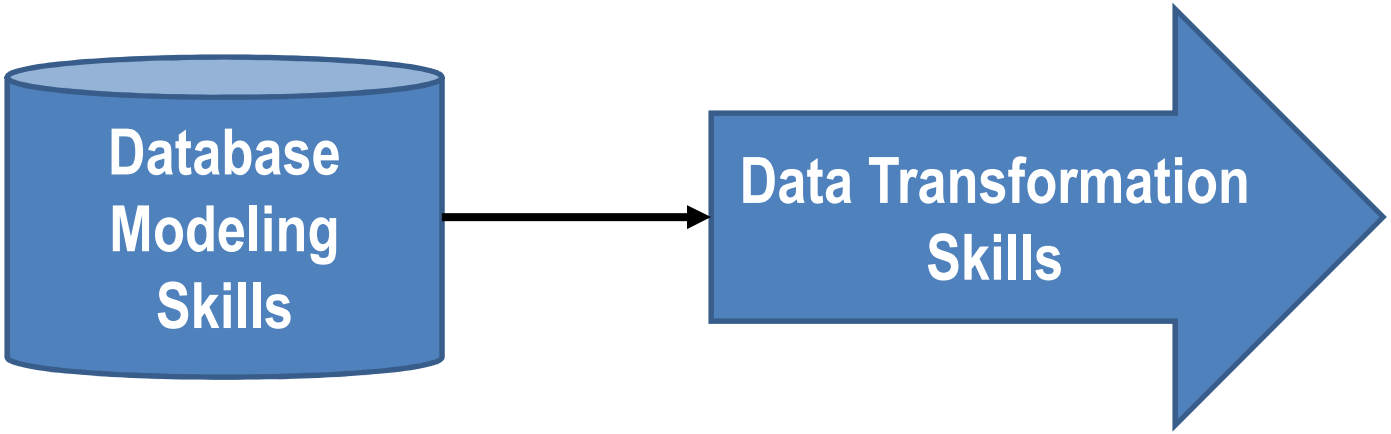
Growth

Diversity

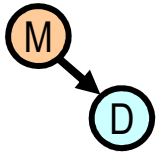
- Will data sets continue to grow at exponential rates?
- Will new system options become more diverse?
- Will new markets have different demands?
- Will some ideas be "absorbed" into existing RDBMS vendors products?
- Will the NoSQL community continue to be the place where new database ideas and products are incubated?
- Will there be standards in NoSQL?
- Will the job of doing high-quality architectural tradeoffs analysis become easier?



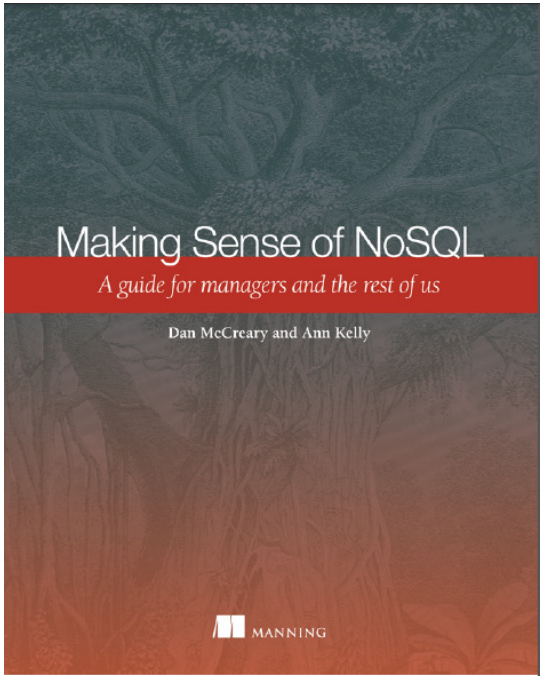
Shift of Skills



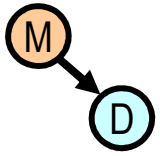
- Import public data sets (no shredding into rows!)
- Natural language processing
- Export (ad-hoc query of unstructured data)
- Search
- Streaming
- Machine learning
- Summarization
- Analysis and discovery



Making Sense of NoSQL



http://manning.com/mccreary



Questions

Thank You!

Dan McCreary
dan@danmccreary.com