TRANSACTIONS!

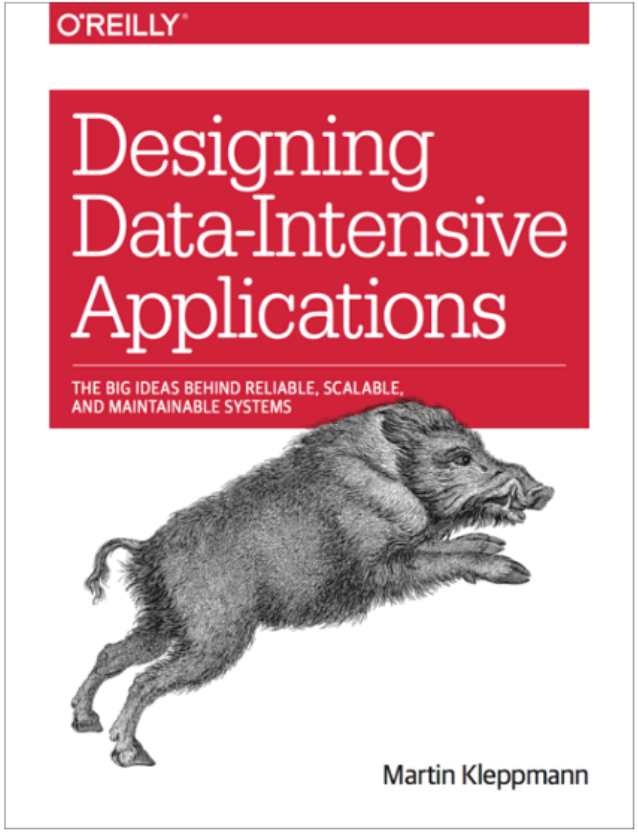
MYTHS, SURPRISES & OPPORTUNITIES

Martin Kleppmann @martinkl

TRANSACTIONS!

JOYS, CHALLENGES & MISERY

"Consistency guarantees? Haha!"
(cue manic laughter)



dataintensive.net

@martinkl

1975 IBM System R

1975 IBM System R

2008? NoSQL

2012? "NewSQL"

1975 IBM System R

(Chamberlin et al, 1981)

2008? No ~~SQL~~ Transactions

2012? "New ~~SQL~~" Transactions

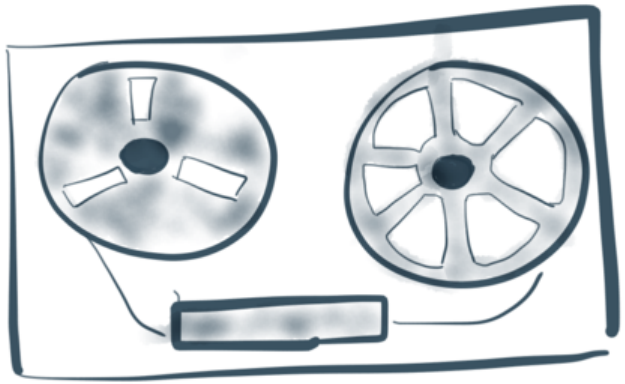

ACID
(Härder & Reuter, 1983)

"more mnemonic than precise"
(Brewer, 2012)

(Härder & Reuter, 1983)

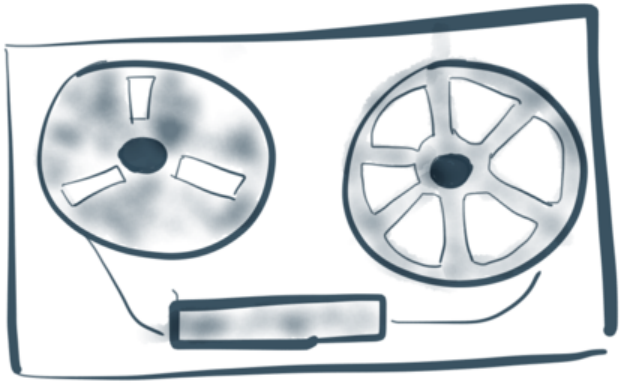

ACID
Durability

Durability

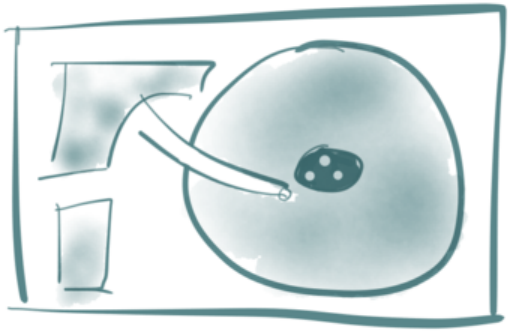


Archive tape

Durability

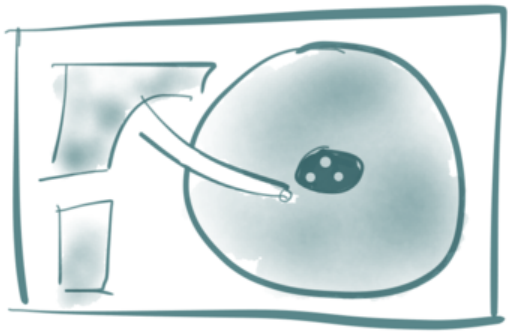


Archive tape



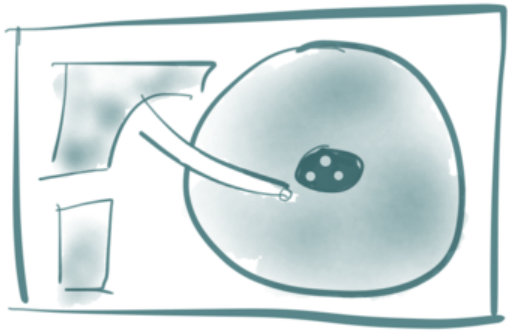
fsync to disk

Durability

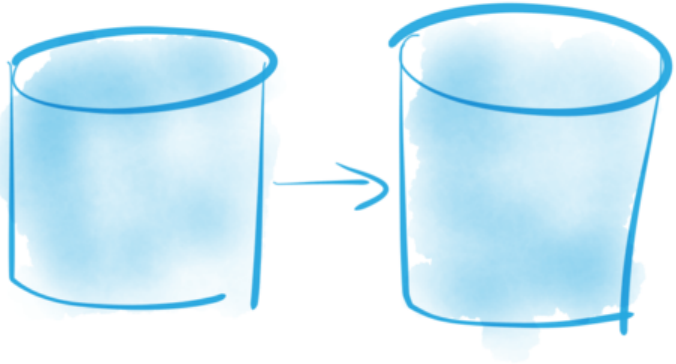


fsync to disk

Durability



fsync to disk



Replication

Consistency

Consistency

≠ C in CAP Theorem

(Brewer, 2012)

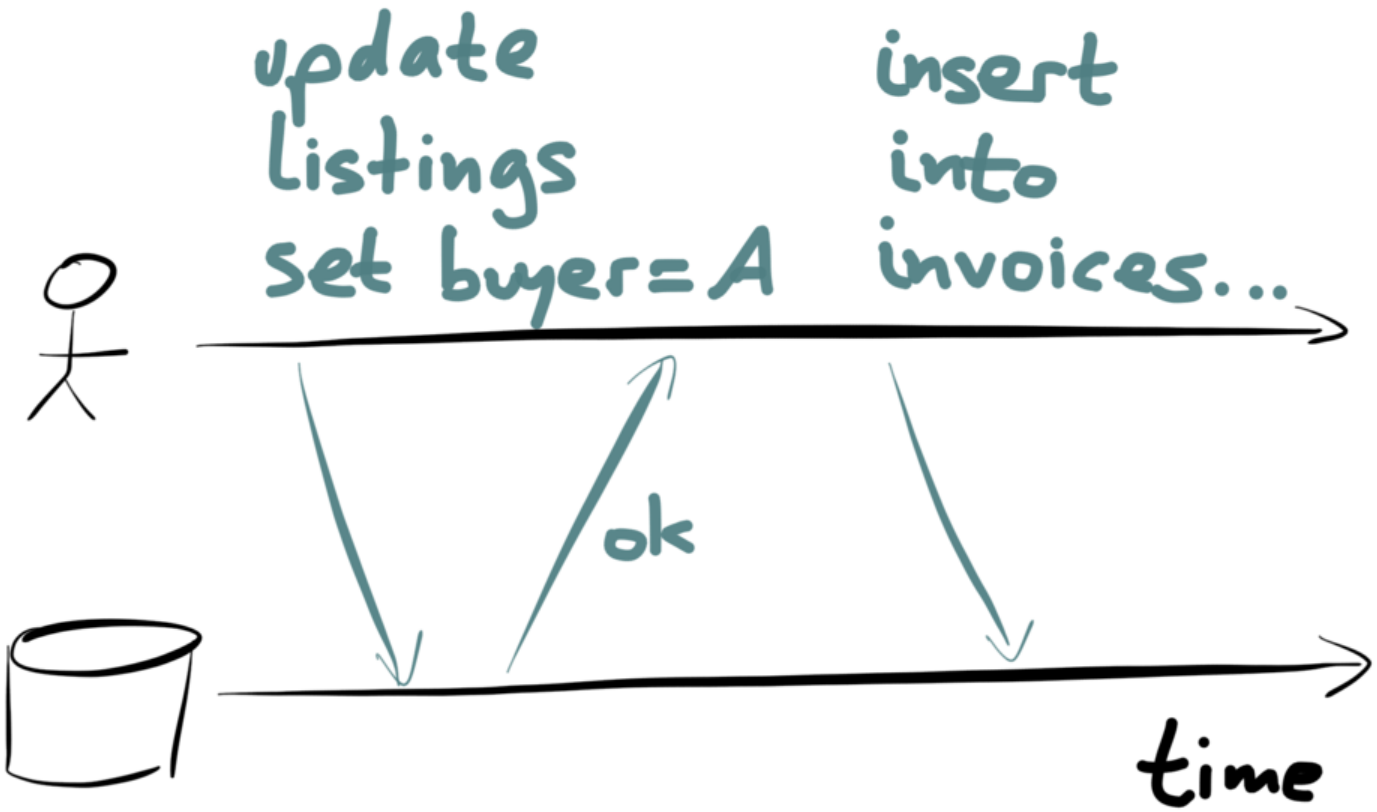
Consistency

"tossed in to make the acronym work" (Hellerstein)

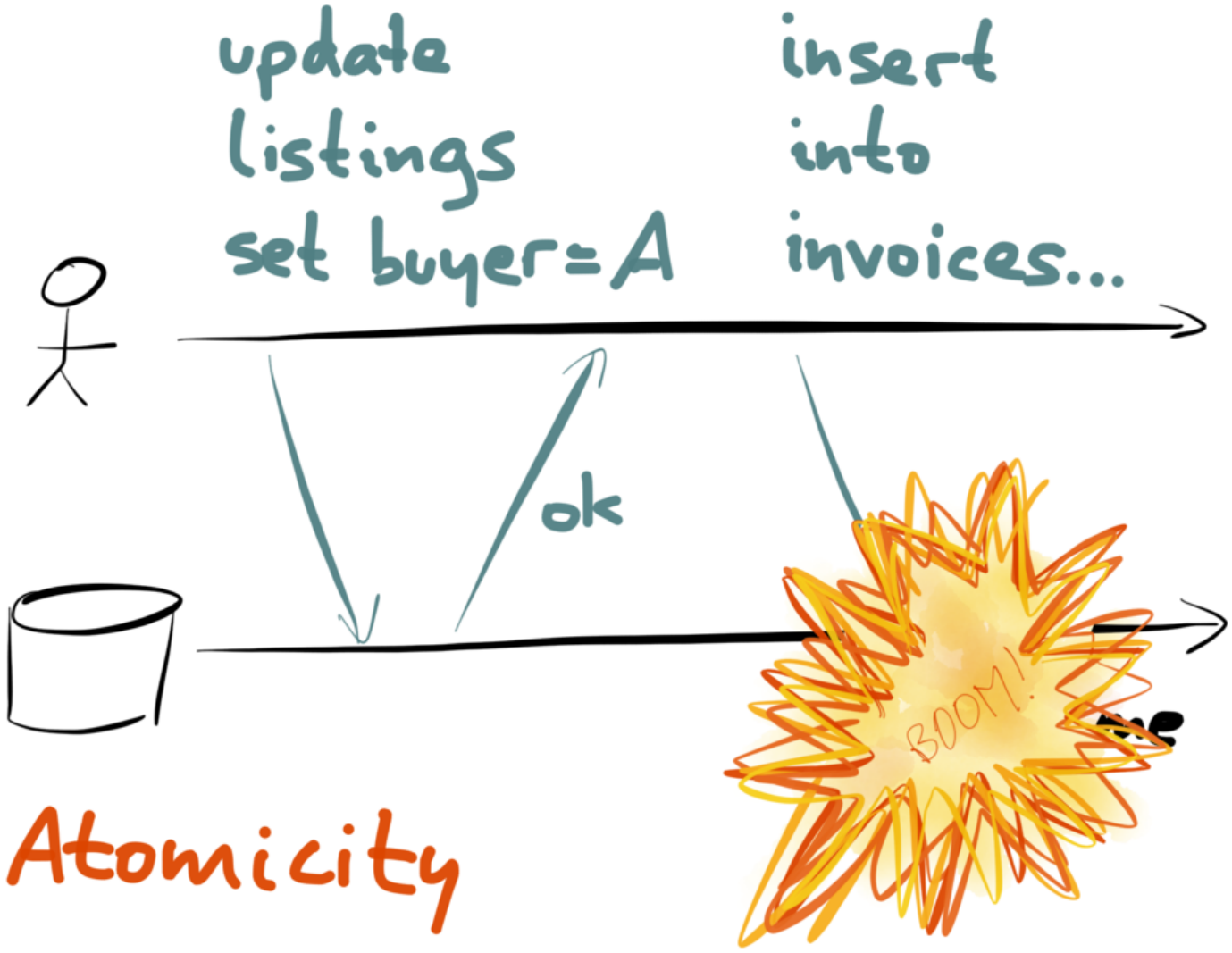
Atomicity

NOT ABOUT CONCURRENCY!


ACID
Handling faults (crashes)



Atomicity



Atomicity

Transactions = multi-object atomicity

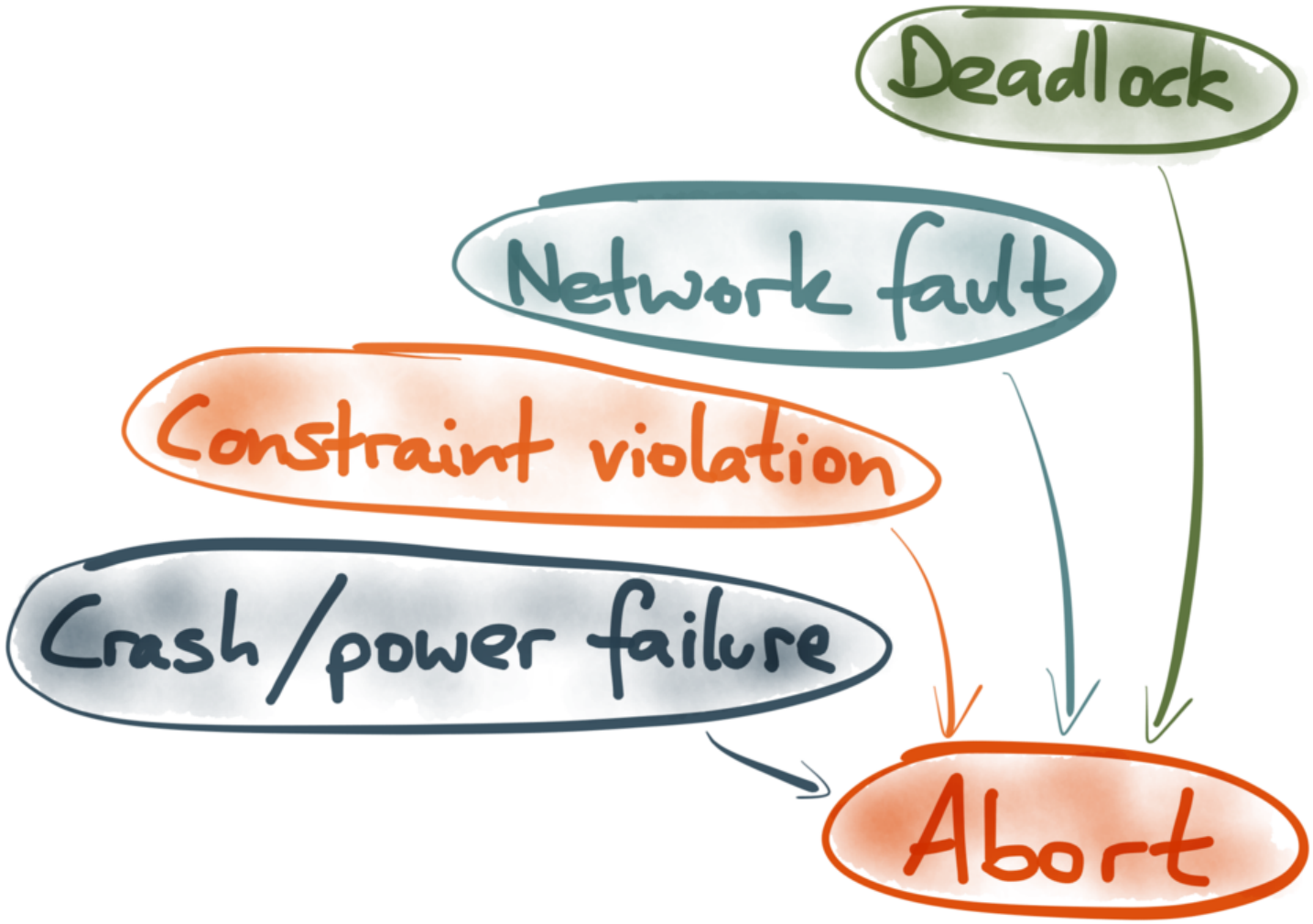
Roll back writes on abort

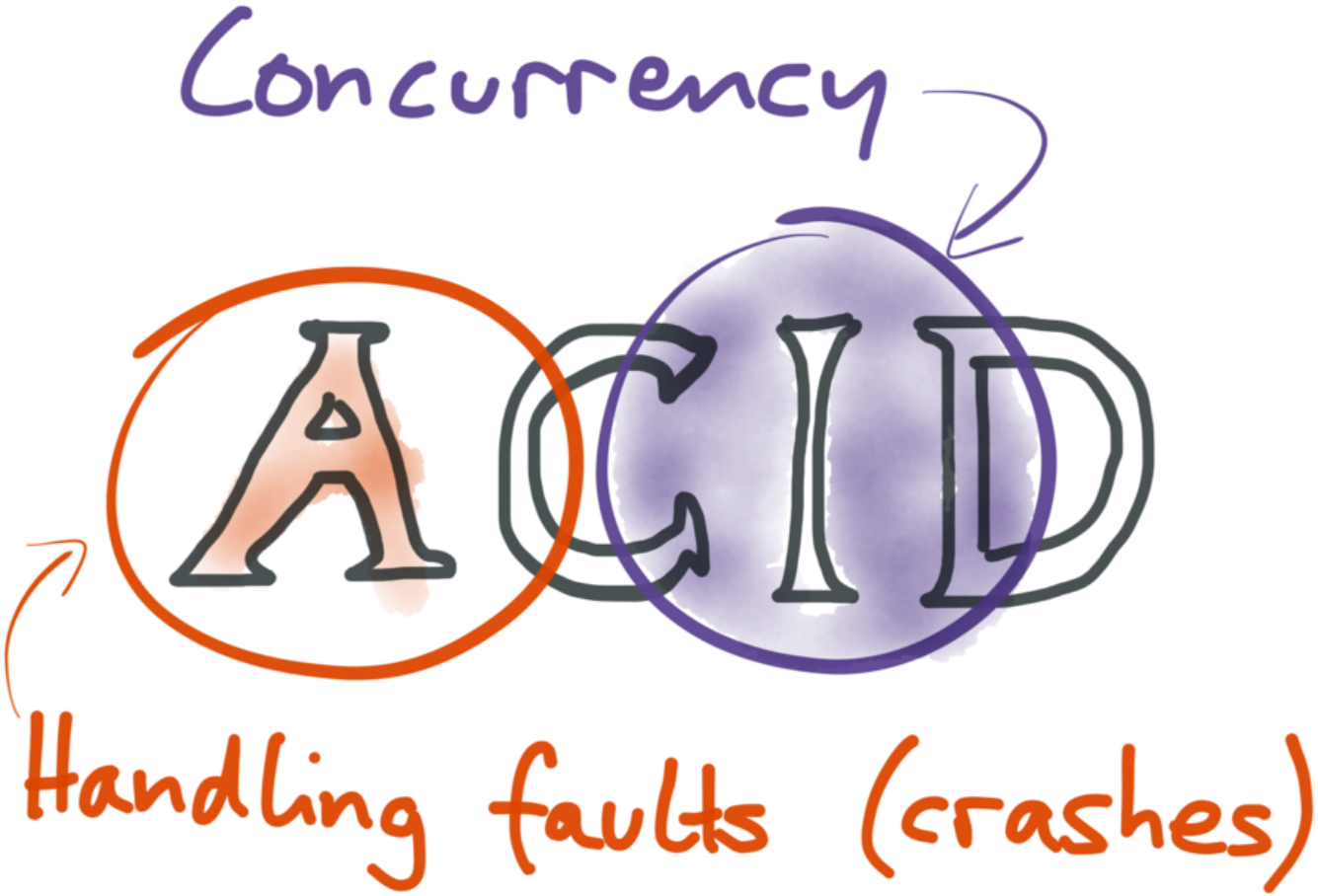
Transactions =

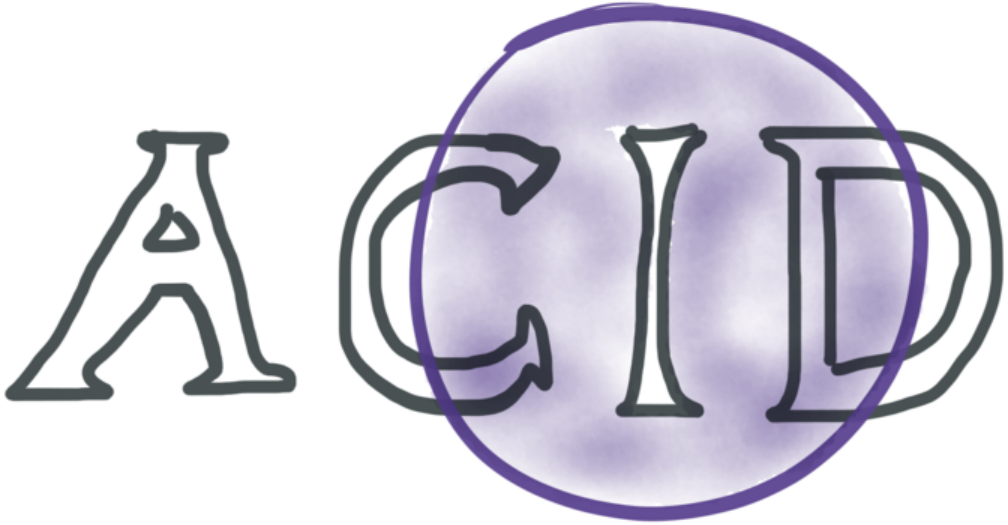
multi-object atomicity

Roll back writes on abort

Abortability

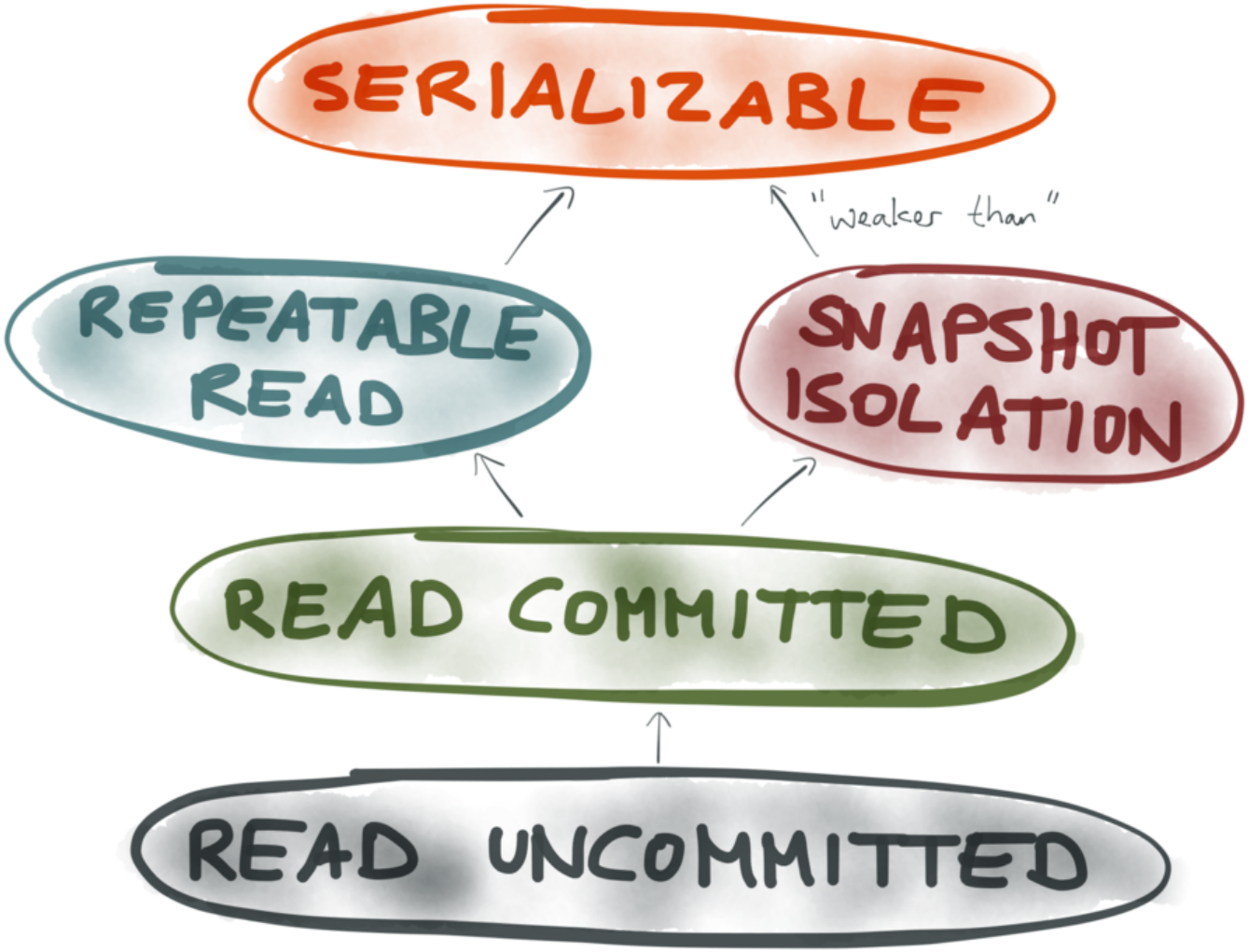

Deadlock
Network fault
Constraint violation
Crash/power failure
Abort


Concurrency
ACID
Handling faults (crashes)



Isolation

SERIALIZABLE ?


SERIALIZABLE
"weaker than"
REPEATABLE READ
SNAPSHOT ISOLATION
READ COMMITTED
READ UNCOMMITTED

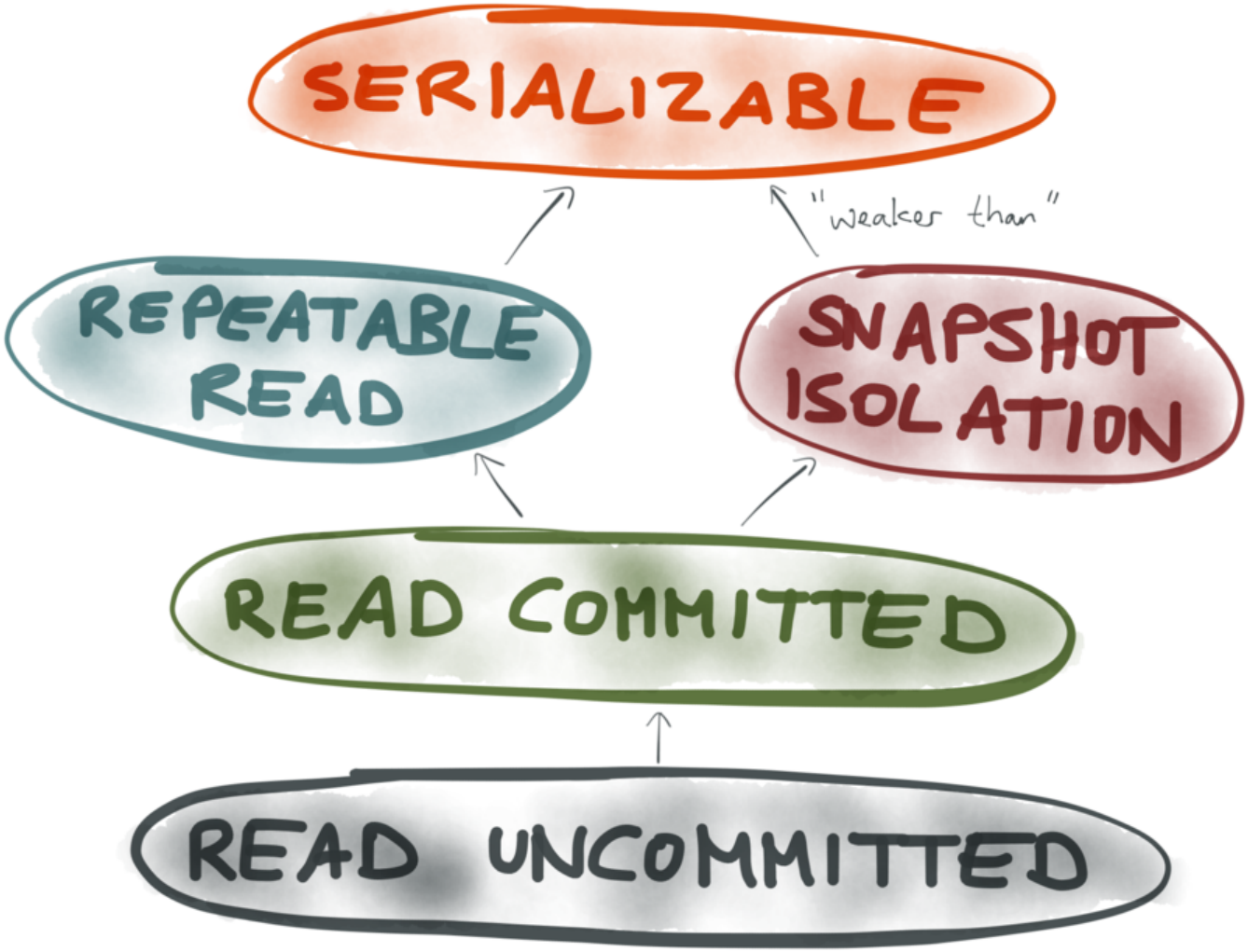
Database	Default	Maximum
Actian Ingres 10.0/10S [1]	S	S
Aerospike [2]	RC	RC
Akiban Persistit [3]	SI	SI
Clustrix CLX 4100 [4]	RR	RR
Greenplum 4.1 [8]	RC	S
IBM DB2 10 for z/OS [5]	CS	S
IBM Informix 11.50 [9]	Depends	S
MySQL 5.6 [12]	RR	S
MemSQL 1b [10]	RC	RC
MS SQL Server 2012 [11]	RC	S
NuoDB [13]	CR	CR
Oracle 11g [14]	RC	SI
Oracle Berkeley DB [7]	S	S
Oracle Berkeley DB JE [6]	RR	S
Postgres 9.2.2 [15]	RC	S
SAP HANA [16]	RC	SI
ScaleDB 1.02 [17]	RC	RC
VoltDB [18]	S	S
RC: read committed, RR: repeatable read, SI: snapshot isolation, S: serializability, CS: cursor stability, CR: consistent read		

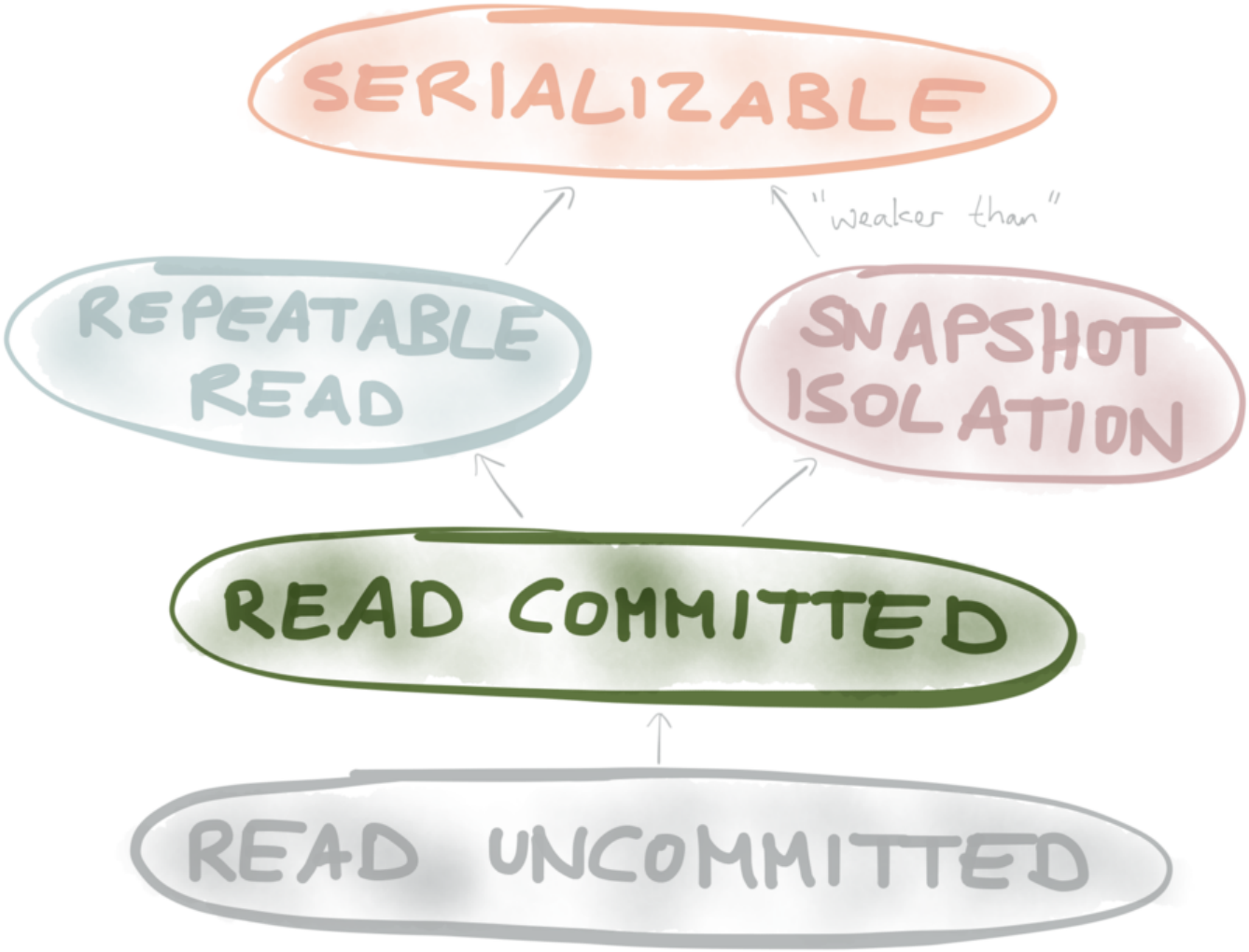
Table 1: Default and maximum isolation levels for ACID and NewSQL databases as of January 2013.

Bailis et al, 2013

Actian Ingres 10.0/10S [1]	S	S
Aerospike [2]	RC	RC
Akiban Persistit [3]	SI	SI
Clustrix CLX 4100 [4]	RR	RR
Greenplum 4.1 [8]	RC	S
IBM DB2 10 for z/OS [5]	CS	S
IBM Informix 11.50 [9]	Depends	S
MySQL 5.6 [12]	RR	S
MemSQL 1b [10]	RC	RC
MS SQL Server 2012 [11]	RC	S
NuoDB [13]	CR	CR
Oracle 11g [14]	RC	SI

Bailis et al, 2013


SERIALIZABLE
"weaker than"
REPEATABLE READ
SNAPSHOT ISOLATION
READ COMMITTED
READ UNCOMMITTED

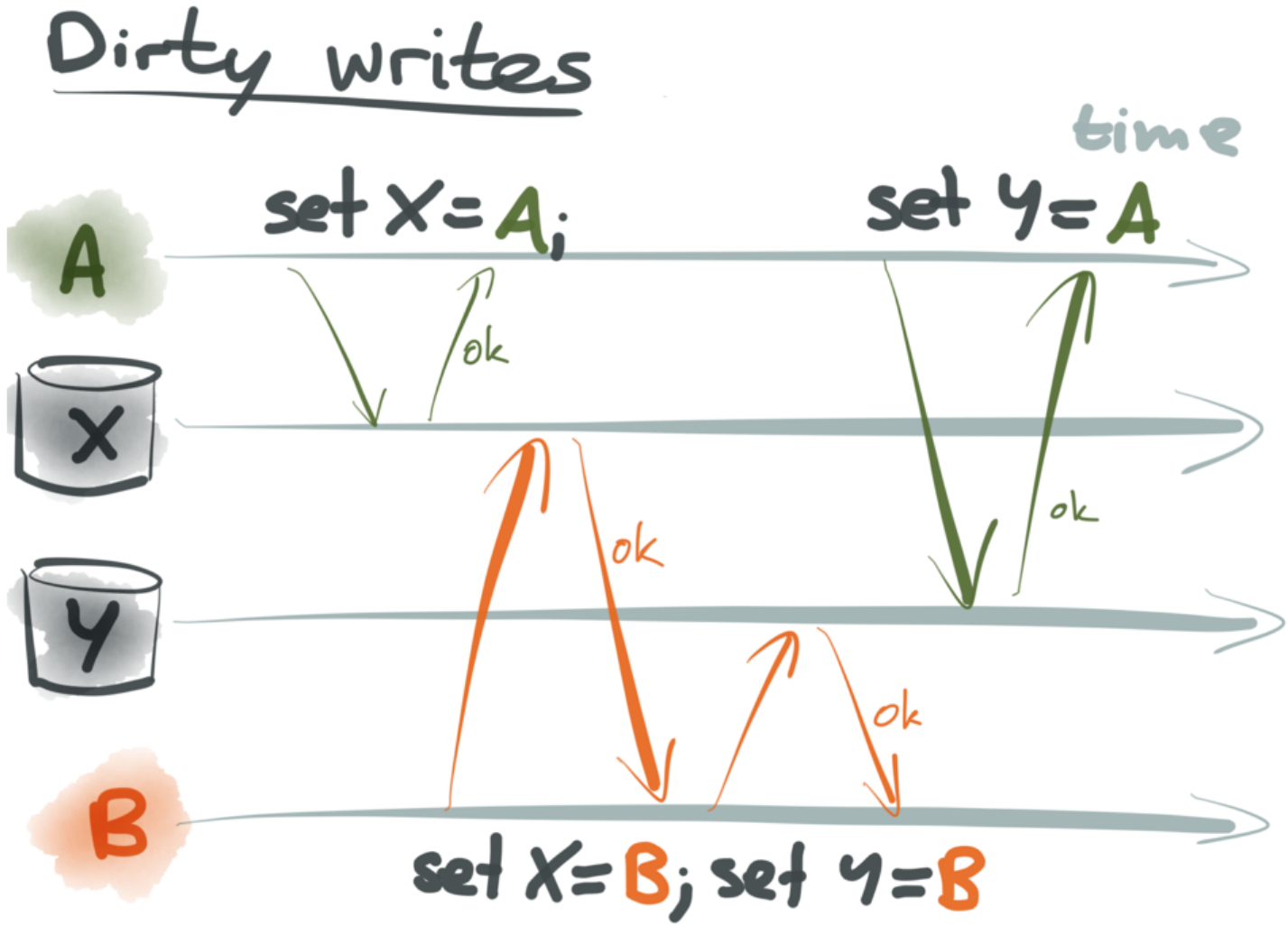

SERIALIZABLE
"weaker than"
REPEATABLE READ
SNAPSHOT ISOLATION
READ COMMITTED
READ UNCOMMITTED

READ COMMITTED

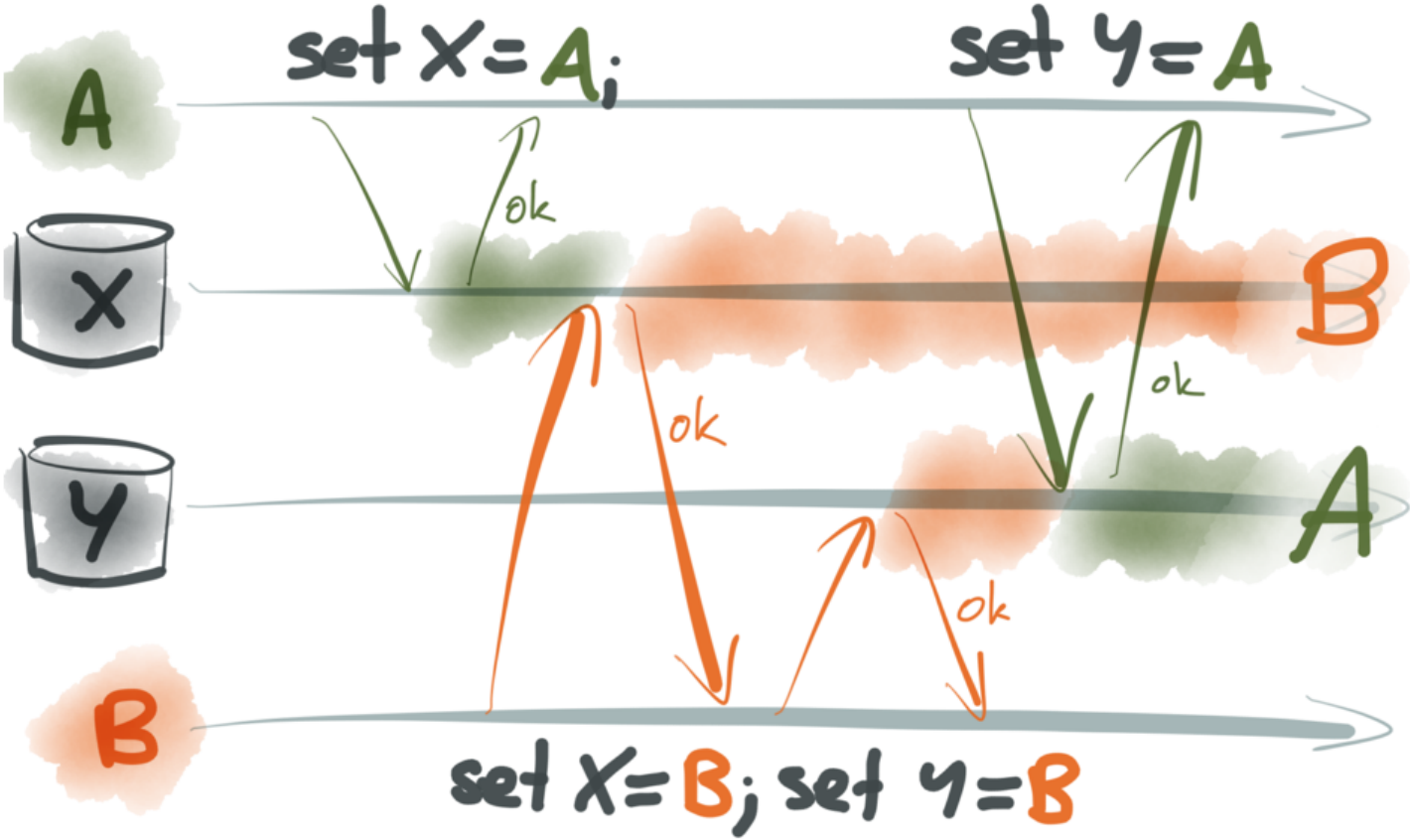
(Default in Postgres, Oracle, SQL Server)

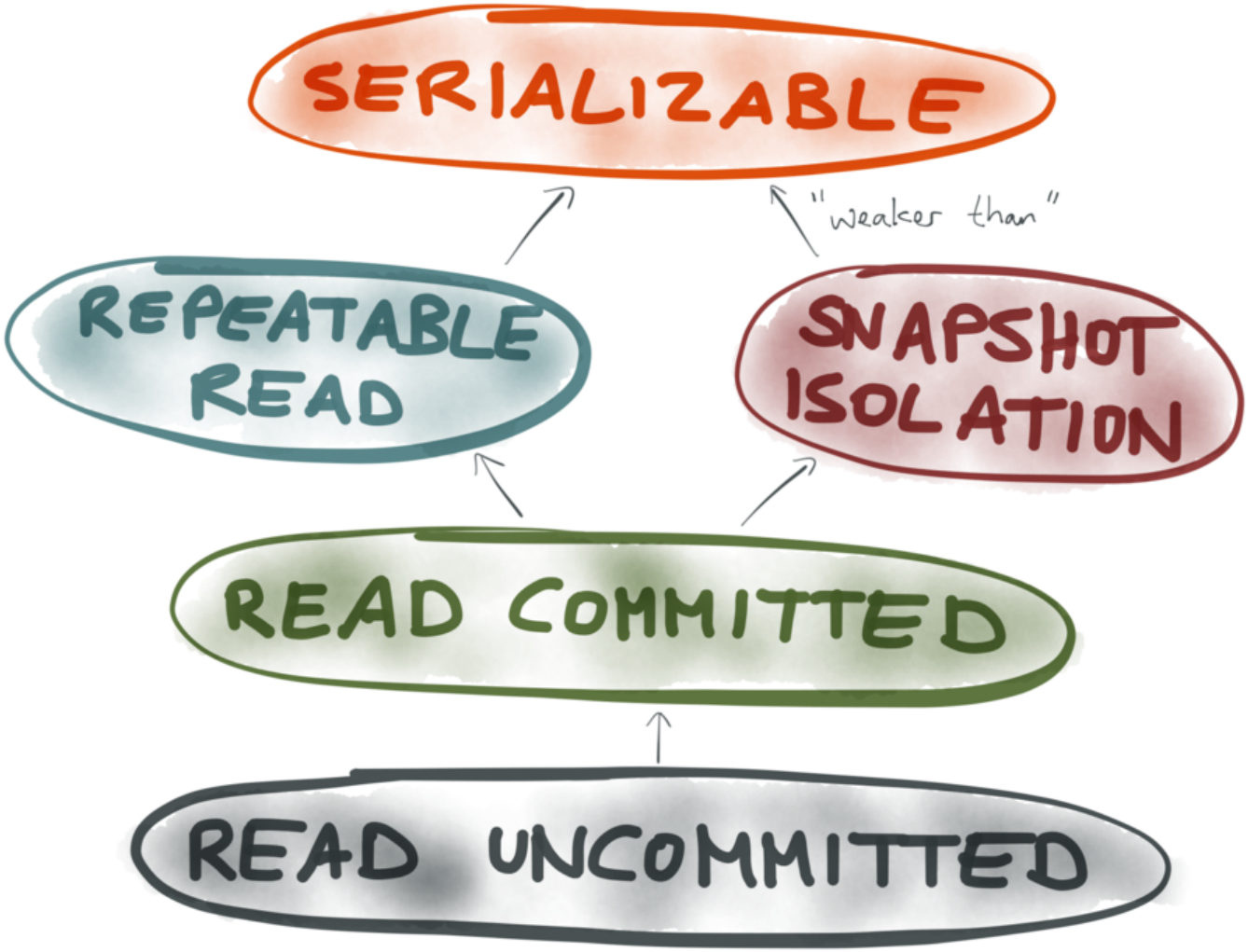
~~Dirty reads~~

~~Dirty writes~~


Dirty writes
time
set X=A;
set Y=A
A
X
Y
B
ok
ok
ok
ok
set X=B; set Y=B

Dirty writes (prevented by READ COMMITTED)

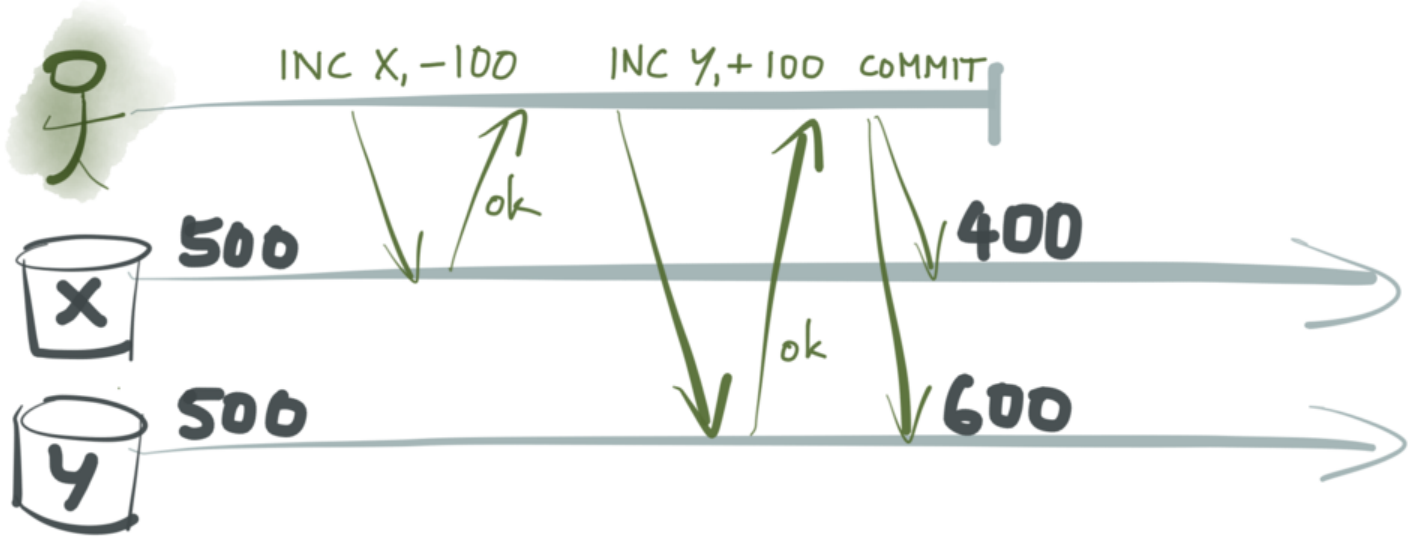



SERIALIZABLE
"weaker than"
REPEATABLE READ
SNAPSHOT ISOLATION
READ COMMITTED
READ UNCOMMITTED

Read skew

(can occur under READ COMMITTED)

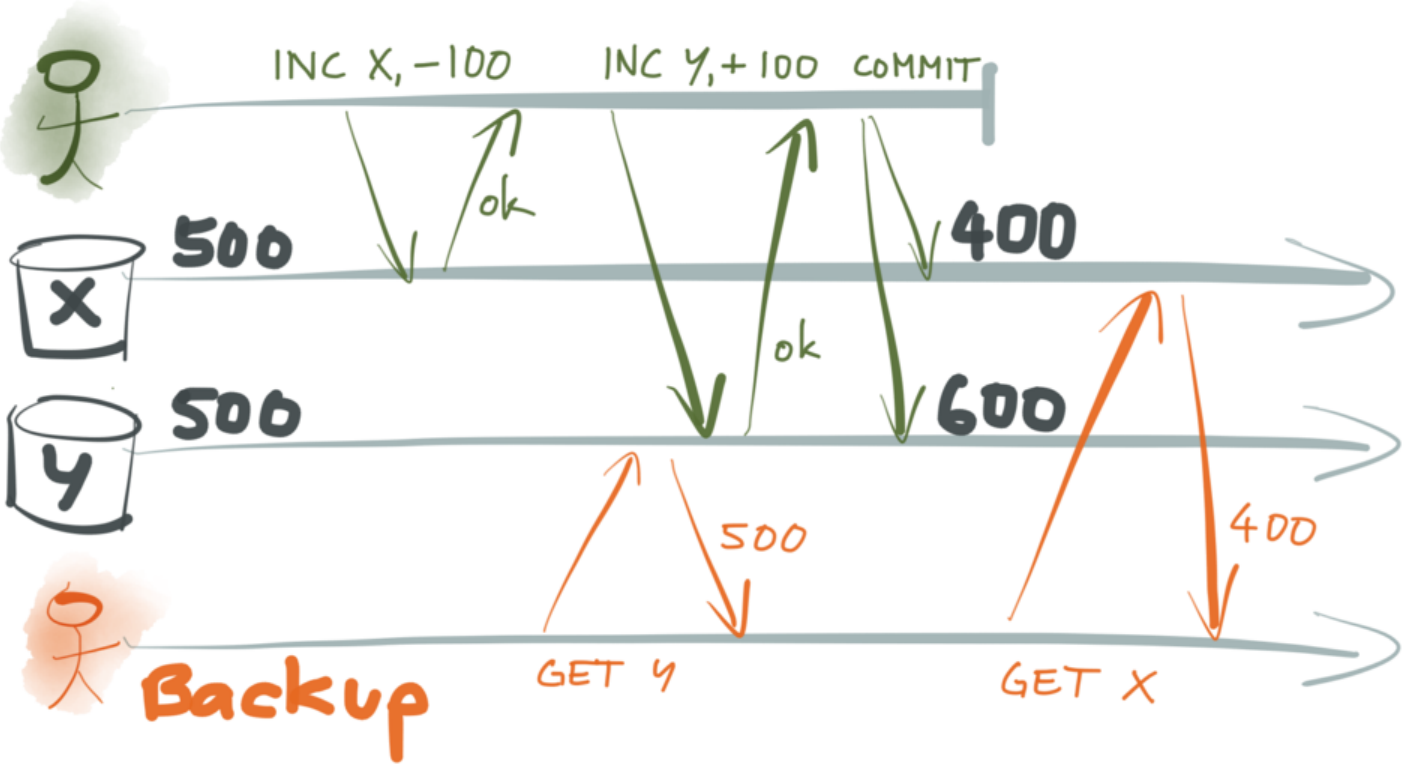
Transfer $100 from X to Y.



Read skew

(can occur under READ COMMITTED)

Transfer $100 from X to Y.

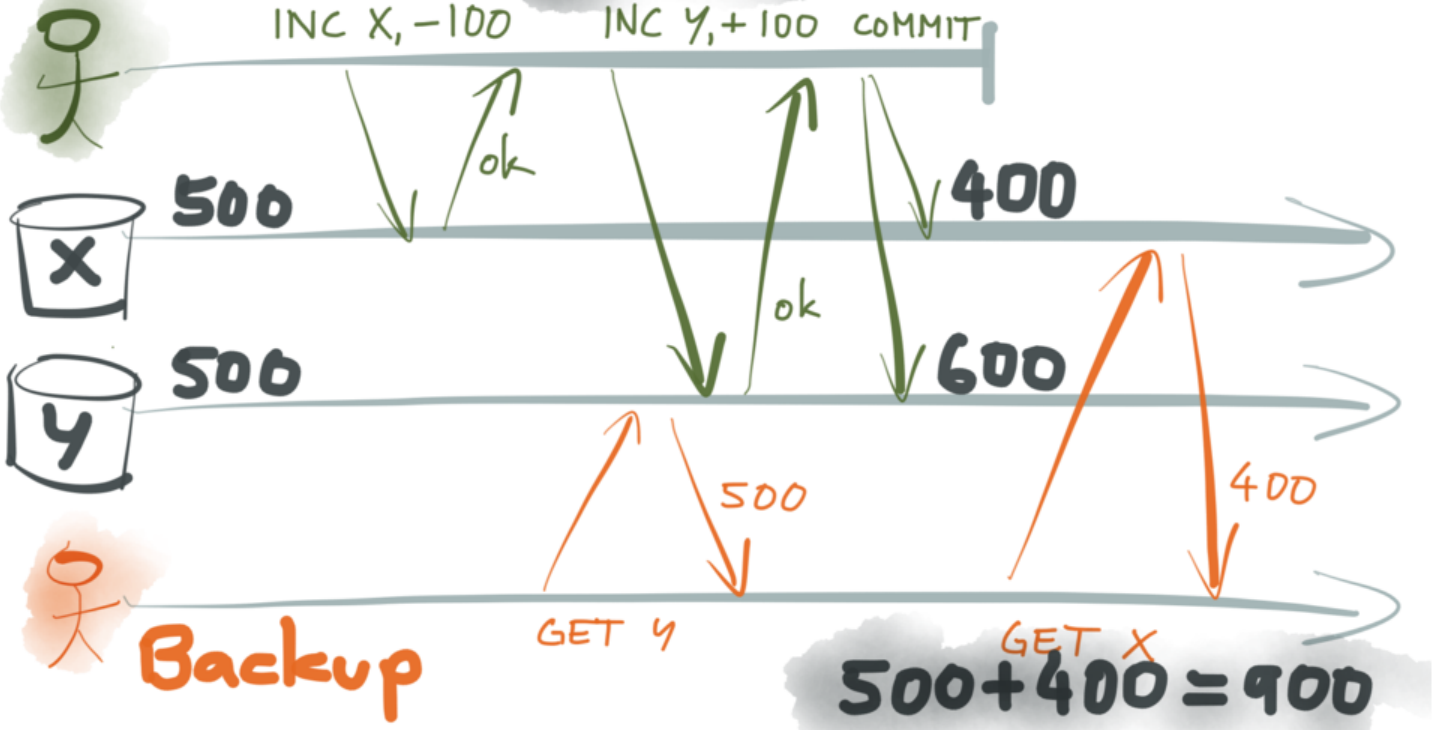


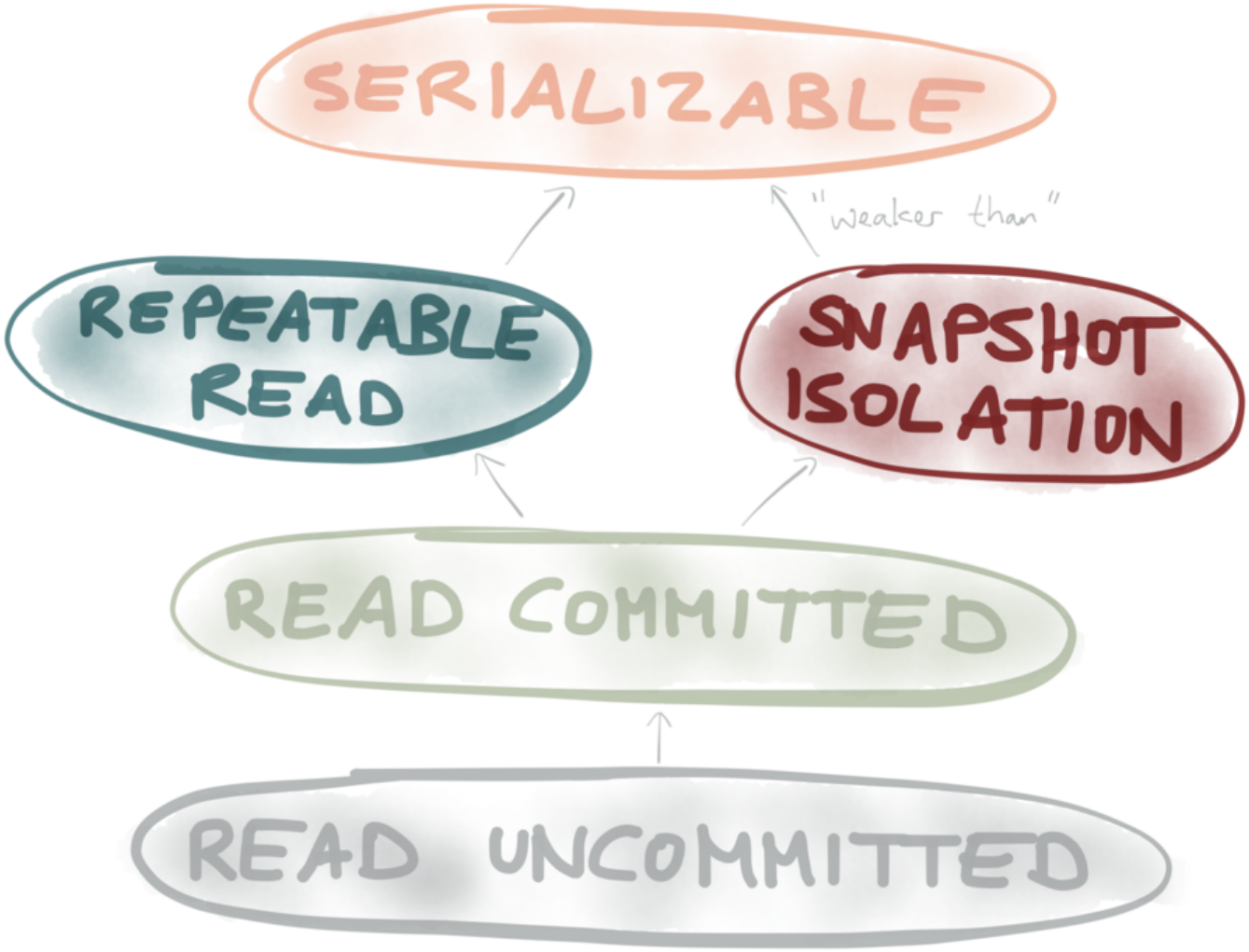
Read skew

(can occur under READ COMMITTED, prevented by SNAPSHOT ISOLATION)

Transfer $100 from X to Y.

500 + 500 = 1000

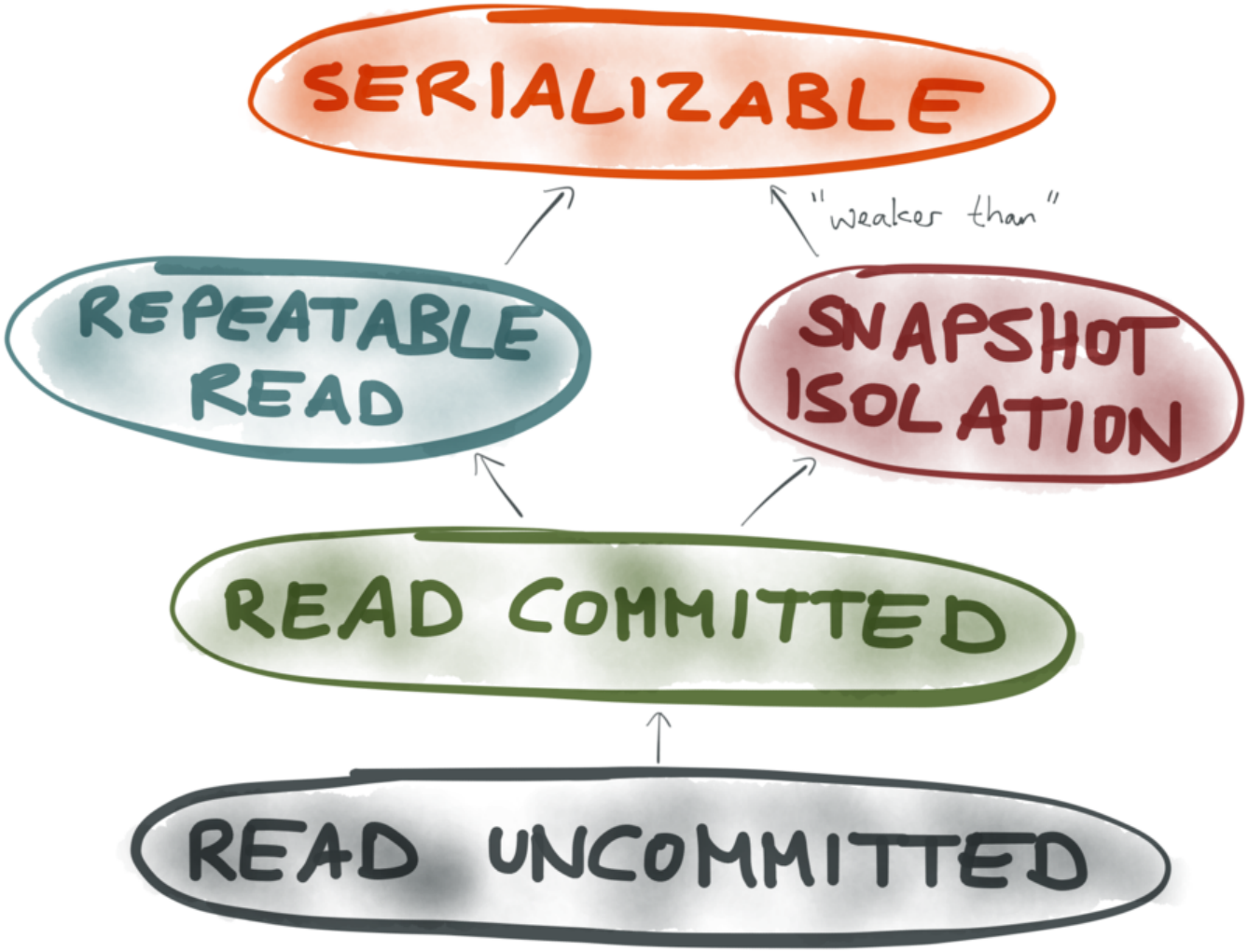


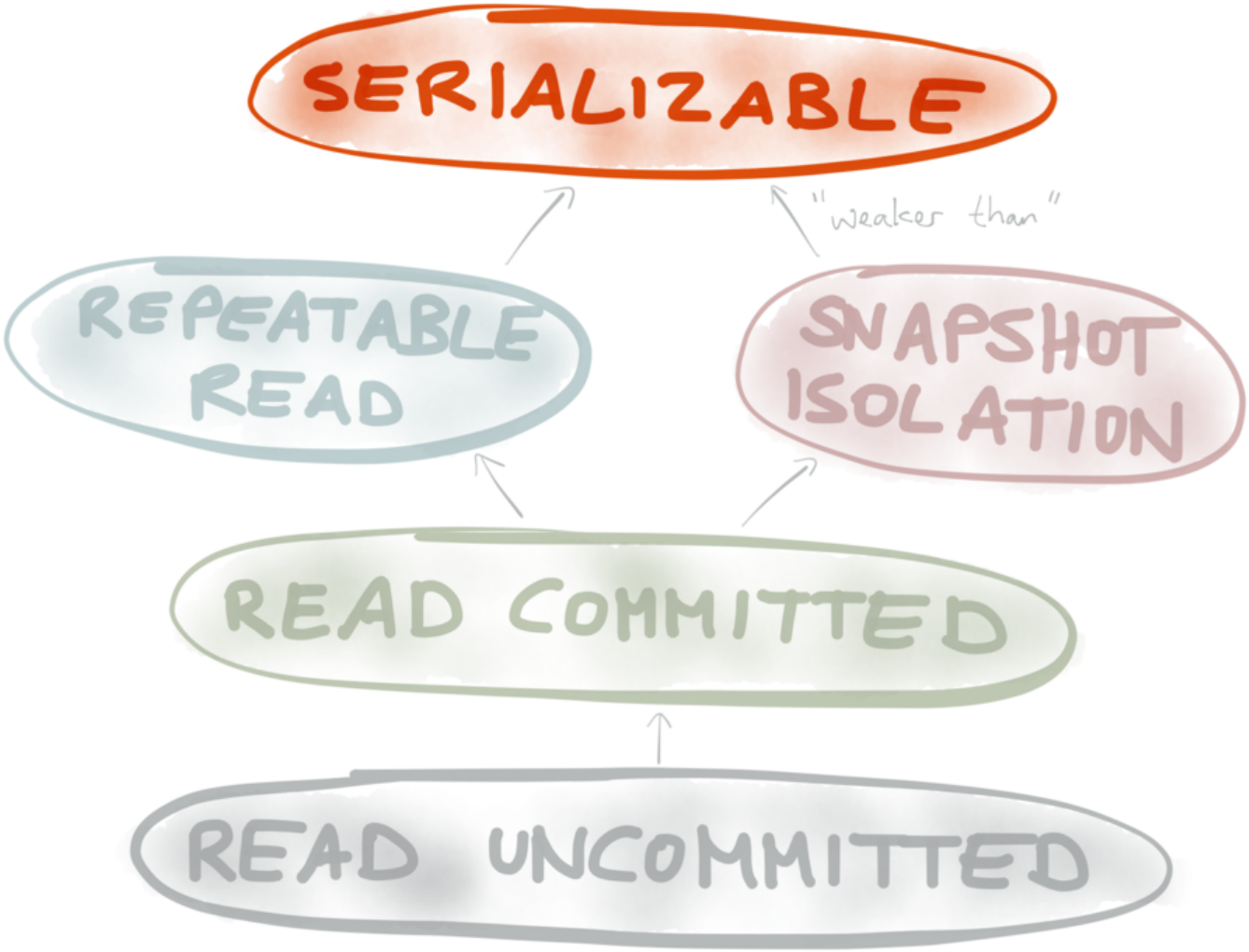

SERIALIZABLE
"weaker than"
REPEATABLE READ
SNAPSHOT ISOLATION
READ COMMITTED
READ UNCOMMITTED

Preventing read skew

- Locking REPEATABLE READ
  (eg. MS SQL Server)

- Snapshot isolation / MVCC
  (eg. Postgres "REPEATABLE READ",
  MySQL "REPEATABLE READ" – for read-only transactions,
  SQL Server "SNAPSHOT",
  Oracle "SERIALIZABLE")


SERIALIZABLE
"weaker than"
REPEATABLE READ
SNAPSHOT ISOLATION
READ COMMITTED
READ UNCOMMITTED

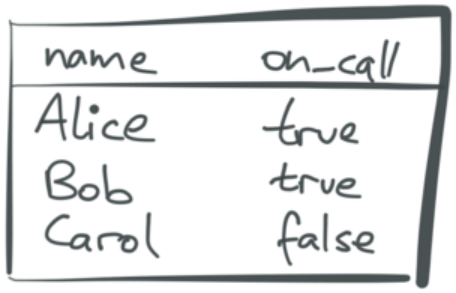

SERIALIZABLE
"weaker than"
REPEATABLE READ
SNAPSHOT ISOLATION
READ COMMITTED
READ UNCOMMITTED

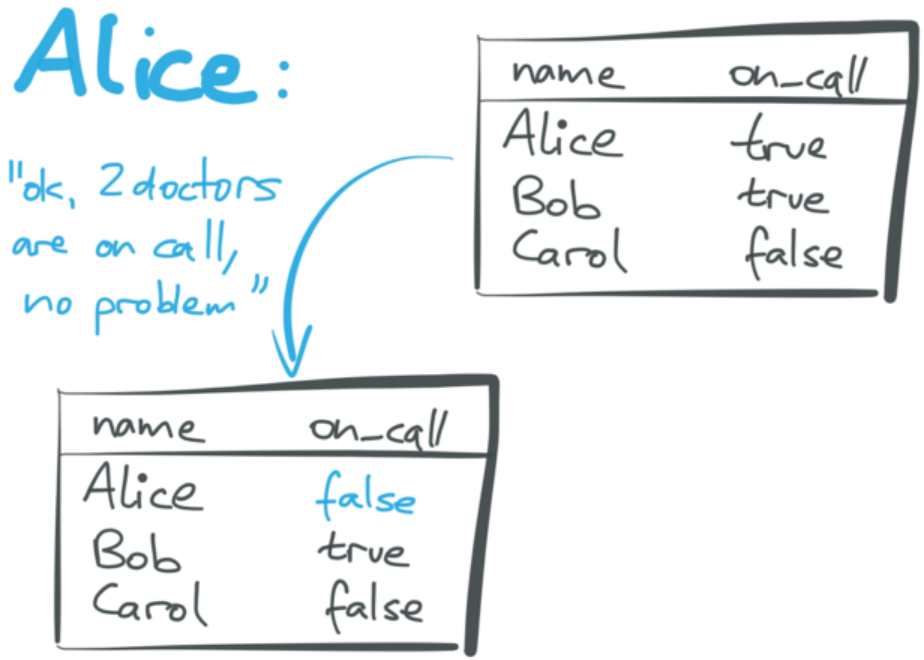
```
BEGIN TRANSACTION;
on_call = $(
  SELECT COUNT(x)
  FROM doctors
  WHERE on_call = true
)

COMMIT;
```

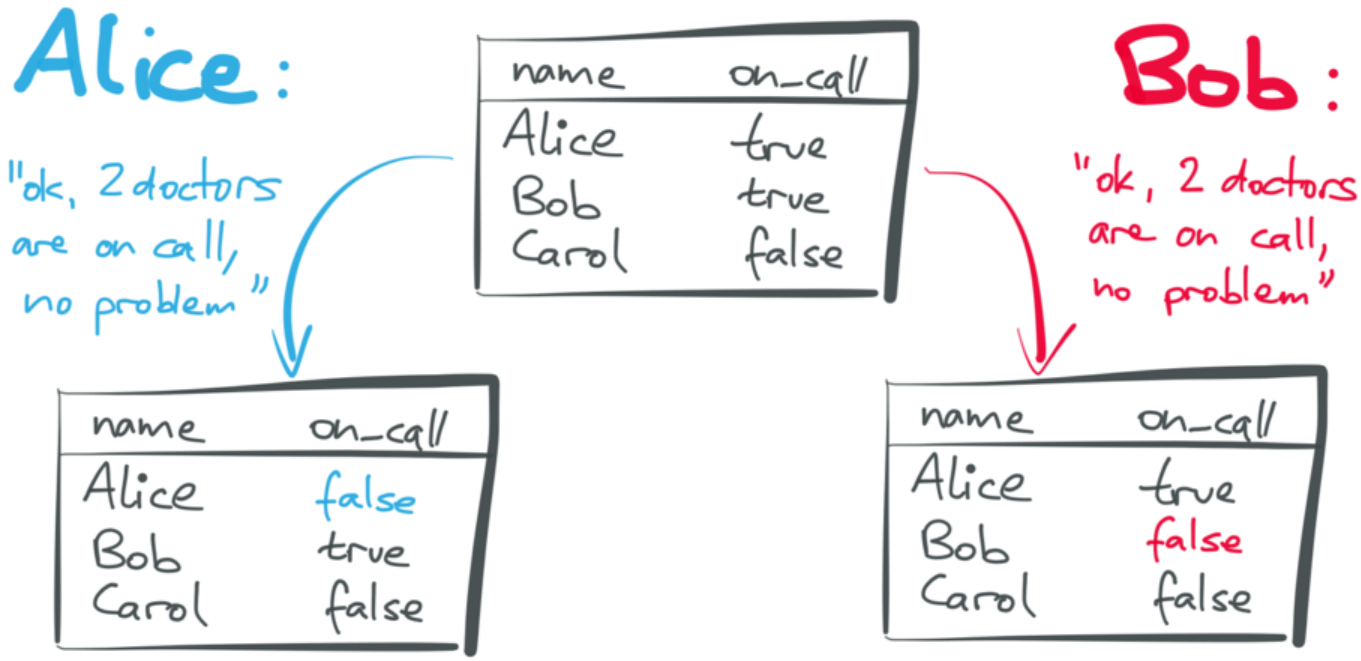
(Cahill, Röhm & Fekete, 2008)

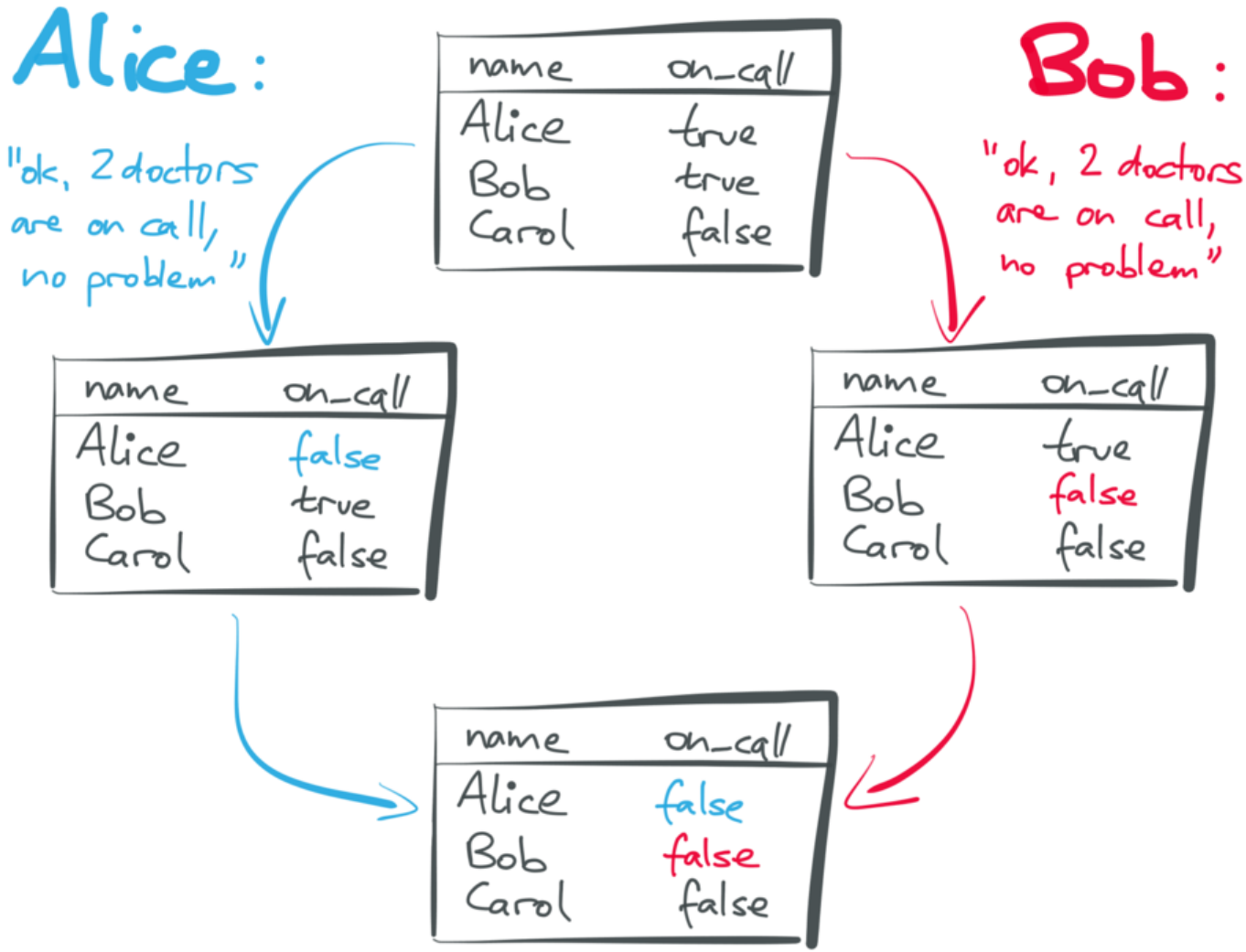
```
BEGIN TRANSACTION;
on_call = $(
  SELECT COUNT(x)
  FROM doctors
  WHERE on_call = true
)
if (on_call >= 2) {
  UPDATE doctors
  SET on_call = false
  WHERE id = 123
}
COMMIT;
```

(Cahill, Röhm & Fekete, 2008)


name
on_call
Alice
true
Bob
true
Carol
false


Alice:
"ok, 2 doctors are on call, no problem"
name on_call
Alice true
Bob true
Carol false
name on_call
Alice false
Bob true
Carol false

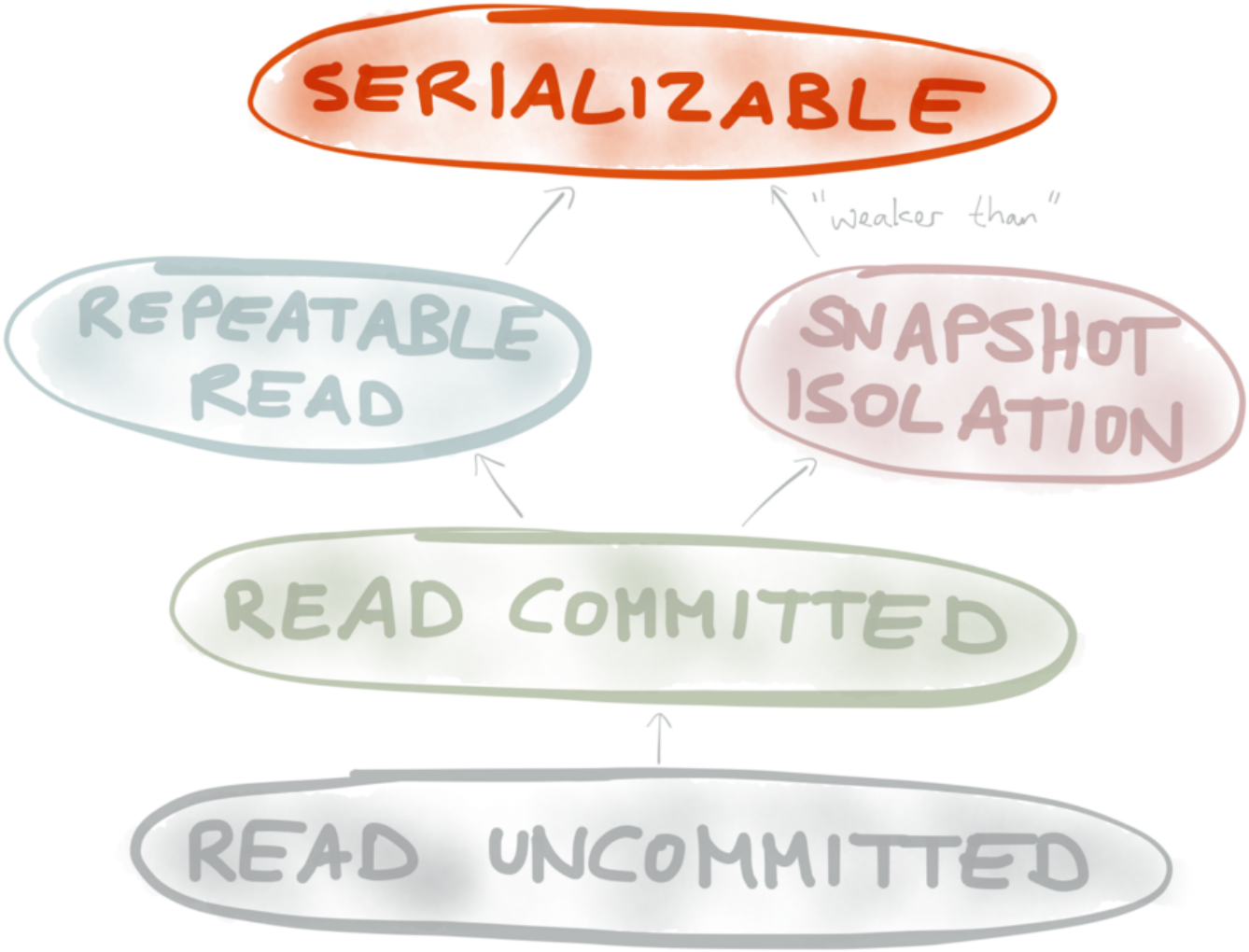

Alice:
"ok, 2 doctors are on call, no problem"
name on_call
Alice true
Bob true
Carol false
Bob:
"ok, 2 doctors are on call, no problem"
name on_call
Alice false
Bob true
Carol false
name on_call
Alice true
Bob false
Carol false


Alice:
"ok, 2 doctors are on call, no problem"
name on_call
Alice true
Bob true
Carol false
Bob:
"ok, 2 doctors are on call, no problem"
name on_call
Alice false
Bob true
Carol false
name on_call
Alice true
Bob false
Carol false
name on_call
Alice false
Bob false
Carol false

Write skew

Pattern: - read something
- make decision
- write decision to DB

By the time the write is committed, the premise of the decision is no longer true!


SERIALIZABLE
"weaker than"
REPEATABLE READ
SNAPSHOT ISOLATION
READ COMMITTED
READ UNCOMMITTED

Solutions (making it serializable)

1. Lock everything you read
   (2-phase Locking, pessimistic)

2-Phase Locking

(InnoDB SERIALIZABLE,
SQL Server SERIALIZABLE.
Note: 2PL ≠ 2PC)

```
SELECT COUNT(x)
FROM doctors
WHERE on_call = true
```

2-Phase Locking

(InnoDB SERIALIZABLE,
SQL Server SERIALIZABLE.
Note: 2PL ≠ 2PC)

```
SELECT COUNT(x)
FROM doctors
WHERE on_call = true
```

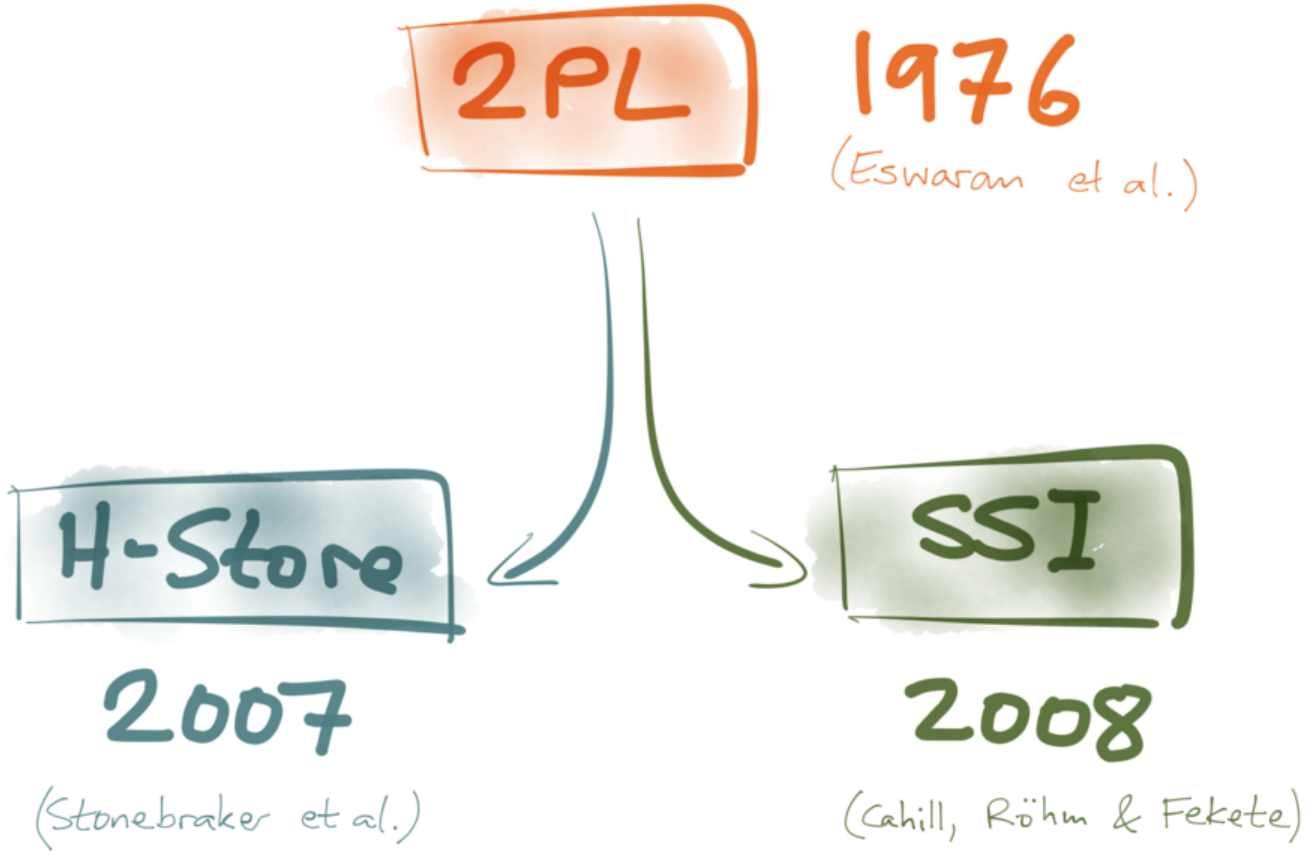
Shared lock (other transactions can read, but cannot modify)

2-Phase Locking (N.B. 2PL ≠ 2PC)

```
SELECT COUNT(x)
FROM doctors
WHERE on_call = true
```

Hold the lock until commit/abort

Read entire DB = stop all writes


2PL
1976
(Eswaran et al.)
H-Store
2007
(Stonebraker et al.)
SSI
2008
(Cahill, Röhm & Fekete)

Solutions (making it serializable)

1. Lock everything you read
   (2-phase locking, pessimistic)
2. Literally serial order
   (Gotta make those transactions fast!)

Literally executing everything in a serial order

(H-Store, VoltDB, Datomic, Redis?)

Literally executing everything in a serial order

(H-Store, VoltDB, Datomic, Redis?)

Only if each txn is really fast!

Literally executing everything in a serial order

(H-Store, VoltDB, Datomic, Redis?)

Only if each txn is really fast!

- keep everything in memory
- locality of data & computation
  (stored procedures, partitioning)

Solutions (making it serializable)

1. Lock everything you read
   (2-phase locking, pessimistic)
2. Literally serial order
   (Gotta make those transactions fast!)
3. Detect conflicts & abort
   (Serializable Snapshot Isolation, optimistic)

Serializable Snapshot Isolation (SSI)

(PostgreSQL SERIALIZABLE – Ports & Grittner, 2012)

Idea: locks like in 2PL

Serializable Snapshot Isolation (SSI)

(PostgreSQL SERIALIZABLE – Ports & Grittner, 2012)

Idea: locks like in 2PL

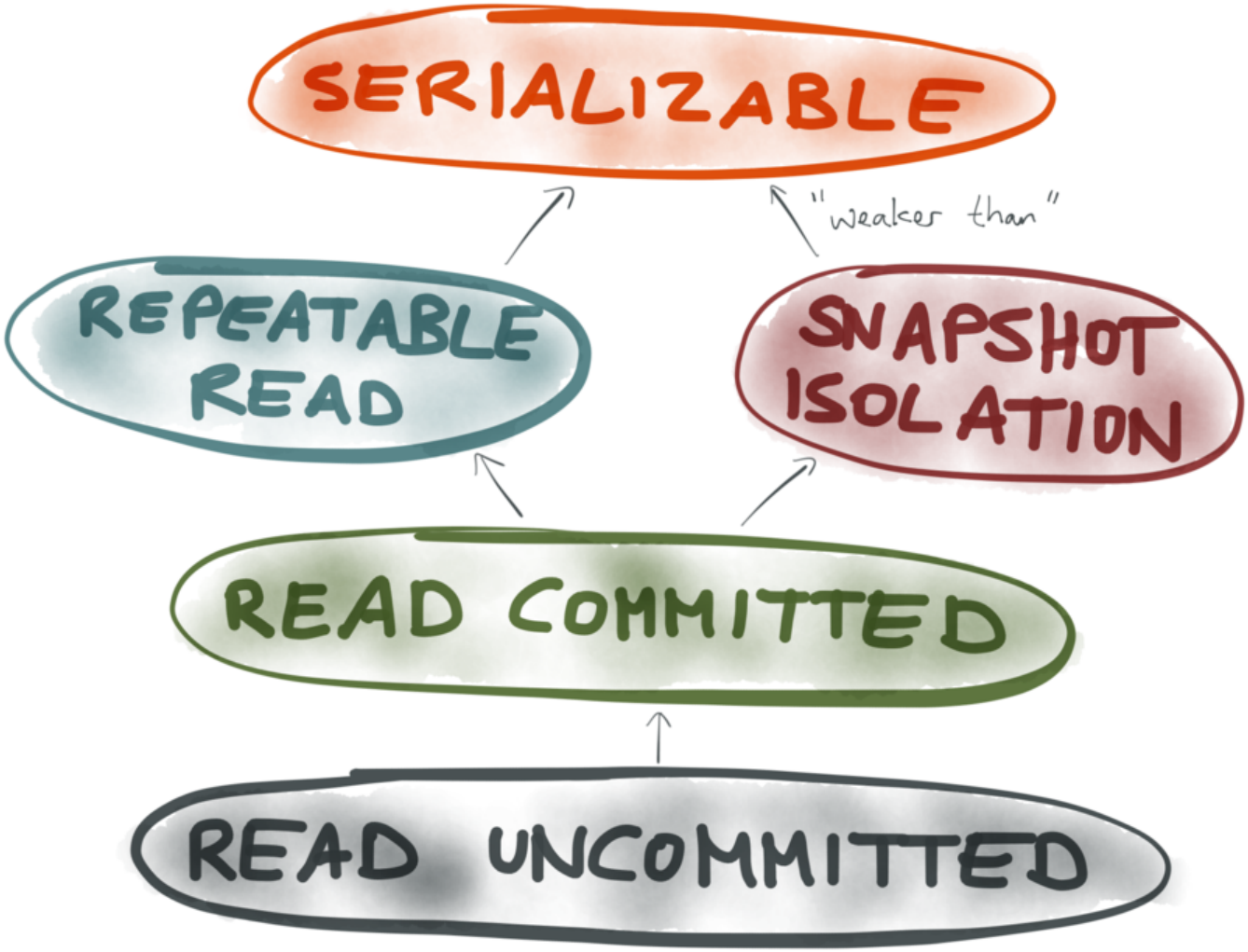
BUT: locks don't block
(only gather information)

Serializable Snapshot Isolation (SSI)

(PostgreSQL SERIALIZABLE – Ports & Grittner, 2012)

Idea: locks like in 2PL

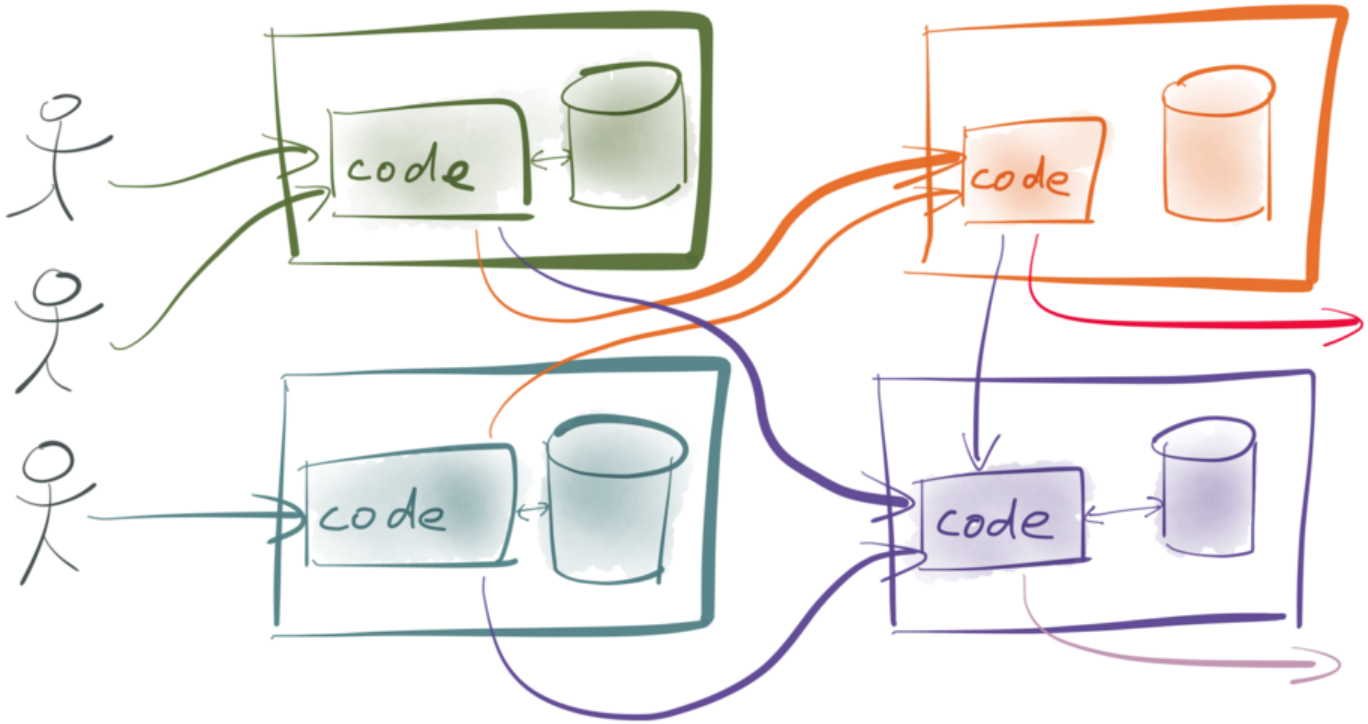
BUT: locks don't block
(only gather information)


SERIALIZABLE
"weaker than"
REPEATABLE READ
SNAPSHOT ISOLATION
READ COMMITTED
READ UNCOMMITTED

...but what about...

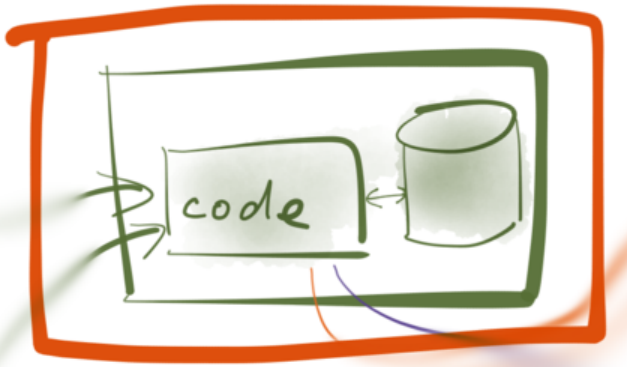
Microservices?

Stream processing?



Microservices: 2-way request/response dataflow via REST/RPC APIs.

Stream processing: 1-way dataflow via queues.



Transaction boundary

Serializability across services?

Serializability across services?

Serializability across services?

Atomic commitment (2PC, 3PC)

Total ordering (Atomic broadcast)

Serializability across services?

Atomic commitment
(2PC, 3PC)

COORDINATION

Total ordering
(Atomic broadcast)

Consensus
(Paxos, Raft, Zab)

(Chandra & Toueg, 1996; Guerraoui, 1995)

Geographic distribution

Without cross-service transactions:

Without cross-service transactions:

A

Compensating transactions

≈ abort/rollback at app level

(Garcia-Molina & Salem, 1987)

Without cross-service transactions:

A

Compensating transactions

≈ abort/rollback at app level

(Garcia-Molina & Salem, 1987)

C

Apologies

(Helland & Campbell, 2009)

detect & fix constraint violations

(after the fact, rather than preventing them)

Every sufficiently large deployment of

microservices

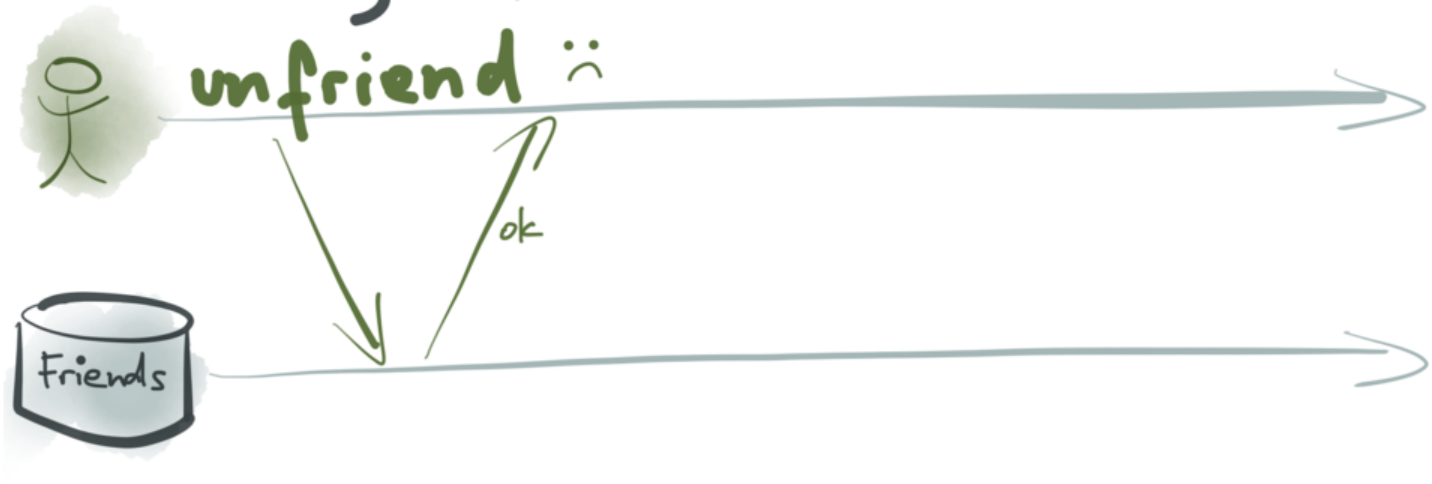
Every sufficiently large deployment of

microservices

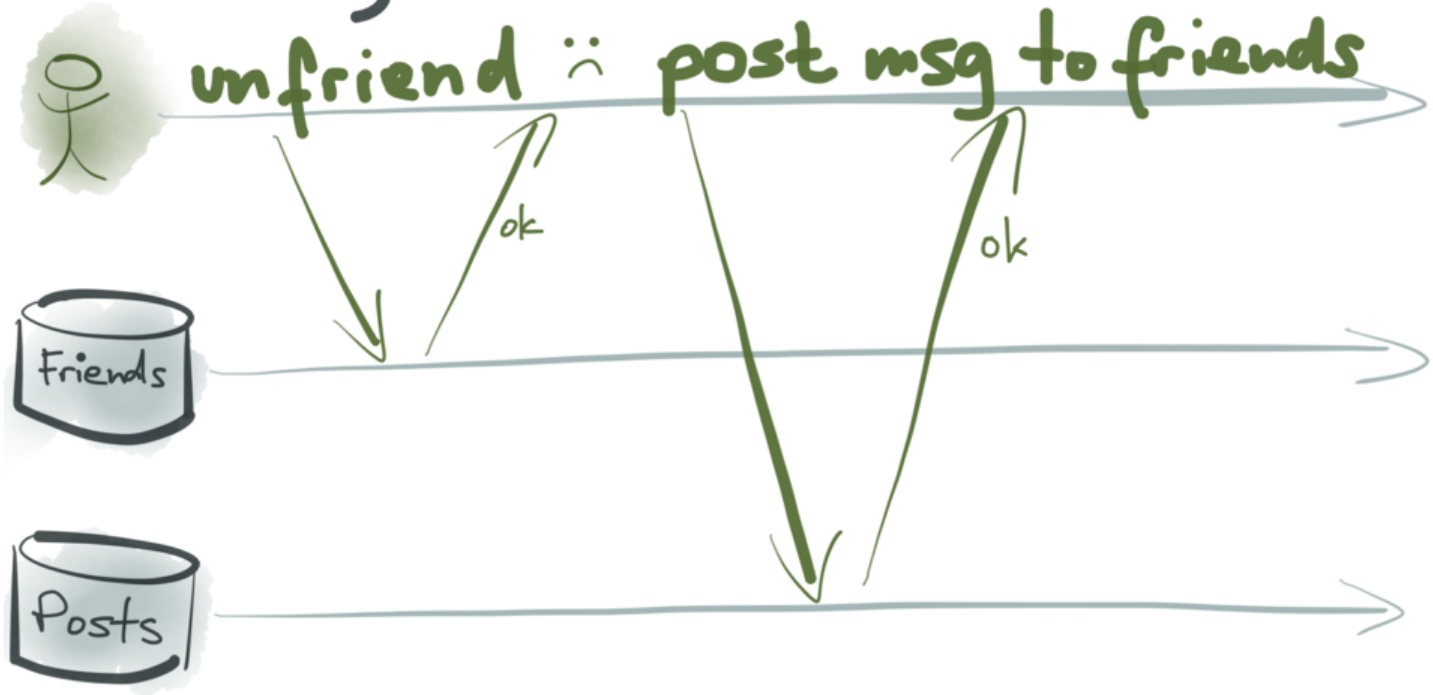
contains an ad-hoc, informally-specified, bug-ridden, slow implementation of half of

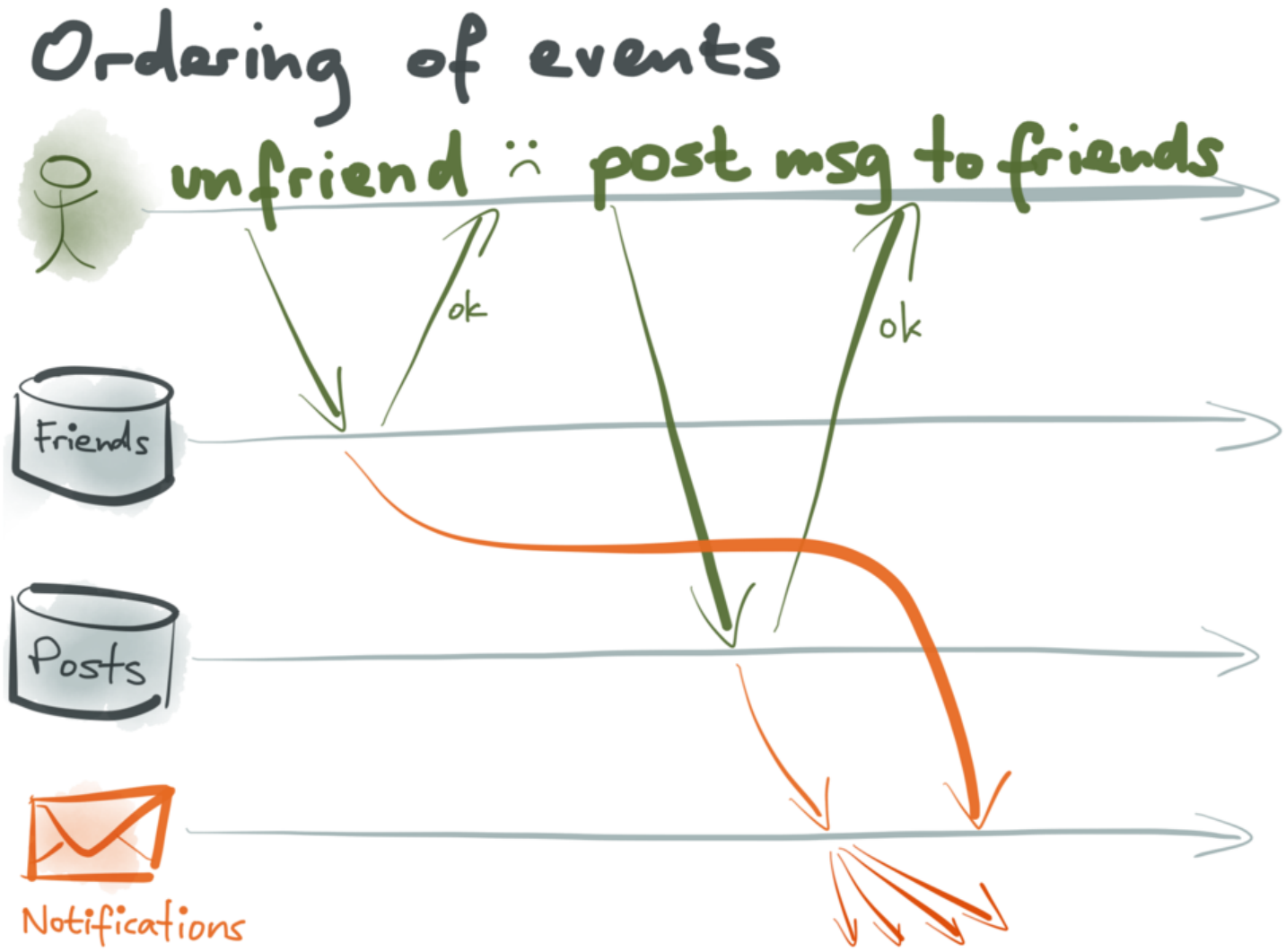
transactions

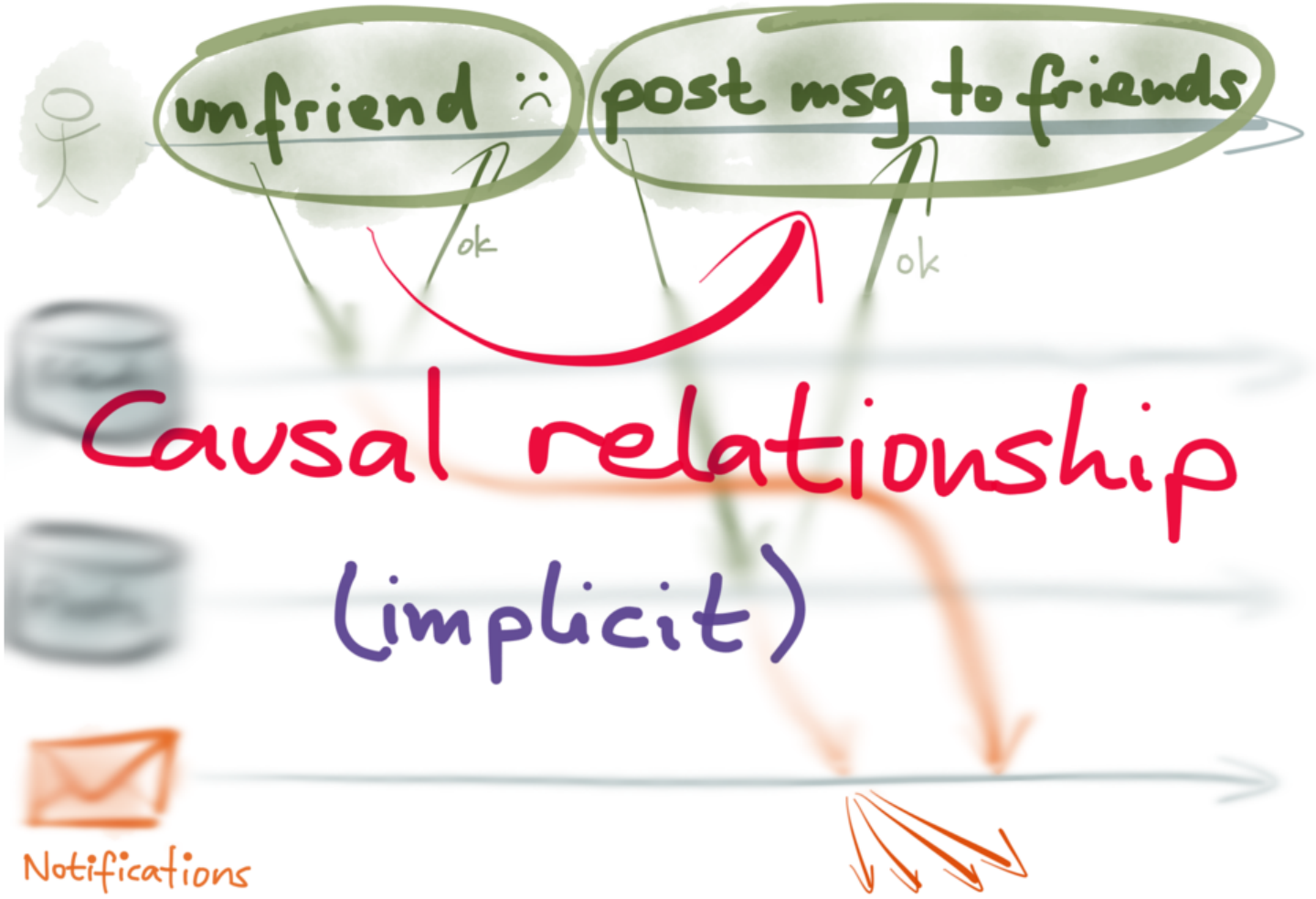
Ordering of events

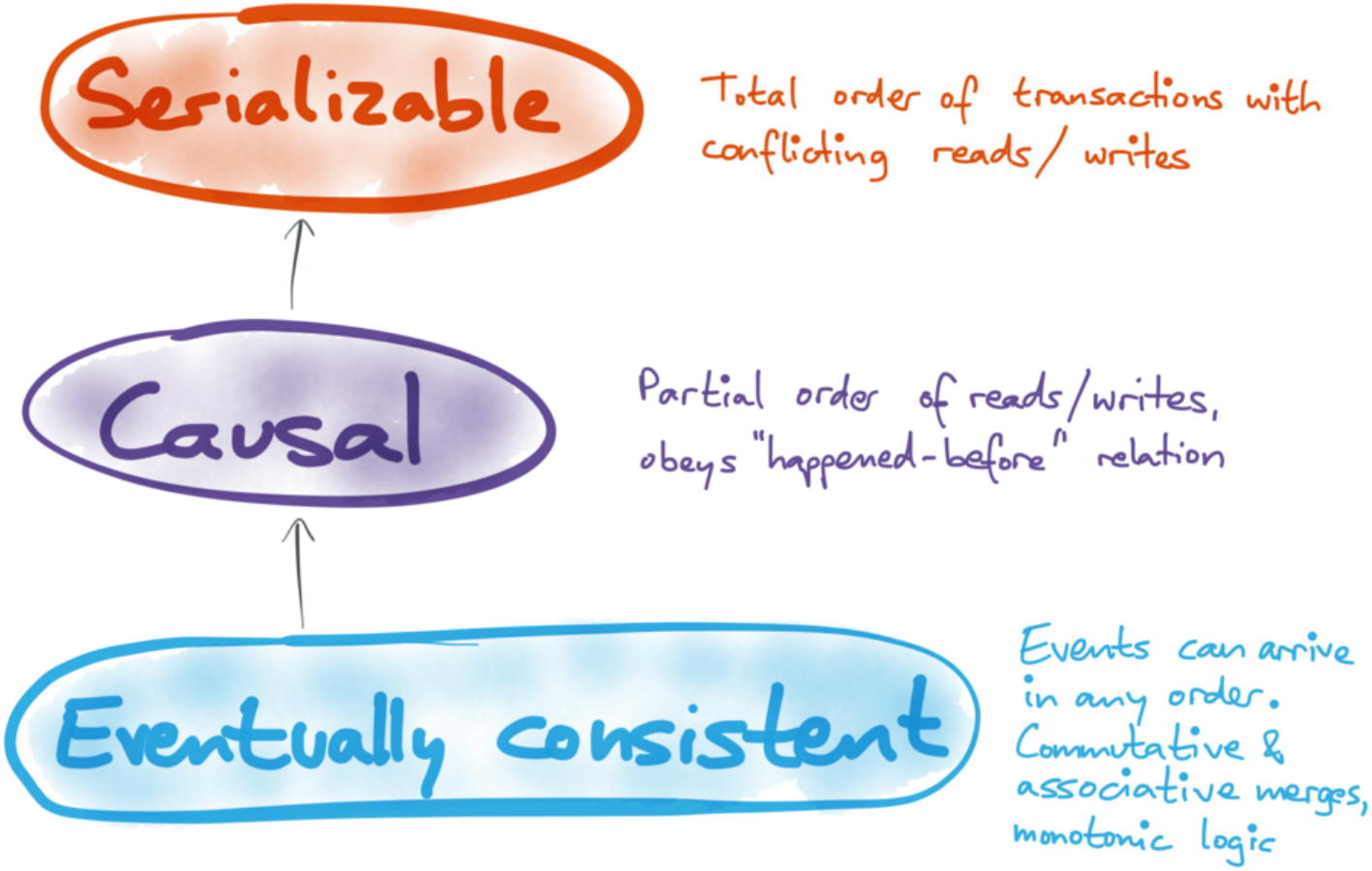


Ordering of events




Ordering of events
unfriend
post msg to friends
ok
ok
Friends
Posts
Notifications


unfriend
post msg to friends
ok
ok
Causal relationship
(implicit)
Notifications



(Bailis et al, SIGMOD 2013 ; Hellerstein, 2010 ; Lloyd et al, 2011 ; Zawirski et al, 2013)

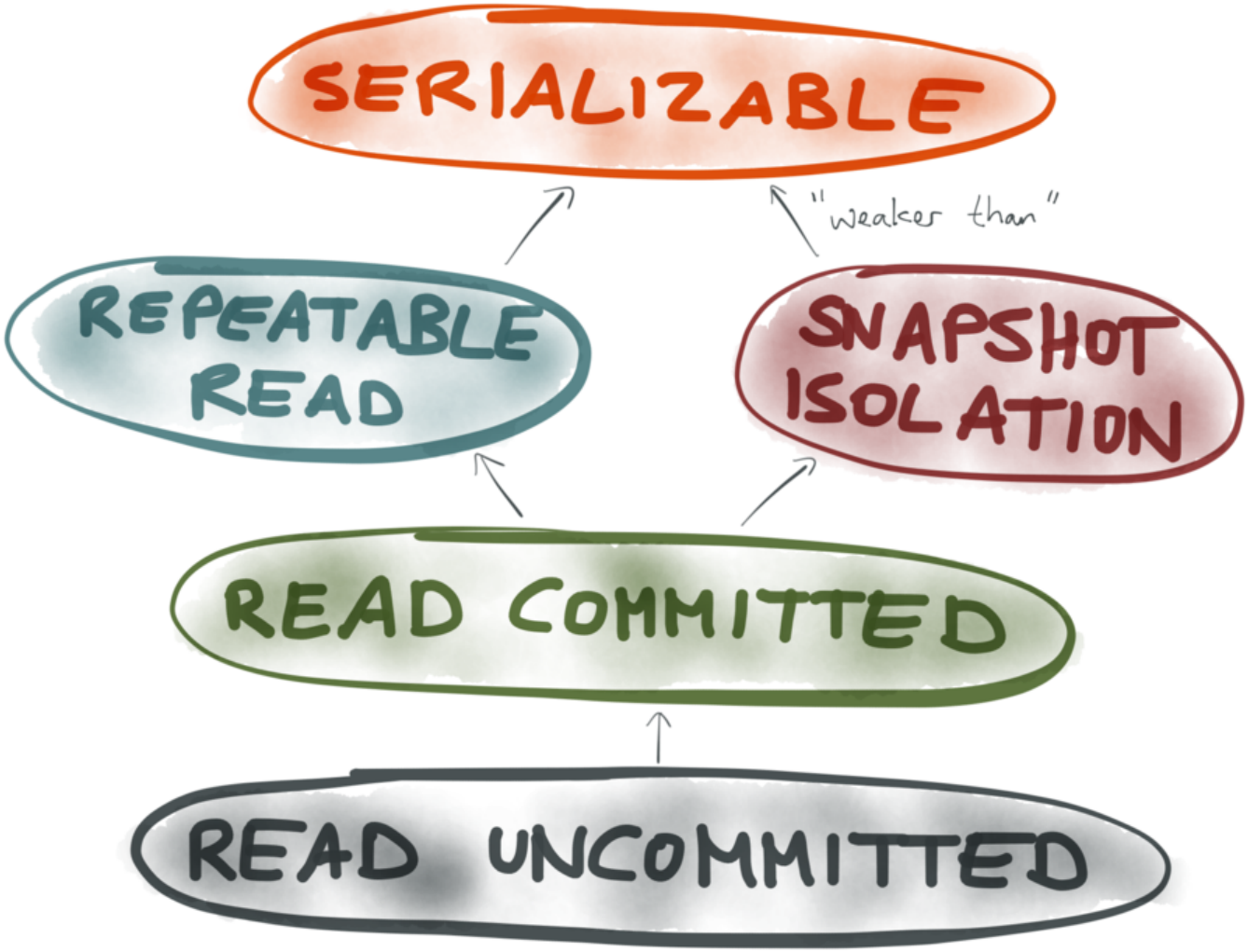
Causality can be maintained without global coordination

"Consistent snapshot" in SI

consistent with causality

But also efficient?

(Attiya, Ellen & Morrison, 2015)


SERIALIZABLE
"weaker than"
REPEATABLE READ
SNAPSHOT ISOLATION
READ COMMITTED
READ UNCOMMITTED


COORDINATION REQUIRED
COORDINATION-FREE
MAV? +causal
(Bailis et al, 2014)
READ COMMITTED
READ UNCOMMITTED

References (1/4)

1. Atul Adya: "Weak Consistency: A Generalized Theory and Optimistic Implementations for Distributed Transactions," PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, March 1999. http://pmg.csail.mit.edu/papers/adya-phd.pdf
2. Hagit Attiya, Faith Ellen, and Adam Morrison: "Limitations of Highly-Available Eventually-Consistent Data Stores," at *ACM Symposium on Principles of Distributed Computing* (PODC), July 2015. http://www.cs.technion.ac.il/people/mad/online-publications/podc2015-replds.pdf
3. Peter Bailis, Alan Fekete, Ali Ghodsi, Joseph M Hellerstein, and Ion Stoica: "HAT, not CAP: Towards Highly Available Transactions," at *14th USENIX Workshop on Hot Topics in Operating Systems* (HotOS), May 2013. http://www.bailis.org/papers/hat-hotos2013.pdf
4. Peter Bailis, Ali Ghodsi, Joseph M Hellerstein, and Ion Stoica: "Bolt-on Causal Consistency," at *ACM International Conference on Management of Data* (SIGMOD), June 2013. http://db.cs.berkeley.edu/papers/sigmod13-bolton.pdf
5. Peter Bailis, Aaron Davidson, Alan Fekete, et al.: "Highly Available Transactions: Virtues and Limitations," at *40th International Conference on Very Large Data Bases* (VLDB), September 2014. http://www.bailis.org/papers/hat-vldb2014.pdf
6. Hal Berenson, Philip A Bernstein, Jim N Gray, et al.: "A Critique of ANSI SQL Isolation Levels," at *ACM International Conference on Management of Data* (SIGMOD), May 1995. http://research.microsoft.com/pubs/69541/tr-95-51.pdf

References (2/4)

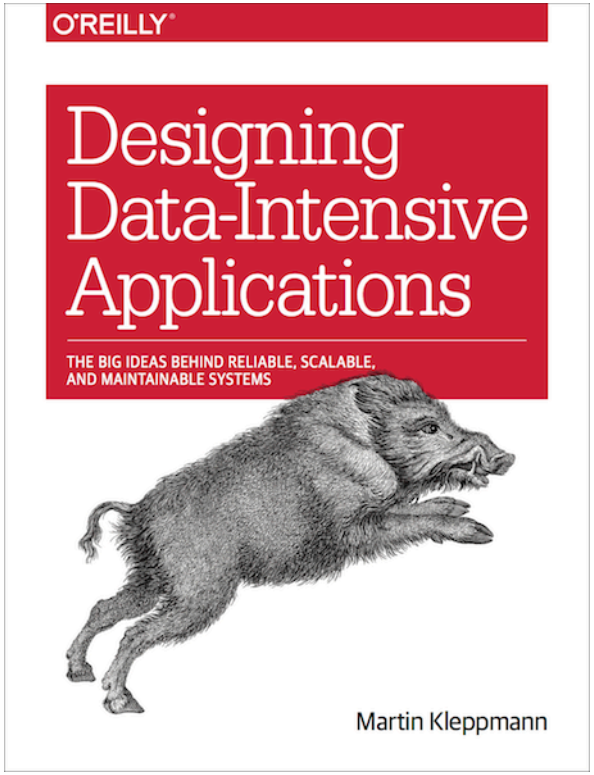
7. Eric A Brewer: "CAP Twelve Years Later: How the "Rules" Have Changed," *IEEE Computer Magazine*, volume 45, number 2, pages 23–29, February 2012. http://cs609.cs.ua.edu/CAP12.pdf
8. Michael J Cahill, Uwe Röhm, and Alan Fekete: "Serializable Isolation for Snapshot Databases," at *ACM International Conference on Management of Data* (SIGMOD), pages 729–738, June 2008. http://www.cs.nyu.edu/courses/fall12/CSCI-GA.2434-001/p729-cahill.pdf
9. Donald D Chamberlin, Morton M Astrahan, Michael W Blasgen, et al.: "A History and Evaluation of System R," *Communications of the ACM*, volume 24, number 10, pages 632–646, October 1981. http://diaswww.epfl.ch/courses/adms07/papers/p632-chamberlin.pdf
10. Tushar Deepak Chandra and Sam Toueg: "Unreliable Failure Detectors for Reliable Distributed Systems," *Journal of the ACM*, volume 43, number 2, pages 225–267, March 1996. http://courses.csail.mit.edu/6.852/08/papers/CT96-JACM.pdf
11. Kapali P Eswaran, Jim N Gray, Raymond A Lorie, and Irving L Traiger: "The Notions of Consistency and Predicate Locks in a Database System," *Communications of the ACM*, volume 19, number 11, pages 624–633, November 1976. http://paul.rutgers.edu/cs545/S02/papers/eswaran-transaction.pdf
12. Hector Garcia-Molina and Kenneth Salem: "Sagas," at *ACM International Conference on Management of Data* (SIGMOD), May 1987. http://www.cs.cornell.edu/andru/cs711/2002fa/reading/sagas.pdf

References (3/4)

13. Jim N Gray, Raymond A Lorie, Gianfranco R Putzolu, and Irving L Traiger: "Granularity of Locks and Degrees of Consistency in a Shared Data Base," in *Modelling in Data Base Management Systems: Proceedings of the IFIP Working Conference on Modelling in Data Base Management Systems*, G.M. Nijssen, Editor. Elsevier/North Holland Publishing, pages 364–394, 1976. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.92.8248
14. Rachid Guerraoui: "Revisiting the relationship between non-blocking atomic commitment and consensus," at *9th International Workshop on Distributed Algorithms* (WDAG), pages 87–100, September 1995. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.27.6456
15. Theo Härder and Andreas Reuter: "Principles of Transaction-Oriented Database Recovery," *ACM Computing Surveys*, volume 15, number 4, pages 287–317, December 1983. http://web.stanford.edu/class/cs340v/papers/recovery.pdf
16. Pat Helland and Dave Campbell: "Building on Quicksand," at *4th Biennial Conference on Innovative Data Systems Research* (CIDR), January 2009. https://database.cs.wisc.edu/cidr/cidr2009/Paper_133.pdf
17. Joseph M Hellerstein: "The Declarative Imperative: Experiences and Conjectures in Distributed Logic," Technical Report, University of California at Berkeley, UCB/EECS-2010-90, June 2010. http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-90.pdf
18. Martin Kleppmann: "Hermitage: Testing the 'I' in ACID," 25 November 2014. http://martin.kleppmann.com/2014/11/25/hermitage-testing-the-i-in-acid.html

References (4/4)

19. Martin Kleppmann: "A Critique of the CAP Theorem," Preprint arXiv:1509.05393 [cs.DC], Sep 2015. http://arxiv.org/abs/1509.05393
20. Martin Kleppmann: *Designing Data-Intensive Applications*. O'Reilly Media, to appear. ISBN 1-4493-7332-1. http://dataintensive.net/
21. Wyatt Lloyd, Michael J Freedman, Michael Kaminsky, and David G Andersen: "Don't Settle for Eventual: Scalable Causal Consistency for Wide-Area Storage with COPS," at *23rd ACM Symposium on Operating Systems Principles* (SOSP), pages 401–416, October 2011. https://www.cs.cmu.edu/~dga/papers/cops-sosp2011.pdf
22. Dan R K Ports and Kevin Grittner: "Serializable Snapshot Isolation in PostgreSQL," at *38th International Conference on Very Large Data Bases* (VLDB), volume 5, number 12, pages 1850–1861, August 2012. http://drkp.net/papers/ssi-vldb12.pdf
23. Michael Stonebraker, Samuel Madden, Daniel J Abadi, et al.: "The End of an Architectural Era (It's Time for a Complete Rewrite)," at *33rd International Conference on Very Large Data Bases* (VLDB), pages 1150–1160, September 2007. http://www.vldb.org/conf/2007/papers/industrial/p1150-stonebraker.pdf
24. Marek Zawirski, Annette Bieniusa, Valter Balegas, et al.: "SwiftCloud: Fault-Tolerant Geo-Replication Integrated all the Way to the Client Machine," INRIA Research Report 8347, August 2013. http://arxiv.org/abs/1310.3107



dataintensive.net

Free signed copies!

Today 12.30pm
O'Reilly booth
@Peabody

Slides that I didn’t have time for

In short, the current situation in the DBMS community reminds us of the period 1970 - 1985 where there was a "group grope" for the best way to build DBMS engines and dramatic changes in commercial products and DBMS vendors ensued. The 1970 - 1985 period was a time of intense debate, a myriad of ideas, and considerable upheaval.

We predict the next fifteen years will have the same feel.

(Stonebraker et al, 2007)

2-Phase Locking

(InnoDB SERIALIZABLE,
SQL Server SERIALIZABLE.
Note: 2PL ≠ 2PC)

```
SELECT COUNT(*)
FROM doctors
WHERE on_call = true
```

Shared lock (other transactions can read, but cannot modify)

Predicate lock? (Eswaran et al, 1976)

Index-range locks.

Think you don't need multi-object transactions?

Think you don't need multi-object transactions?

Foreign key references

Think you don't need multi-object transactions?

Foreign key references

Secondary indexes

Think you don't need multi-object transactions?

Foreign key references

Secondary indexes

Updating denormalized data

Granularity of Locks and Degrees of Consistency in a Shared Data Base

J.N. Gray, R.A. Lorie, G.R. Putzolu, I.L. Traiger

IBM Research Laboratory
San Jose, California

The problem of choosing the appropriate granularity (size) of lockable objects is introduced and the tradeoff between concurrency and overhead is discussed. A locking protocol which allows simultaneous locking at various granularities by different transactions is presented. It is based on the introduction of additional lock modes besides the conventional share mode and exclusive mode. A proof is given of the equivalence of this protocol to a conventional one.

Gray et al, 1976

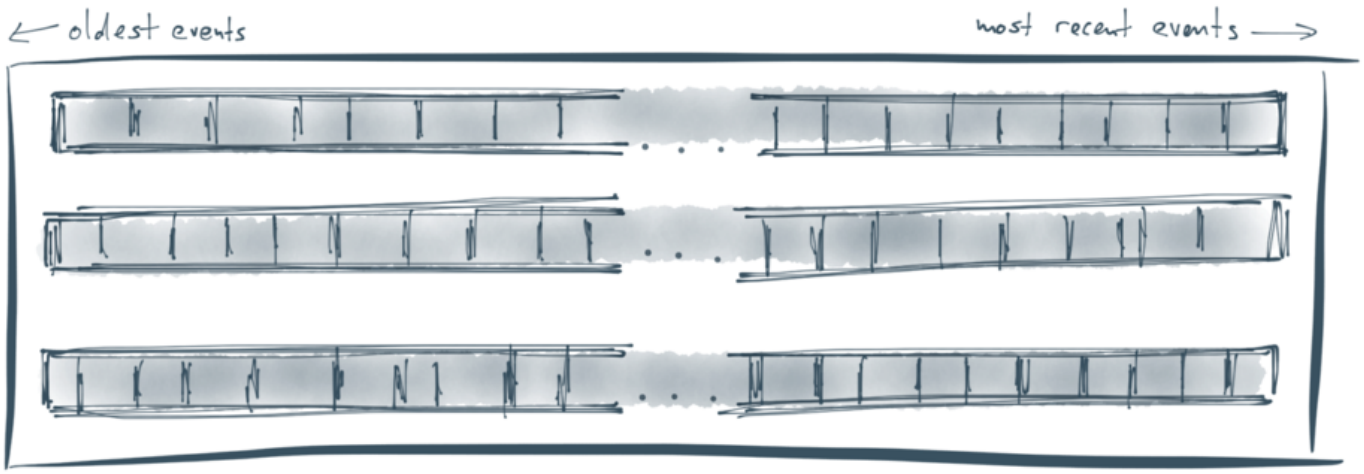
Hermitage

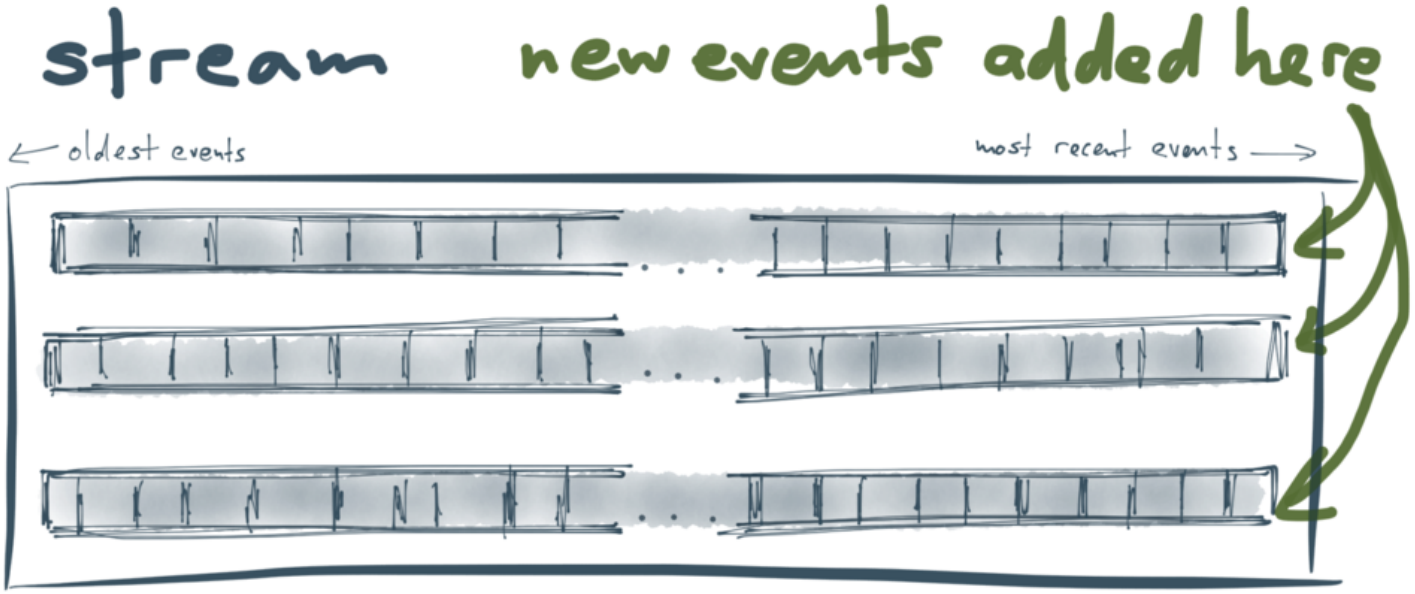
DBMS	So-called isolation level	Actual isolation level	G0	G1a	G1b	G1c	OTV	PMP	P4	G-single	G2-item	G2
PostgreSQL	"read committed" ★	monotonic atomic views	✓	✓	✓	✓	✓	–	–	–	–	–
	"repeatable read"	snapshot isolation	✓	✓	✓	✓	✓	✓	✓	✓	–	–
	"serializable"	serializable	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
MySQL/InnoDB	"read uncommitted"	read uncommitted	✓	–	–	–	–	–	–	–	–	–
	"read committed"	monotonic atomic views	✓	✓	✓	✓	✓	–	–	–	–	–
	"repeatable read" ★	monotonic atomic views	✓	✓	✓	✓	✓	R/O	–	R/O	–	–
	"serializable"	serializable	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Oracle DB	"read committed" ★	monotonic atomic views	✓	✓	✓	✓	✓	–	–	–	–	–
	"serializable"	snapshot isolation	✓	✓	✓	✓	✓	✓	✓	✓	–	some
MS SQL Server	"read uncommitted"	read uncommitted	✓	–	–	–	–	–	–	–	–	–
	"read committed" (locking) ★	monotonic atomic views	✓	✓	✓	✓	✓	–	–	–	–	–
	"read committed" (snapshot)	monotonic atomic views	✓	✓	✓	✓	✓	–	–	–	–	–
	"repeatable read"	repeatable read	✓	✓	✓	✓	✓	–	✓	some	✓	–
	"snapshot"	snapshot isolation	✓	✓	✓	✓	✓	✓	✓	✓	–	–
	"serializable"	serializable	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
FDB SQL Layer	"serializable" ★	serializable	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

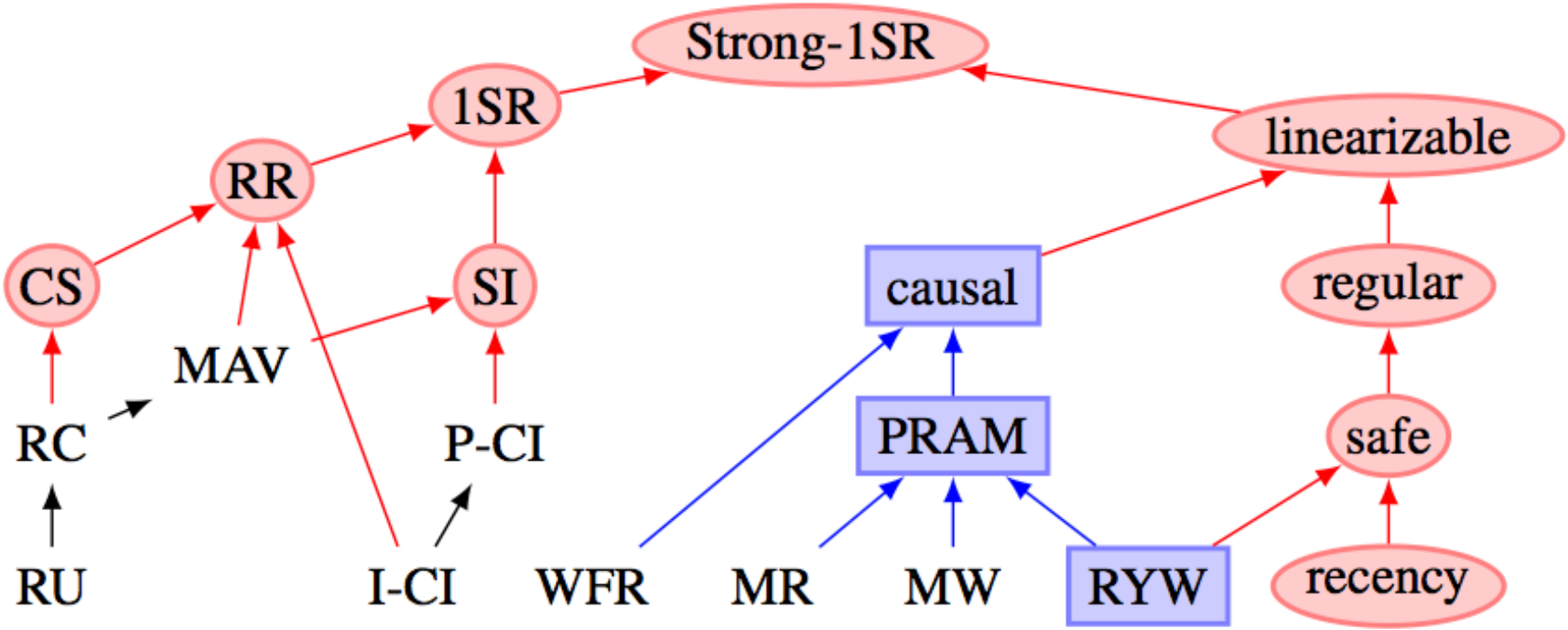
github.com/ept/hermitage

```
success = false;
while (!success) {
    try {
        doTransaction();
        success = true;
    } catch (AbortException e){
        ...
    }
}
```

stream




stream
new events added here
← oldest events
most recent events →



Bailis et al, 2014

CONCURRENCY BUGS

Hard to find in app testing

Hard to specify invariants

Memories, guesses, apologies

(Helland & Campbell, 2009)