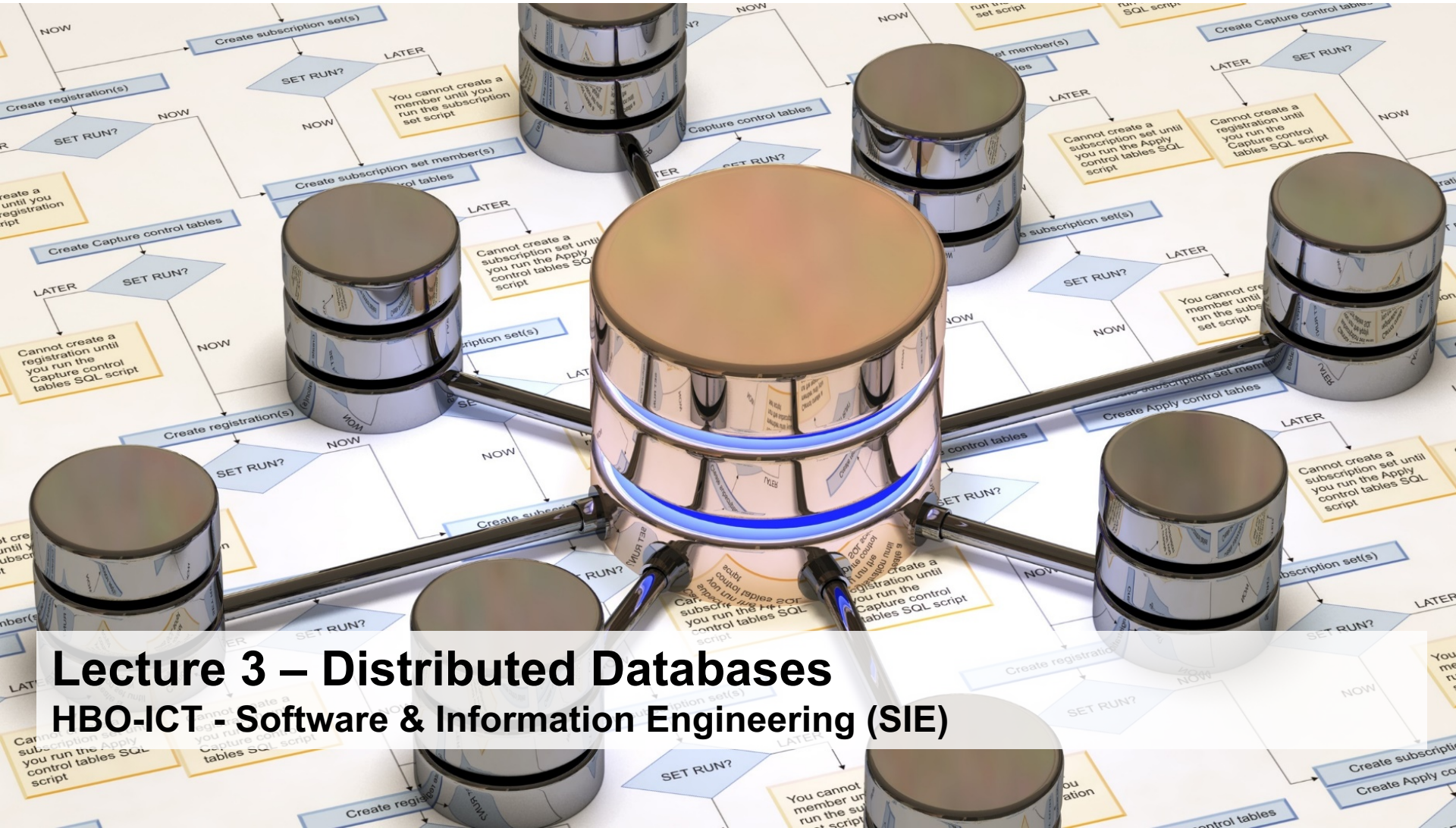
Big Data System Design

(TICT-VKBDSD-17), 11 februari 2019



Lecture 3 – Distributed Databases

HBO-ICT - Software & Information Engineering (SIE)

Learning outcomes

Purpose & content
RDBMD
Con'cy
Scaling
NoSQL

- Describe the impact of infrastructuur on Big Data systems.
- Describe and determine distributed persistence-solution.
- Evaluate existing persistence-solution to Big Data requirements.

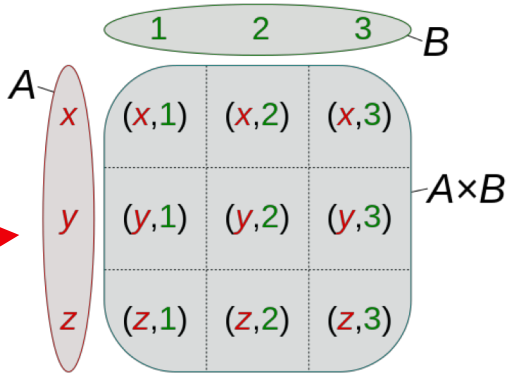
Content

- Relational Databases
- Concurrency in RDBMS
- Scaling RDBMS
- NoSQL Solutions

Relational (or *Table-oriented*) Data Model

- Based on *Set theory*
  - Selection (*which rows*)
  - Projection (*which columns*)
  - Join *(Cartesian product)*
  - Classical (*union, intersection, difference*)
- Data integrity only ensured by *design (e.g. normalization)!*
  - Prevents anomalies by eliminating *redundancies*
  - Increases amount of tables

Very Generic!



Purpose & content
RDBMD
Con'cy
Scaling
NoSQL

Attribute (*Column*)

Value (*Cell*)

Attribute names

Data types

Tuple (*Row/Record*)

Schema (*Table*)

Extension

Student		
Name:	**Sex:**	**Date of birth:**
varchar(60)	**char(1)**	**date**
Tim	M	12-09-1999
Roslin	F	29-02-2000
Marietje	F	06-11-1999
Erik	M	03-05-2000

ACID Properties of RDBMS

Purpose & content
RDBMD
Con'cy
Scaling
NoSQL

Atomicity

- All-or-nothing
- Transaction can not be subdivided

Consistency

- Consistent state + Transaction → Consistent state
  1. Entity integrity (e.g. PKs)
  2. Referential integrity (e.g. FKs)
  3. Domain integrity (e.g. atomic-types)
  4. User-defined

Integrity constraints (ICs) (schema)

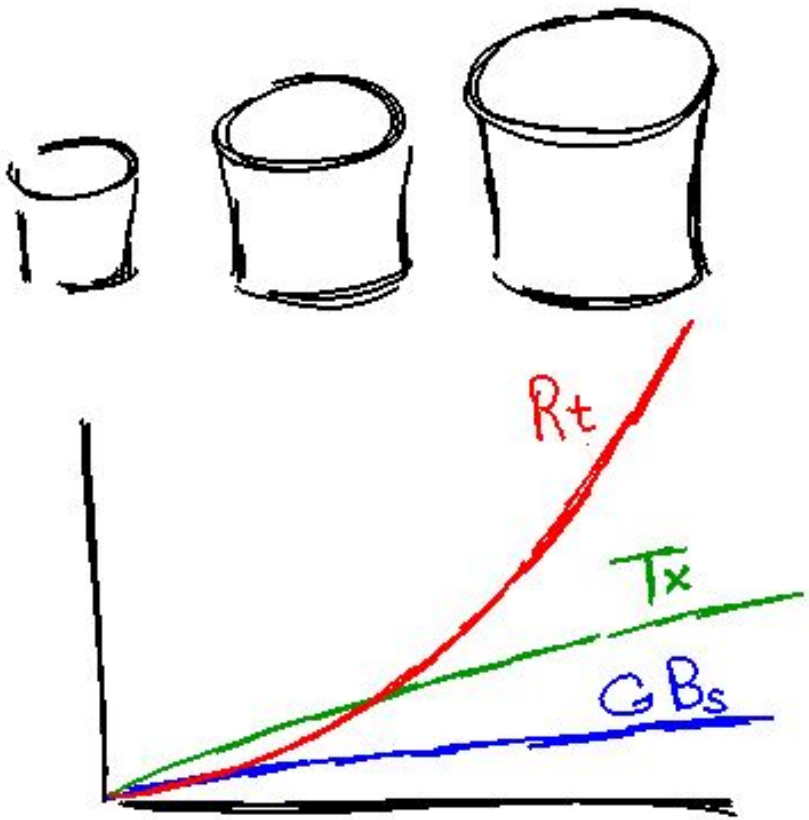
Isolation

- Concurrent transactions only see *commited* transactions

Durability

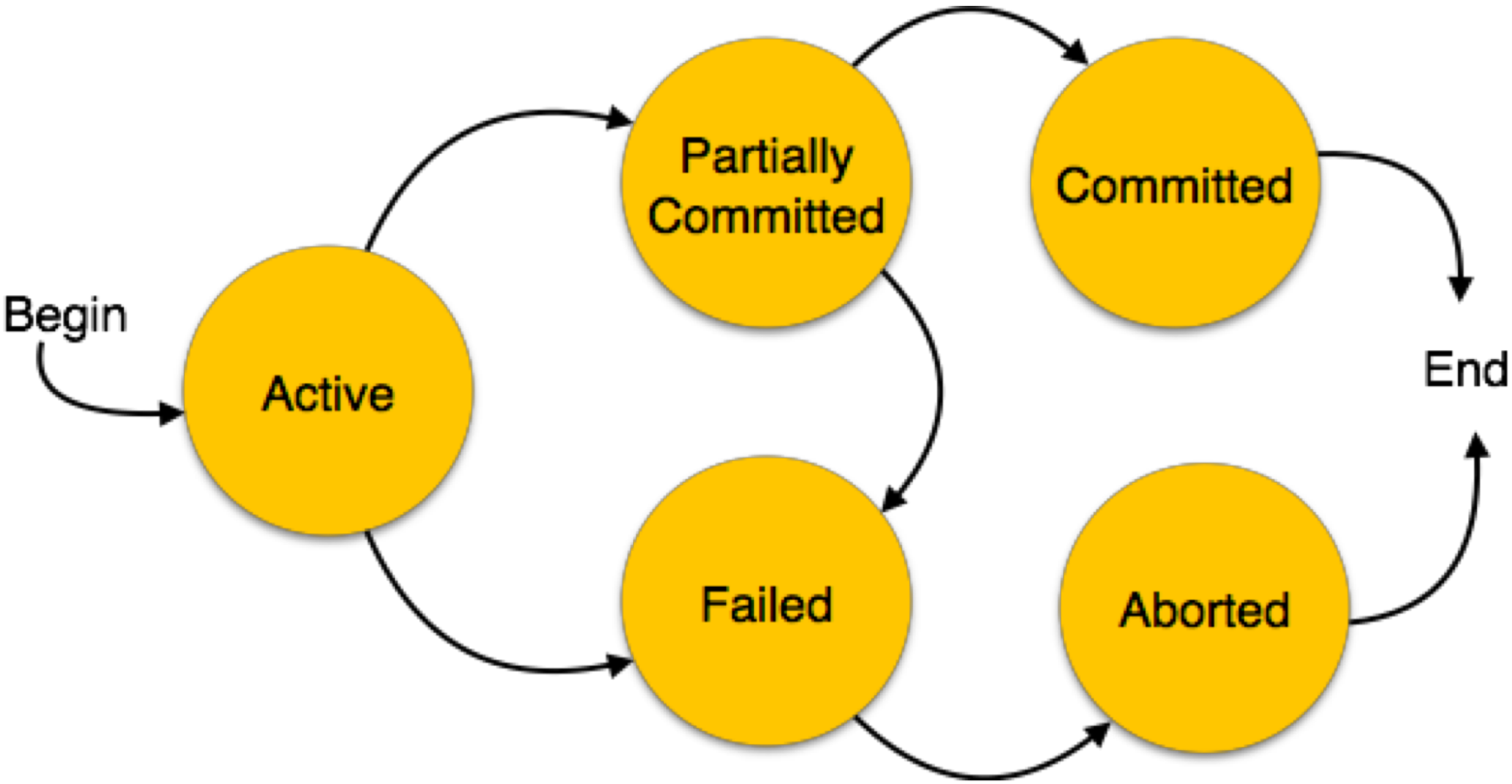
- Protect commits against system failure (e.g. *transaction log* recovery)

Transaction scaling



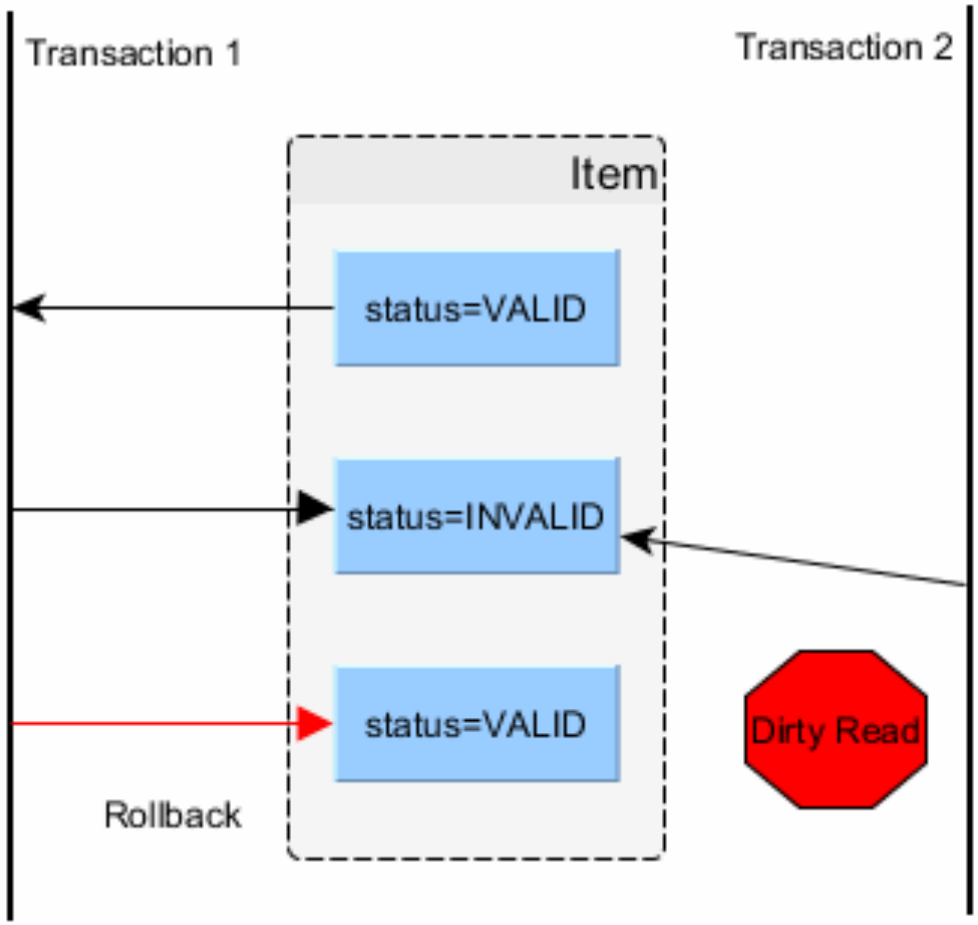
- Rt = Response Time
- Tx = Transactions
- GBs = Size of database

Transaction life-cycle



Concurrency Issues

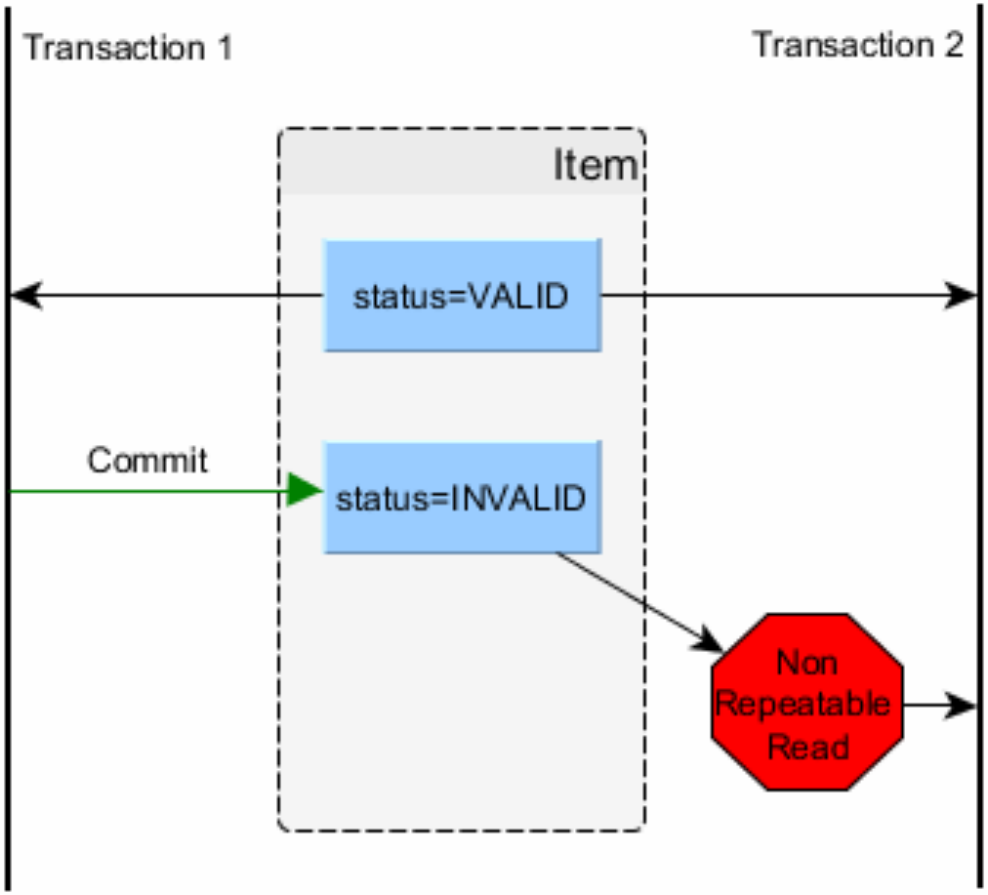
- Interleaving of transactions can lead to anomalies
  - ***Dirty Read***



Transaction accesses uncommitted changes by different transactions

Concurrency Issues

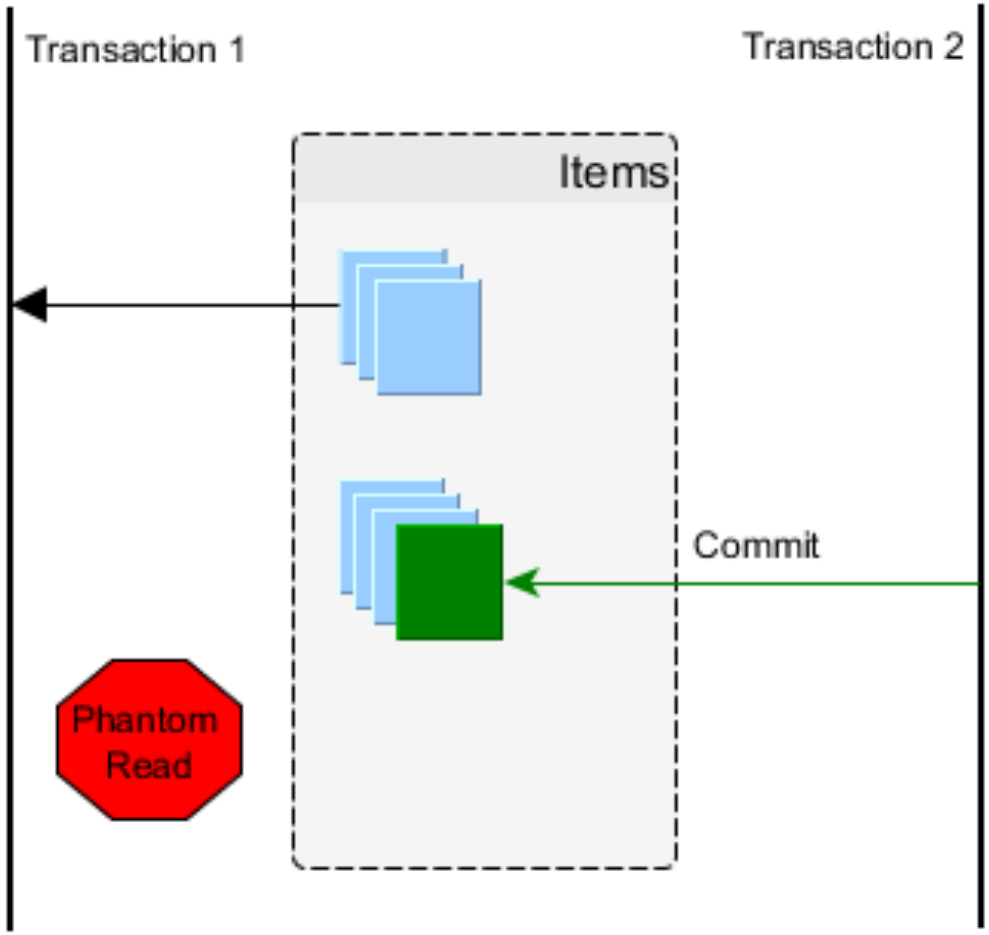
- Interleaving of transactions can lead to anomalies
  - Dirty Read
  - ***Non-repeatable Read***



Same row-read within transaction returns different results

Concurrency Issues

- Interleaving of transactions can lead to anomalies
  - Dirty Read
  - Non-repeatable Read
  - ***Phantom Read***



Same range-read within transaction returns different results

Concurrency Issues

- Interleaving of transactions can lead to anomalies
  - Dirty Read
  - Non-repeatable Read
  - Phantom Read

- Most RDBMs allow control over Isolation
  - Allows balancing between *consistency* and *scalability*

Isolation level	Phenomena		
	Dirty Reads	Non-repeatable Reads	Phantom Reads
Serializible	No	No	No
Repeatable Reads	No	No	Yes
Read Commited	No	Yes	Yes
Read Uncommited	Yes	Yes	Yes

- Enforced by *Concurrency Control*

Scalable & Distributed Systems

- Distributed System
  - Software system
  - Components on networked computers (nodes)
  - Communicate & Coordinate through messages

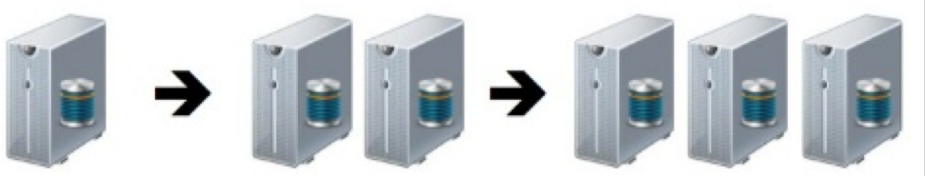
Purpose & content
RDBMD
Con'cy
Scaling
NoSQL

- Scalability
  - Ability to handle growing amount of work
  - Ability to enlarged to accommodate that growth (*elasticity*)
  - Performant
    - Short response-times
    - High throughput
  - Available
    - Uptime / (uptime + downtime)
    - 99.999% often guaranteed (but what does it really mean…)

Scaling

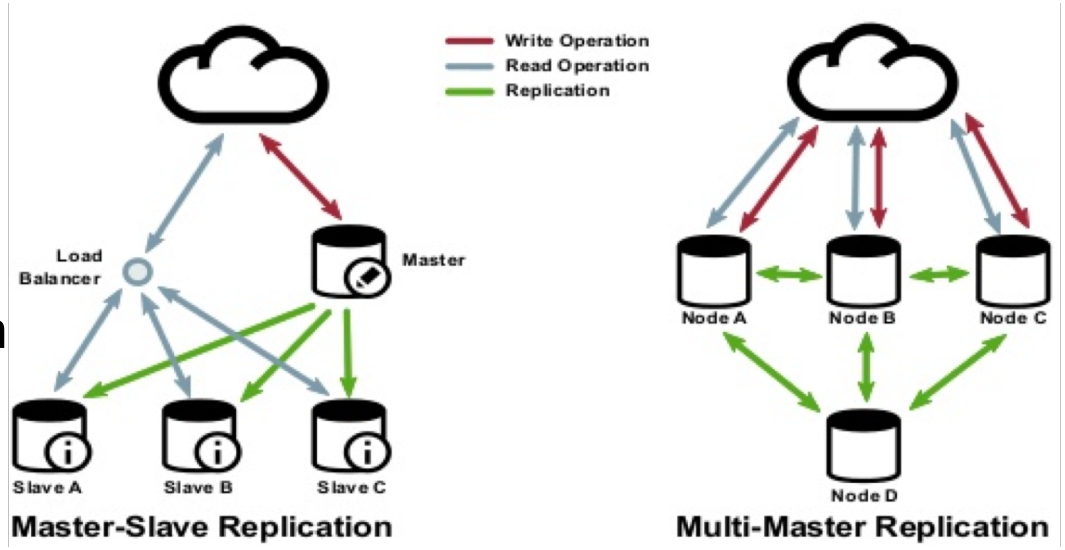
- Vertical Scaling
  - Add resources to a node
  - Increase *capacity*
  - *Load* unaffected
  - System *complexity* unaffected
  - AKA Scale-Up

- Horizontal Scaling
  - Add nodes to a cluster
  - *Capacity* unaffected
  - Decreased *Load*
  - Availability and throughput with increased *complexity*
  - AKA Scale-Out



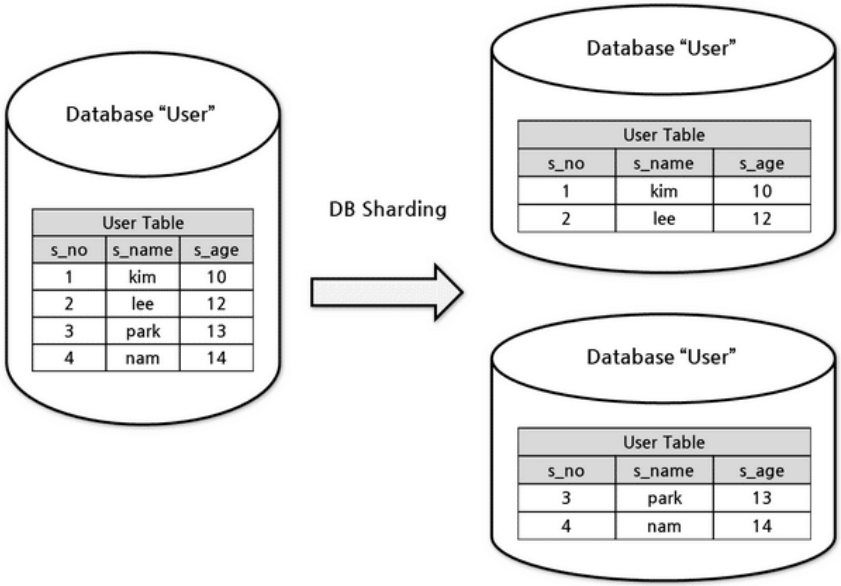
Horizontally Scaling RDBMS

- Replication
  - Auto-backup
  - Master = SPoF + bottleneck
  - Synchronization



Purpose & content
RDBMD
Con'cy
Scaling
NoSQL

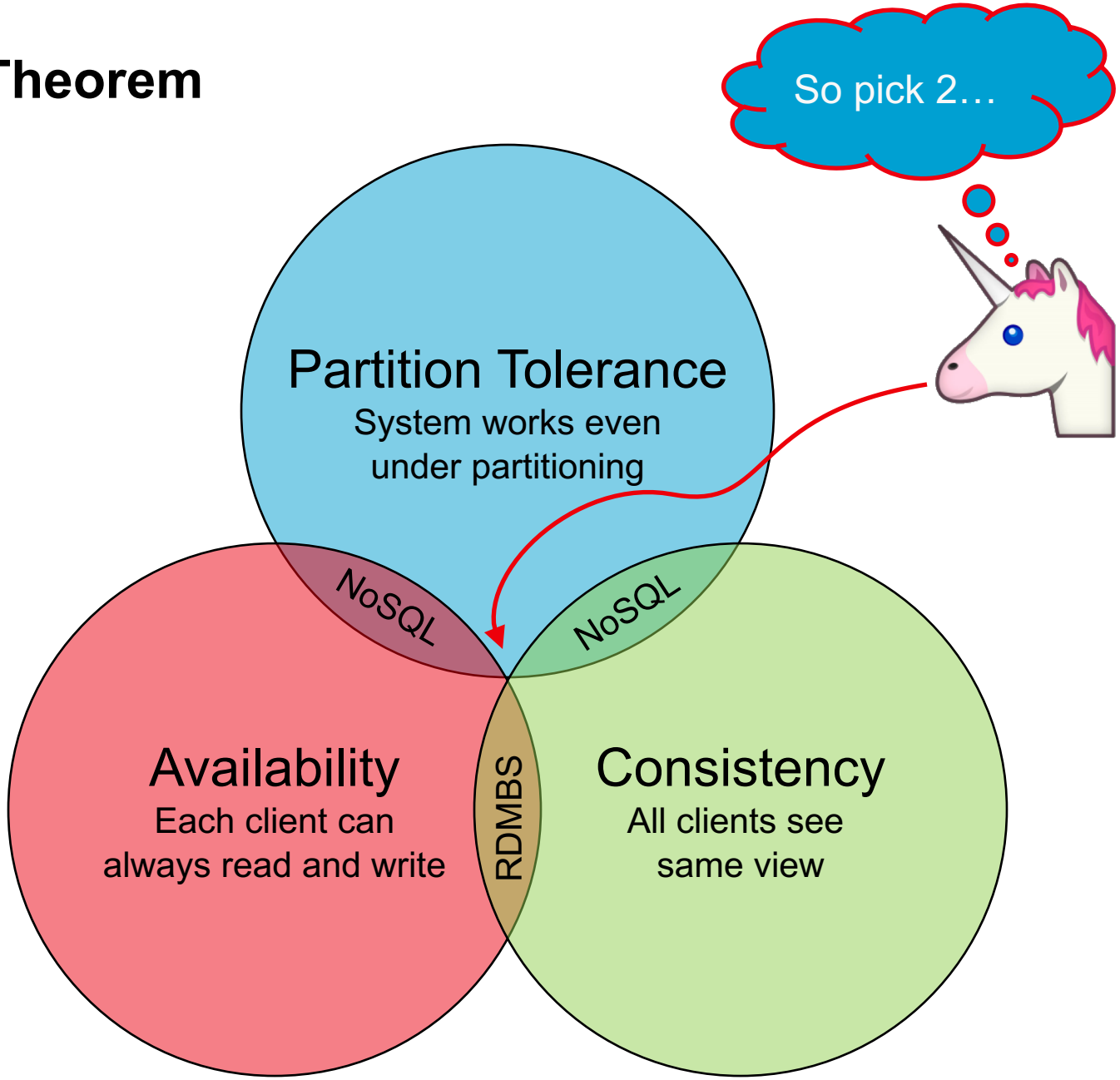
- Sharding
  - Horizontal partitioning
  - Split rows over nodes
  - Cross-shard joins difficult
  - Optimal sharding?
  - Locality of reference
  - Concurrency is complex
  - Requires *coordination*



Conclusion…

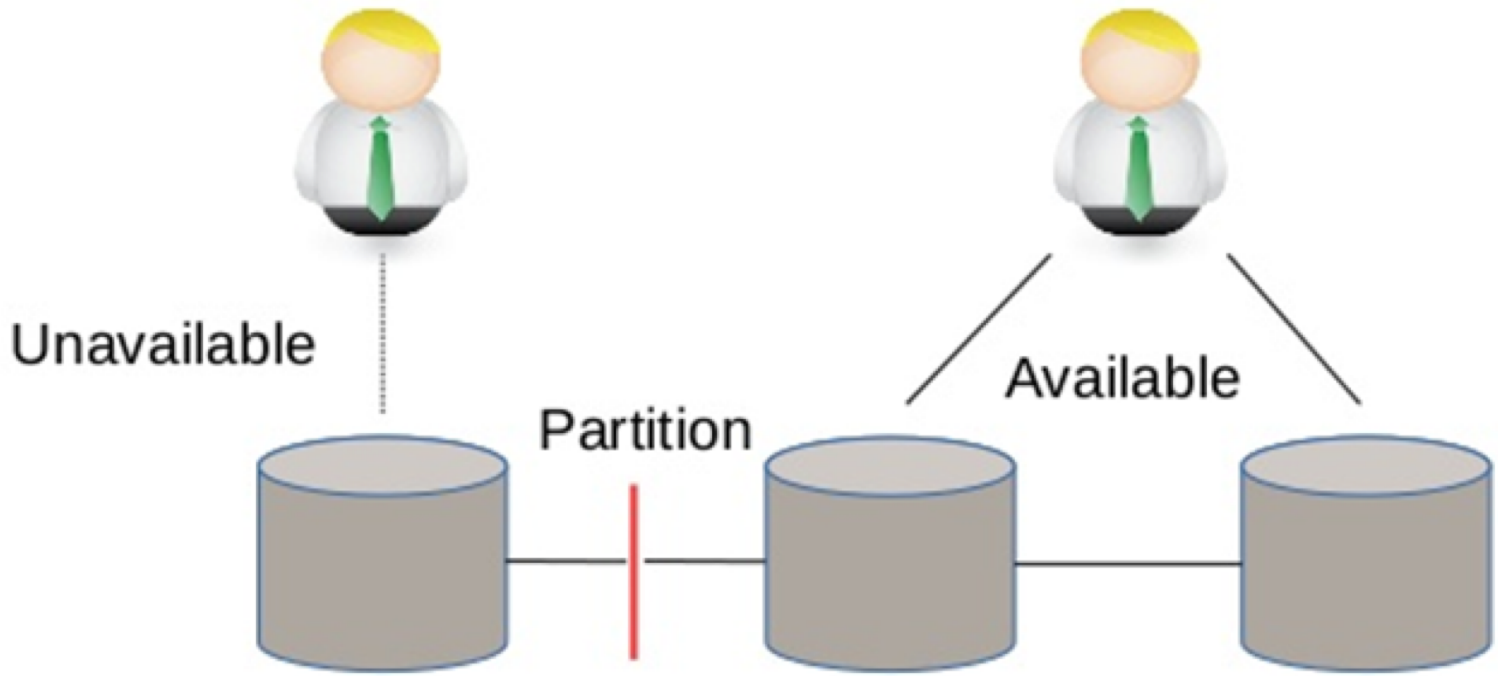
ACID + Distributed = PAIN

CAP Theorem

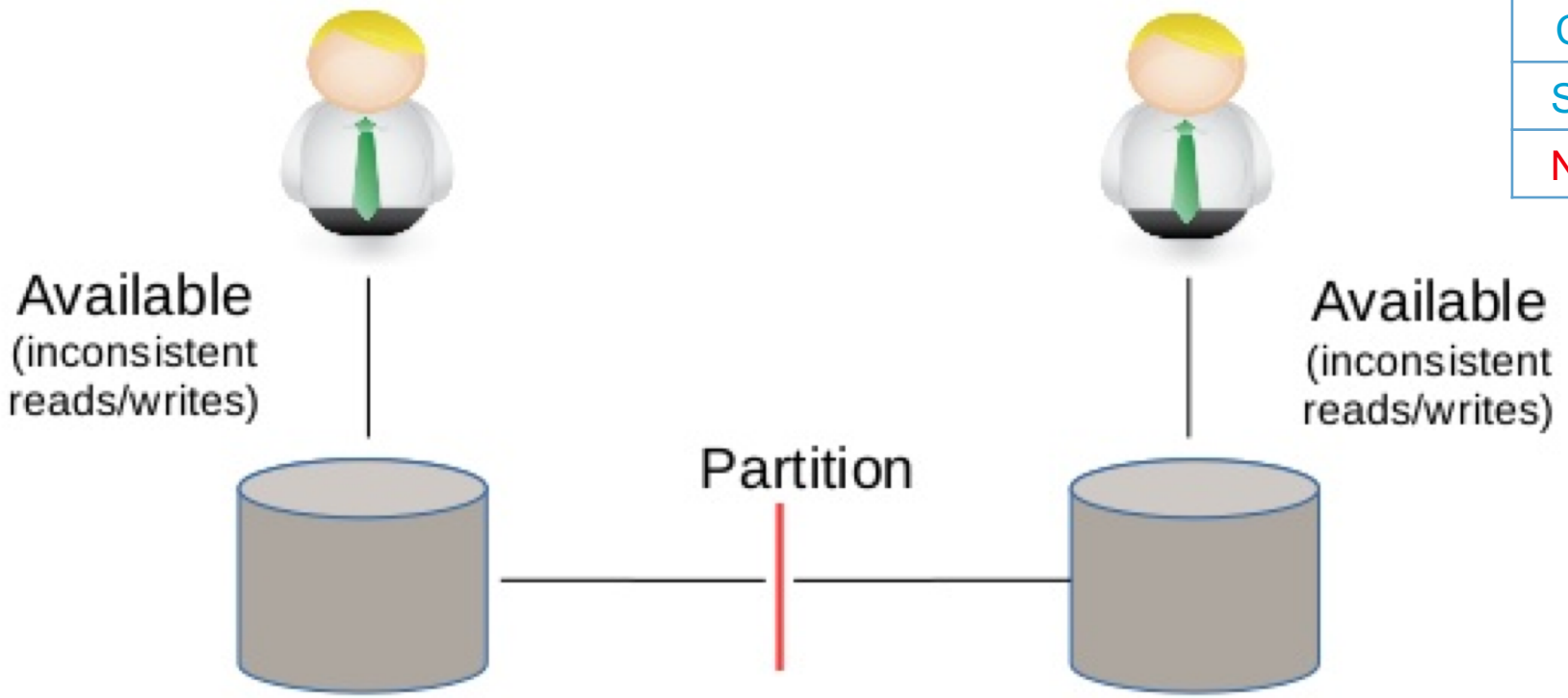


Purpose & content
RDBMD
Con'cy
Scaling
NoSQL

CP Systems



AP Systems


Available
(inconsistent
reads/writes)
Partition
Available
(inconsistent
reads/writes)

NoSQL

- Not Only SQL
  - Not anti-SQL
  - 4 Major categories:
    - Key-Value Stores (e.g. Dynamo, Cassandra, Riak, Redis)
    - Document Stores (e.g. CouchDB, MongoDB, Elasticsearch)
    - Wide Column Stores (e.g. HBase, Cassandra, BigTable)
    - Graph Databases (e.g. Neo4j, OrientDB)
- Relax on
  1. Relational Data-model, e.g.
     - Schema-less (AKA schema-on-read)
     - Unstructured
     - Opaque
  2. ACID properties, instead conform to BASE
     - Basically Available
     - Soft State
     - Eventually Consistent

Most AP-systems

Updating distributed data through Quorum Voting

- AP systems often allow consistency levels
  - N = # of nodes with replica
  - W = minimal # of replicas that must ACK the update
  - R = minimal # of replicas that must ACK successful read

Quora

  - W + R > N → Strong consistency
    - W=N, R=1 → Optimized for reads
    - W=1, R=N → Optimized for writes
  - W + R <= N → Weak consistency

Key-Value Store example - Redis

- In-memory
- Optional Durability
- List, Set and hash operations

```
> SET newkey "the redis string begins"
OK
> GET newkey
"the redis string begins"
```

Purpose & content
RDBMD
Con'cy
Scaling
NoSQL

```
> hmset user:1 username jsmith password 4bAc0s email jsmith@gmail.com
OK
> hgetall user:1
1) "username"
2) "jsmith"
3) "password"
4) "4bAc0s"
5) "email"
6) "jsmith@gmail.com"
```

Key	Value
K1	AAA,BBB,CCC
K2	AAA,BBB
K3	AAA,DDD
K4	AAA,2,01/01/2015
K5	3,ZZZ,5623

Wide Column Store example - HBase

- Stores by column
- Tightly coupled to HDFS
- Time-stamped cells

Purpose & content
RDBMD
Con'cy
Scaling
NoSQL

COLUMN FAMILIES

Row key	personal data		professional data	
empid	name	city	designation	salary
1	raju	hyderabad	manager	50,000
2	ravi	chennai	sr.engineer	30,000
3	rajesh	delhi	jr.engineer	25,000

```
hbase(main):012:0> get 'emp', '1'
   COLUMN                          CELL
personal:city timestamp = 1417521848375, value = hyderabad
personal:name timestamp = 1417521785385, value = ramu
professional:designation timestamp = 1417521885277, value = manager
professional:salary timestamp = 1417521903862, value = 50000

4 row(s) in 0.0270 seconds
```

Document Store example - MongoDB

- Stores BSON (Binary JSON)
- Circumvents Object-Relation Impedance Mismatch

Purpose & content
RDBMD
Con'cy
Scaling
NoSQL

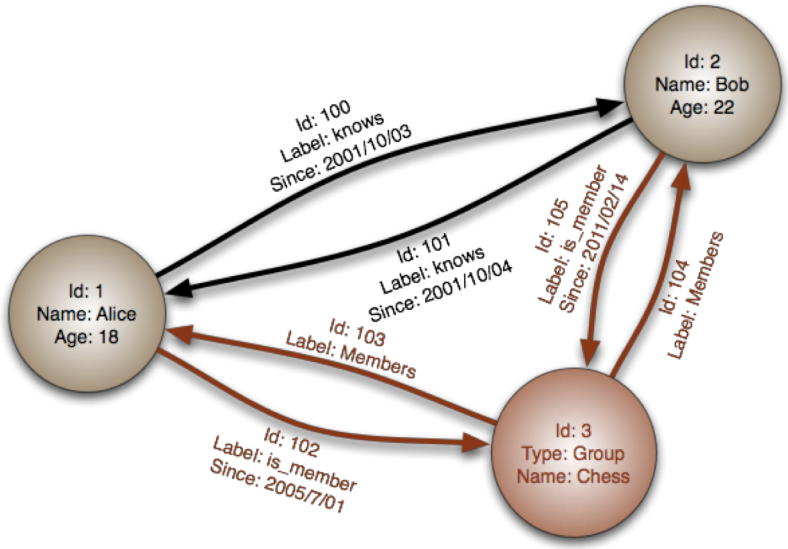
```
mongo> use myDatabase
Mongo> db.books.find(
  {
     title: 'MongoDB Overview',
     likes: 100
  }
)
```

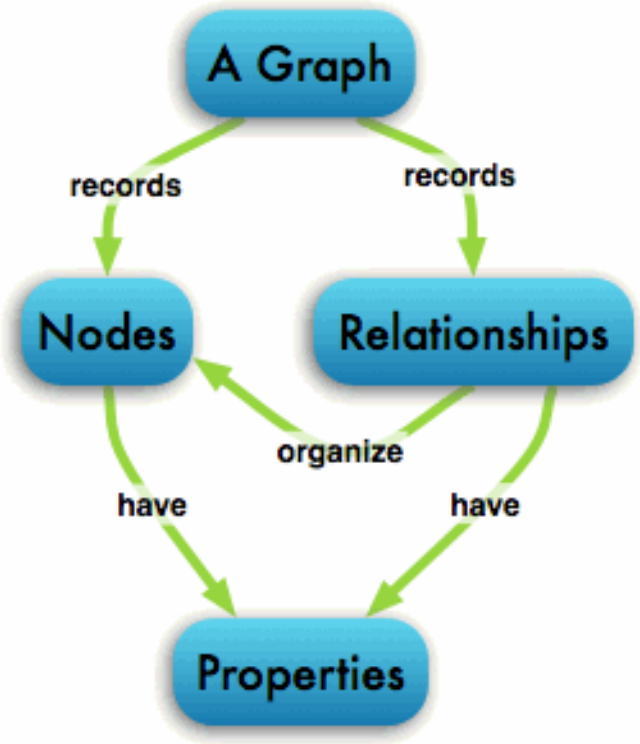
```
_id: ObjectId(7df78ad8902c)
title: 'MongoDB Overview',
description: 'MongoDB is no sql database',
by: 'tutorials point',
url: 'http://www.tutorialspoint.com',
tags: ['mongodb', 'database', 'NoSQL'],
likes: 100,
comments: [
   {
      user:'user1',
      message: 'My first comment',
      dateCreated: new Date(2011,1,20,2,15),
      like: 0
   },
   {
      user:'user2',
      message: 'My second comments',
      dateCreated: new Date(2011,1,25,7,45),
      like: 5
   }
]
}
```

Graph Database example - Neo4j

- Very generic model (graphs)
- *Cypher* query language

Purpose & content
RDBMD
Con'cy
Scaling
NoSQL






```
MATCH (tom:Person {name:"Tom Hanks"})-[:ACTED_IN]->(m)<-[:ACTED_IN]-(coActors),
      (coActors)-[:ACTED_IN]->(m2)<-[:ACTED_IN]-(cocoActors)
WHERE NOT (tom)-[:ACTED_IN]->(m2)
RETURN cocoActors.name AS Recommended, count(*) AS Strength ORDER BY Strength DESC
```