Database Architectures

Charles Severance

Usually

Database Normalization (3NF)

There is *tons* of database theory - way too much to understand without excessive predicate calculus

- Do not replicate data. Instead, reference data. Point at data.
- Use integers for keys and for references.
- Add a special “key” column to each table, which you will make references to.

http://en.wikipedia.org/wiki/Database_normalization

To SQL or no to SQL?

That is the question.. Or is it?

Relational or Not?

Rows and Columns vs. Documents, Keys, and Values

ACID or BASE?

Probably the best question to ask.

A	••Atomicity
C	••Consistency
I	••Isolation
D	••Durability

https://en.wikipedia.org/wiki/ACID

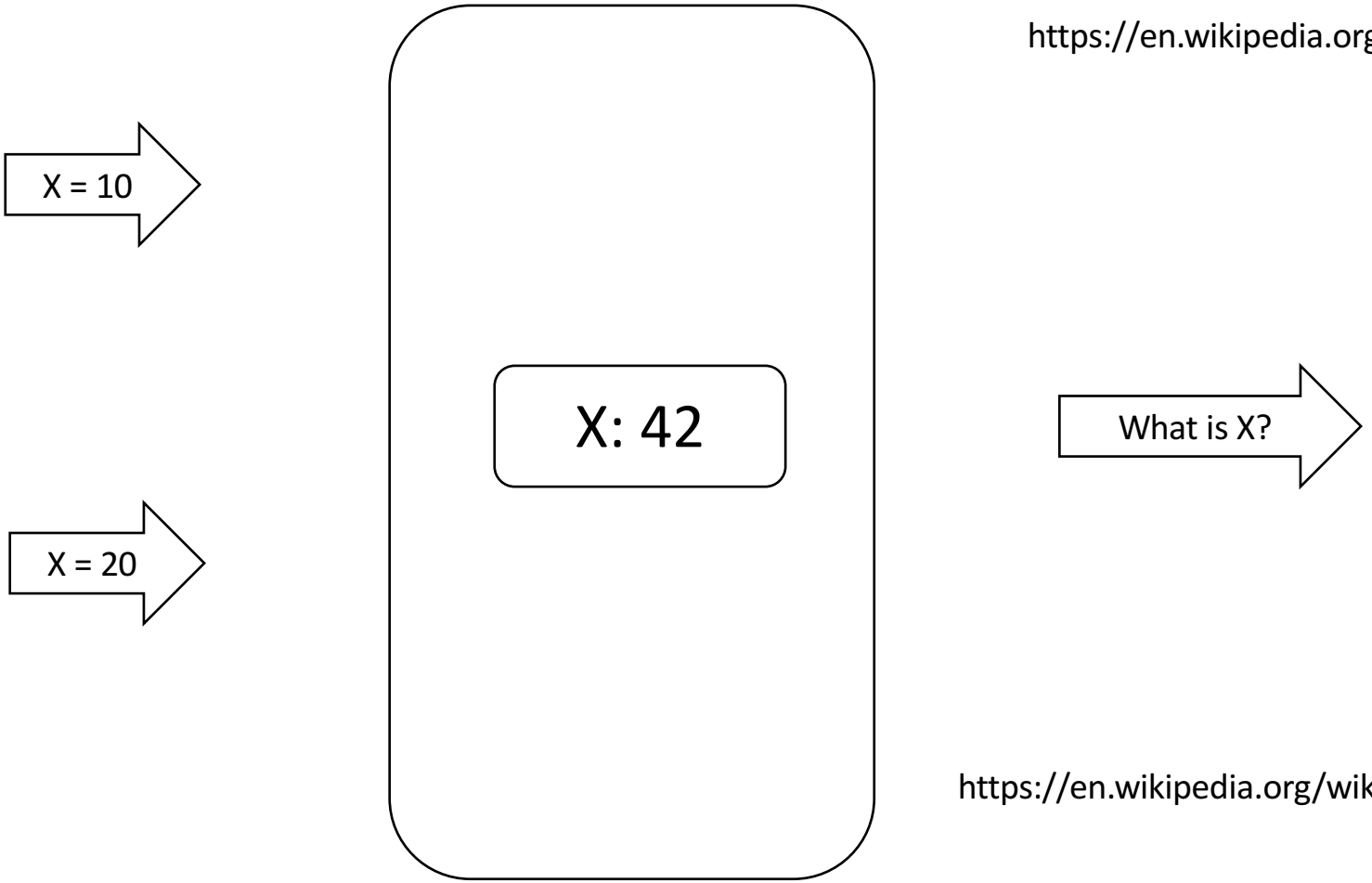
B ••Basically

A ••Available

S ••Soft state

E ••Eventual consistency

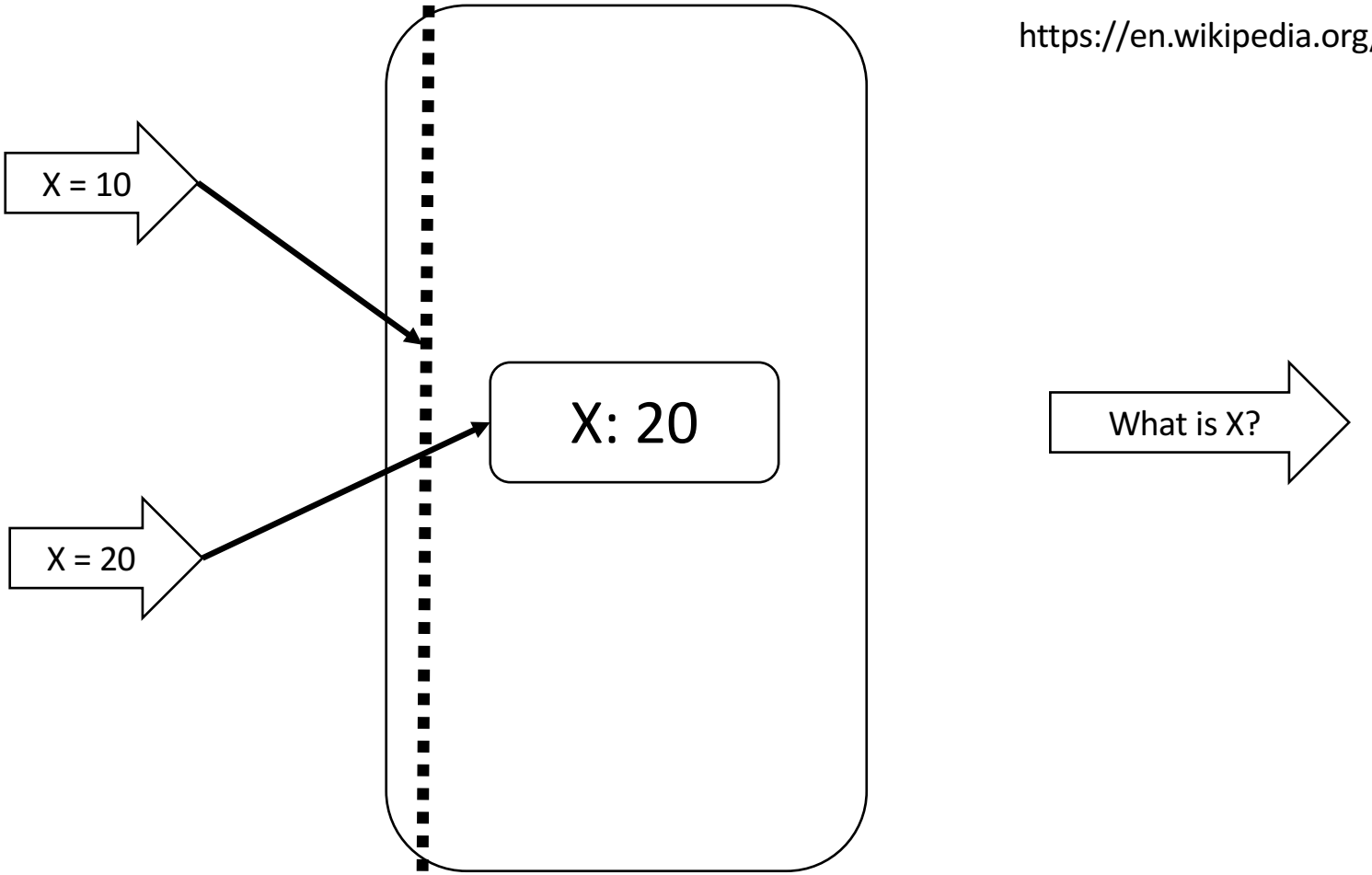
https://en.wikipedia.org/wiki/Eventual_consistency



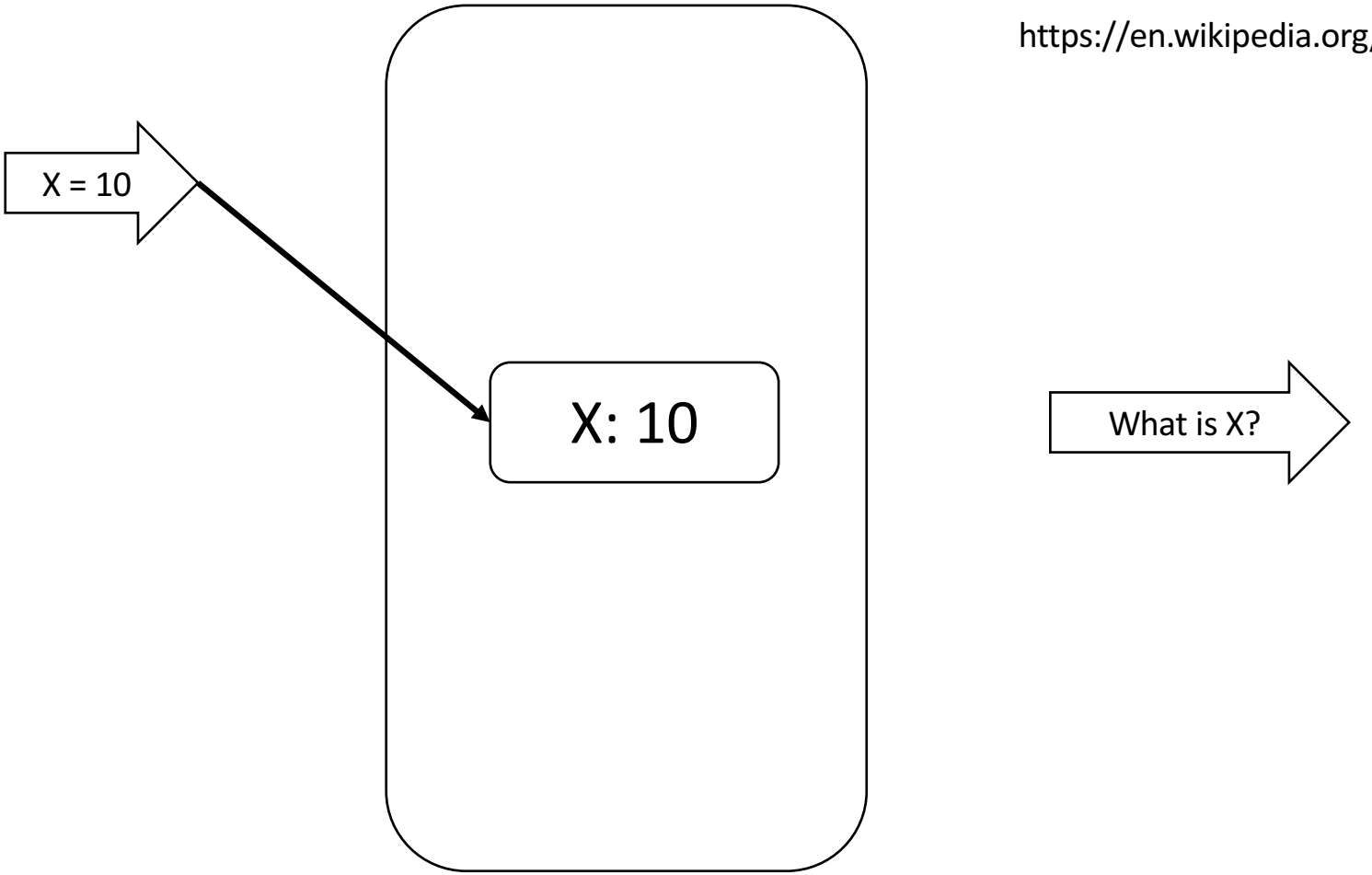
https://en.wikipedia.org/wiki/ACID

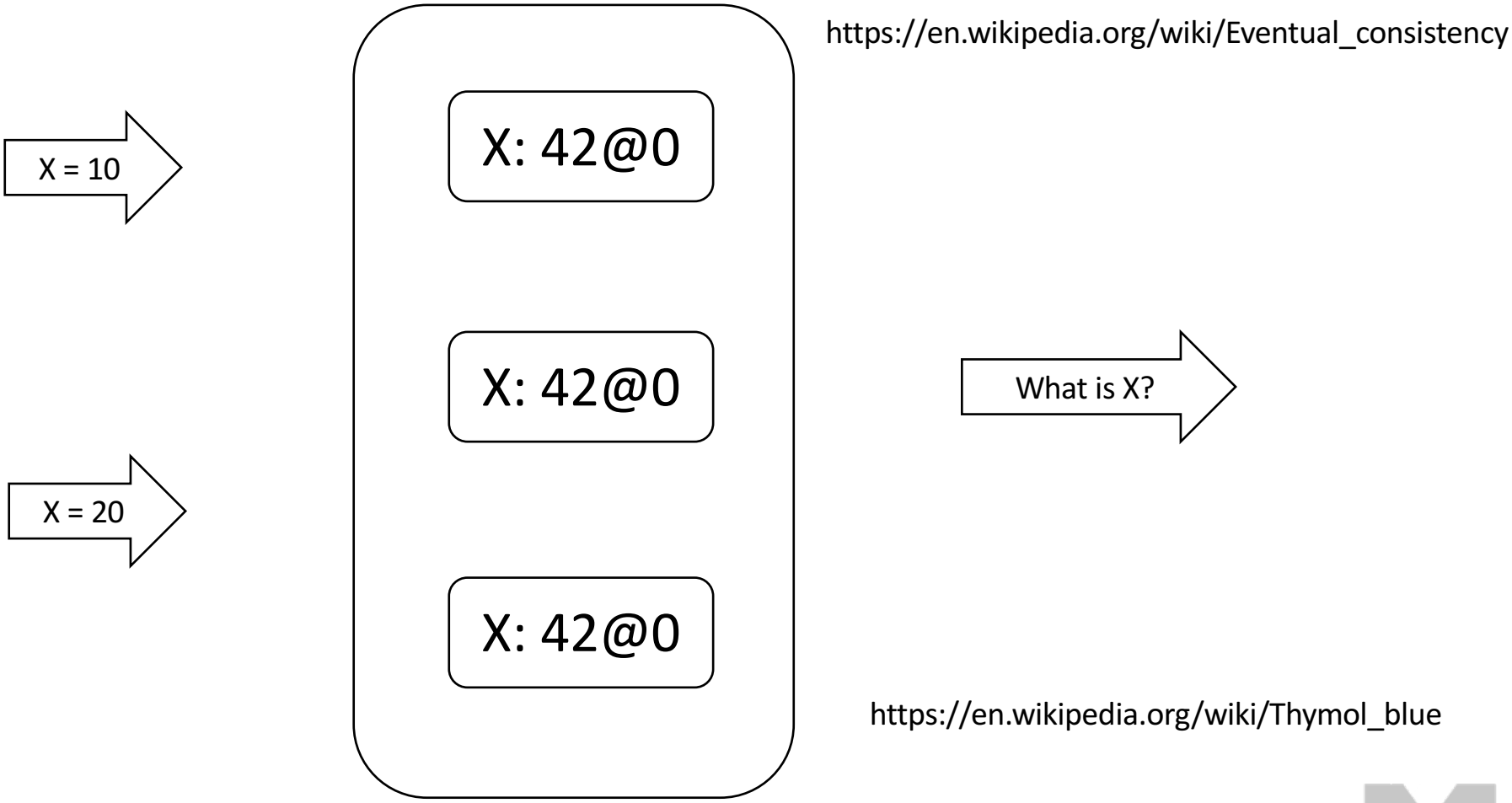
https://en.wikipedia.org/wiki/Thymol_blue

https://en.wikipedia.org/wiki/ACID



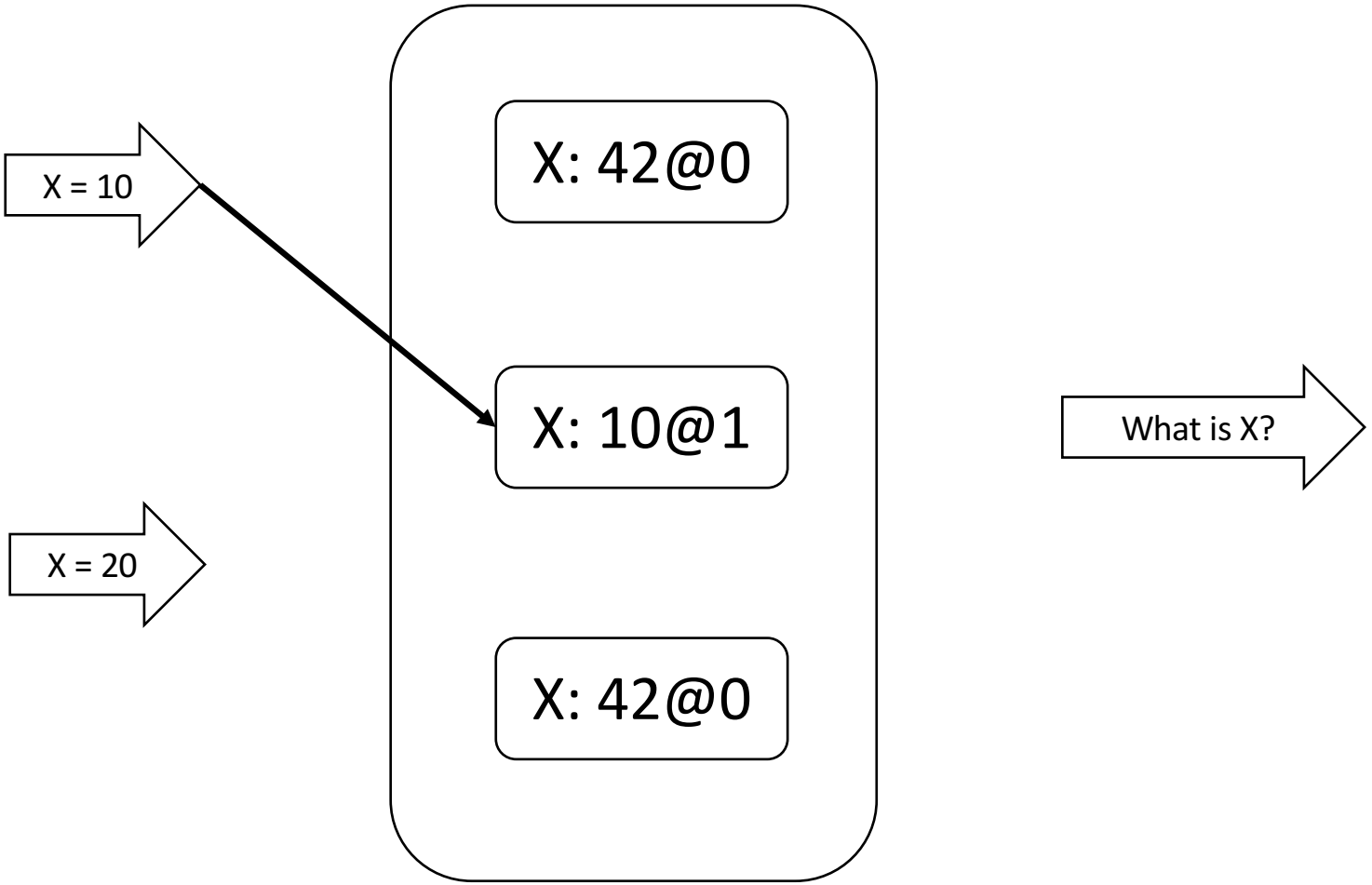
https://en.wikipedia.org/wiki/ACID

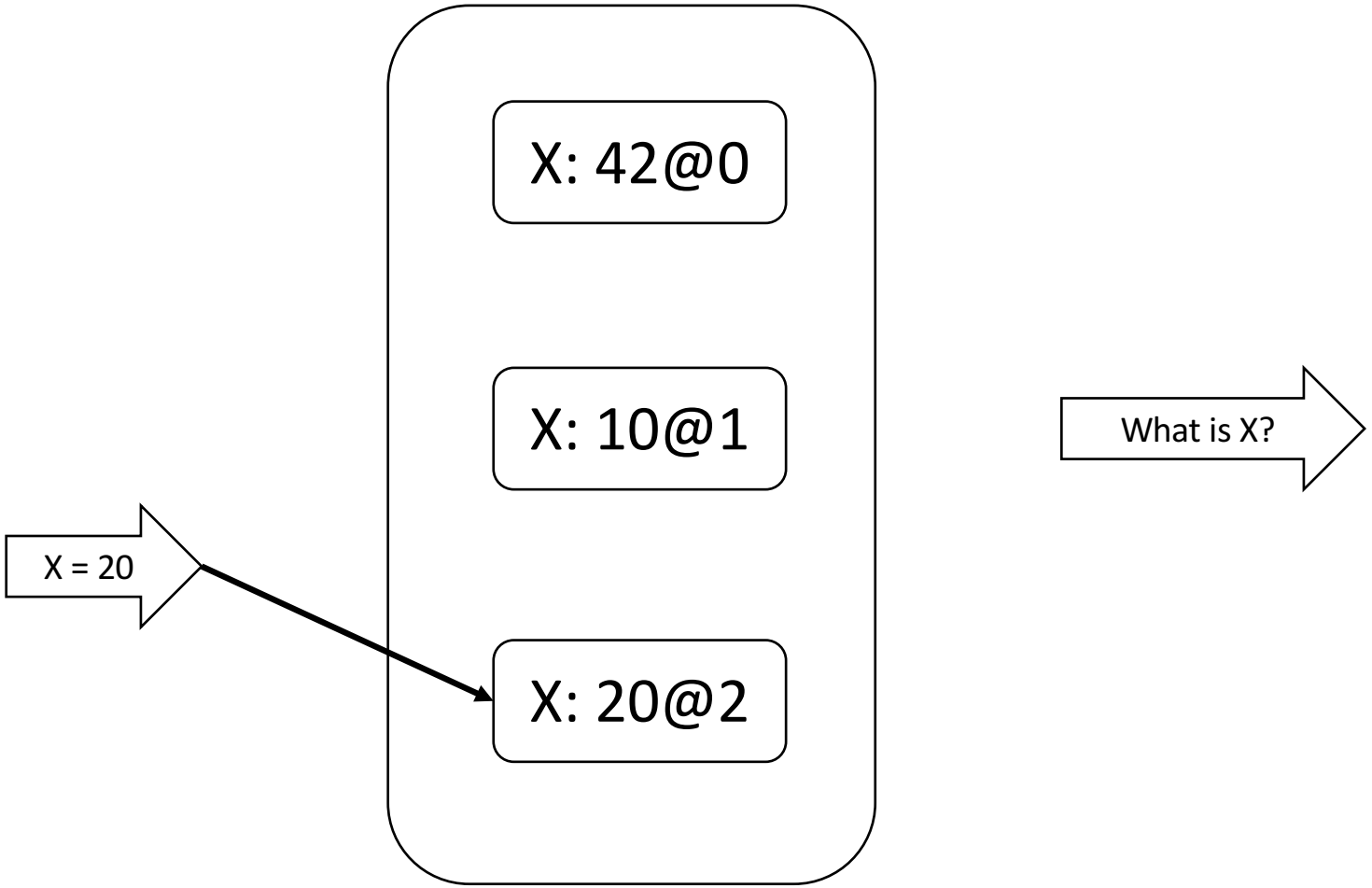


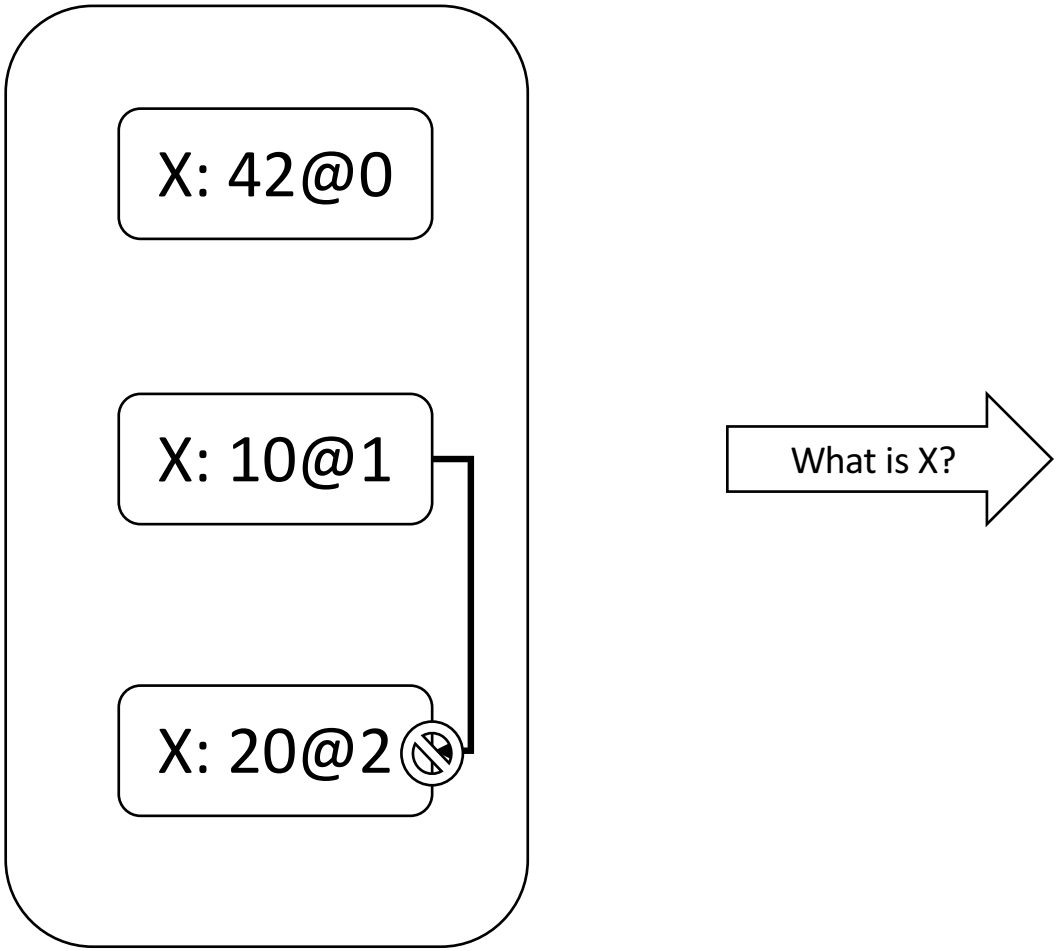


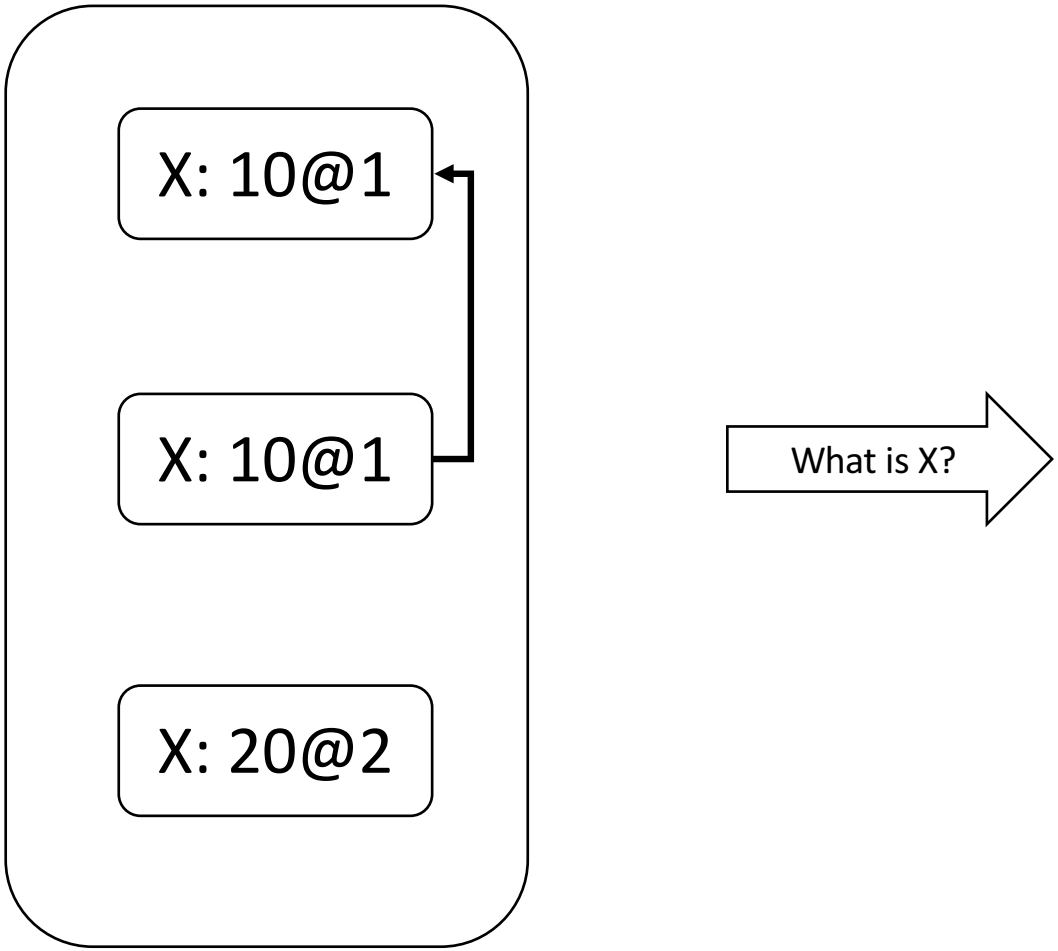
https://en.wikipedia.org/wiki/Eventual_consistency

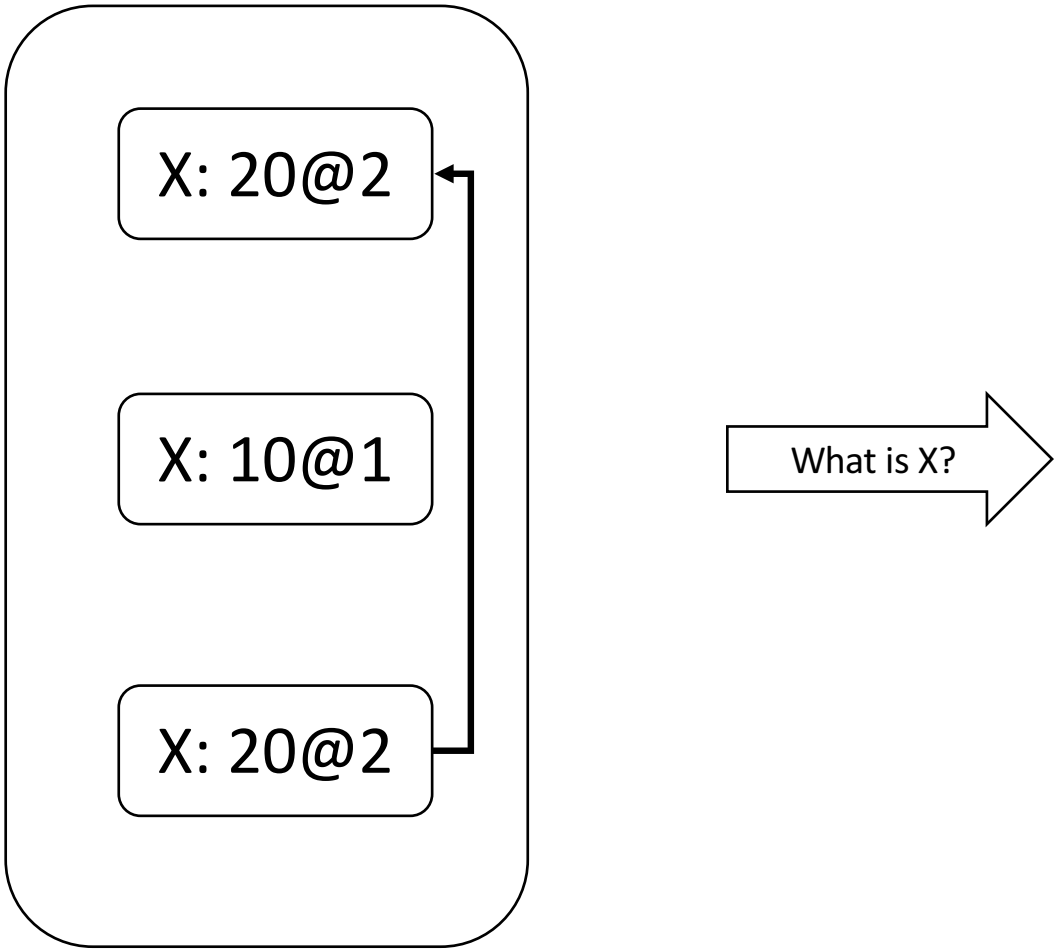
https://en.wikipedia.org/wiki/Thymol_blue

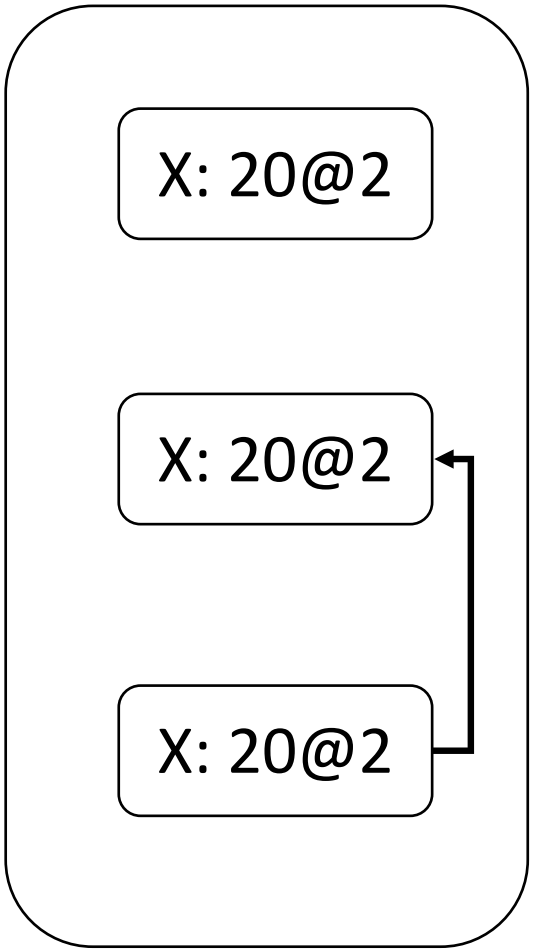

X = 10
X = 20
X: 42@0
X: 10@1
X: 42@0
What is X?


X: 42@0
X: 10@1
X: 20@2
X = 20
What is X?


X: 42@0
X: 10@1
X: 20@2
What is X?


X: 10@1
X: 10@1
X: 20@2
What is X?


X: 20@2
X: 10@1
X: 20@2
What is X?



https://en.wikipedia.org/wiki/Eventual_consistency

What is X?

https://en.wikipedia.org/wiki/Thymol_blue

Database Software

- ACID (Atomic)
  - Oracle
  - PostgreSQL
  - MySQL
  - SQLite
  - SQLServer
- BASE (Eventual)
  - Mongo
  - Casandra
  - BigTable

Compromises

- ACID (Atomic)
  - SERIAL INTEGER keys
  - Transactions
  - UNIQUE Constraints
  - "One perfect SQL Statement"
- BASE (Eventual)
  - GUIDs – Globally Unique IDs
  - Design for stale data in application
  - Application post-check and resolve
  - Retrieve and throw away

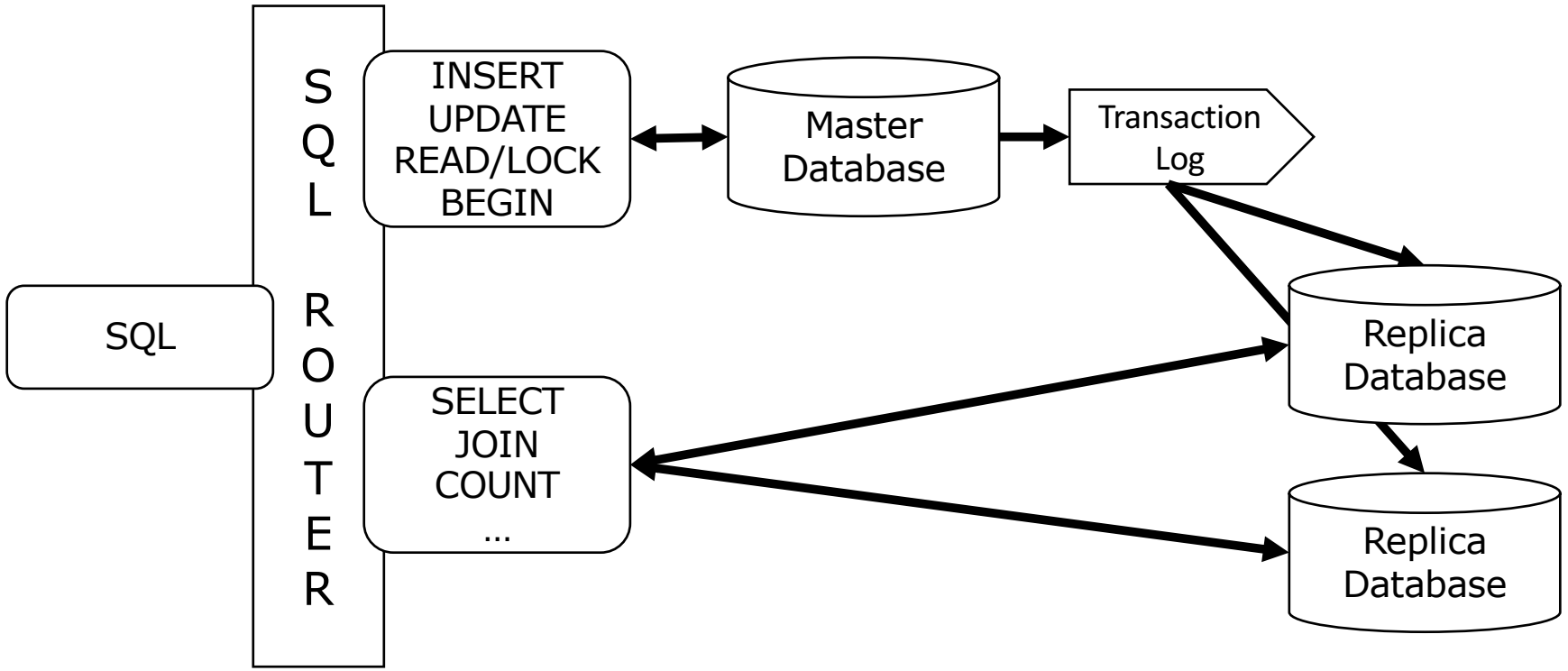
Scaling ACID Databases

Why did we look at BASE at all?

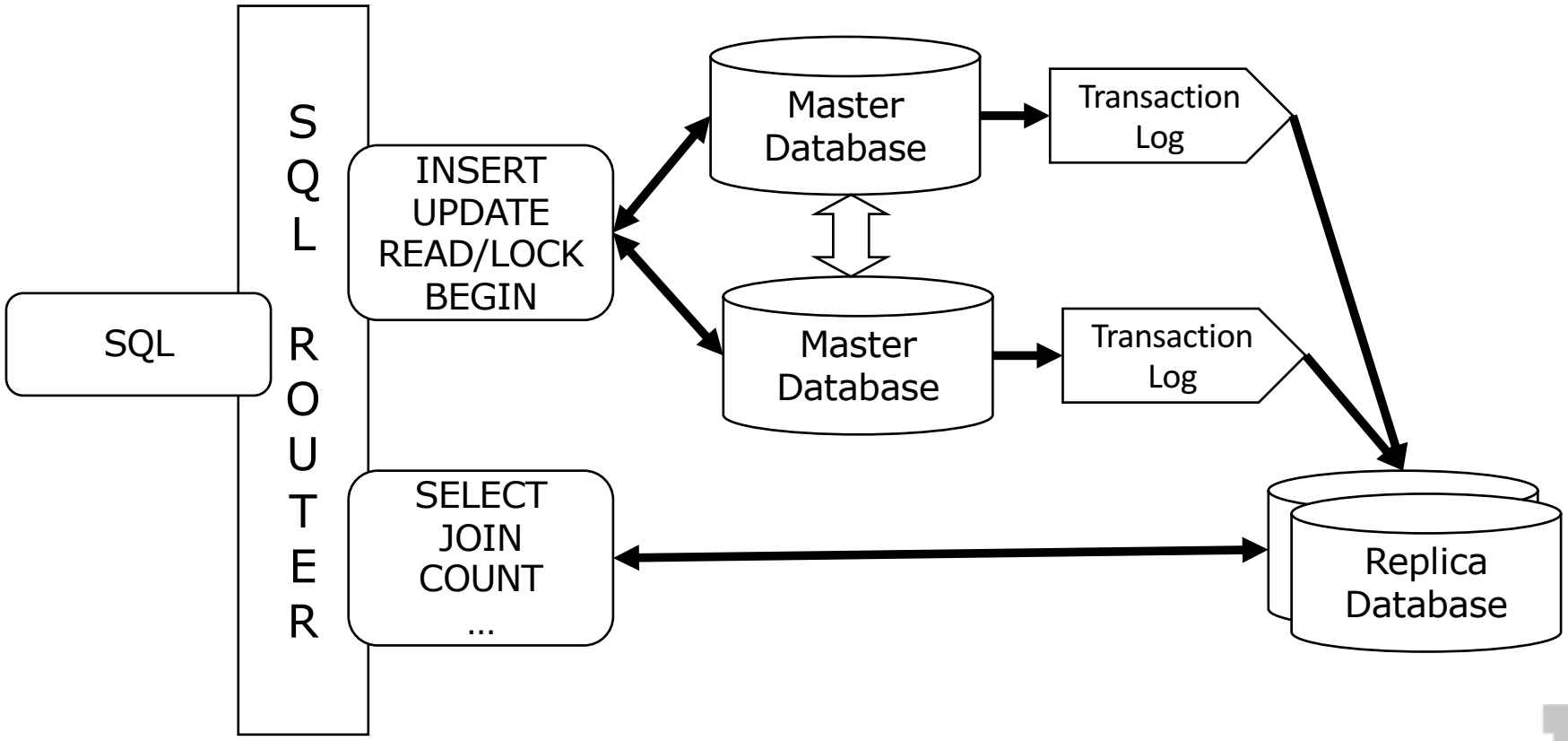
Vertical Scaling

- More disk drives or disk arrays / RAID
- More processors
- More memory
- Switch from spinning to solid state drives
  - Modern SSD drives have scatter / gather
- Has been solidly successful over the years

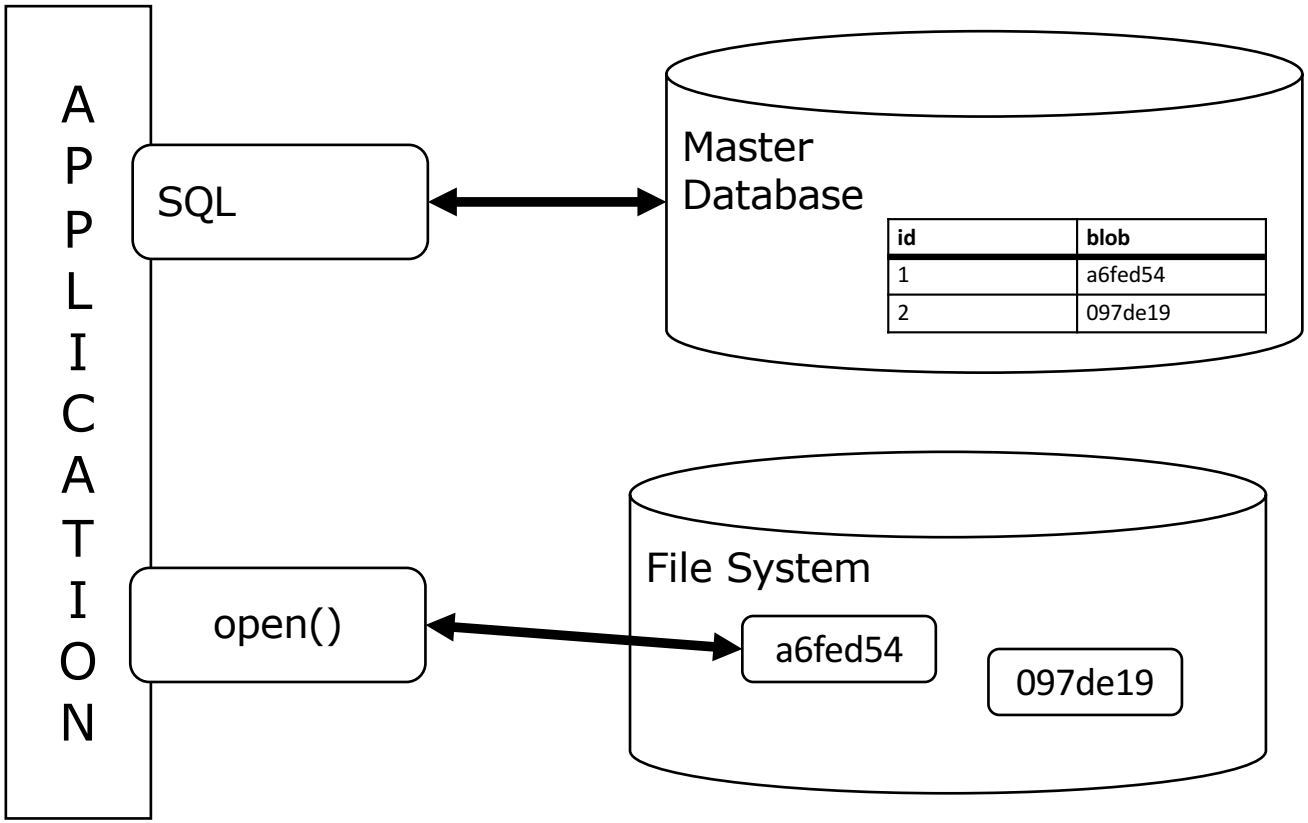
Master / Read Only Replicas



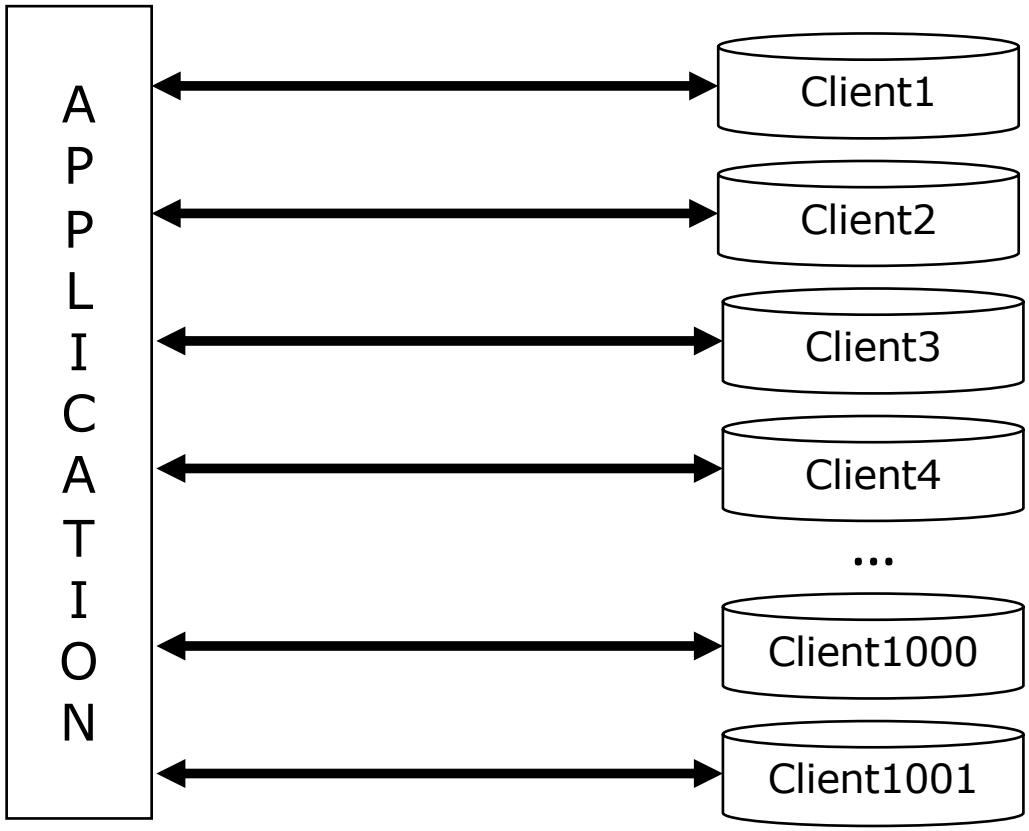
Multi-Master

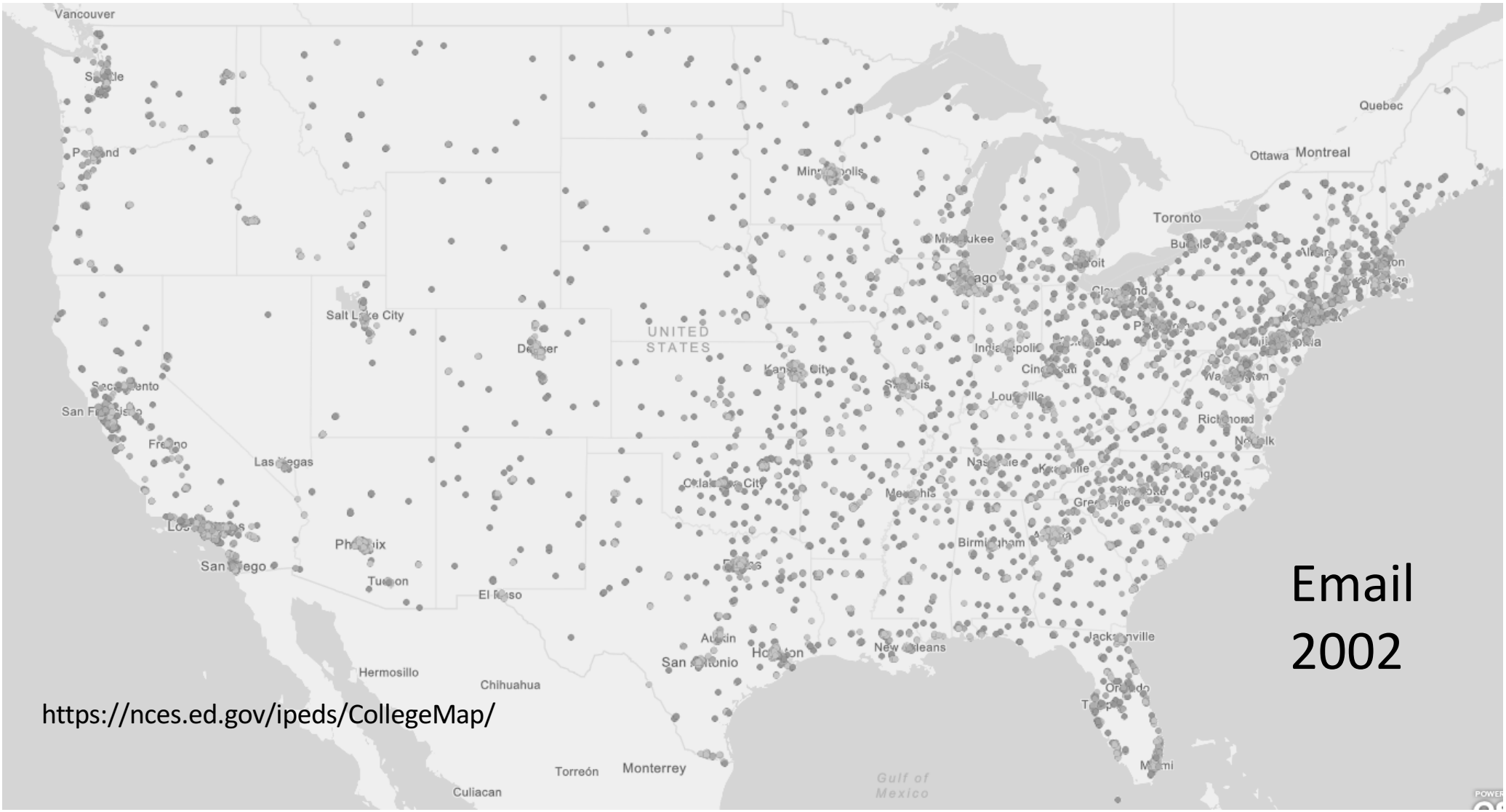


Multiple Store Types



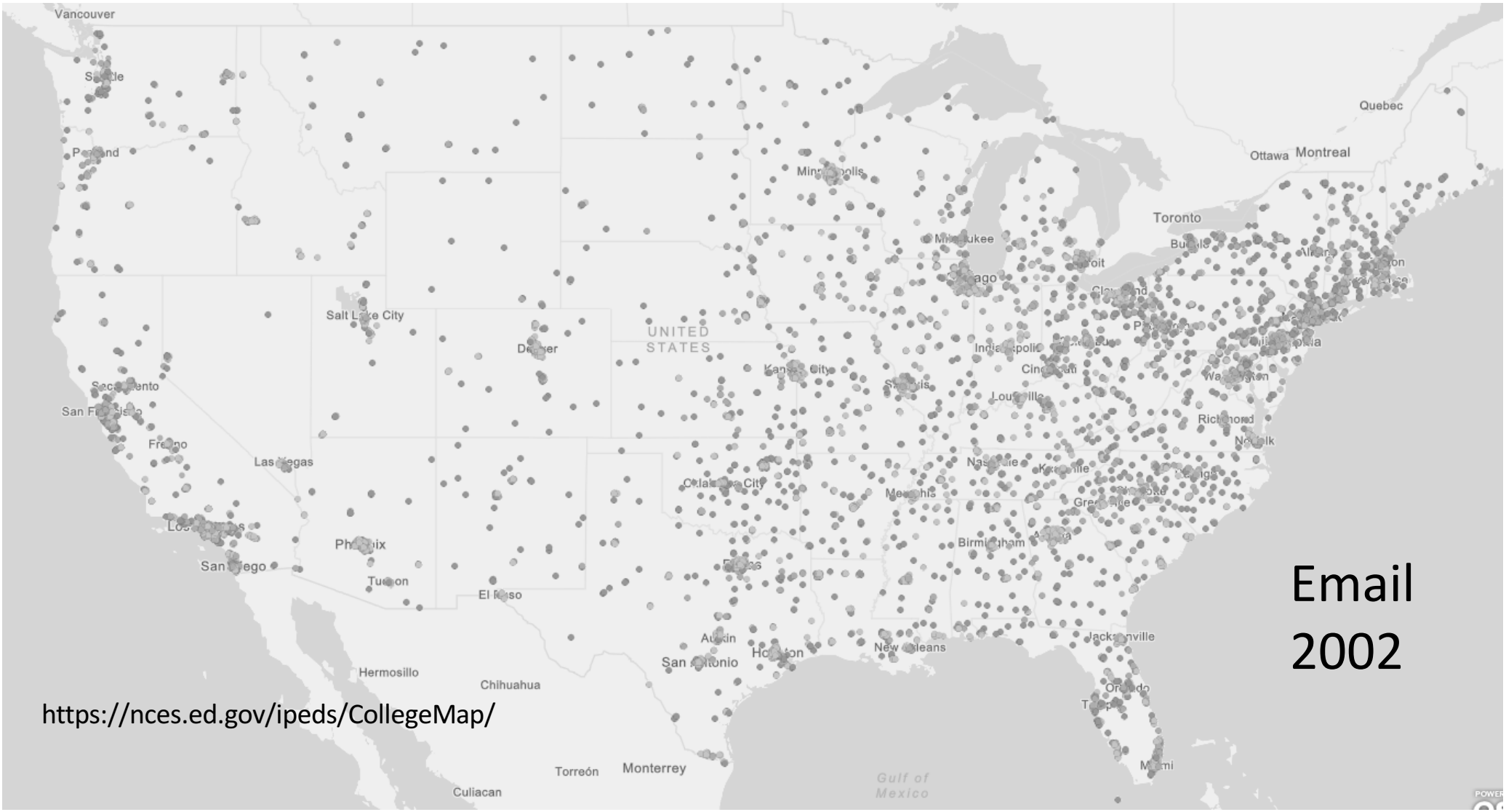
Multi-Tenant / "Pretend Cloud"





https://nces.ed.gov/ipeds/CollegeMap/

First Generation True Cloud Applications….



https://nces.ed.gov/ipeds/CollegeMap/

https://web.archive.org/web/19990428171538/http://google.com/

Email
2009

Google Could Not use RDBMS

- They also chose applications that did not need transactions
  - Everything was free – or "the first ~100Mb was free"
  - Updates were widely distributed – even to email
- Early Google Applications were not FaceBook or Twitter
- They could use cleverly named files and folders and sharding / hashing across servers

Searching / Scatter - Gather

- Google I/O June 2008 Keynote
- Marissa Mayer

https://www.youtube.com/watch?v=6x0cAzQ7PVs

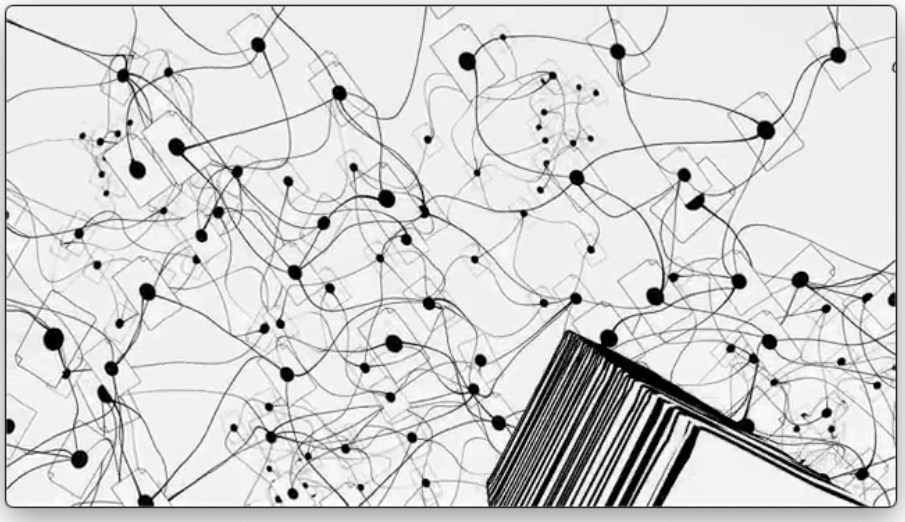
Google Container Tour

- Google Efficient Data Centers Summit April 1, 2009.

https://www.youtube.com/watch?v=zRwPSFpLX8I

Google – How Search Works

- Matt Cutts – March 2010



https://www.youtube.com/watch?v=BNHR6IQJGZs

Watch the Cloud Videos

Searching / Scatter - Gather

- Google I/O June 2008 Keynote
- Marissa Mayer

https://www.youtube.com/watch?v=6x0cAzQ7PVs

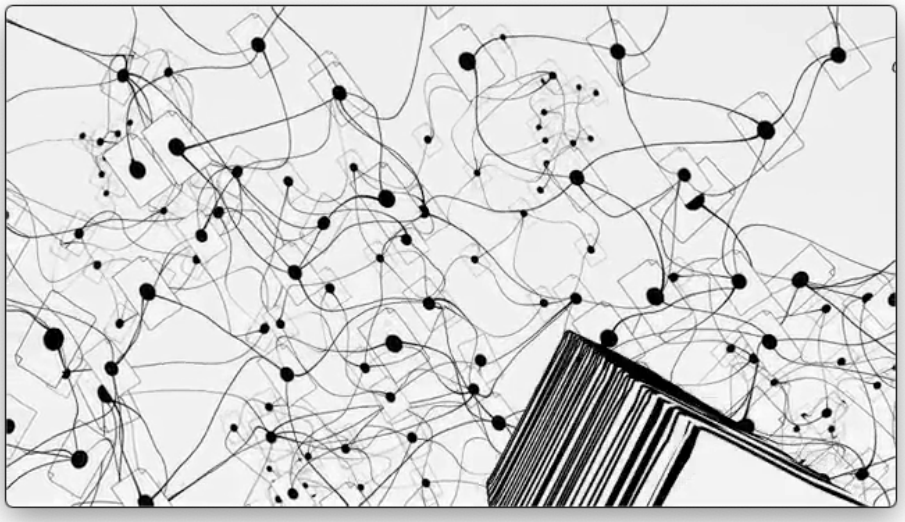
Google Container Tour

- Google Efficient Data Centers Summit April 1, 2009.

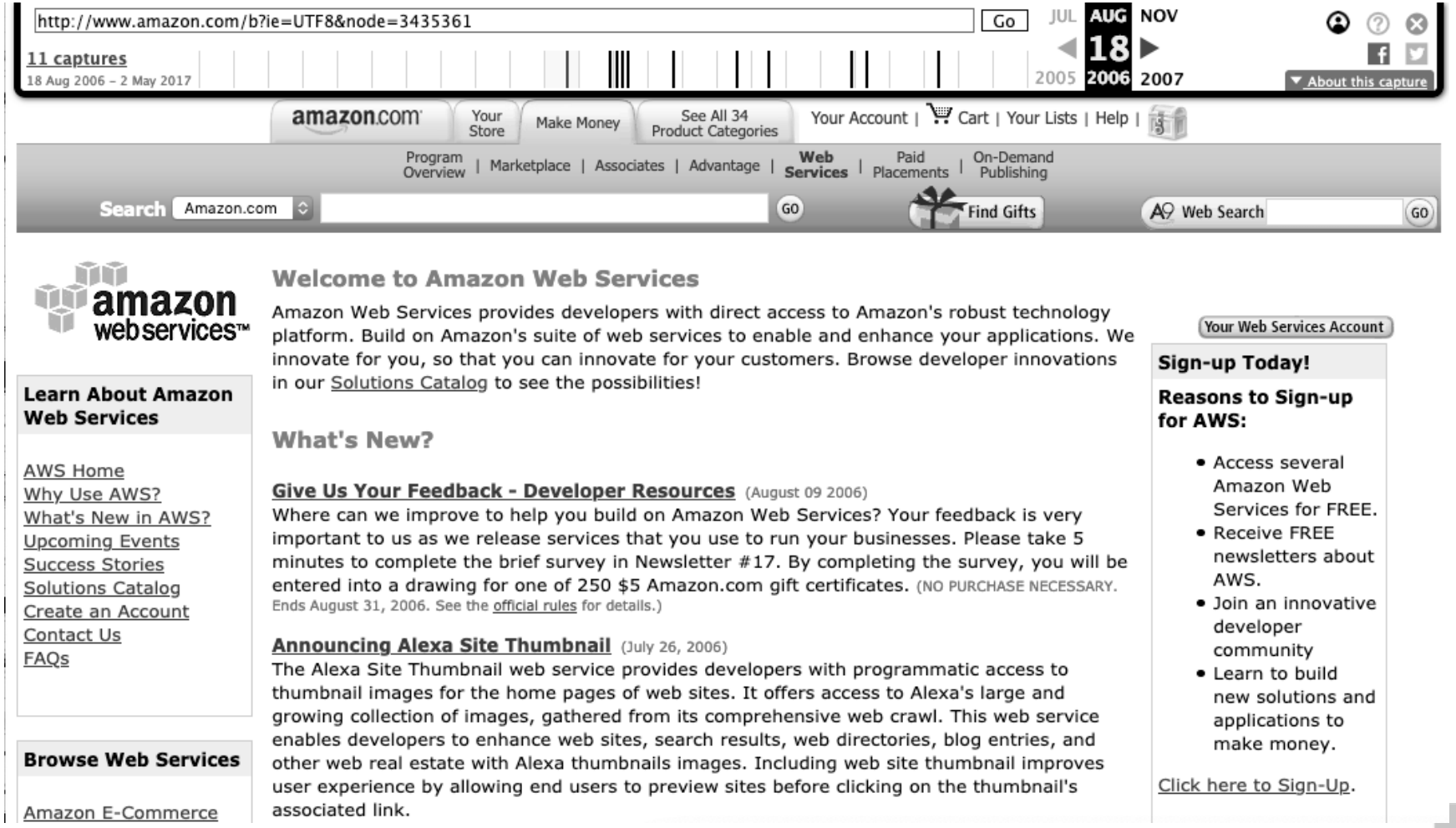
https://www.youtube.com/watch?v=zRwPSFpLX8I

Google – How Search Works

- Matt Cutts – March 2010



https://www.youtube.com/watch?v=BNHR6IQJGZs



https://web.archive.org/web/20060818023744/http://www.amazon.com/b?ie=UTF8&node=3435361

Early Amazon Web Services Pricing

- Large / slow disks were inexpensive
- Small quick CPUs with small amounts of memory were inexpensive

- Applications that responded to load by dynamically adding small servers and slow disk were ideal

https://pages.mtu.edu/~steve/CSERI/

Efficient use of "carpet clusters"

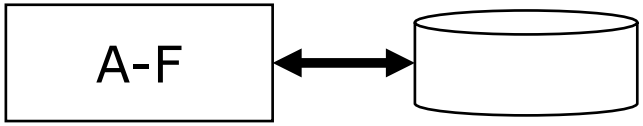
- Spread data out across many system
- Scatter the query to all the systems
- Gather the results
- (a.k.a. Map-Reduce)
- A single query might be 1-2 seconds
- Many queries could be "in flight" at the same time (need a fast network)
- You might just run a RDBMS on each node and shard

Second Generation Cloud Scale Applications

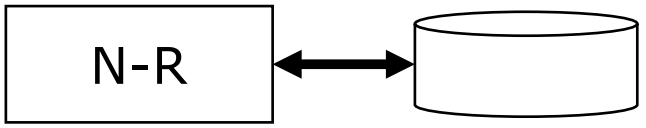
FaceBook is More Challenging

- Friend lists – edit / add / drop / find
- Privacy
- Everyone sees a very different view
- Everyone searches a different corpus

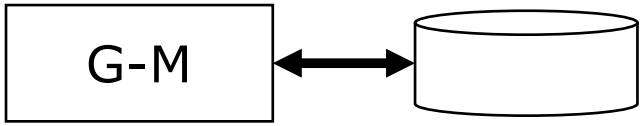
- Data locking for predictable update is replaced by data sharding and replication
- Migrate data "to be close" to the viewer



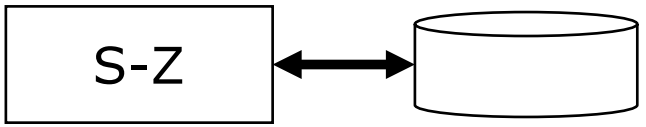

```
Annie
 friends: Greg, Sarah
  status
    Greg: Pizza
```



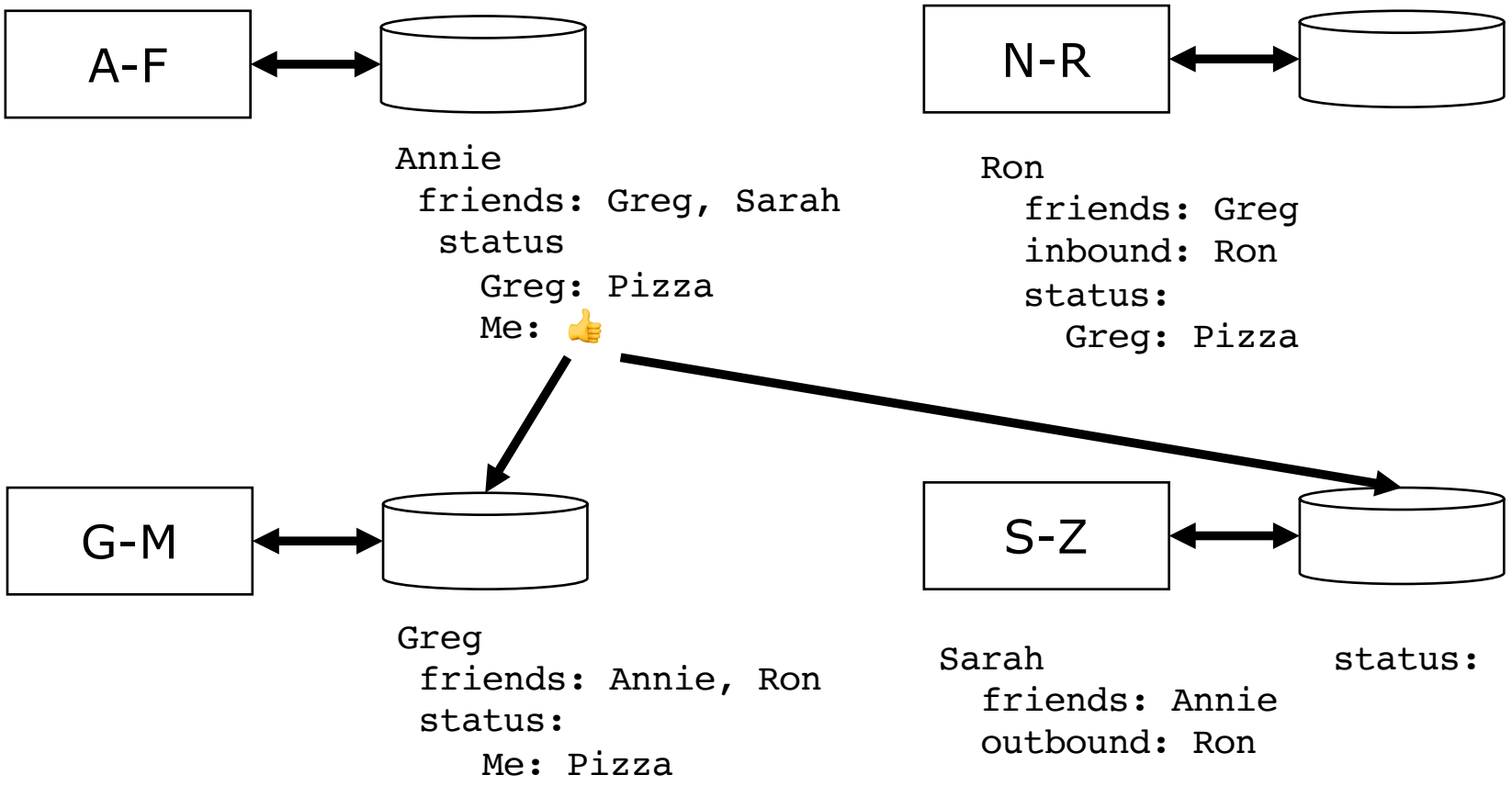

```
Ron
  friends: Greg
  inbound: Ron
  status:
    Greg: Pizza
```

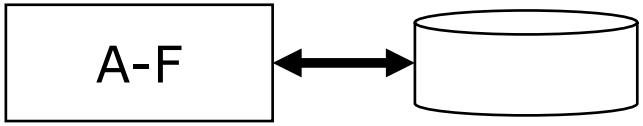



```
Greg
 friends: Annie, Ron
 status:
    Me: Pizza
```

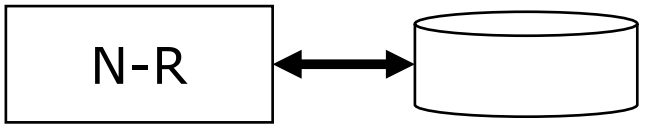



```
Sarah                status:
  friends: Annie
  outbound: Ron
```

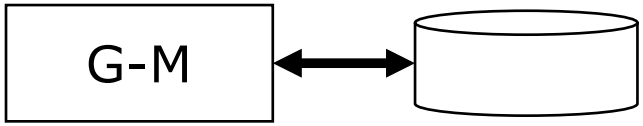

A-F
N-R
Annie
friends: Greg, Sarah
status
Greg: Pizza
Me: 👍
Ron
friends: Greg
inbound: Ron
status:
Greg: Pizza
G-M
S-Z
Greg
friends: Annie, Ron
status:
Me: Pizza
Sarah
friends: Annie
outbound: Ron
status:



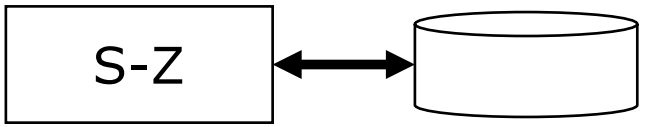
Annie
friends: Greg, Sarah
status
Greg: Pizza
Me: 👍



Ron
friends: Greg
inbound: Ron
status:
Greg: Pizza
Annie: 👍 (??)



Greg
friends: Annie, Ron
status:
Me: Pizza
Anne: 👍



Sarah
friends: Annie
outbound: Ron

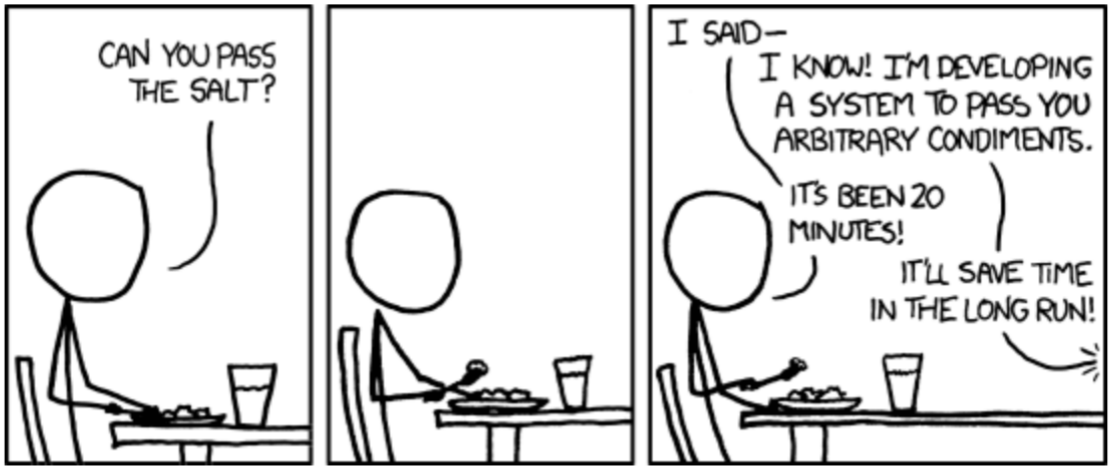
status:
Greg: Pizza (??)
Annie: 👍 (??)

Problems to Solve

- Clever non-locking solutions to distribution
  - GUIDs for primary keys
  - Hashing / Sharding for predictable data placement / lookup
- Some central control – mostly "what is where"
- Perhaps use one or more RDBMS for taking money or new accounts

THE GENERAL PROBLEM

|< < PREV RANDOM NEXT > >|



|< < PREV RANDOM NEXT > >|

PERMANENT LINK TO THIS COMIC: HTTPS://XKCD.COM/974/
IMAGE URL (FOR HOTLINKING/EMBEDDING): HTTPS://IMGS.XKCD.COM/COMICS/THE_GENERAL_PROBLEM.PNG

The Emergence of BASE Solutions (i.e. NoSQL)

The basic principles of BASE DBMS

- Everything is distributed – fast network
- No locks (*)
- Lots of fast / small memory CPUs
- Lots of disks
- Indexes follow data shards
- Documents not rows / columns
- Schema on read – not schema on write(*)

JSON Ascending

- JSON is a great way to represent / move / store structured data
- Fast parsers in every programming language
- Easily compressed to save storage and transfer

```
{
   "superlongkey" : 42;
}

{
   "superlongkey" : 43;
}
```

Open Source NoSQL databases

- CouchDB (2008)
  - **C**luster **O**f **U**nreliable **C**ommodity **H**ardware
- MongoDB – 2009
  - Distributed JSON storage
- Cassandra – 2008
  - From FaceBook
  - Also Apache Hadoop – Map / Reduce
- ElasticSearch – 2010
  - Initially full text search Apache Lucene
  - Evolved into JSON database

Proprietary / Software AS a Service (SAAS) NoSQL Databases

- Amazon DynamoDB
  - Backed the Amazon catalog
- Google BigTable
  - Stored Google's copy of the web
- Azure Table Storage
  - Catching up ☺

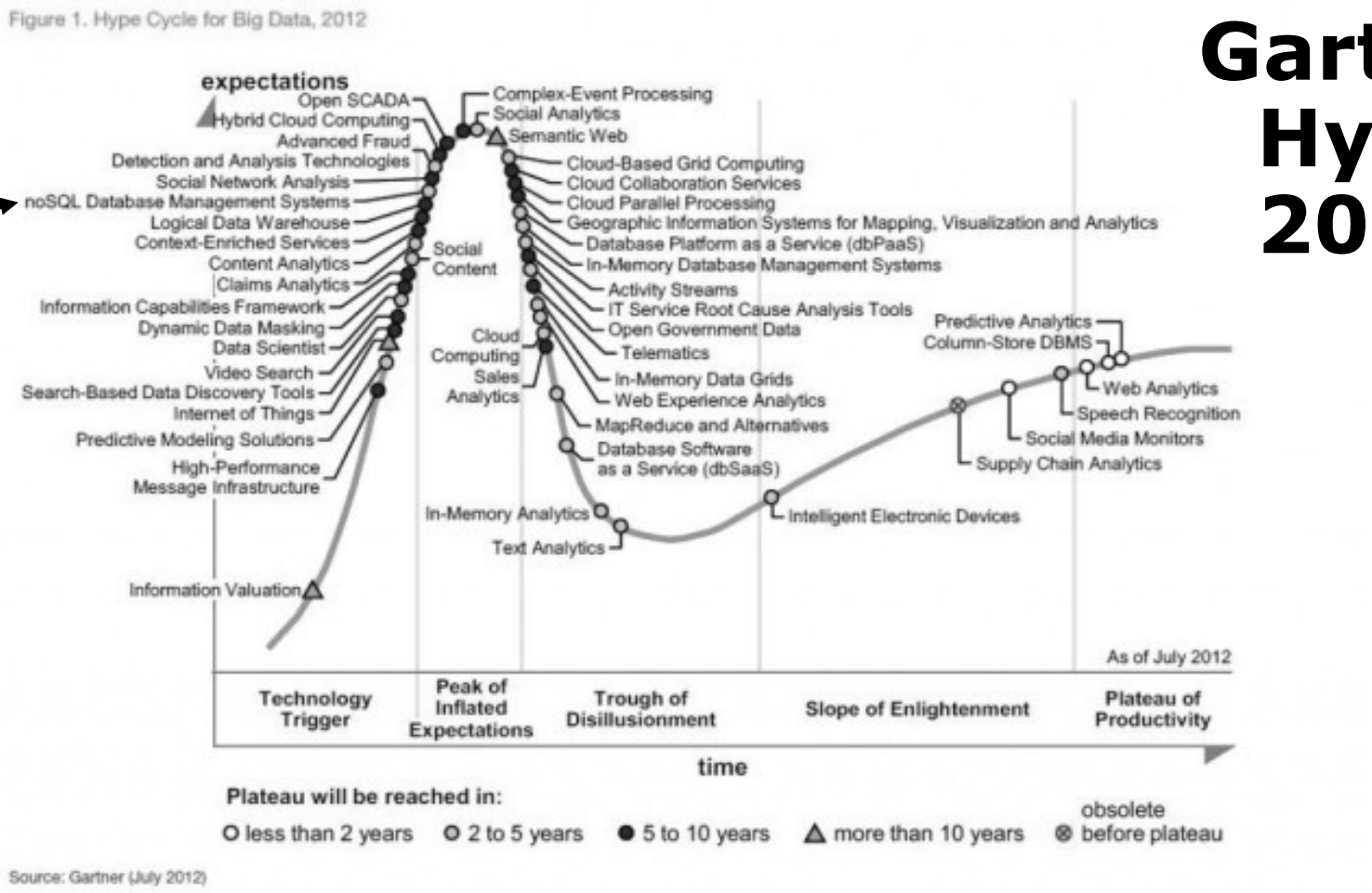
Every Startup 2010-Present

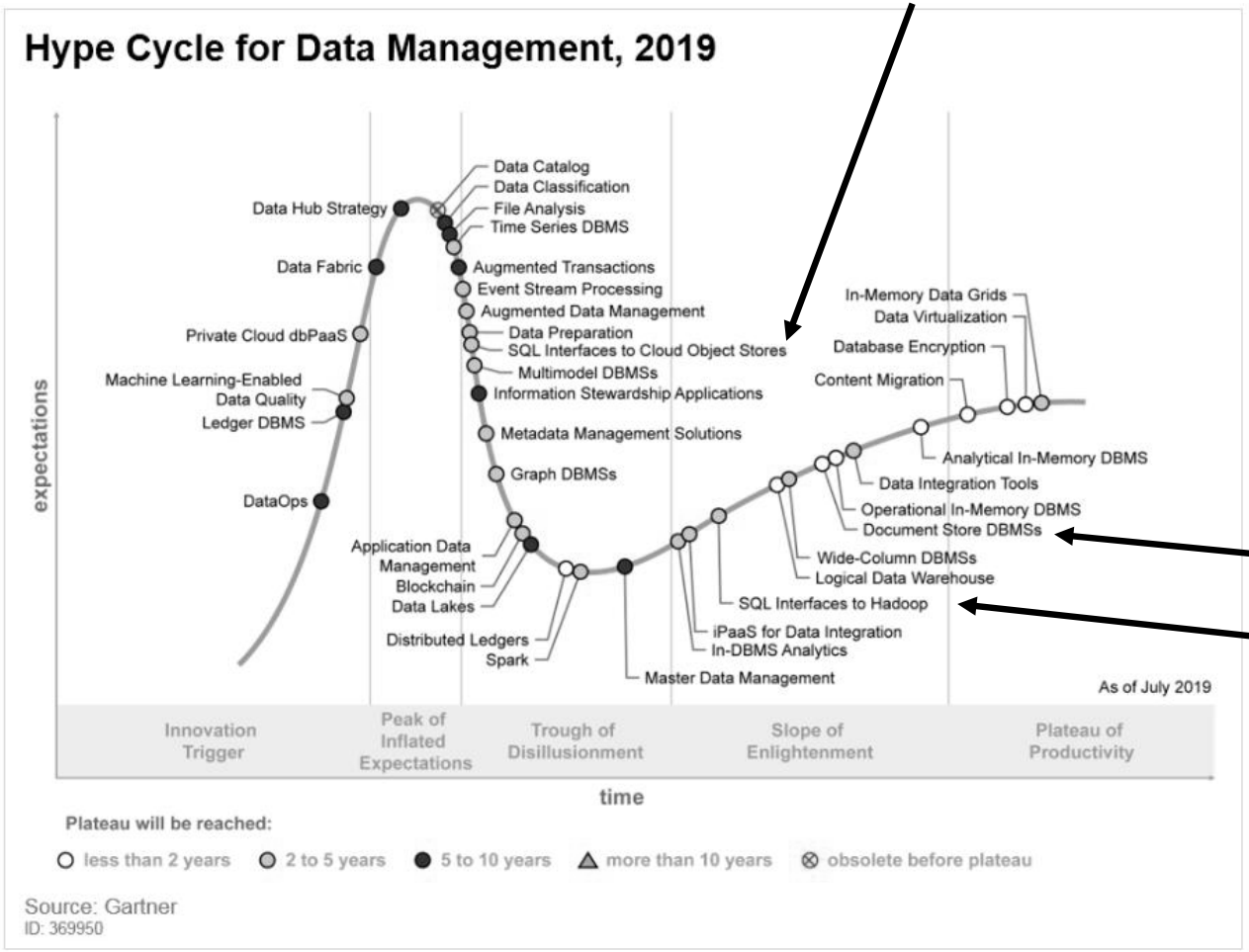
https://commons.wikimedia.org/wiki/File:Gold_Pan.jpg

Be like FaceBook – Make Money

- Emergence of client-side applications
  - Backbone, Angular, React, Vue …
- Emergence of JavaScript in the server
  - node.js – great at asynch / micro services
- NoSQL databases
  - Distributed, scalable, inexpensive resources
- Lots of startups / fresh ground up development

Gartner Hype 2012




Hype Cycle for Data Management, 2019
expectations
Data Hub Strategy
Data Fabric
Private Cloud dbPaaS
Machine Learning-Enabled
Data Quality
Ledger DBMS
DataOps
Data Catalog
Data Classification
File Analysis
Time Series DBMS
Augmented Transactions
Event Stream Processing
Augmented Data Management
Data Preparation
SQL Interfaces to Cloud Object Stores
Multimodel DBMSs
Information Stewardship Applications
Metadata Management Solutions
Graph DBMSs
Application Data Management
Blockchain
Data Lakes
Distributed Ledgers
Spark
Master Data Management
iPaaS for Data Integration
In-DBMS Analytics
SQL Interfaces to Hadoop
Wide-Column DBMSs
Logical Data Warehouse
Operational In-Memory DBMS
Document Store DBMSs
Data Integration Tools
Analytical In-Memory DBMS
Content Migration
Database Encryption
Data Virtualization
In-Memory Data Grids
As of July 2019
Innovation Trigger
Peak of Inflated Expectations
Trough of Disillusionment
Slope of Enlightenment
Plateau of Productivity
time
Plateau will be reached:
less than 2 years
2 to 5 years
5 to 10 years
more than 10 years
obsolete before plateau
Source: Gartner
ID: 369950

Case Study - Vericite

- Startup founded in 2014 – expected 100TB
  - Cloud / multi-tenant / document based
- Used MySQL for POC – Did not want to shard
- Built on Cassandra and "owned hardware"
- Cassandra fell down at scale - consultant
- Switched to Amazon DynamoDB
  - Works – expensive but cheaper than consultants
- NoSQL database competed against larger firm using custom storage on physical hardware

Reacting to the rise of NoSQL

But That's Not All…

- The ACID vendors saw market share slipping away circa 2013
- As NoSQL applications matured they found that application developers wanted "a few" transactions and JOINs

- ACID + BASE became the new sweet spot

Technology Changes 2009-2019

- AWS Could sell you 32 CPU systems with large amounts of RAM cheaper than you could own them
- Solid State Disk developed scatter / gather on a single drive with 32 + simultaneous reads to different areas of the drive

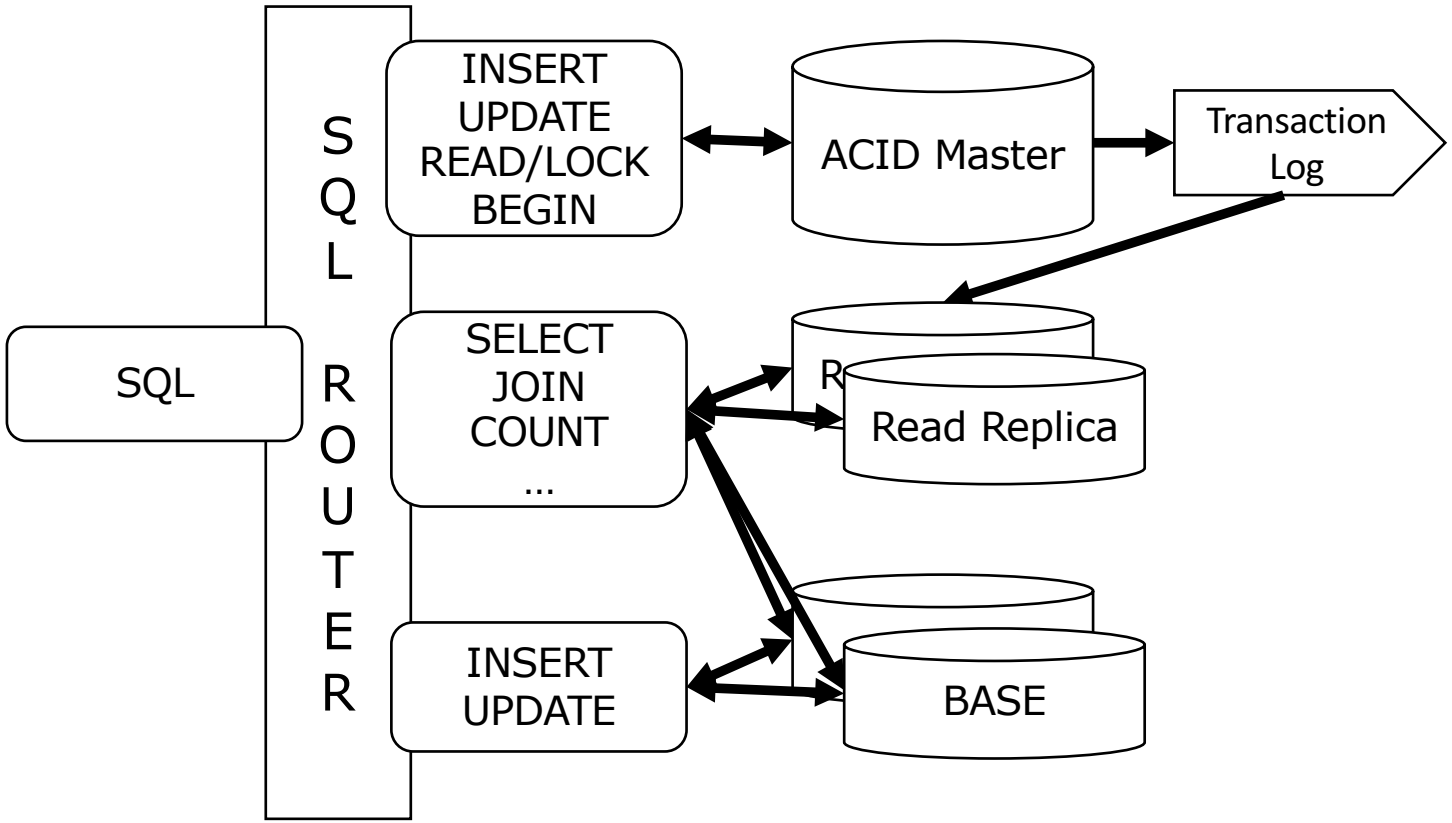
RDBMS Vendors reacted

- Oracle
  - JSON Columns
  - NoSQL Features
- MySQL 8.0 – JSON Columns
- PostgreSQL
  - 8.3 HSTORE Columns (2008 and 2014)
  - 9.2 JSON Columns (2012)
  - 9.4 JSONB Columns (2014)
- Amazon Redshift is based on a "modified" PostgreSQL 8.0 (2013)

ACID + BASE or BASE + ACID

- It turns out to be easier to relax ACID than to do the research and development to implement ACID in a system that is distributed at its core
- SQL does not imply ACID
- BASE runtime databases are adopting SQL syntax for some of their operations to make it easier for developers

Hybrid (Hypothetical)



Being BASE-Like in ACID RDBMS

- Do not normalize – Replicate
- Don't use SERIAL - use UUID
- Columns are for indexing
- Do not use foreign keys or don't mark them as such
- Design your schema / indexes to enable reading a single row on query

https://www.wix.engineering/post/scaling-to-100m-mysql-is-a-better-nosql

Being BASE-Like in ACID RDBMS

- Use software migrations instead of ALTER
- Query for records by primary key or by indexed column
- Do not use JOINs
- Do not use aggregations (COUNT ??)

https://www.wix.engineering/post/scaling-to-100m-mysql-is-a-better-nosql

Summary

- NoSQL is doing well
  - More for specialized applications
  - Less conversation about the "end of SQL"
  - Breathless is becoming pragmatic
  - There is a learning curve - production experience
  - SASS from cloud vendors makes it "easier"
- Some applications converting back
  - "Move from MongoDB to PostgreSQL"
- Review: Why PostgreSQL for this course?

Acknowledgements / Contributions

Initial Development: Charles R. Severance, University of Michigan School of Information

Insert new Contributors and Translators here including names and dates

Continue new Contributors and Translators here