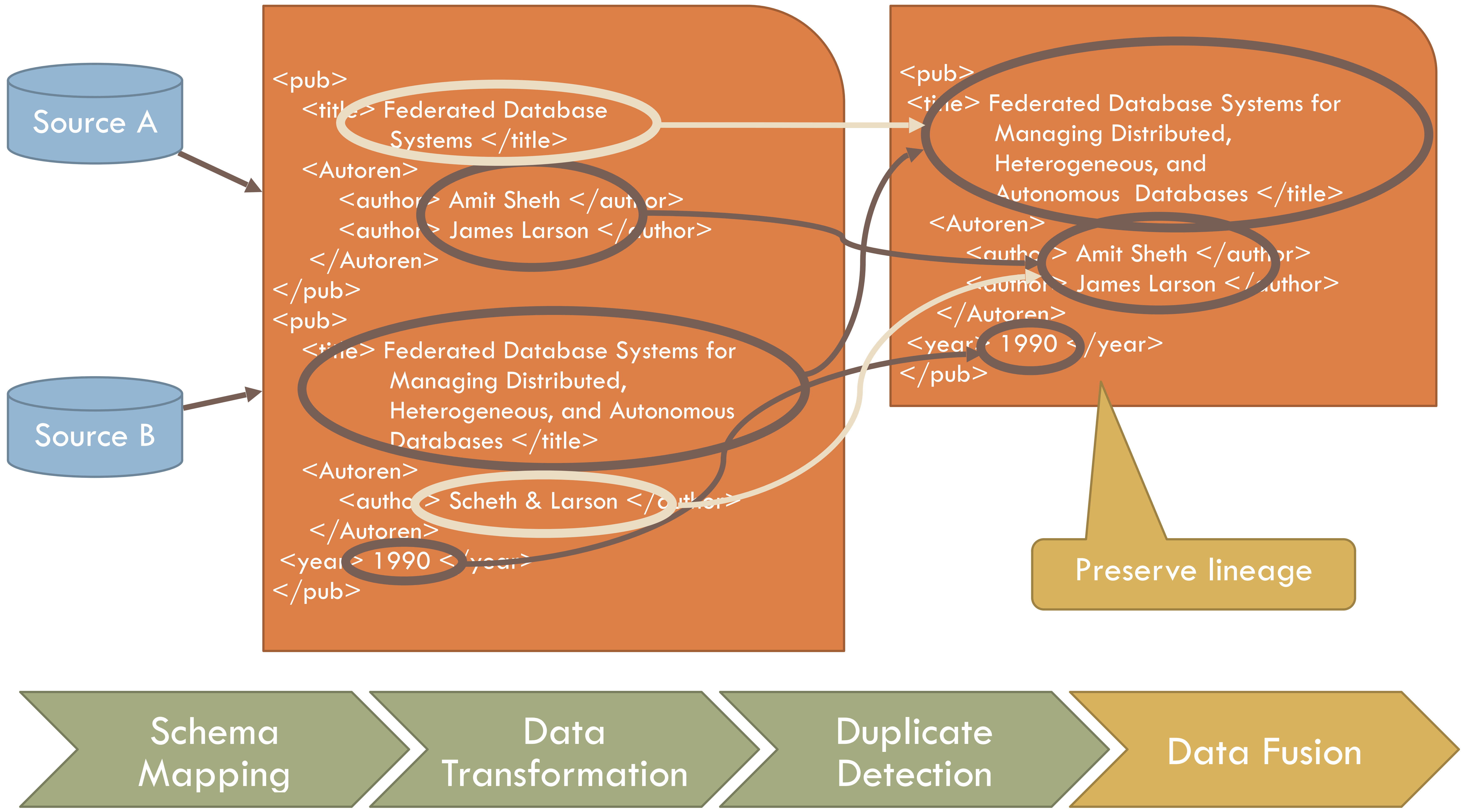
Advanced Data Management (CSCI 640/490)

Scalable Databases

Dr. David Koop

Information Integration



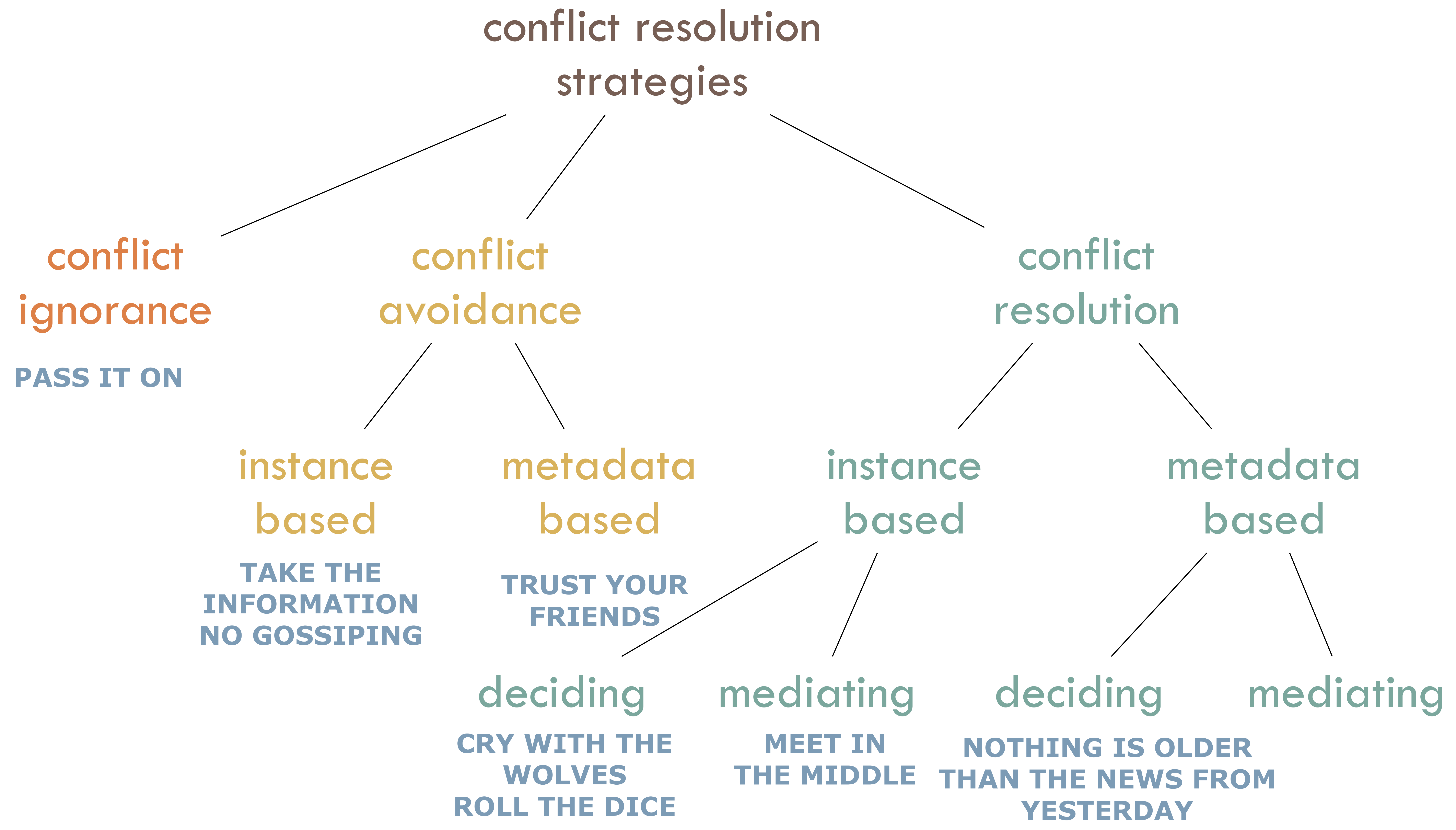
Source A
Source B
<pub>
<title> Federated Database Systems </title>
<Autoren>
<author> Amit Sheth </author>
<author> James Larson </author>
</Autoren>
</pub>
<pub>
<title> Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases </title>
<Autoren>
<author> Scheth & Larson </author>
</Autoren>
<year> 1990 </year>
</pub>
<pub>
<title> Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases </title>
<Autoren>
<author> Amit Sheth </author>
<author> James Larson </author>
</Autoren>
<year> 1990 </year>
</pub>
Preserve lineage
Schema Mapping
Data Transformation
Duplicate Detection
Data Fusion

[L. Dong and F. Naumann, 2009]

Data Fusion

- Problem: Given a duplicate, create a single object representation while resolving conflicting data values.
- Difficulties:
  - Null values: Subsumption and complementation
  - Contradictions in data values
  - Uncertainty & truth: Discover the true value and model uncertainty in this process
  - Metadata: Preferences, recency, correctness
  - Lineage: Keep original values and their origin
  - Implementation in DBMS: SQL, extended SQL, UDFs, etc.

Conflict Resolution Strategies



[L. Dong and F. Naumann, 2009]

Example Problem

[X L Dong et al., 2009]

Example Problem

	S1	S2	S3
Stonebraker	MIT	Berkeley	MIT
Dewitt	MSR	MSR	UWisc
Bernstein	MSR	MSR	MSR
Carey	UCI	AT&T	BEA
Halevy	Google	Google	UW

[X L Dong et al., 2009]

Naive Voting Works

	S1	S2	S3
Stonebraker	MIT	Berkeley	MIT
Dewitt	MSR	MSR	UWisc
Bernstein	MSR	MSR	MSR
Carey	UCI	AT&T	BEA
Halevy	Google	Google	UW

[X L Dong et al., 2009]

Naive Voting Only Works if Data Sources are Independent

[X L Dong et al., 2009]

Naive Voting Only Works if Data Sources are Independent

	S1	S2	S3	S4	S5
Stonebraker	MIT	Berkeley	MIT	MIT	MS
Dewitt	MSR	MSR	UWisc	UWisc	UWisc
Bernstein	MSR	MSR	MSR	MSR	MSR
Carey	UCI	AT&T	BEA	BEA	BEA
Halevy	Google	Google	UW	UW	UW

[X L Dong et al., 2009]

S4 and S5 copy from S3

	S1	S2	S3	S4	S5
Stonebraker	MIT	Berkeley	MIT	MIT	MS
Dewitt	MSR	MSR	UWisc	UWisc	UWisc
Bernstein	MSR	MSR	MSR	MSR	MSR
Carey	UCI	AT&T	BEA	BEA	BEA
Halevy	Google	Google	UW	UW	UW

[X L Dong et al., 2009]

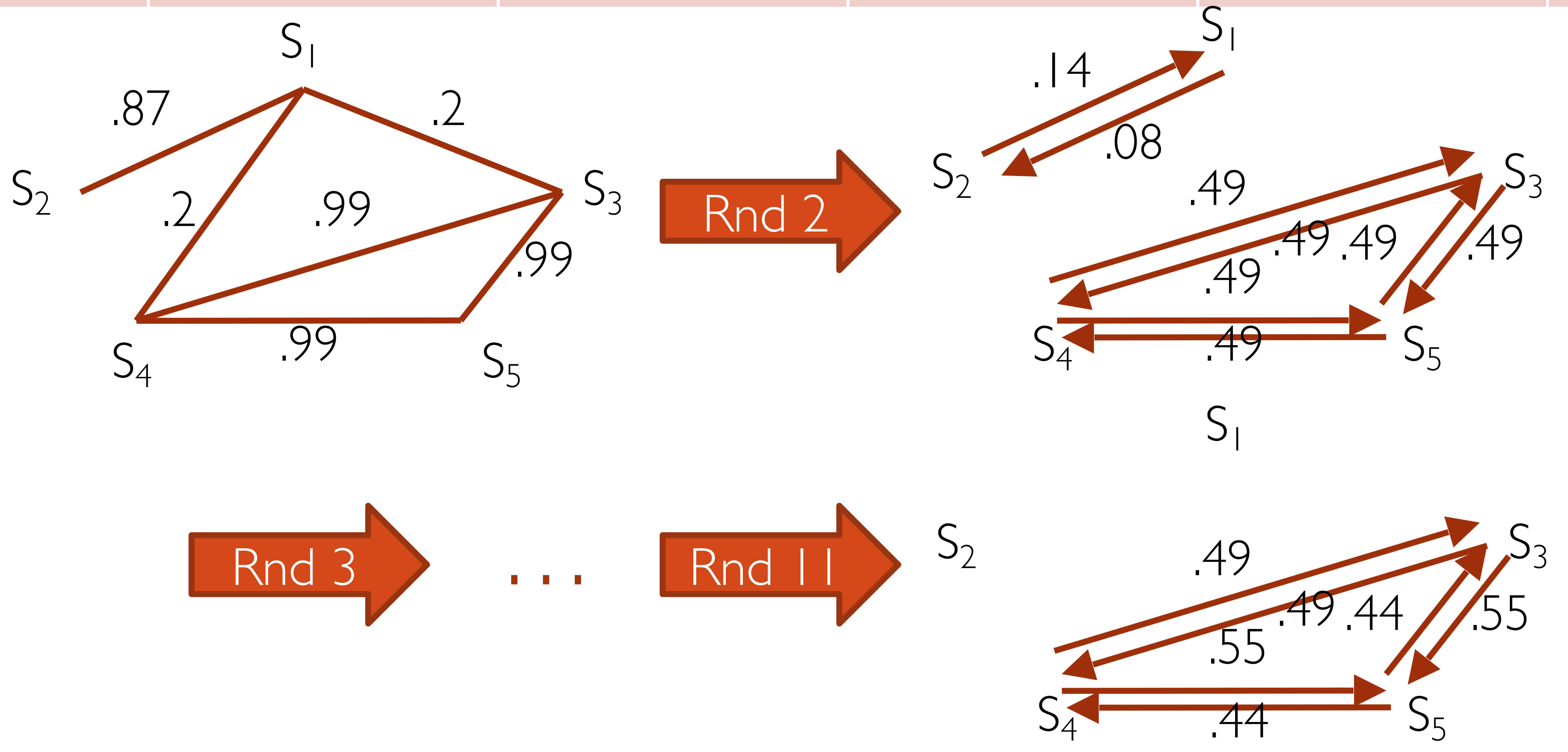
S4 and S5 copy from S3

	S1	S2	S3	S4	S5
Stonebraker	MIT	Berkeley	MIT	MIT	MS
Dewitt	MSR	MSR	UWisc	UWisc	UWisc
Bernstein	MSR	MSR	MSR	MSR	MSR
Carey	UCI	AT&T	BEA	BEA	BEA
Halevy	Google	Google	UW	UW	UW

[X L Dong et al., 2009]

Example

	S1	S2	S3	S4	S5
Stonebraker	MIT	Berkeley	MIT	MIT	MS
Dewitt	MSR	MSR	UWisc	UWisc	UWisc
Bernstein	MSR	MSR	MSR	MSR	MSR
Carey	UCI	AT&T	BEA	BEA	BEA
Halevy	Google	Google	UW	UW	UW



[X L Dong et al., 2009]

Example

Accuracy	S1	S2	S3	S4	S5
Round 1	.52	.42	.53	.53	.53
Round 2	.63	.46	.55	.55	.55
Round 3	.71	.52	.53	.53	.37
Round 4	.79	.57	.48	.48	.31
…	…	…	…	…	…
Round 11	.97	.61	.40	.40	.21

Value Confidence	Carey			Halevy	
	UCI	AT&T	BEA	**Google**	UW
Round 1	1.61	1.61	2.0	2.1	2.0
Round 2	1.68	1.3	2.12	2.74	2.12
Round 3	2.12	1.47	2.24	3.59	2.24
Round 4	2.51	1.68	2.14	4.01	2.14
…	…	…	…	…	…
Round 11	4.73	2.08	1.47	6.67	1.47

[X L Dong et al., 2009]

Assignment 3

- Ask a Manager Salary Data
- Use Polars & OpenRefine
- Due Tomorrow

Assignment 4

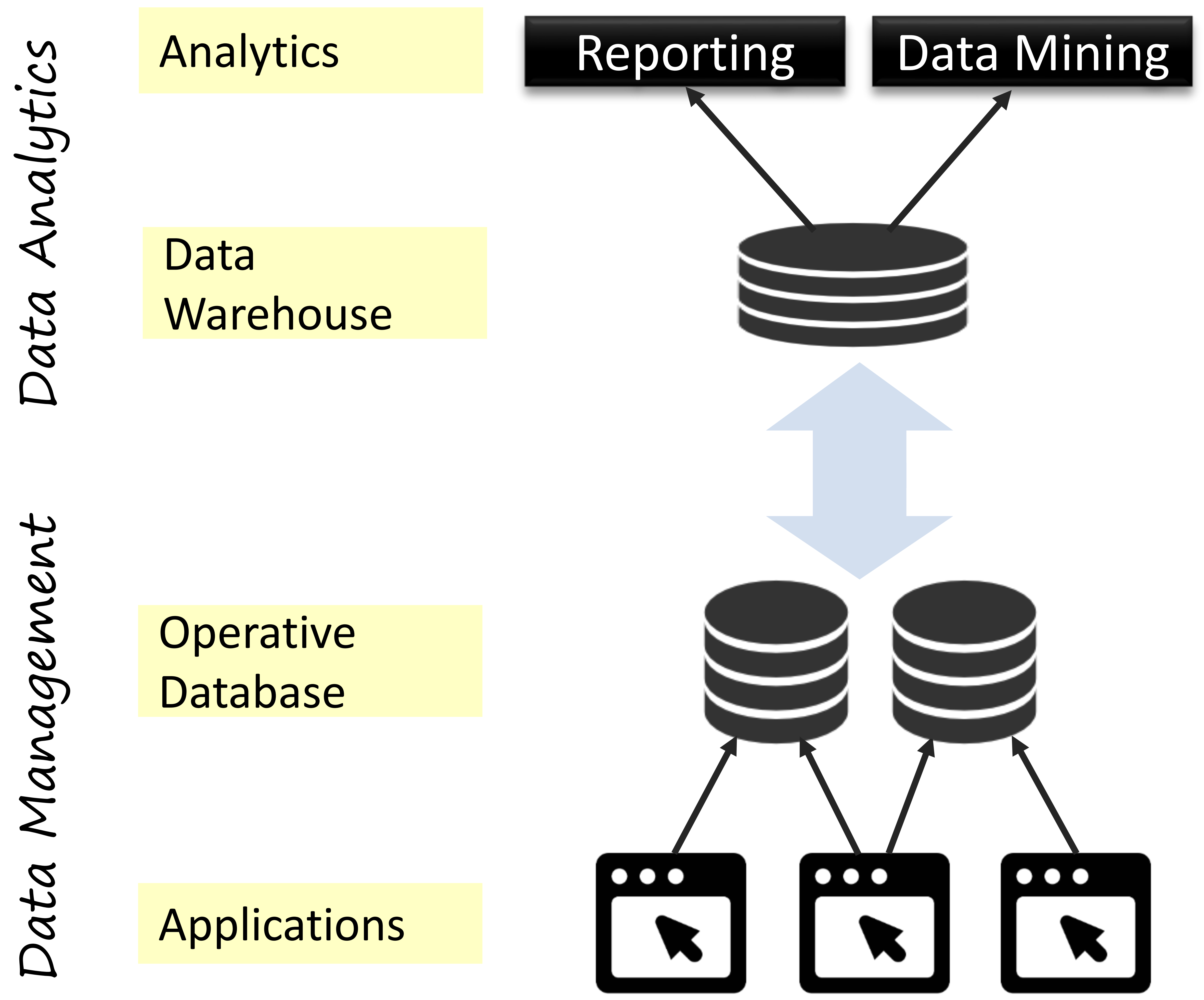
- Data Integration & Data Fusion
- Out soon

Paper Critique

- Read What's Really New with NewSQL?
- Submit critique **before class** on Wednesday, October 22
- Discussion ideas:
  - What are the advantages or disadvantages of NewSQL vs NoSQL?
  - Are they really different from standard RDBMS?
  - Which category of NewSQL databases is most promising?

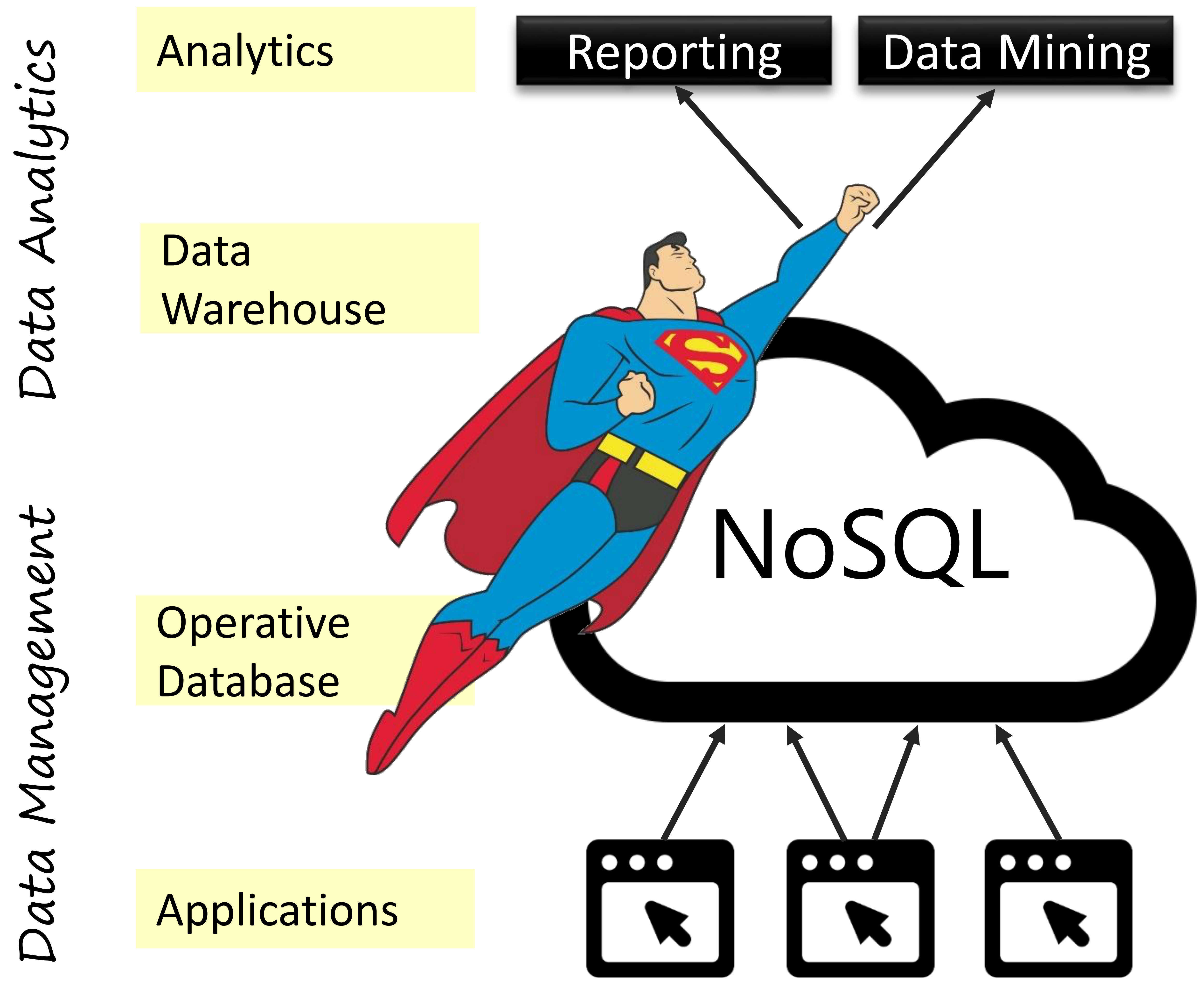
Scalable Database Systems

Database Architecture



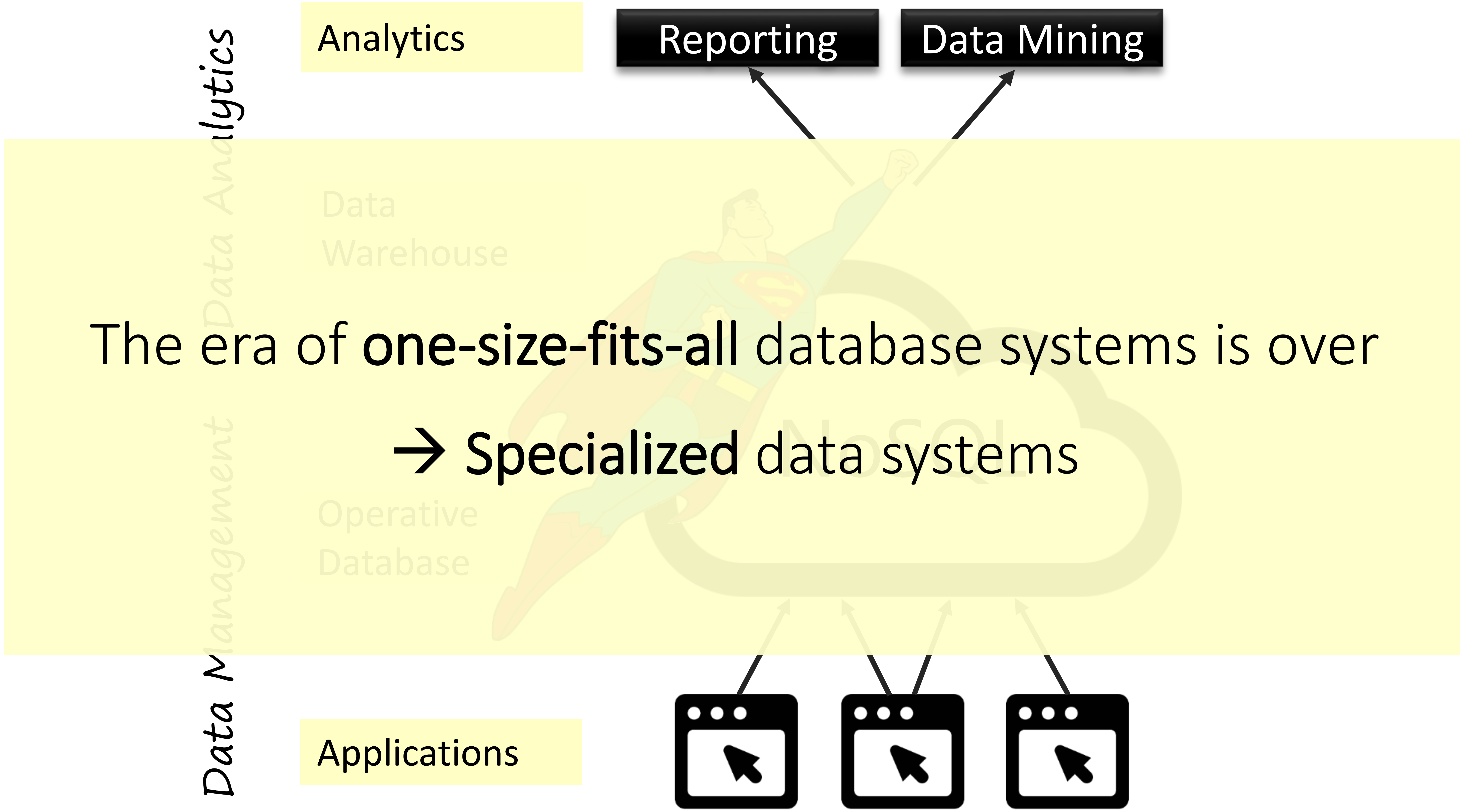
[F. Gessert et al., 2017]

Database Architecture



[F. Gessert et al., 2017]

Database Architecture



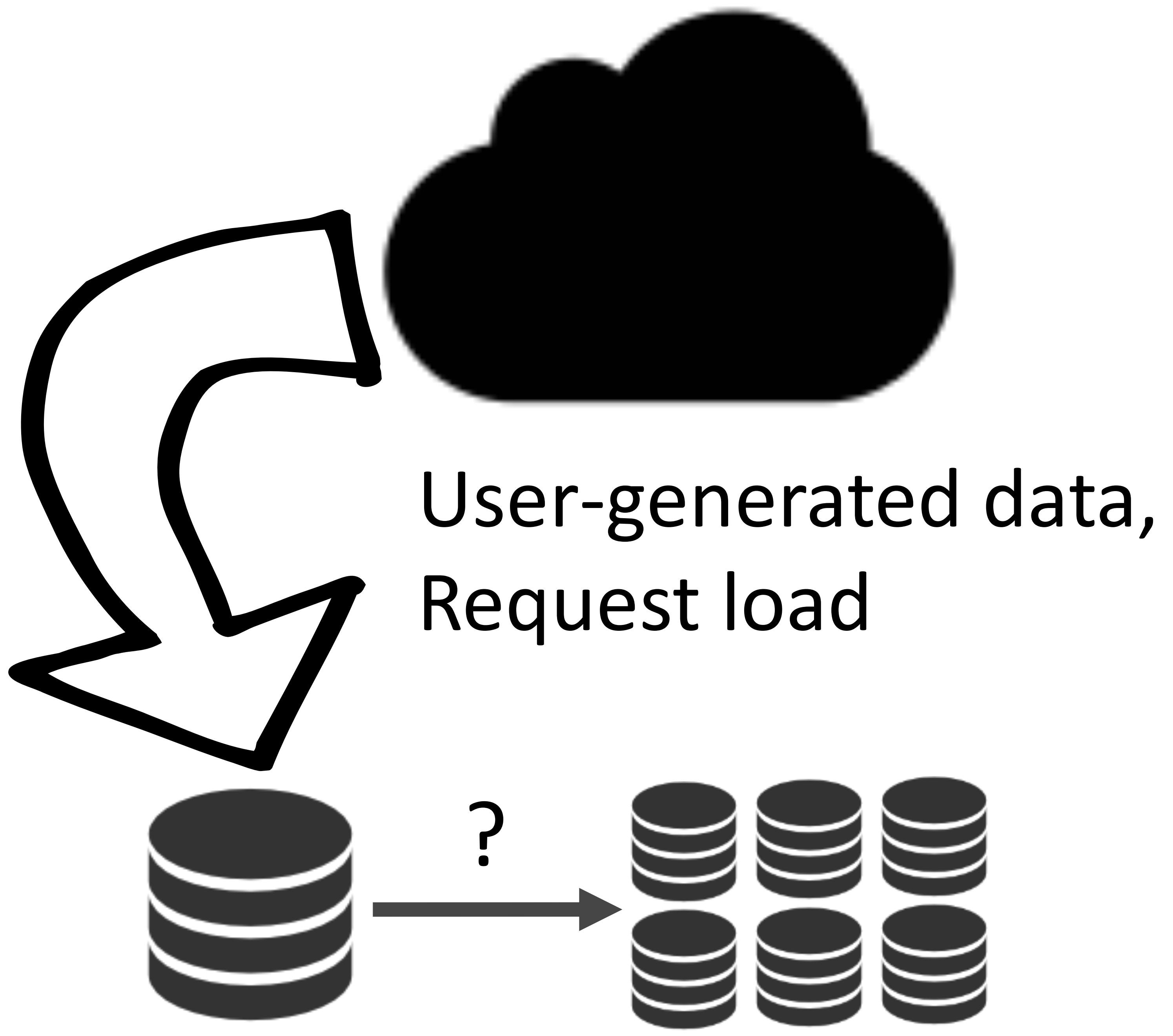
The era of **one-size-fits-all** database systems is over

→ **Specialized** data systems

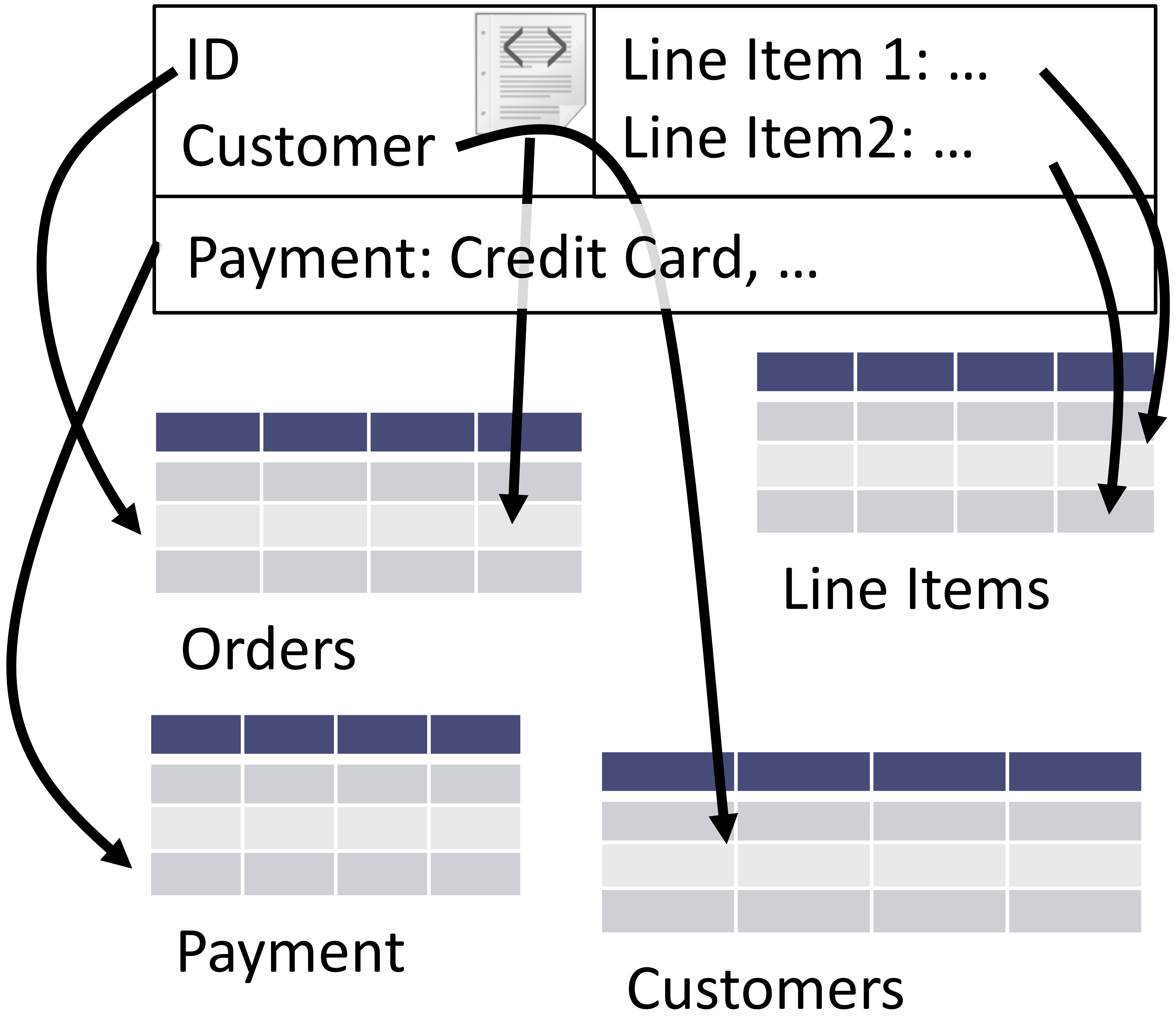
[F. Gessert et al., 2017]

NoSQL Motivation

Scalability

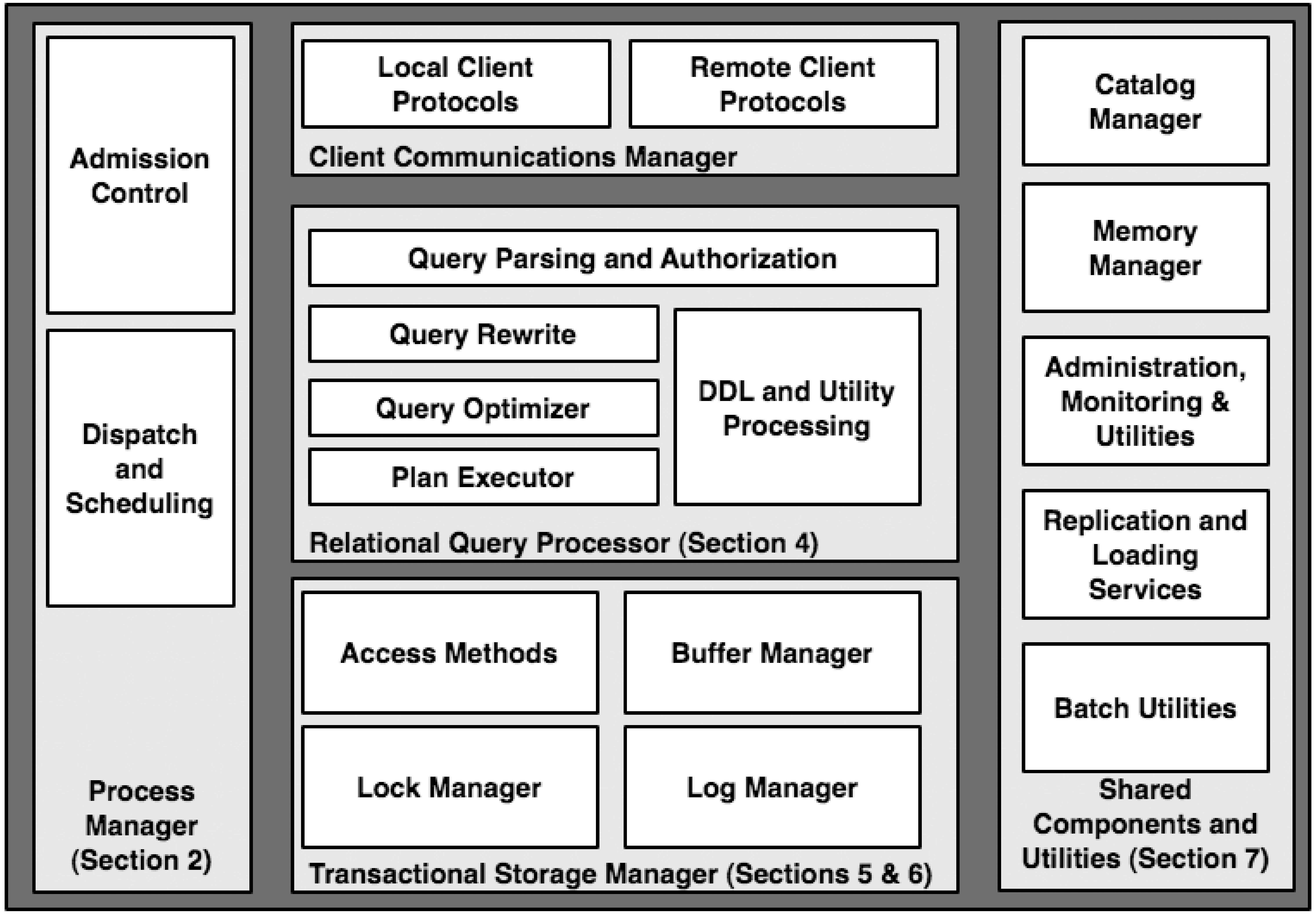


Impedance Mismatch



[F. Gessert et al., 2017]

Relational Database Architecture


Admission Control
Dispatch and Scheduling
Process Manager (Section 2)
Local Client Protocols
Remote Client Protocols
Client Communications Manager
Query Parsing and Authorization
Query Rewrite
Query Optimizer
Plan Executor
DDL and Utility Processing
Relational Query Processor (Section 4)
Access Methods
Buffer Manager
Lock Manager
Log Manager
Transactional Storage Manager (Sections 5 & 6)
Catalog Manager
Memory Manager
Administration, Monitoring & Utilities
Replication and Loading Services
Batch Utilities
Shared Components and Utilities (Section 7)

[Hellerstein et al., Architecture of a Database System]

Relational Databases: One size fits all?

- Lots of work goes into relational database development:
  - B-trees
  - Cost-based query optimizers
  - ACID (Atomicity, Consistency, Isolation, Durability)
- Vendors largely stuck with this model from the 1980s through 2000s
- Having different systems leads to business problems:
  - cost problem
  - compatibility problem
  - sales problem
  - marketing problem

[Stonebraker and Çetinetmel, 2005]

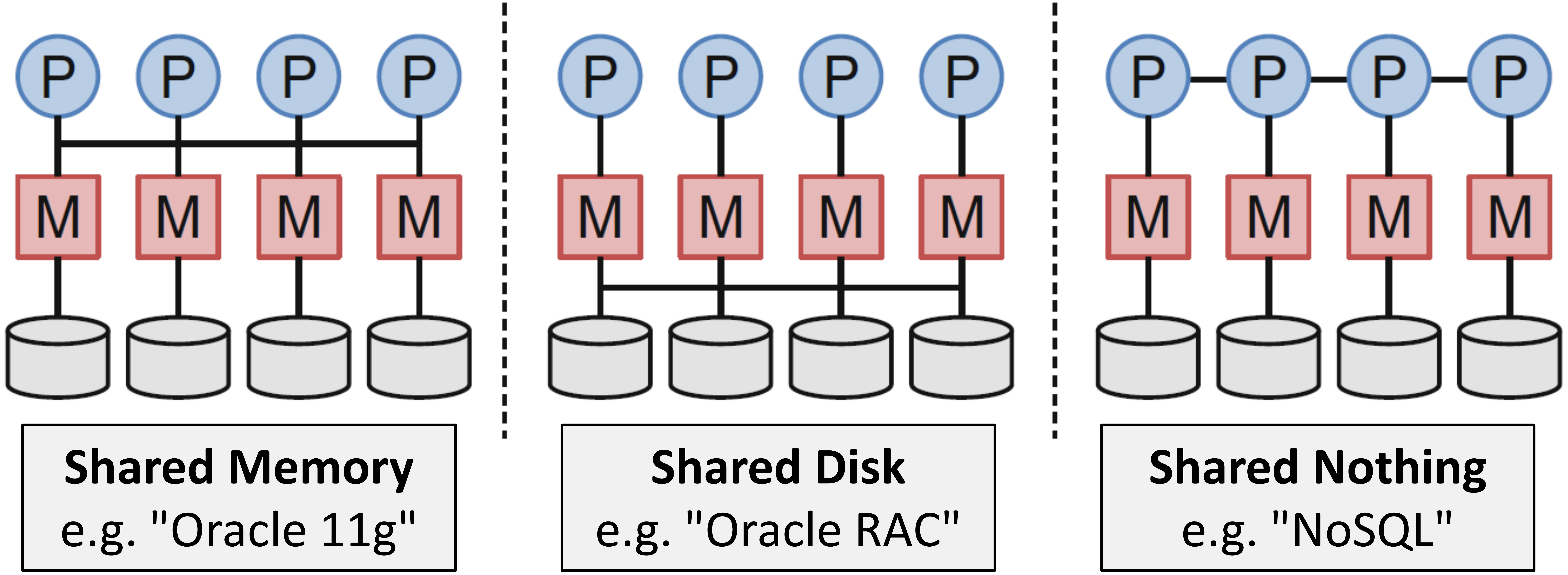
ACID Transactions

- Make sure that transactions are processed reliably
- **A**tomicity: leave the database as is if some part of the transaction fails (e.g. don't add/remove only part of the data) using rollbacks
- **C**onsistency: database moves from one valid state to another
- **I**solation: concurrent execution matches serial execution
- **D**urability: endure hardware failures, make sure changes hit disk

How to Scale Relational Databases?

Shared Nothing Architecture

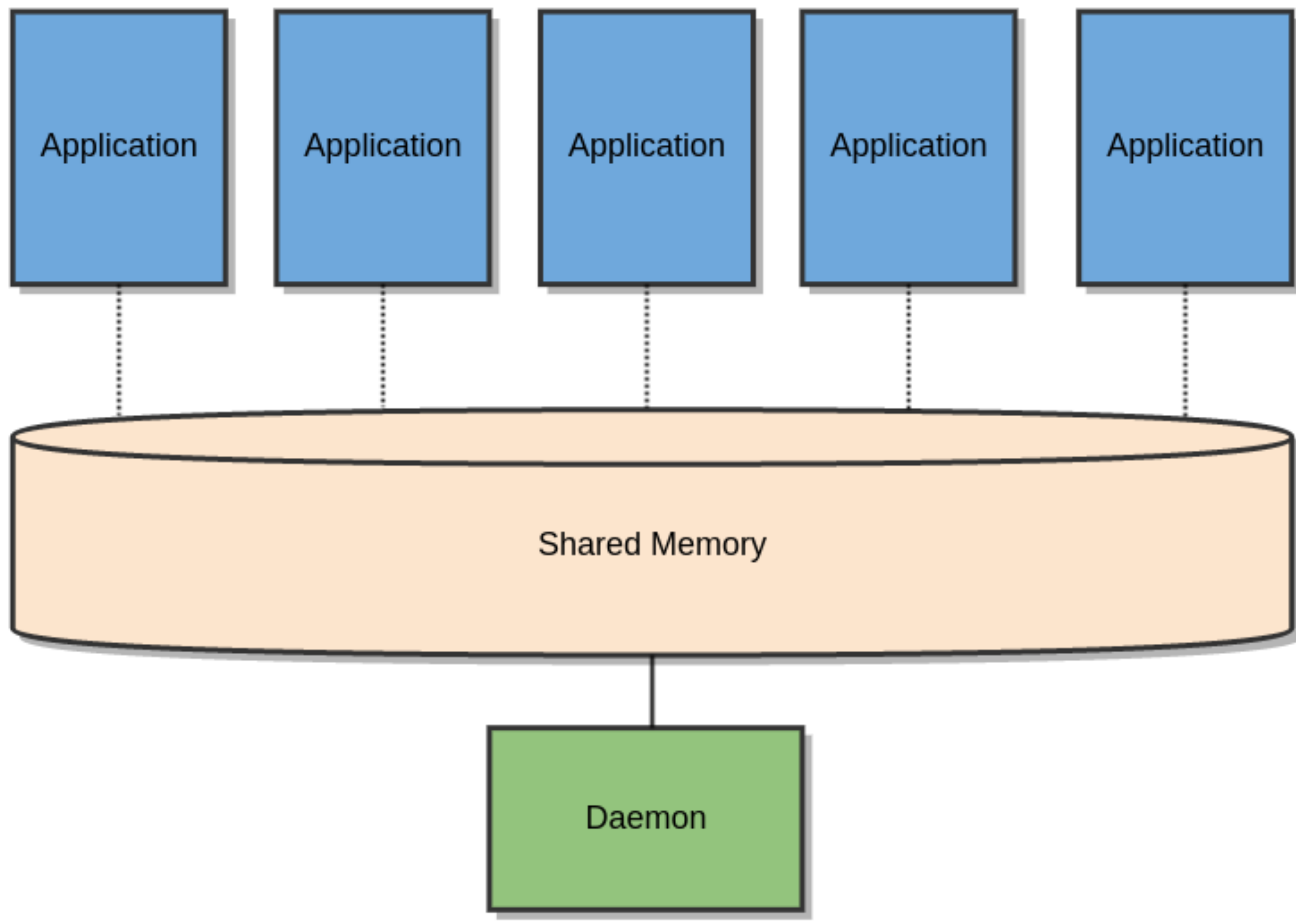
Shift towards higher distribution & less coordination:



[F. Gessert et al., 2017]

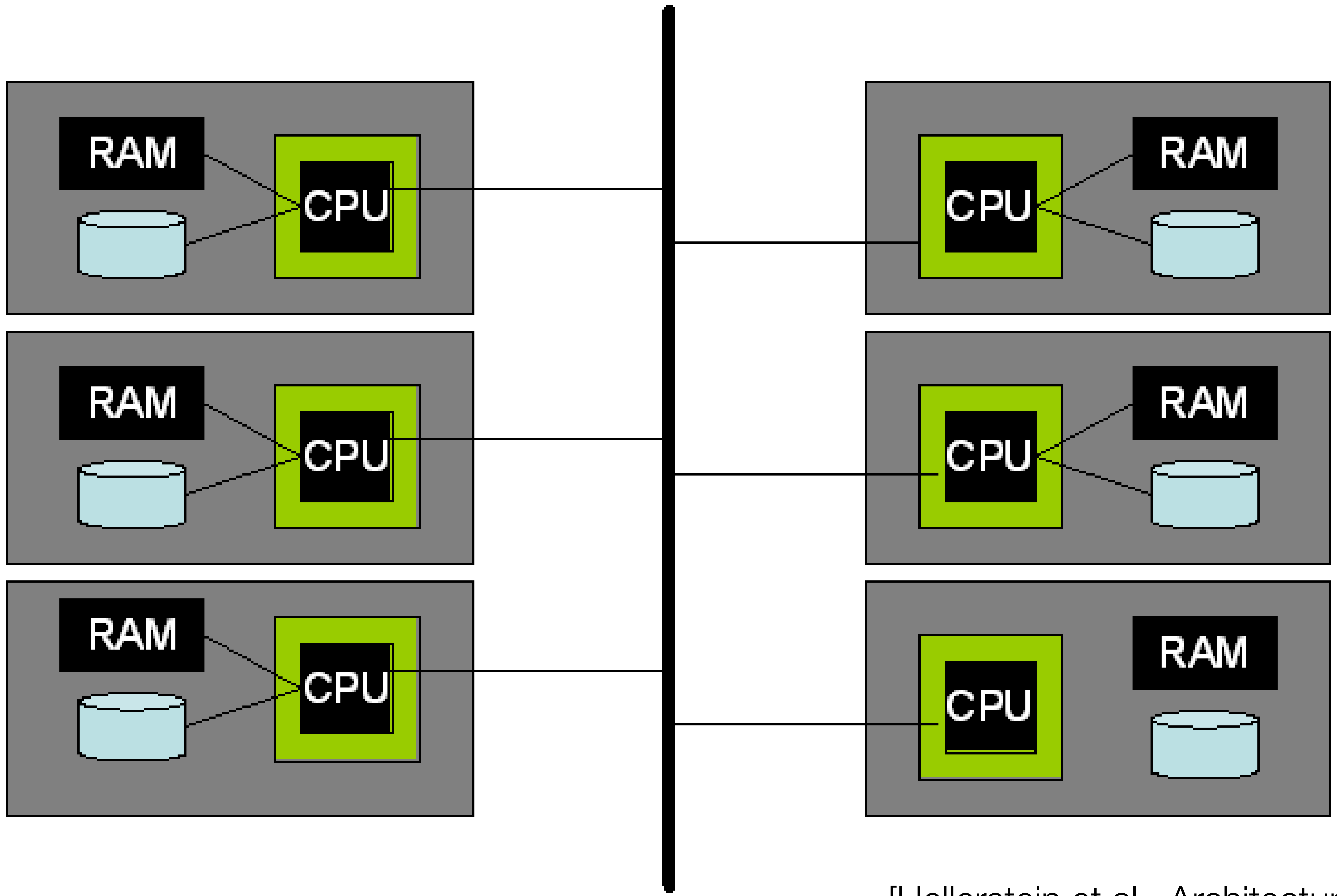
TrafficDB: Shared-Memory Data Store

- Traffic-aware route planning
- Want up-to-date data for all
- Thousands of requests per second
  - High-Frequency Reads
  - Low-Frequency Writes
- "Data must be stored in a region of RAM that can be shared and efficiently accessed by *several* different application processes"



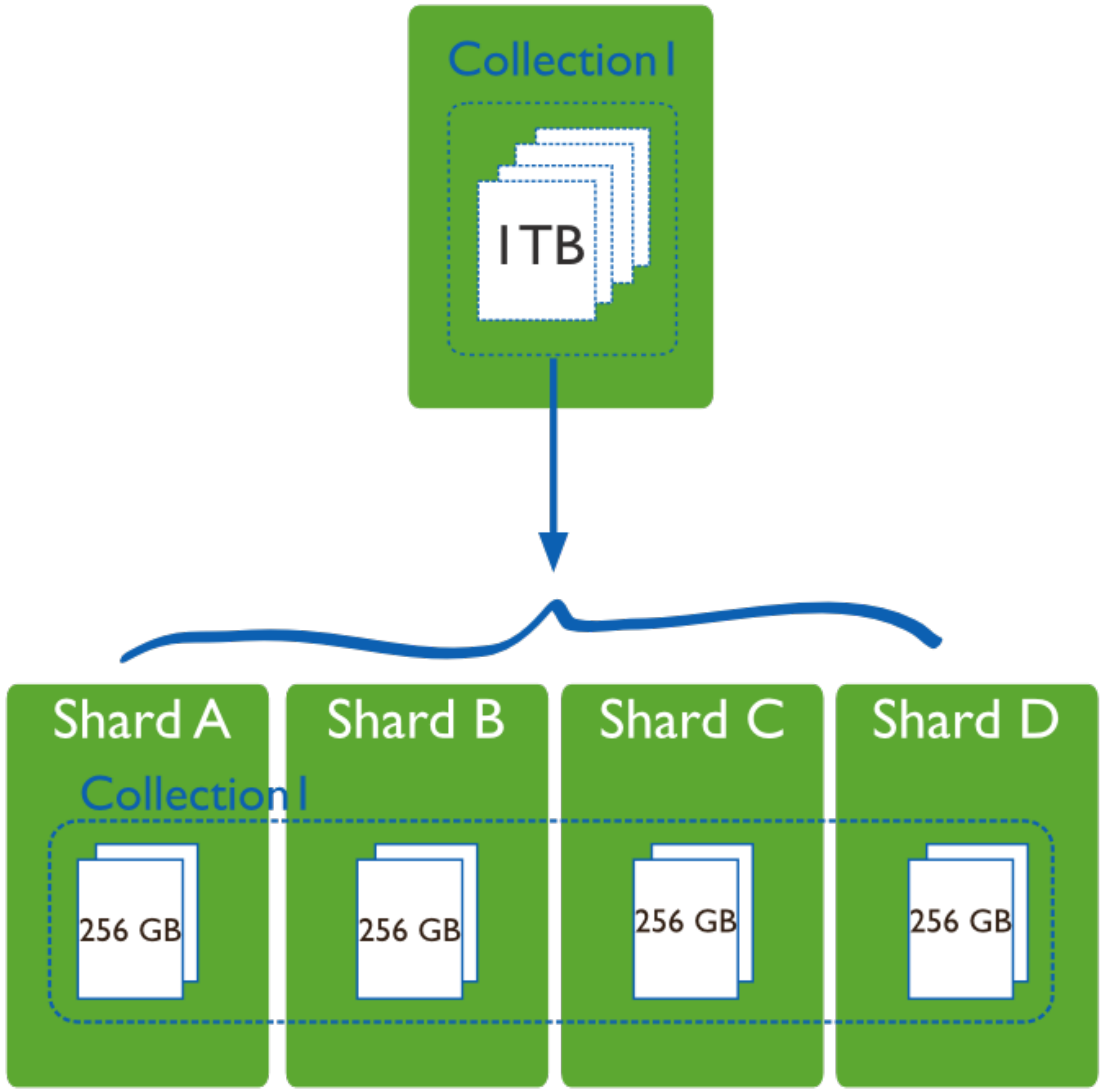
[R. Fernandes et al., 2016]

Parallel DB Architecture: Shared Nothing



[Hellerstein et al., Architecture of a Database System]

Sharding



[MongoDB]

Stonebraker: The End of an Architectural Era

- "RDBMSs were designed for the business data processing market, which is their sweet spot"
- "They can be beaten handily in most any other market of significant enough size to warrant the investment in a specialized engine"
- Changes in markets (science), necessary features (scalability), and technology (amount of memory)
- RDBMS Overhead: Logging, Latching, and Locking
- Relational model is not necessarily the answer
- SQL is not necessarily the answer

OLTP vs. OLAP

- Online Transactional Processing (OLTP) often used in business applications, data entry and retrieval transactions
- OLTP Examples:
  - Add customer's shopping cart to the database of orders
  - Find me all information about John Hammond's death
- OLTP is focused on the day-to-day operations while Online Analytical Processing (OLAP) is focused on analyzing that data for trends, etc.
- OLAP Examples:
  - Find the average amount spent by each customer
  - Find which year had the most movies with scientists dying

Row Stores

Primary Key

Row

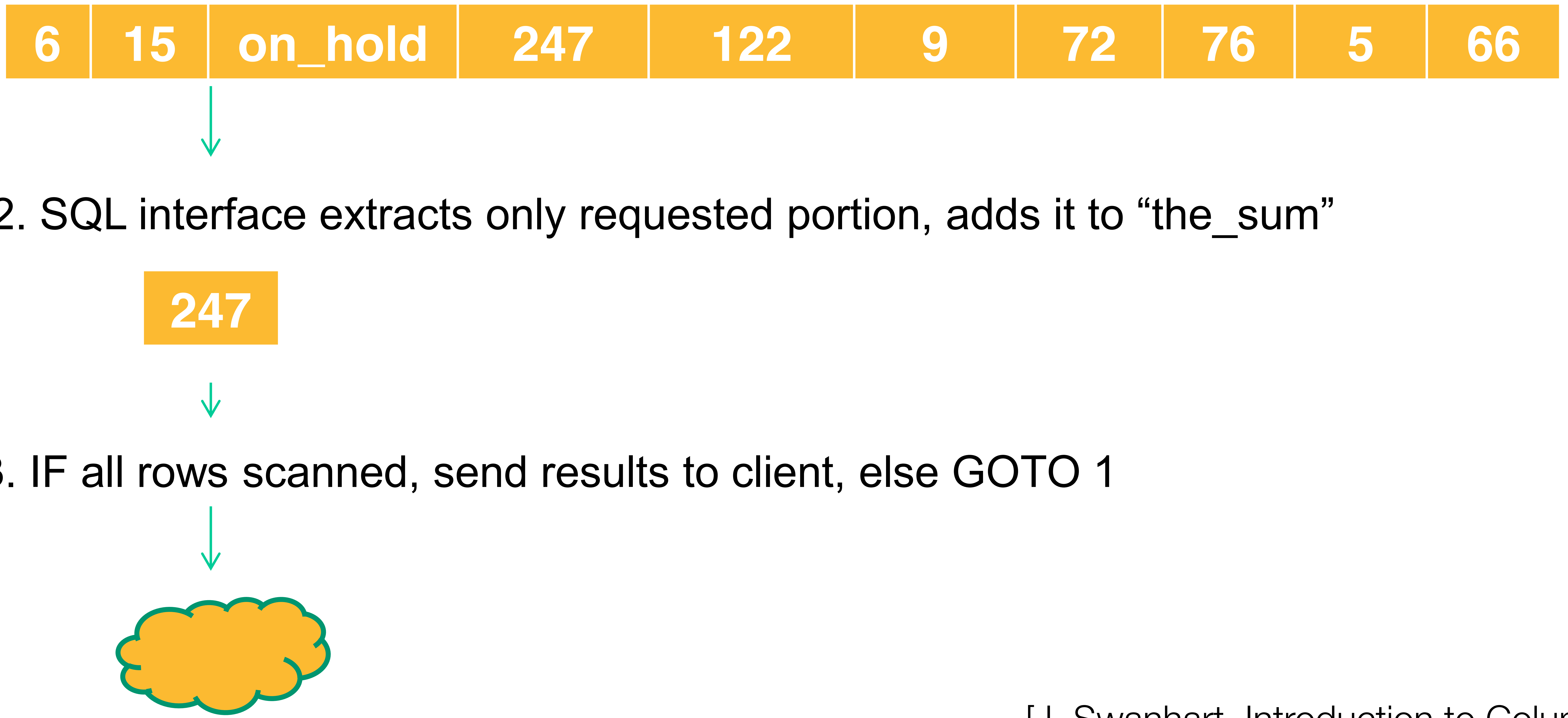
id	scientist	death_by	movie_name
1	Reinhardt	Crew	The Black Hole
2	Tyrell	Roy Batty	Blade Runner
3	Hammond	Dinosaur	Jurassic Park
4	Soong	Lore	Star Trek: TNG
5	Morbius	The machine	Forbidden Planet
6	Dyson	SWAT	Terminator 2: Judgment Day

[J. Swanhart, Introduction to Column Stores]

Inefficiency in Row Stores for OLAP

select sum(metric) as the_sum from fact

1. Storage engine gets *a whole row* from the table

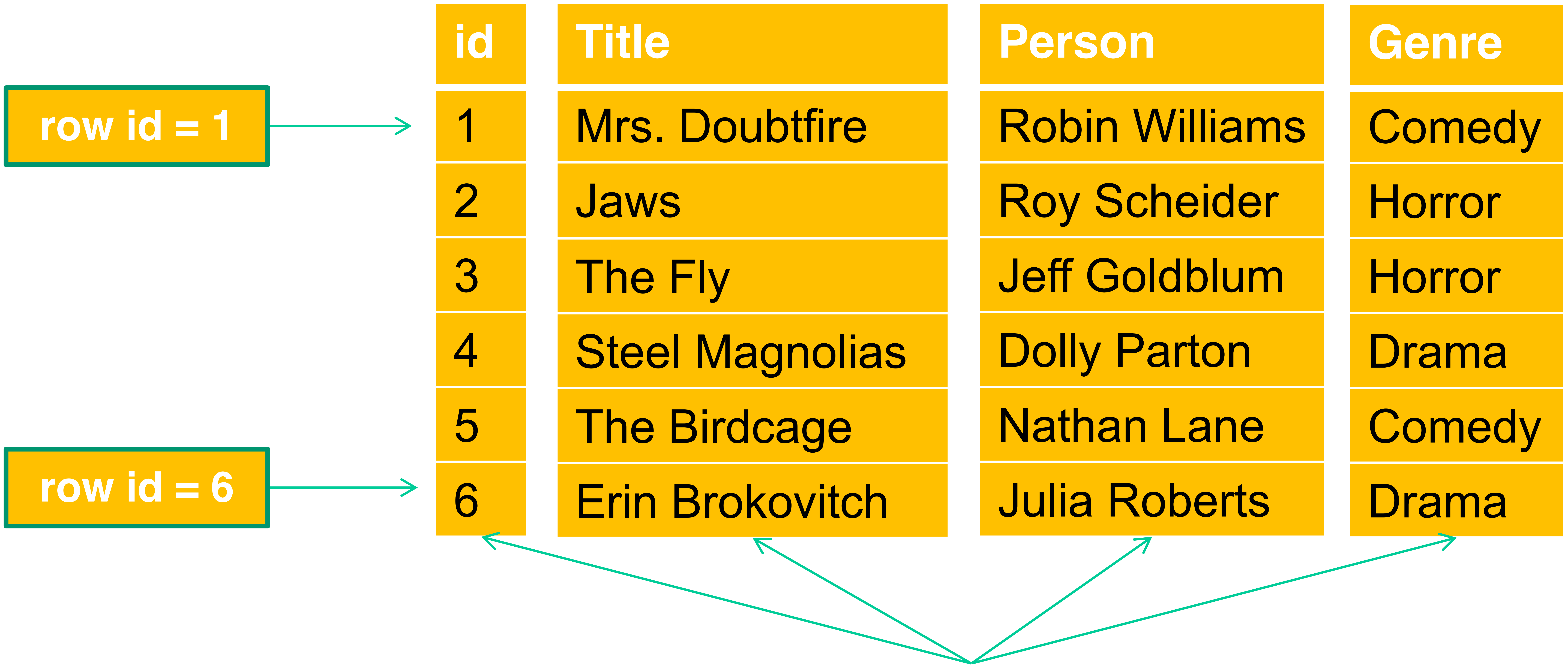


2. SQL interface extracts only requested portion, adds it to "the_sum"

3. IF all rows scanned, send results to client, else GOTO 1

[J. Swanhart, Introduction to Column Stores]

Column Stores



id	Title	Person	Genre
1	Mrs. Doubtfire	Robin Williams	Comedy
2	Jaws	Roy Scheider	Horror
3	The Fly	Jeff Goldblum	Horror
4	Steel Magnolias	Dolly Parton	Drama
5	The Birdcage	Nathan Lane	Comedy
6	Erin Brokovitch	Julia Roberts	Drama

Each column has a file or segment on disk

[J. Swanhart, Introduction to Column Stores]

Horizontal Partitioning vs. Vertical Partitioning

Original Table

CUSTOMER ID	FIRST NAME	LAST NAME	FAVORITE COLOR
1	TAEKO	OHNUKI	BLUE
2	O.V.	WRIGHT	GREEN
3	SELDA	BAĞCAN	PURPLE
4	JIM	PEPPER	AUBERGINE

[M. Drake]

Horizontal Partitioning vs. Vertical Partitioning

Vertical Partitions

VP1

CUSTOMER ID	FIRST NAME	LAST NAME
1	TAEKO	OHNUKI
2	O.V.	WRIGHT
3	SELDA	BAĞCAN
4	JIM	PEPPER

VP2

CUSTOMER ID	FAVORITE COLOR
1	BLUE
2	GREEN
3	PURPLE
4	AUBERGINE

Original Table

CUSTOMER ID	FIRST NAME	LAST NAME	FAVORITE COLOR
1	TAEKO	OHNUKI	BLUE
2	O.V.	WRIGHT	GREEN
3	SELDA	BAĞCAN	PURPLE
4	JIM	PEPPER	AUBERGINE

Horizontal Partitions

HP1

CUSTOMER ID	FIRST NAME	LAST NAME	FAVORITE COLOR
1	TAEKO	OHNUKI	BLUE
2	O.V.	WRIGHT	GREEN

HP2

CUSTOMER ID	FIRST NAME	LAST NAME	FAVORITE COLOR
3	SELDA	BAĞCAN	PURPLE
4	JIM	PEPPER	AUBERGINE

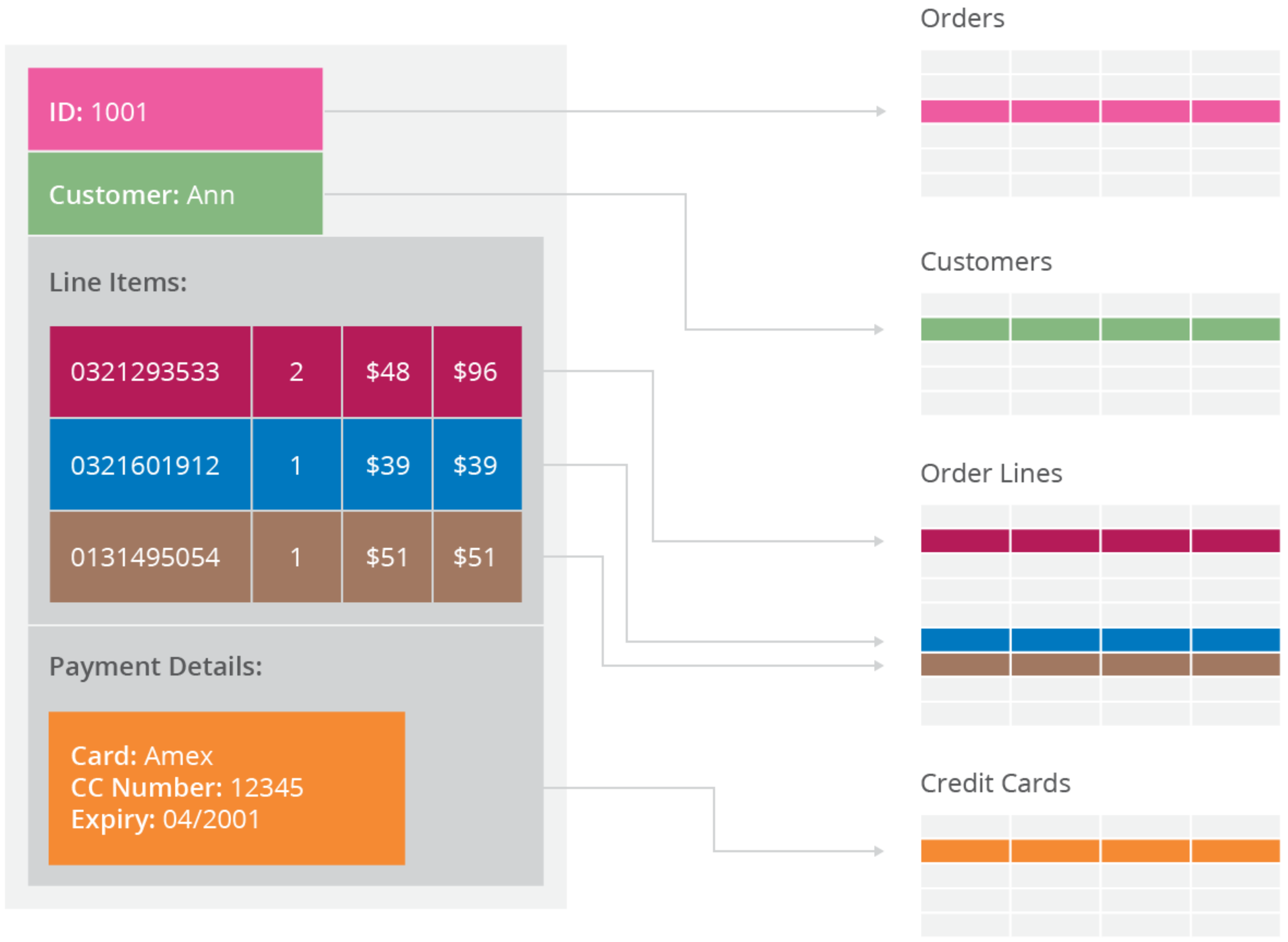
[M. Drake]

NoSQL Paradigm Shift

Commercial DBMS	➡	Open-Source DBMS
Specialized DB hardware (Oracle Exadata, etc.)	➡	Commodity hardware
Highly available network (Infiniband, Fabric Path, etc.)	➡	Commodity network (Ethernet, etc.)
Highly Available Storage (SAN, RAID, etc.)	➡	Commodity drives (standard HDDs, JBOD)

[F. Gessert et al., 2017]

Problems with Relational Databases



[P. Sadalage]

NoSQL Classification Criteria

Data Model

- Key-Value
- Wide-Column
- Document
- Graph

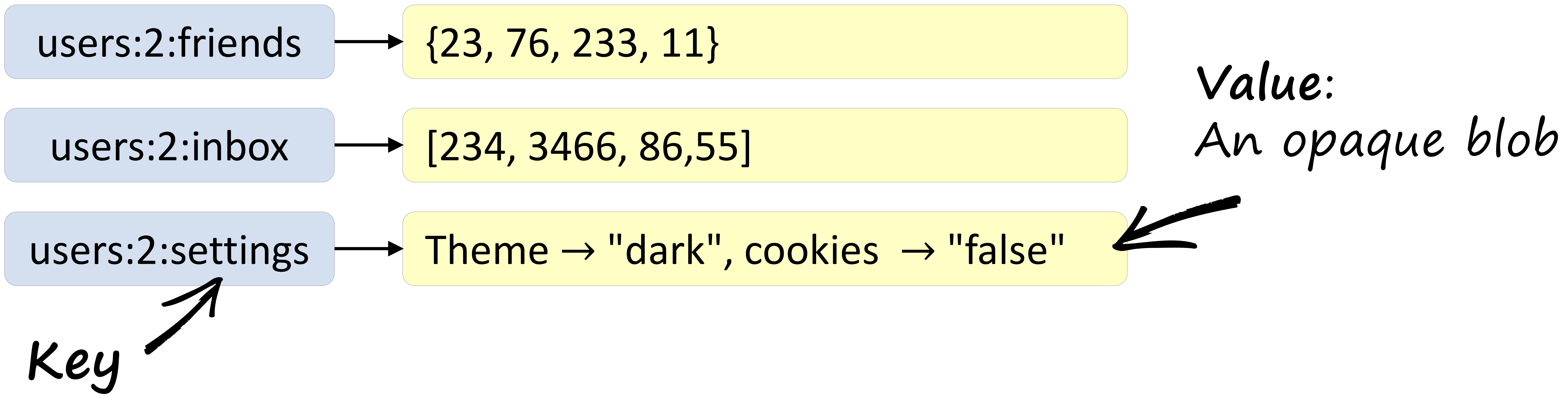
Consistency/Availability Trade-Off

- **AP**: Available & Partition Tolerant
- **CP**: Consistent & Partition Tolerant
- **CA**: Not Partition Tolerant

[F. Gessert et al., 2017]

Key-Value Stores

- **Data model:** (key) -> value
- **Interface**: CRUD (Create, Read, Update, Delete)

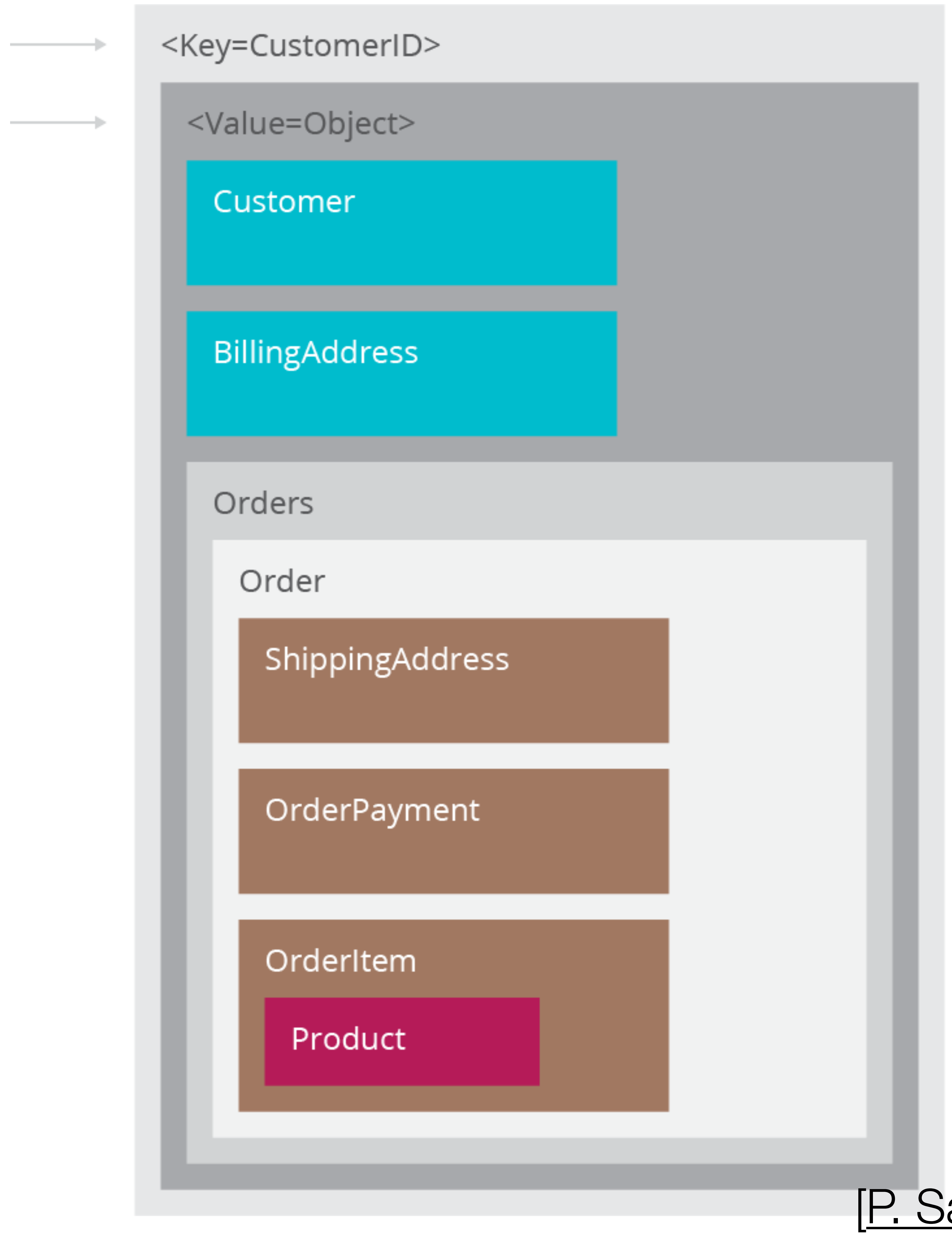


- Examples: Amazon Dynamo (AP), Riak (AP), Redis (CP)

[F. Gessert et al., 2017]

Key-Value Stores

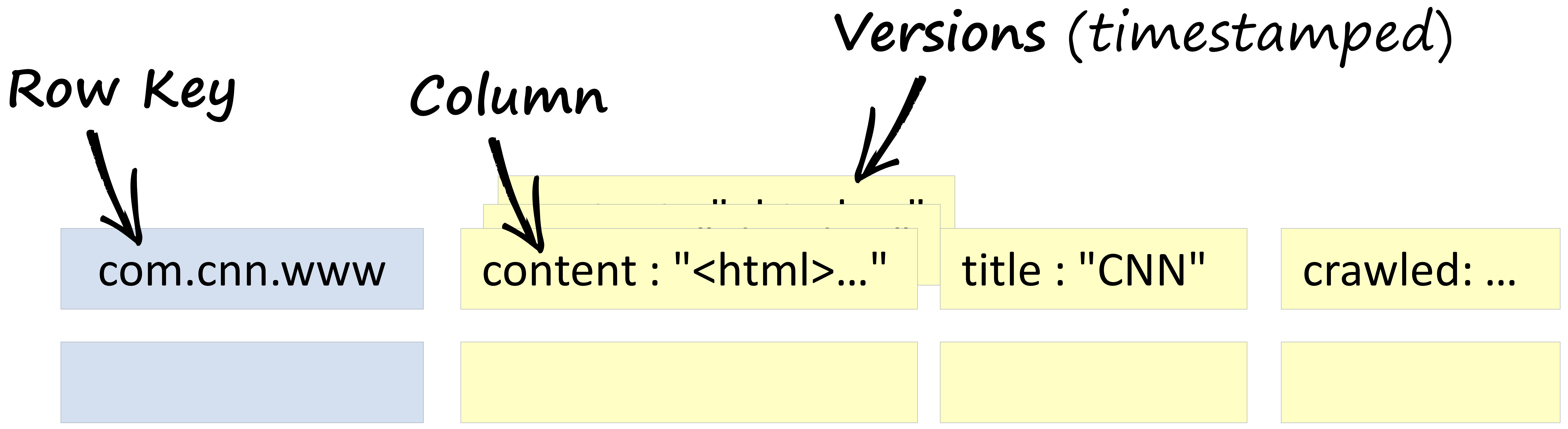
- Always use primary-key access
- Operations:
  - Get/put value for key
  - Delete key



[P. Sadalage]

Wide-Column Stores

- **Data model:** (rowkey, column, timestamp) -> value
- **Interface**: CRUD, Scan

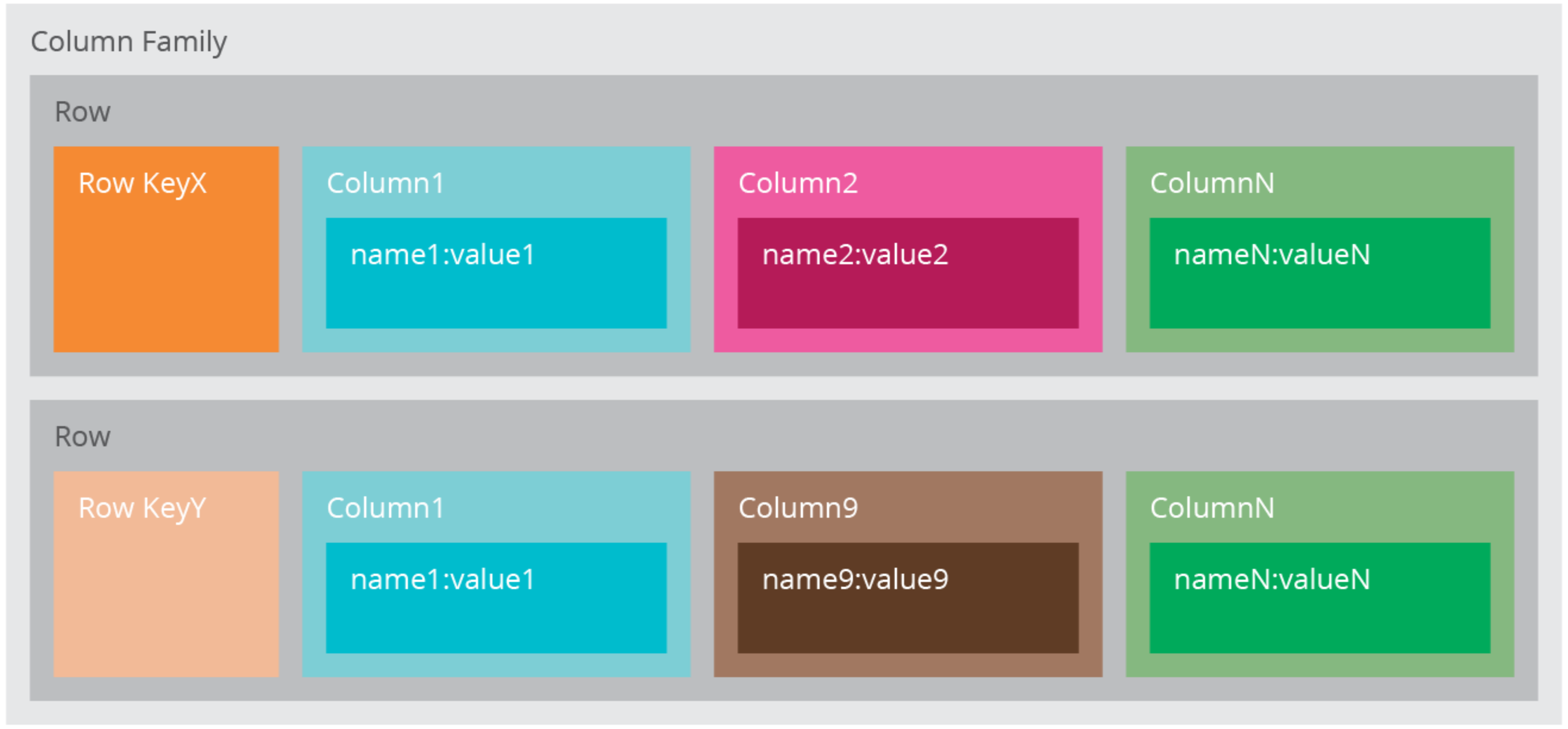


- Examples: Cassandra (AP), Google BigTable (CP), HBase (CP)

[F. Gessert et al., 2017]

Column Stores

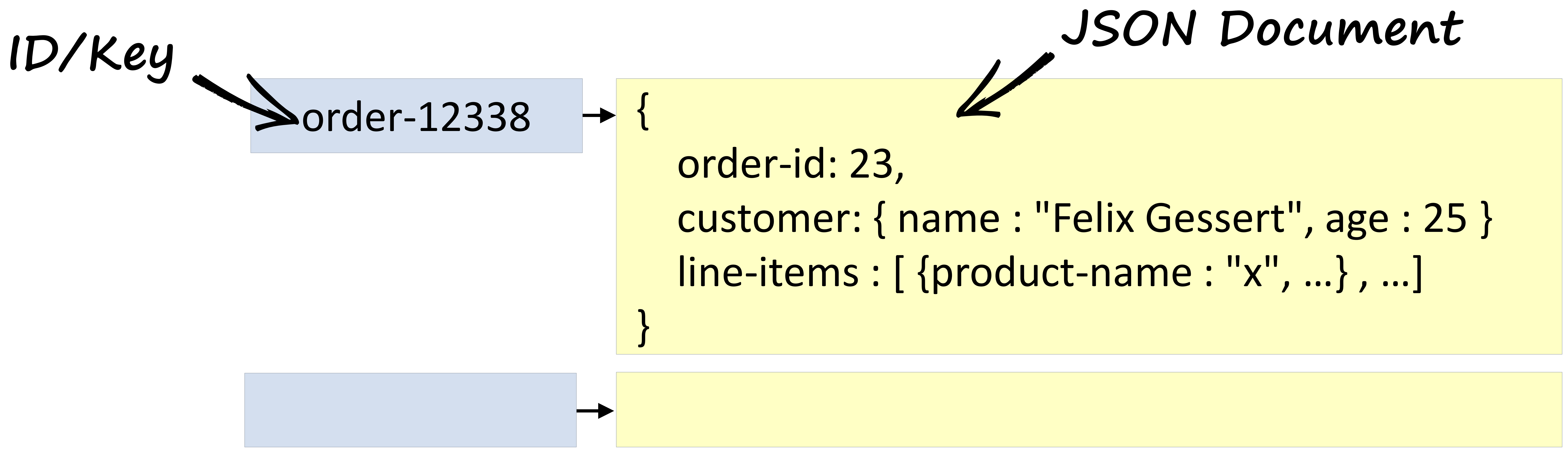
- Instead of having rows grouped/sharded, we group columns
- …or families of columns
- Put similar columns together



[P. Sadalage]

Document Stores

- **Data model:** (collection, key) -> document
- **Interface**: CRUD, Querys, Map-Reduce



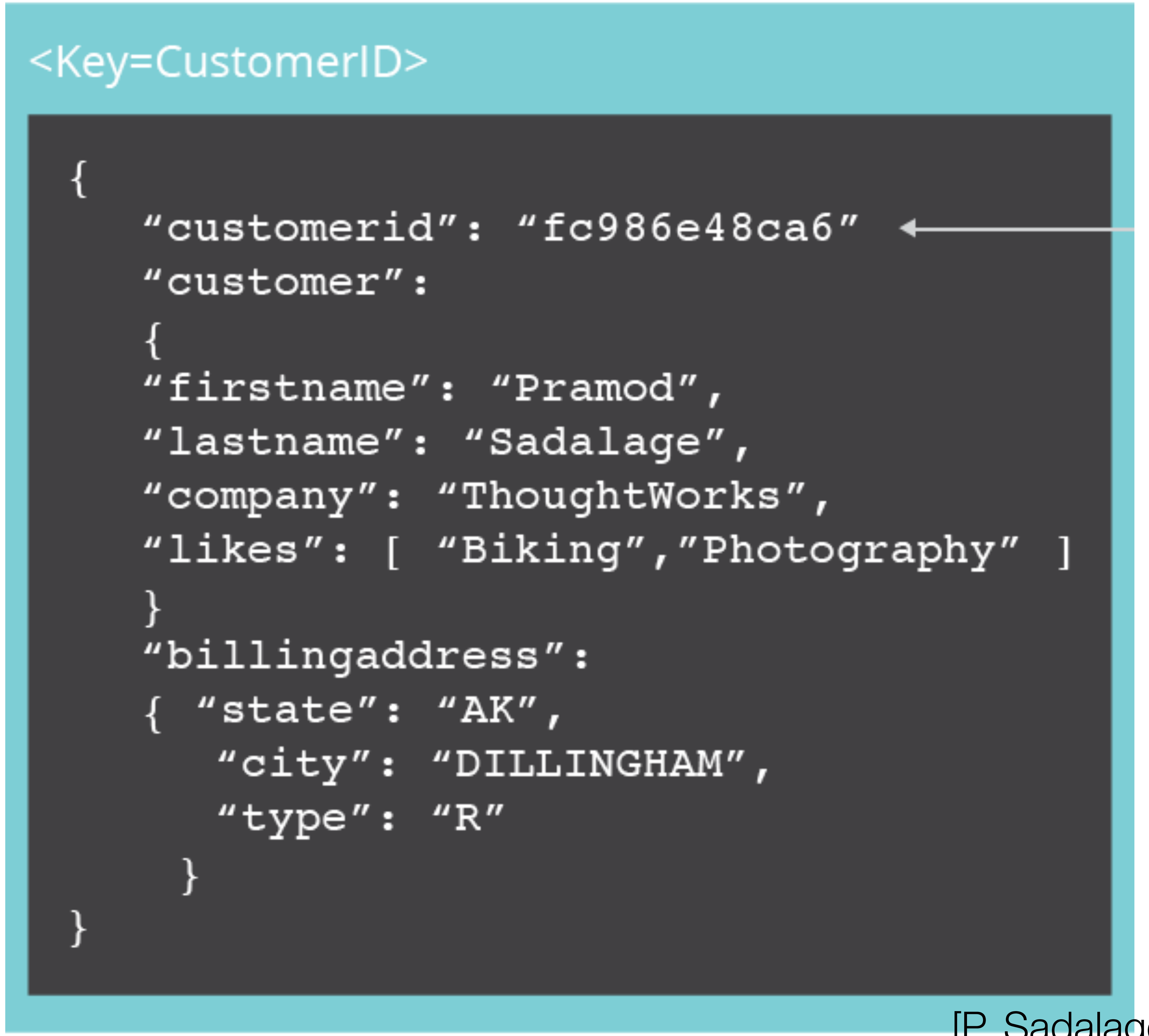
```
{
   order-id: 23,
   customer: { name : "Felix Gessert", age : 25 }
   line-items : [ {product-name : "x", …} , …]
}
```

- Examples: CouchDB (AP), RethinkDB (CP), MongoDB (CP)

[F. Gessert et al., 2017]

Document Stores

- Documents are the main entity
  - Self-describing
  - Hierarchical
  - Do not have to be the same
- Could be XML, JSON, etc.
- Key-value stores where values are "examinable"
- Can have query language and indices overlaid



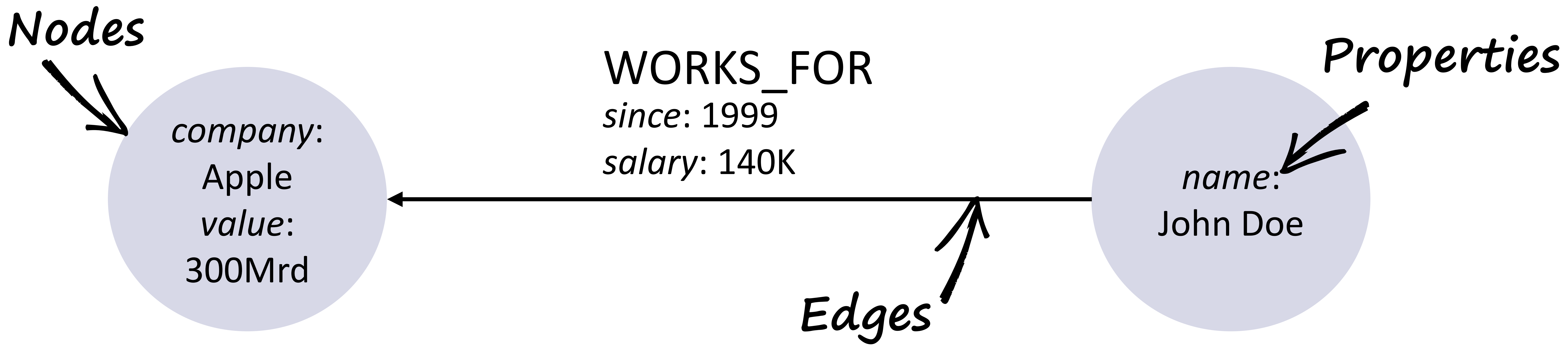
<Key=CustomerID>

```
{
    "customerid": "fc986e48ca6"
    "customer":
    {
    "firstname": "Pramod",
    "lastname": "Sadalage",
    "company": "ThoughtWorks",
    "likes": [ "Biking","Photography" ]
    }
    "billingaddress":
    { "state": "AK",
      "city": "DILLINGHAM",
      "type": "R"
     }
}
```

[P. Sadalage]

Graph Databases

- **Data model:** G = (V, E): Graph-Property Modell
- **Interface**: Traversal algorithms, querys, transactions

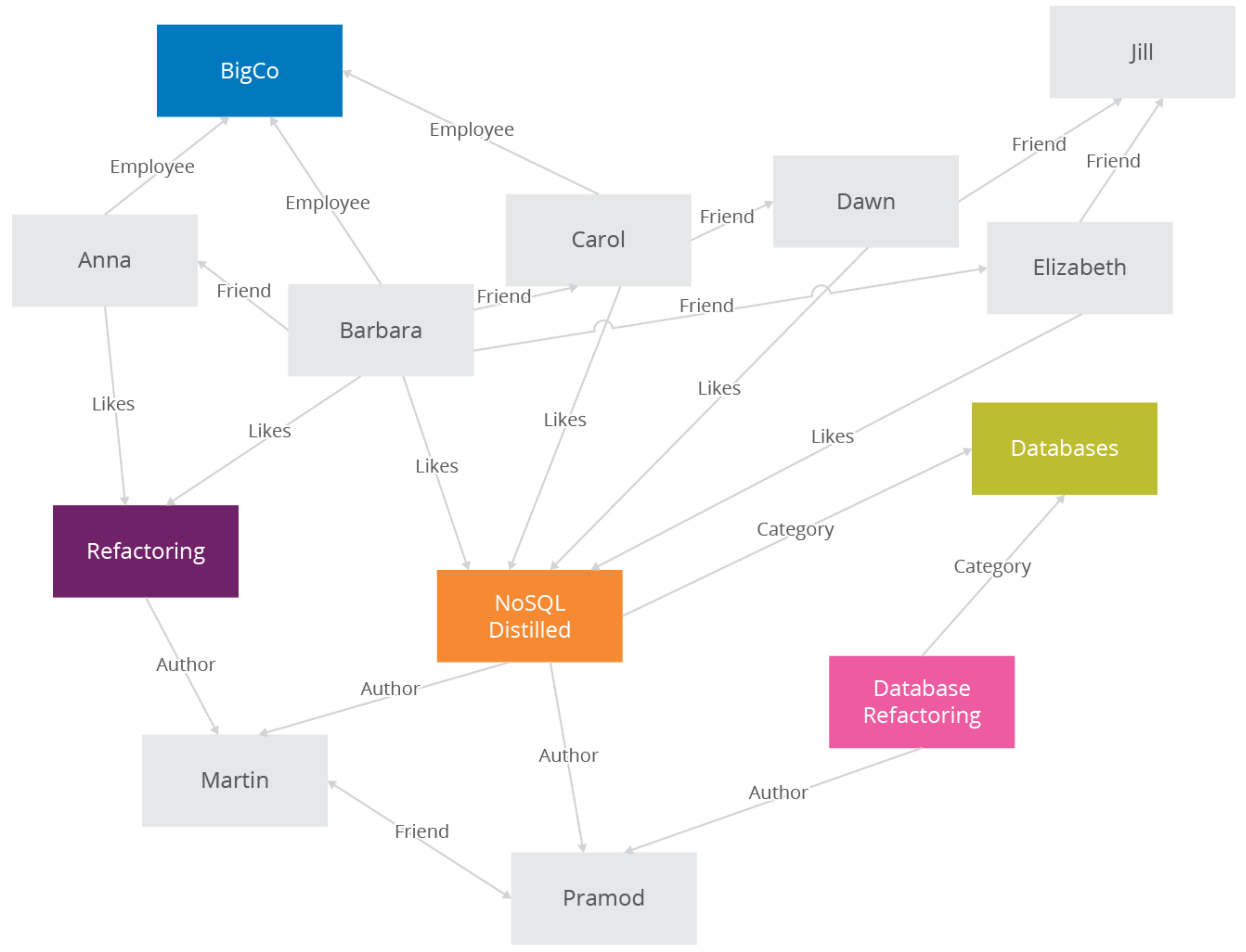


- Examples: Neo4j (CA), InfiniteGraph (CA), OrientDB (CA)

[F. Gessert et al., 2017]

Graph Databases

- Focus on entities and relationships
- Edges may have properties
- Relational databases required a set traversal
- Traversals in Graph DBs are faster



[P. Sadalage]

NoSQL Classification Criteria

Data Model

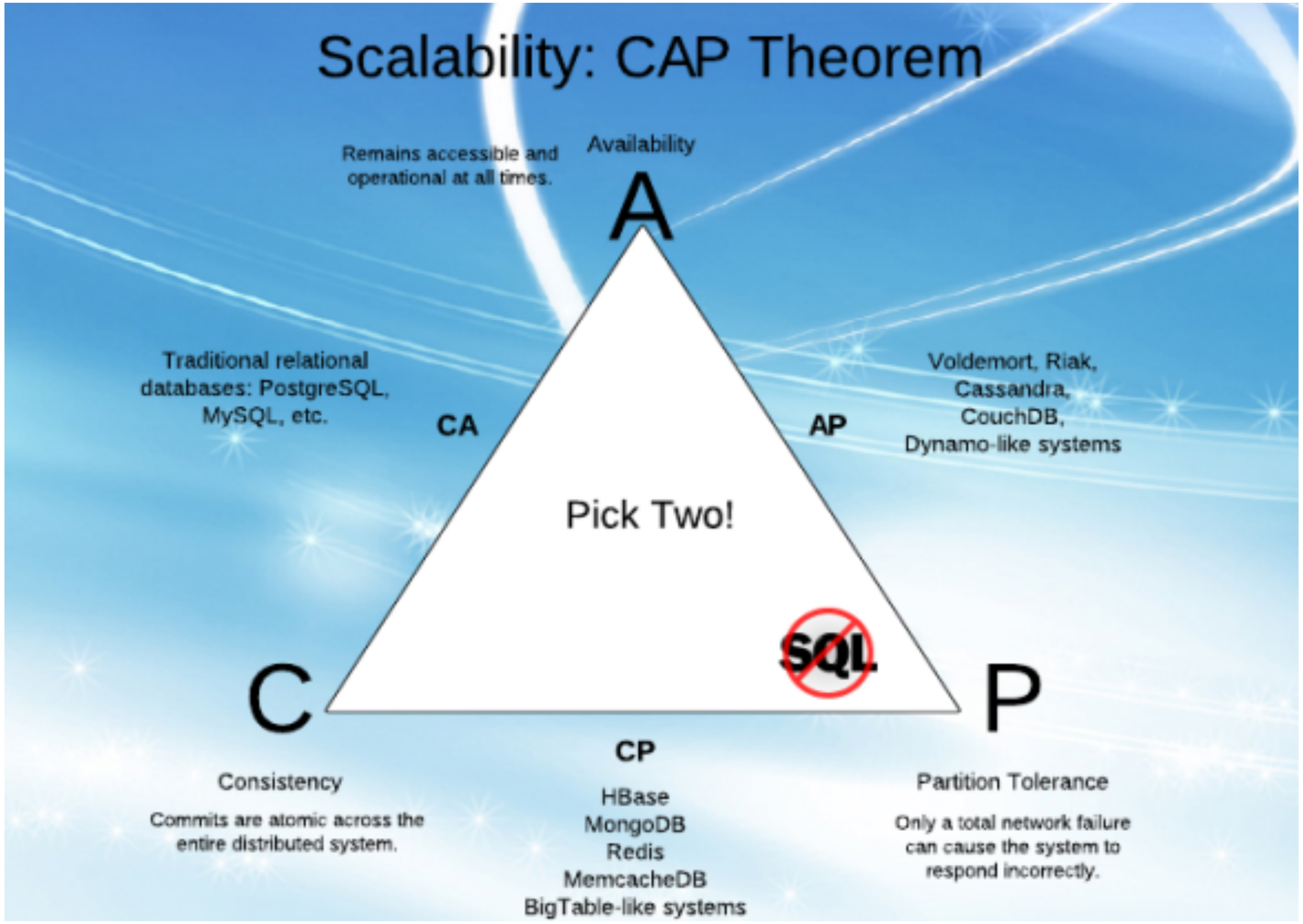
- Key-Value
- Wide-Column
- Document
- Graph

Consistency/Availability Trade-Off

- **AP**: Available & Partition Tolerant
- **CP**: Consistent & Partition Tolerant
- **CA**: Not Partition Tolerant

[F. Gessert et al., 2017]

CAP Theorem


Scalability: CAP Theorem
Availability
Remains accessible and operational at all times.
A
Traditional relational databases: PostgreSQL, MySQL, etc.
CA
AP
Voldemort, Riak, Cassandra, CouchDB, Dynamo-like systems
Pick Two!
SQL
C
P
CP
Consistency
Commits are atomic across the entire distributed system.
HBase
MongoDB
Redis
MemcacheDB
BigTable-like systems
Partition Tolerance
Only a total network failure can cause the system to respond incorrectly.

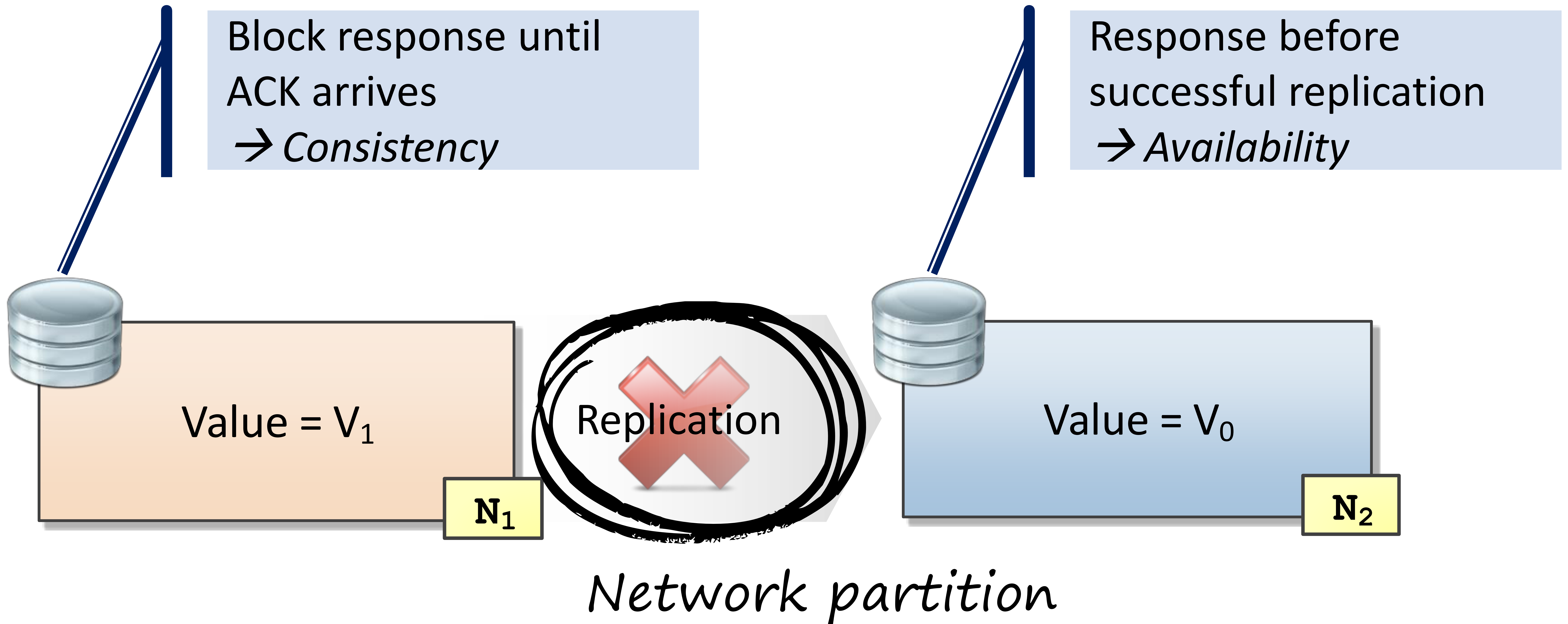
[E. Brewer]

CAP Theorem

- Consistency: every read would get you the most recent write
- Availability: every node (if not failed) always executes queries
- Partition tolerance: system continues to work even if nodes are down
- Theorem (Brewer): It is impossible for a distributed data store to simultaneously provide more than two of Consistency, Availability, and Partition Tolerance

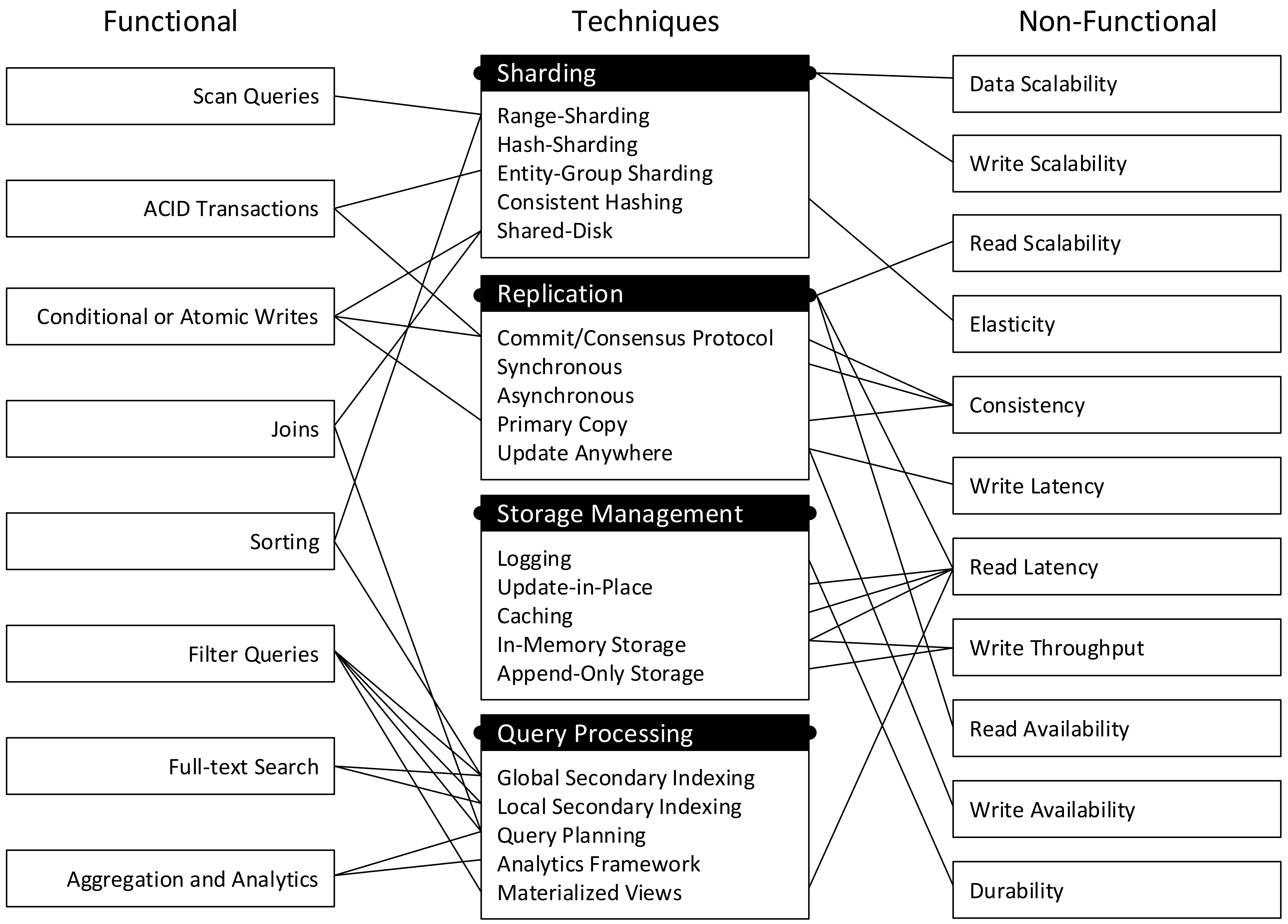
CAP Theorem "Proof"

- If there is a network partition, one of consistency or availability will not be possible



[F. Gessert et al., 2017]

NoSQL Techniques



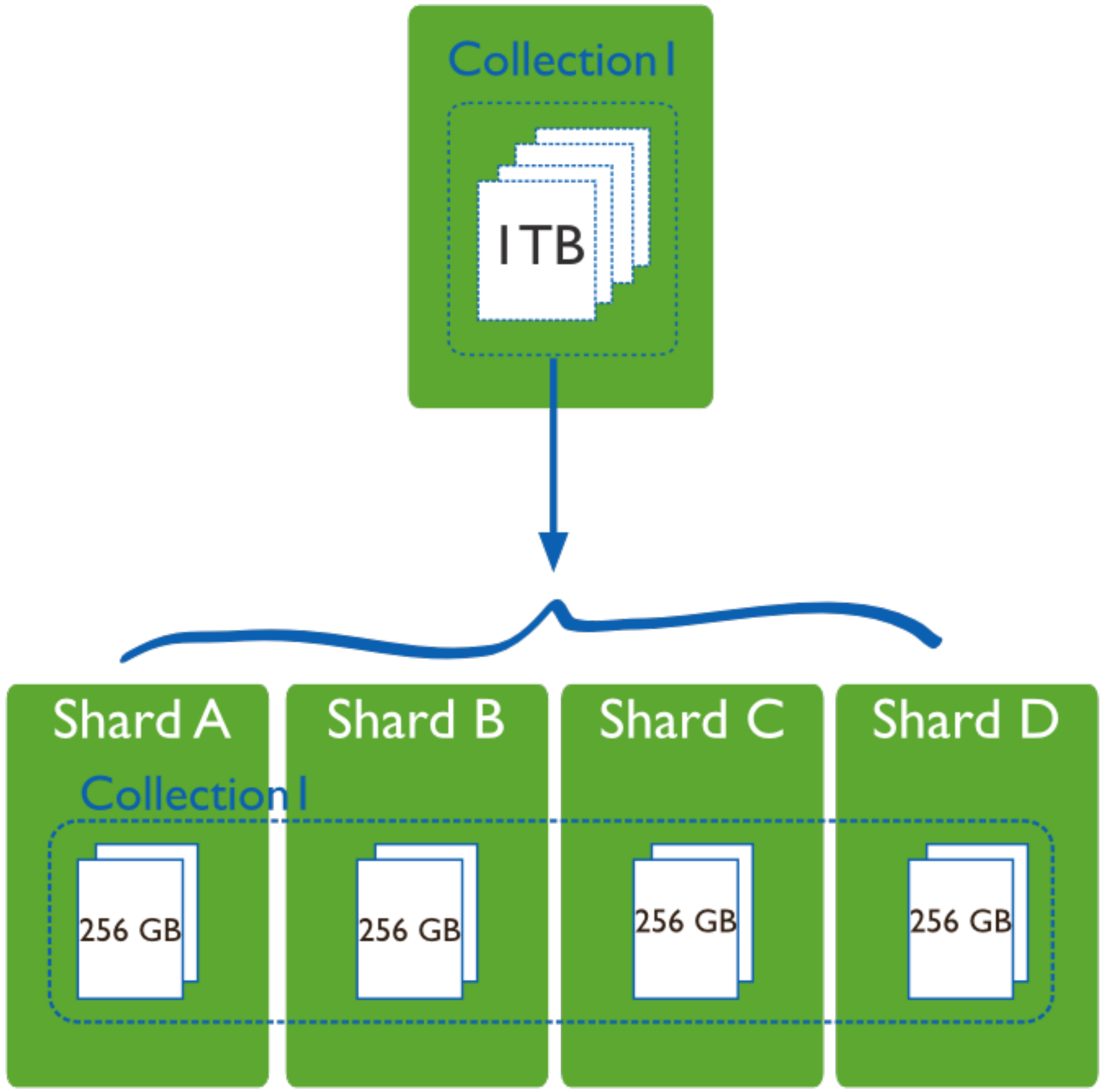
[F. Gessert et al., 2017]

Distributing Data

- Aggregate-oriented databases
- Sharding (horizontal partitioning): Sharding distributes different data across multiple servers, so each server acts as the single source for a subset of data
- Replication: Replication copies data across multiple servers, so each bit of data can be found in multiple places. Replication comes in two forms,
  - Source-replica replication makes one node the authoritative copy that handles writes, replica synchronizes with the source and may handle reads.
  - Peer-to-peer replication allows writes to any node; the nodes coordinate to synchronize their copies of the data.

[P. Sadalage]

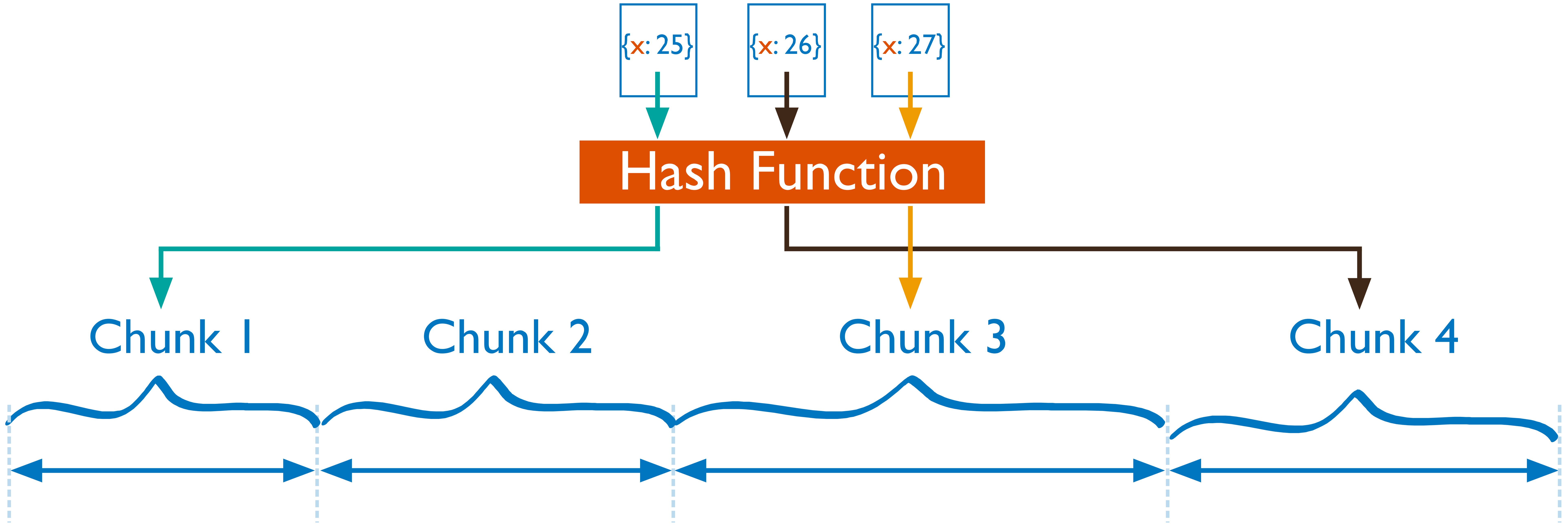
Sharding



[MongoDB]

Sharding Approaches

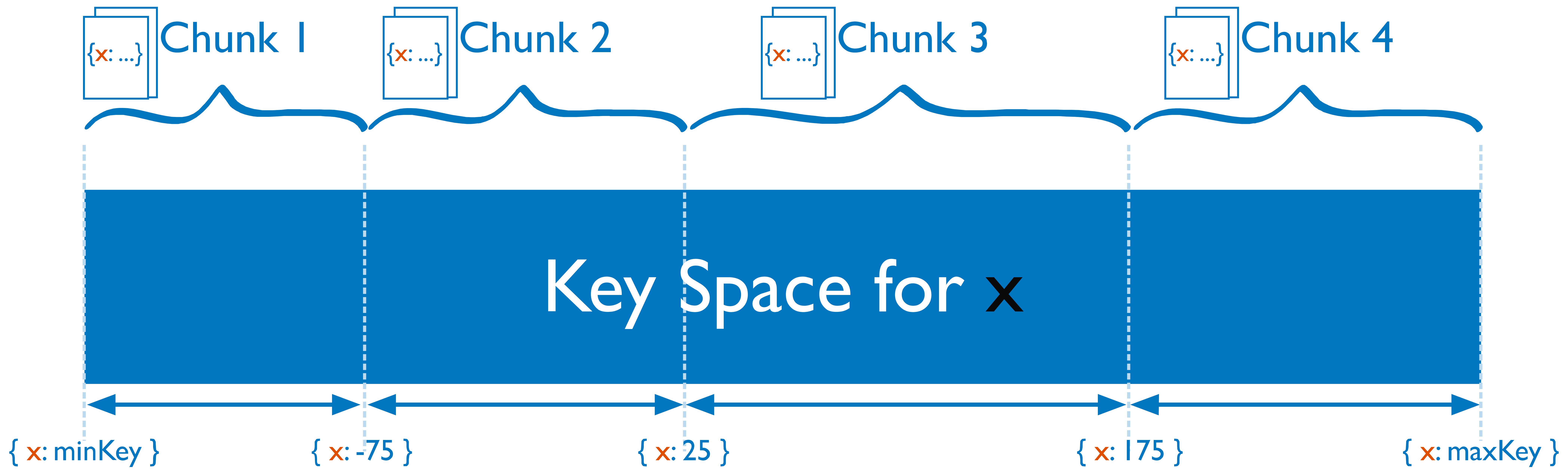
- Hash-based Sharding
  - Hash of data values (e.g. key) determines partition (shard)
  - Pro: Even distribution, Con: No data locality



[D. DeWitt & J. Gray, 1992, via F. Gessert, Image: MongoDB]

Sharding Approaches

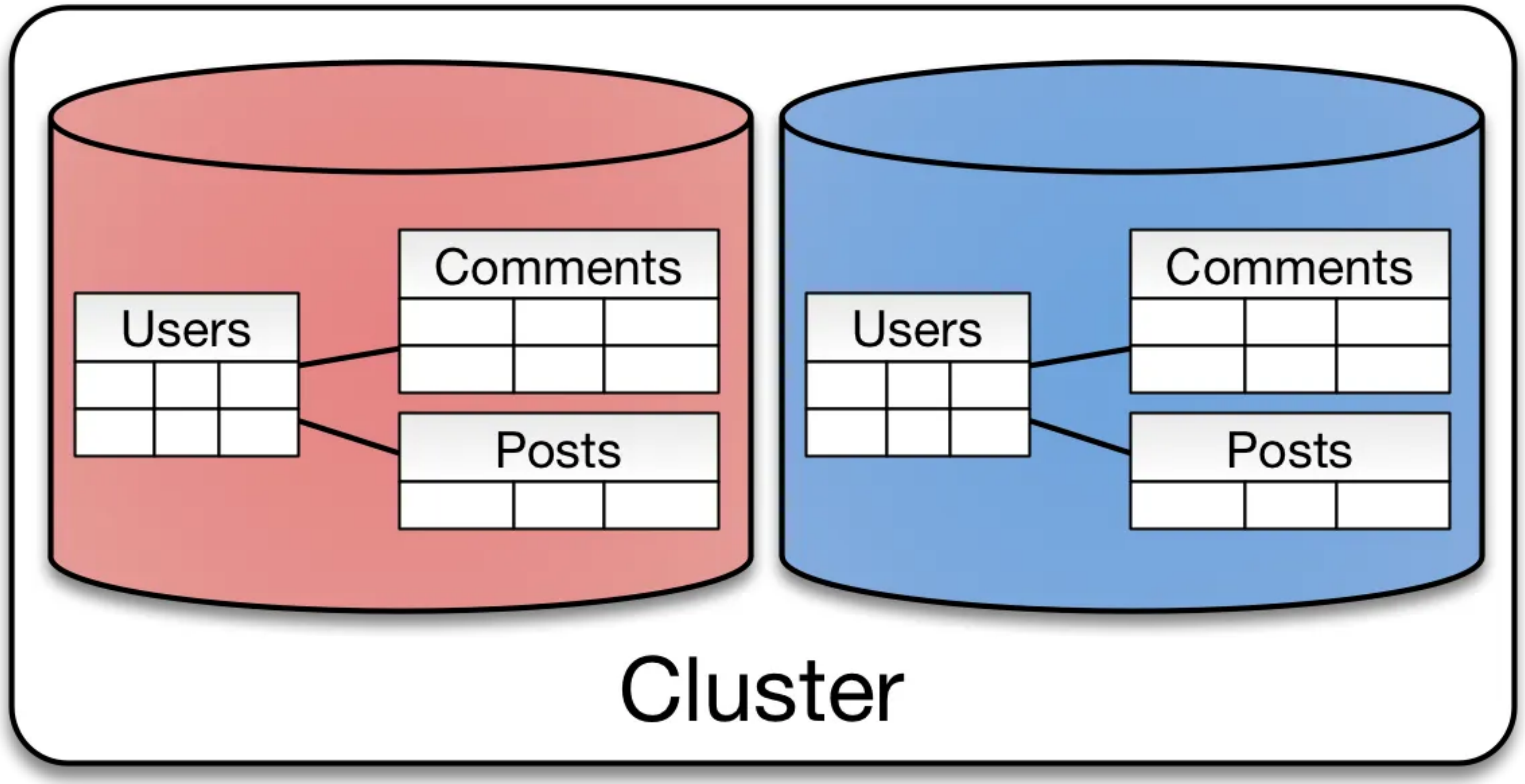
- Range-based Sharding
  - Assigns ranges defined over fields (shard keys) to partitions
  - Pro: Enables Range Scans & Sorting, Con: Repartitioning/balancing req'd



[D. DeWitt & J. Gray, 1992, via F. Gessert, Image: MongoDB]

Sharding Approaches

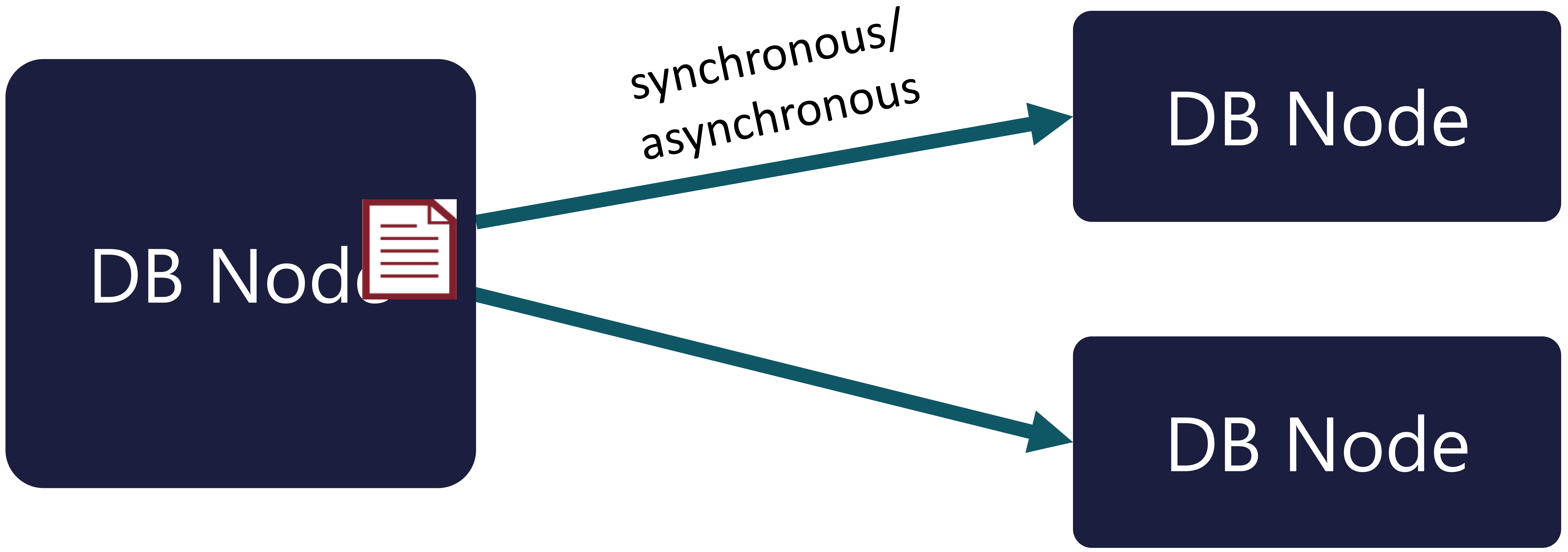
- Entity-Group Sharding
  - Explicit data co-location for single-node-transactions
  - Pro: Enables ACID Transactions, Con: Partitioning not easily changable



[D. DeWitt & J. Gray, 1992, via F. Gessert, Image: J. Kim]

Replication

- Store N copies of each data item
- Consistency model: synchronous vs. asynchronous
- Coordination: Multiple Primary, Primary/Replica



[F. Gessert et al., 2017]

Replication: When

- Asynchronous (lazy)
  - Writes are acknowledged immdediately
  - Performed through log shipping or update propagation
  - Pro: Fast writes, no coordination needed
  - Con: Replica data potentially stale (inconsistent)
- Synchronous (eager)
  - The node accepting writes synchronously propagates updates/transactions before acknowledging
  - Pro: Consistent
  - Con: needs a commit protocol (more roundtrips), unavailable under certain network partitions

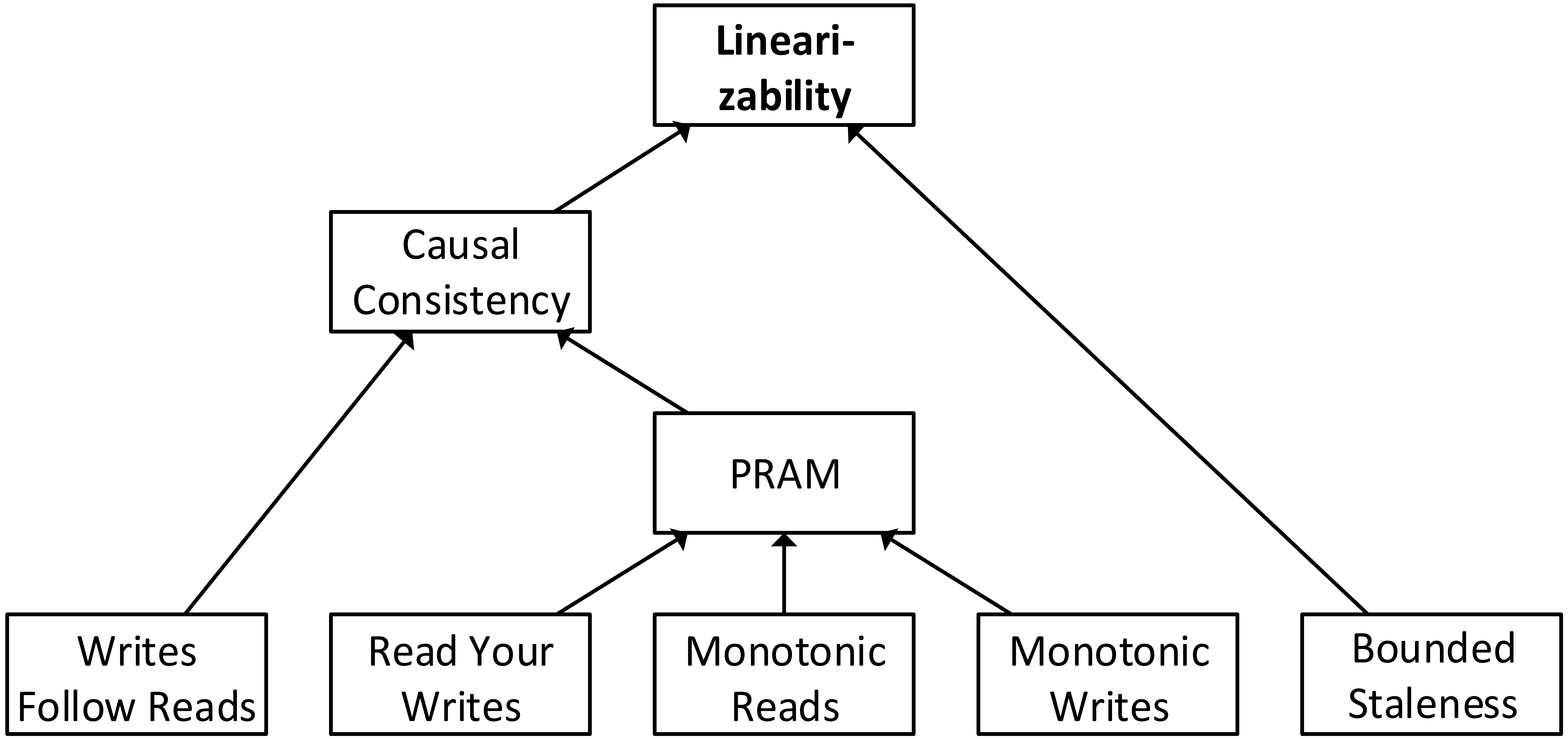
[F. Gessert et al., 2017]

Replication: Where

- Primary-Replica (Primary Copy)
  - Only a dedicated primary is allowed to accept writes, replicas are read-replicas
  - Pro: reads from the primary are consistent
  - Con: primary is a bottleneck and SPOF
- Multi-Primary (Update anywhere)
  - The server node accepting the writes synchronously propagates the update or transaction before acknowledging
  - Pro: fast and highly-available
  - Con: either needs coordination protocols (e.g. Paxos) or is inconsistent

[F. Gessert et al., 2017]

Consistency Levels



[via [F. Gessert et al., 2017]]

Next Class's Paper Critique

- Read What’s Really New with NewSQL?
- Submit critique **before class** on Wednesday, October 22
- Discussion ideas:
  - What are the advantages or disadvantages of NewSQL vs NoSQL?
  - Are they really different from standard RDBMS?
  - Which category of NewSQL databases is most exciting?