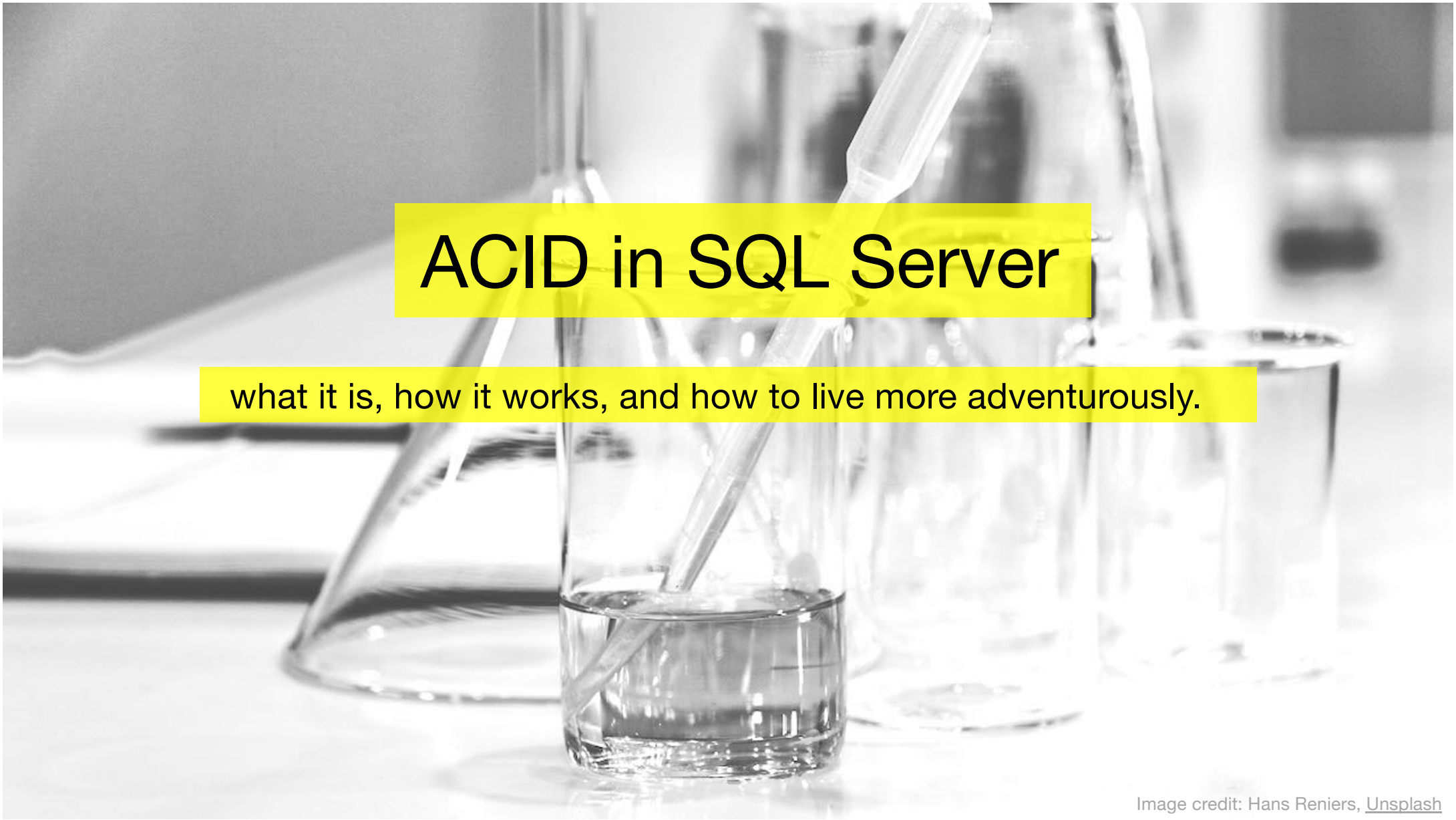


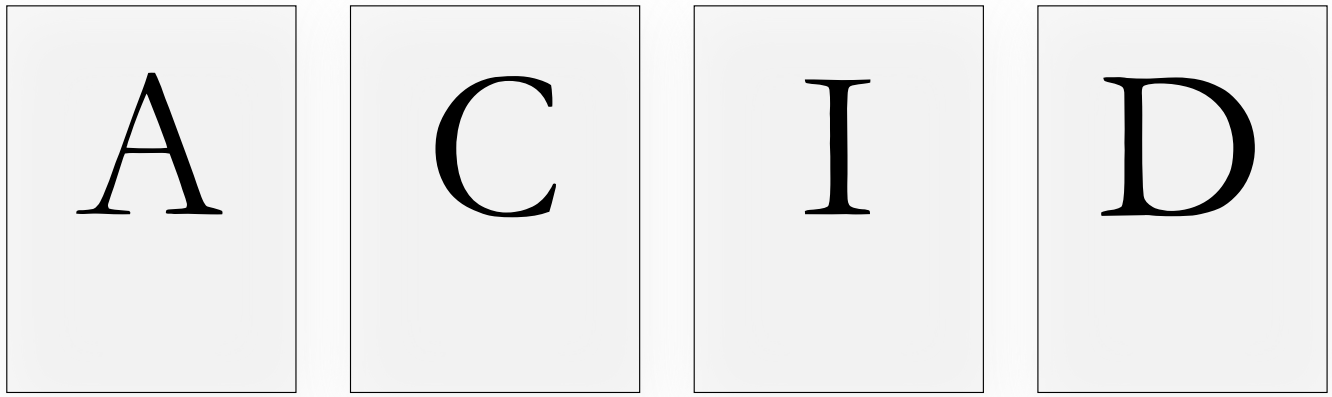
ACID in SQL Server

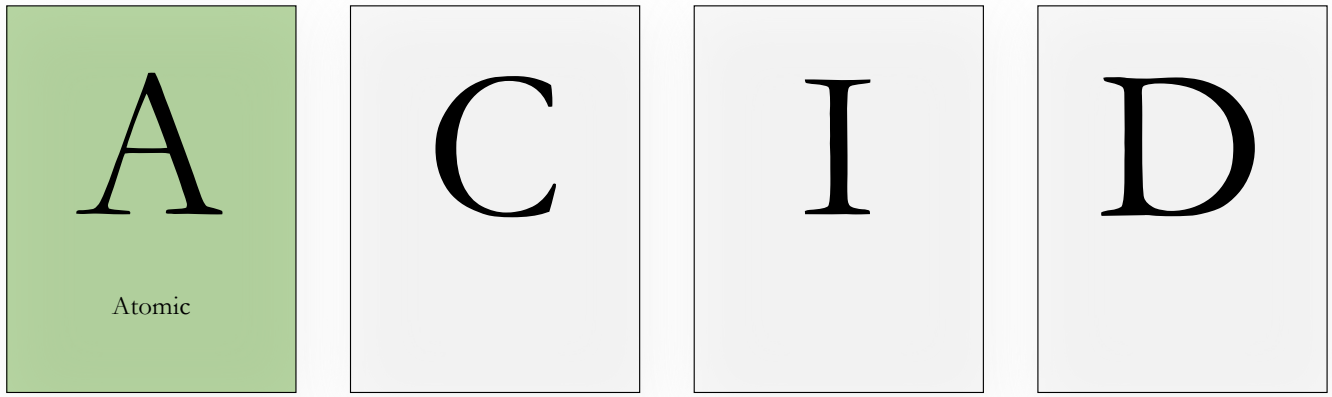
what it is, how it works, and how to live more adventurously.

Image credit: Hans Reniers, Unsplash

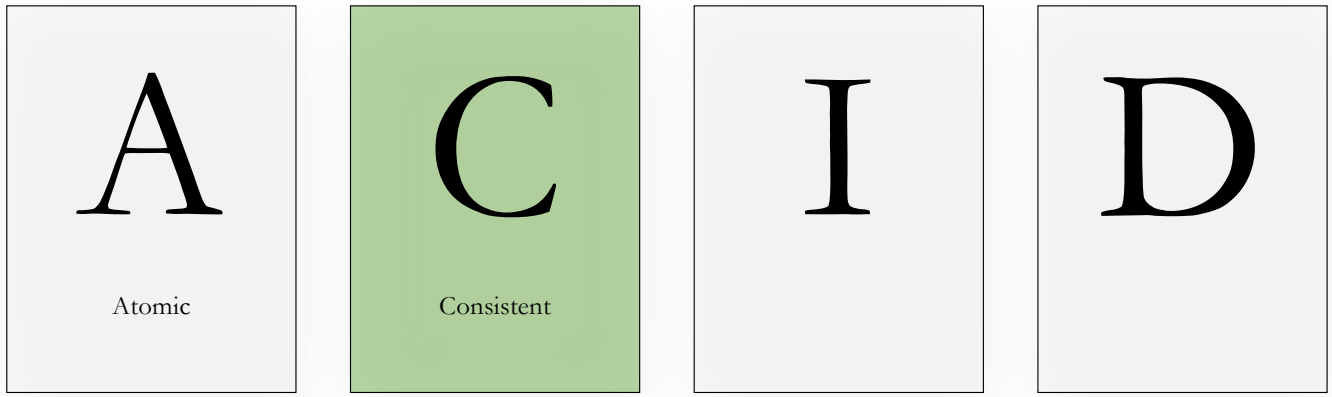
In this session

- What is ACID compliance
- Isolation levels in SQL Server
- Durability in SQL Server

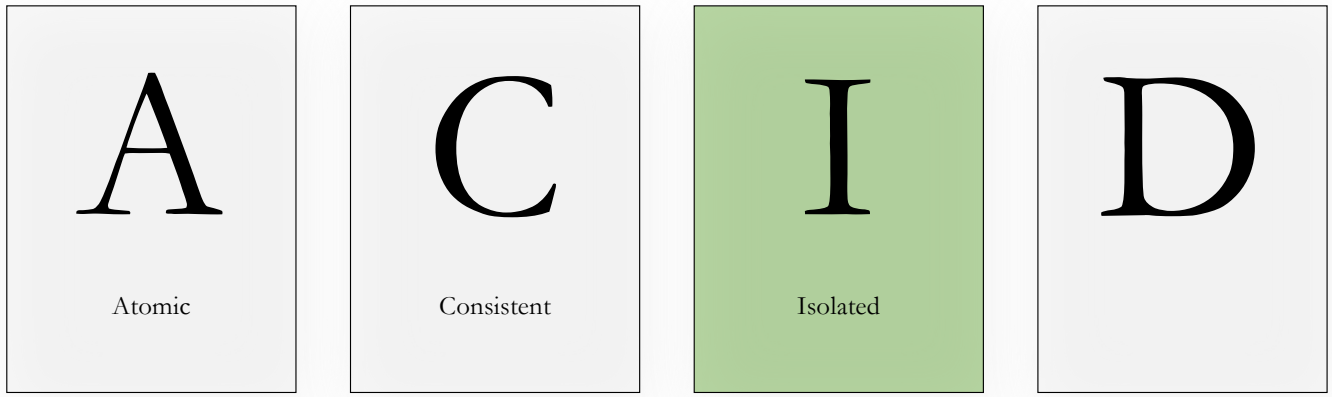

A
C
I
D



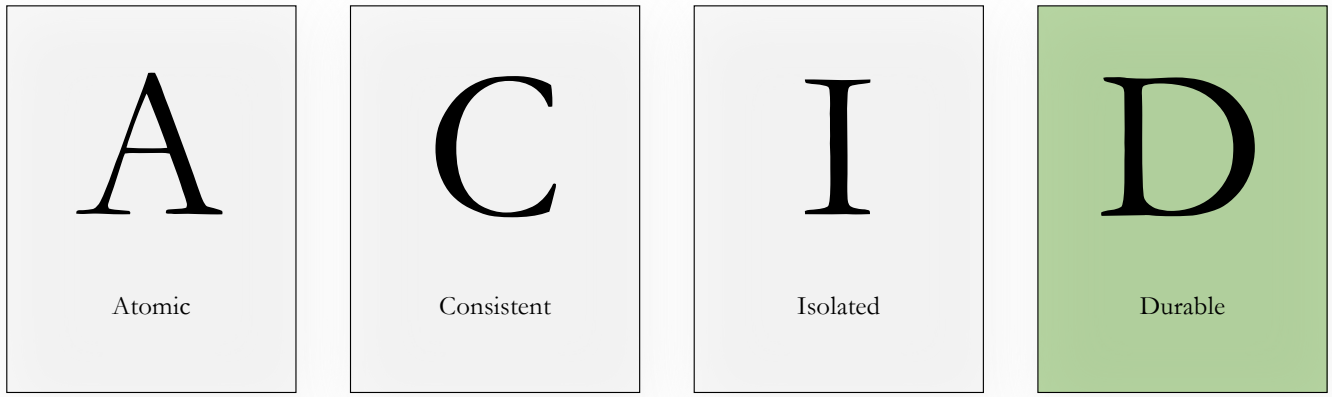
All or nothing.
The *business* logic.



A single truth.
The *database* logic.



No interference
when changing data.



Keep all promises.

Who am I?

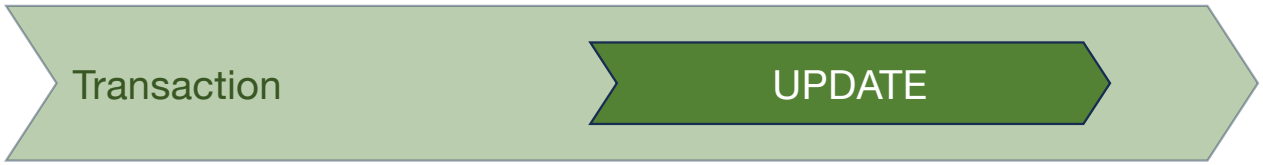
- SQL Server developer since 1997
- Consultant for 25+ years
- Organizer of Data Saturday Stockholm
- Data Platform MVP
- Dog person

@dhmacher on (almost) all the socials.

Isolation: Quirks and features of multi-user databases

Dirty reads

- no Nobel literature prize for you.



Read Uncommitted (a.k.a. NOLOCK)

Dirty reads

- no Nobel literature prize for you.

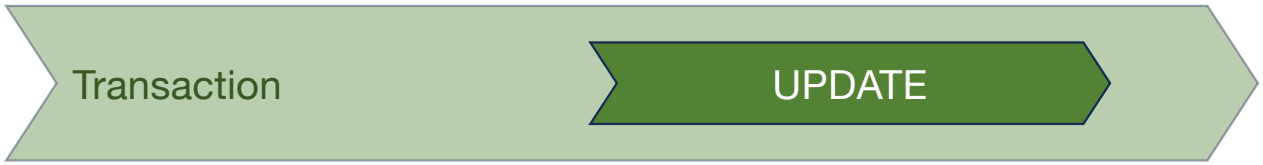
Dirty reads

- no Nobel literature prize for you.

- DML does not respect READ UNCOMMITTED, and cannot make changes to uncommitted data.

Dirty reads

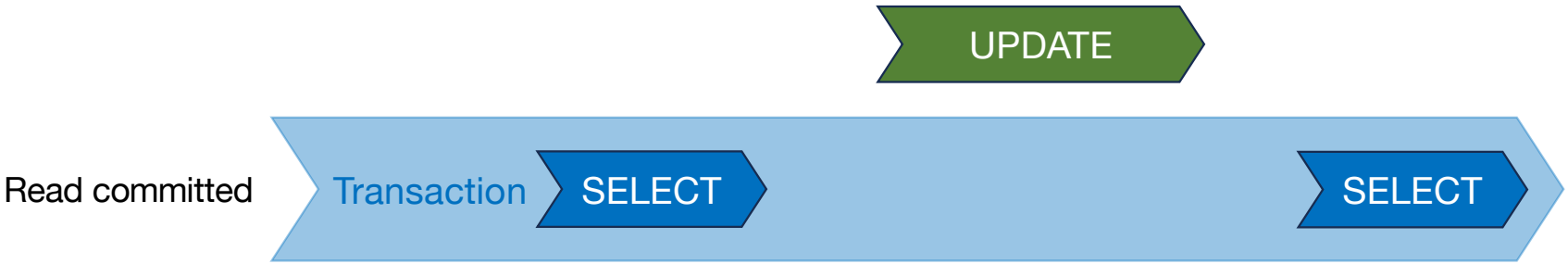
- no Nobel literature prize for you.



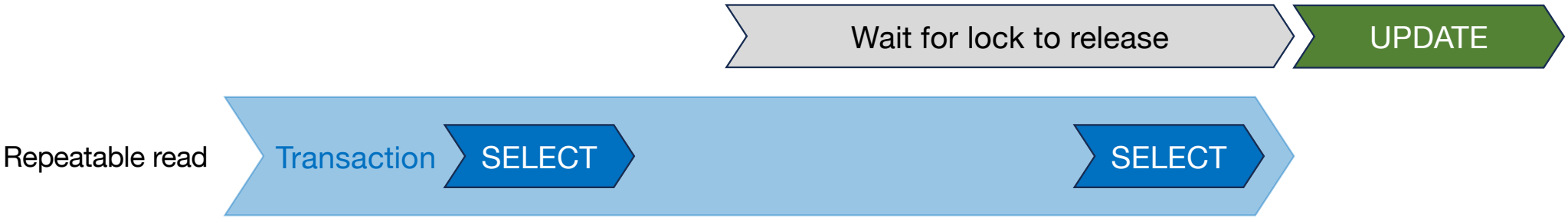
Read Uncommitted (a.k.a. NOLOCK)

DEMO

Non-repeatable reads



Non-repeatable reads



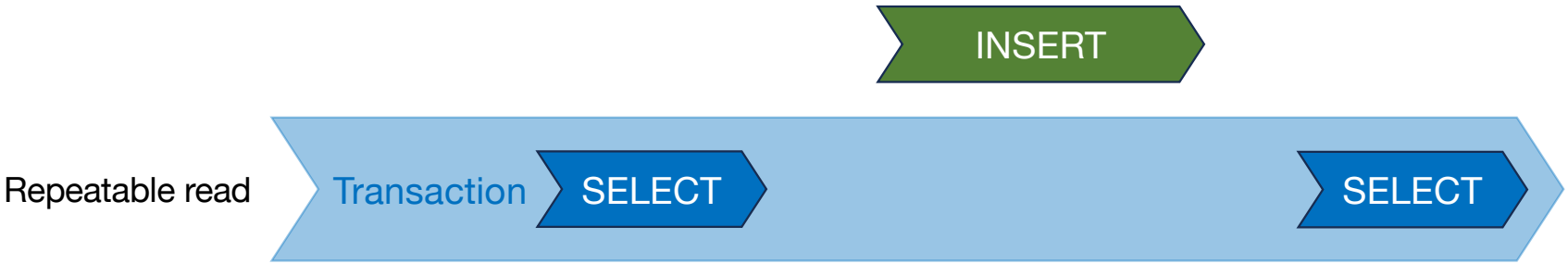
DEMO

Phantom reads

“Where did that come from?”

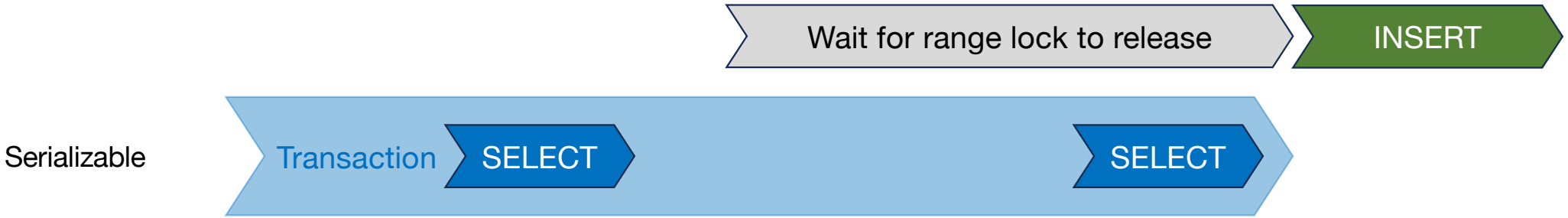
Phantom reads

“Where did that come from?”



Phantom reads

"Where did that come from?"



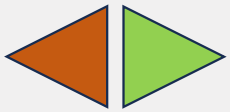
DEMO

What are isolation levels?

What are isolation levels?

Read uncommitted Read committed Repeatable read Serializable

less likely to block, more concurrent



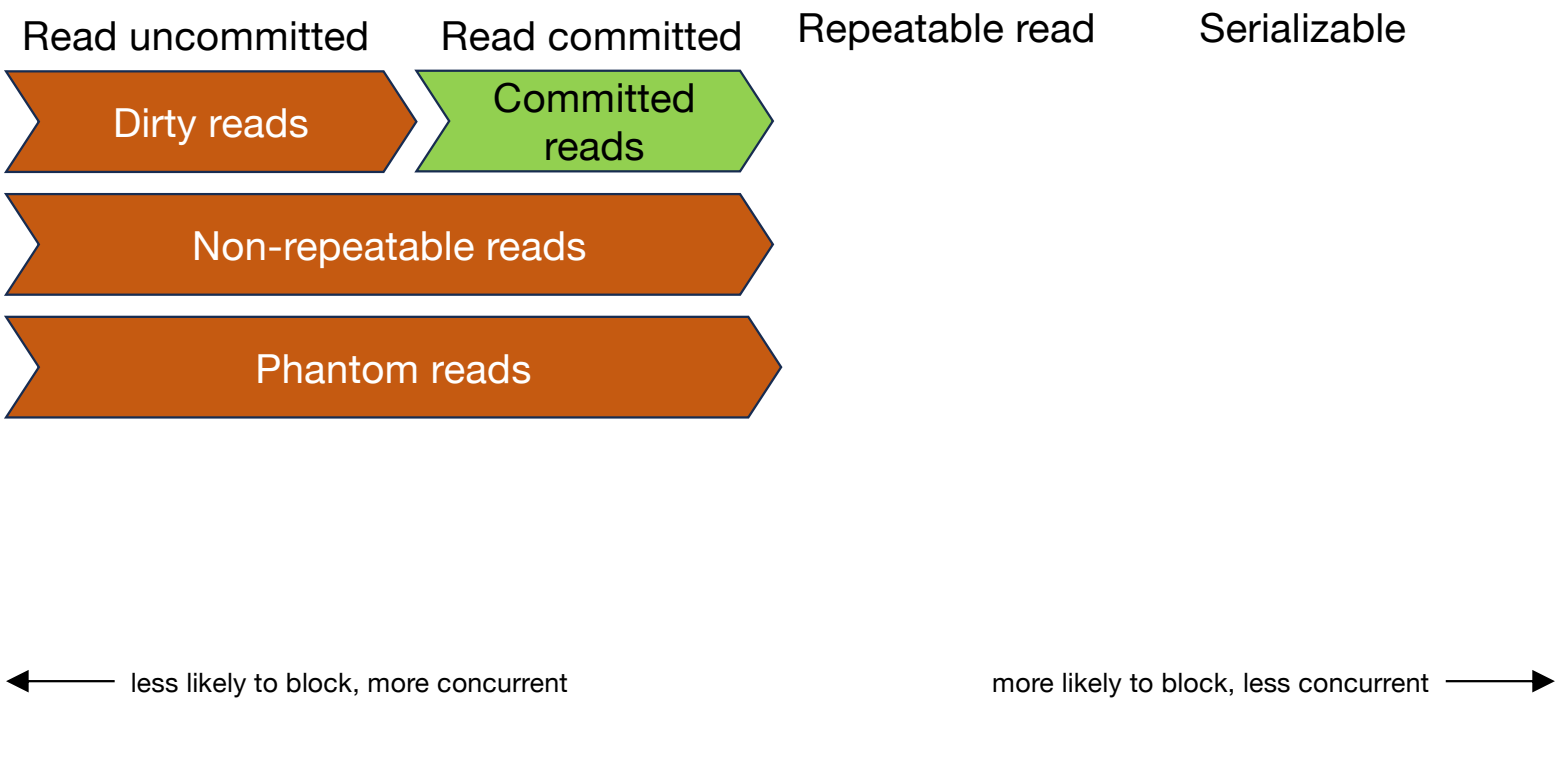
more likely to block, less concurrent

less likely to block, more concurrent

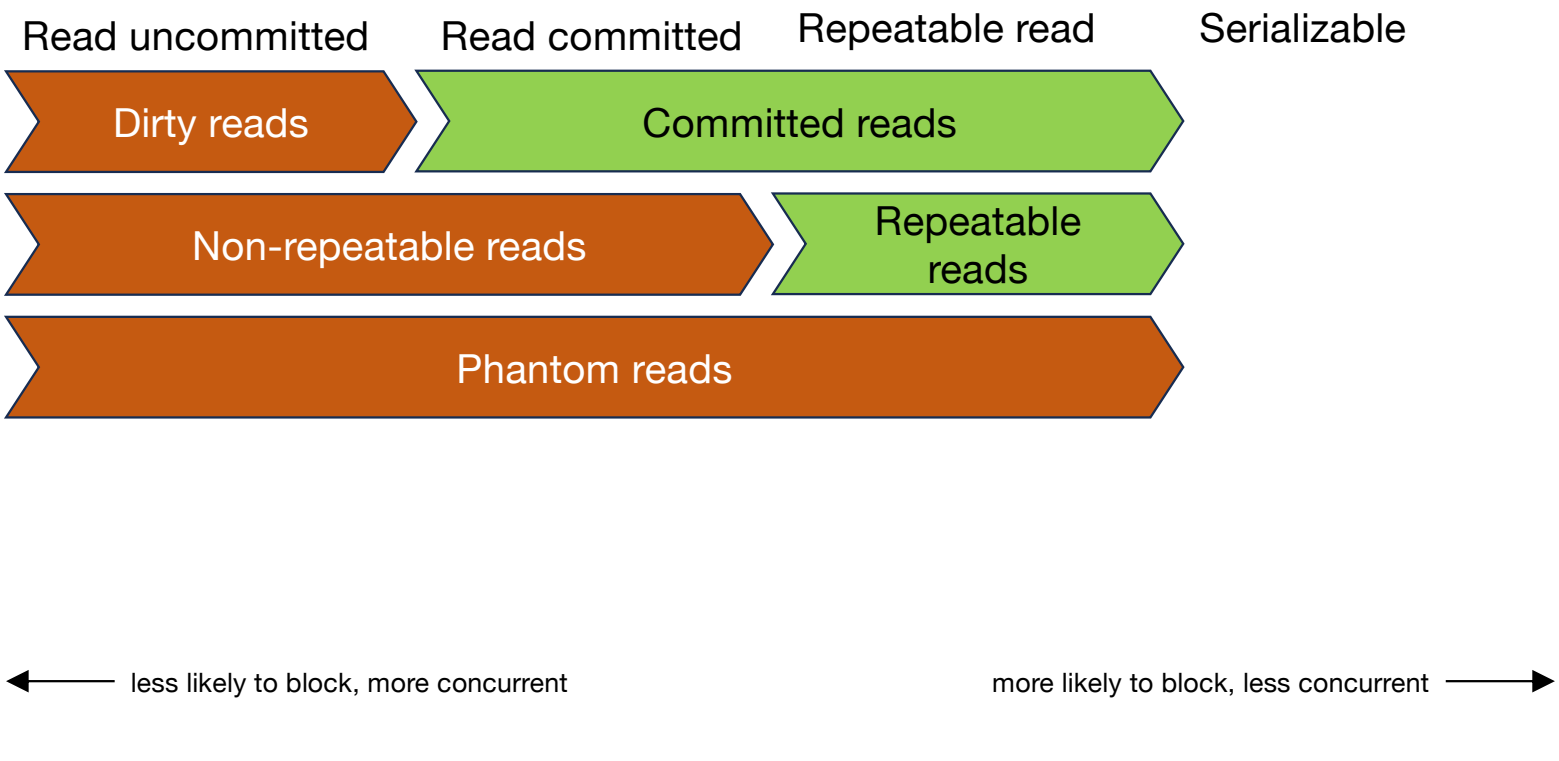
more likely to block, less concurrent

What are isolation levels?

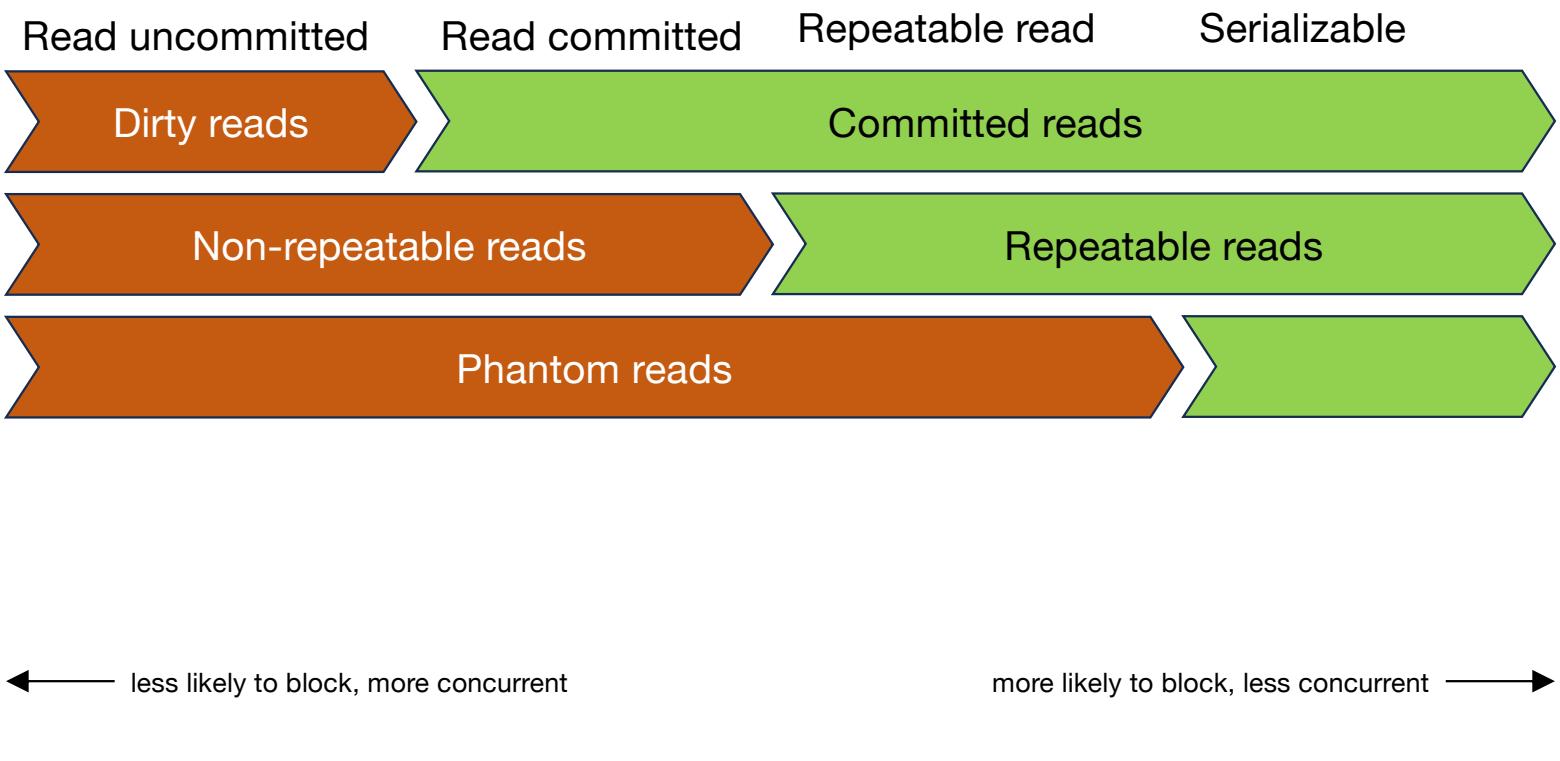
What are isolation levels?



What are isolation levels?

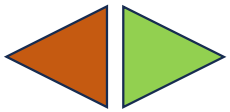


What are isolation levels?



So… Serialize all the things, then?

Less locking



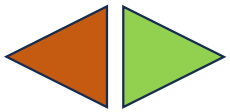
More locking

- Less isolation
- Less predictable

- Better isolation
- More predictable

So… Serialize all the things, then?

Less locking



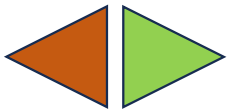
More locking

- Less isolation
- Less predictable
- Better concurrency

- Better isolation
- More predictable
- Lower concurrency

So… Serialize all the things, then?

Less locking

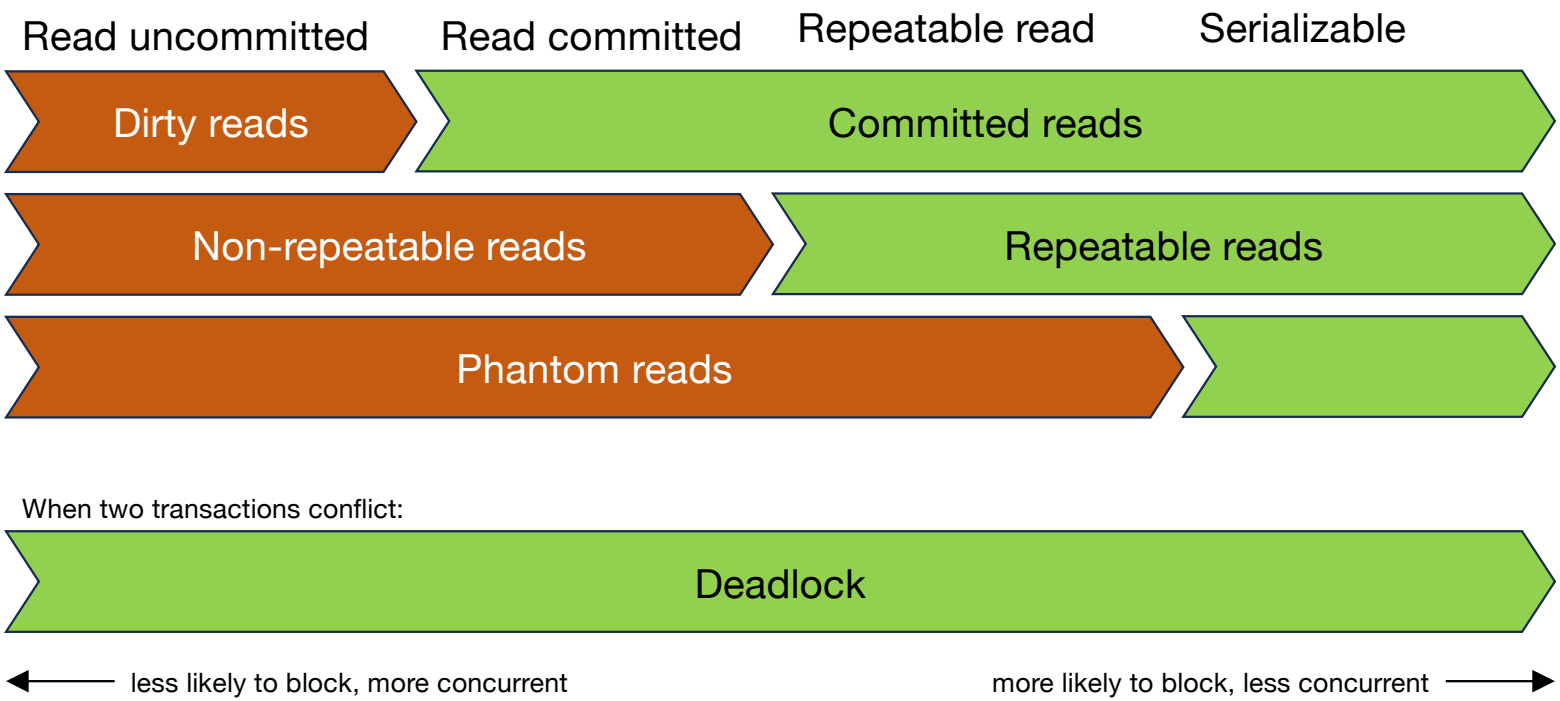


More locking

- Less isolation
- Less predictable
- Better concurrency
- Fewer conflicts

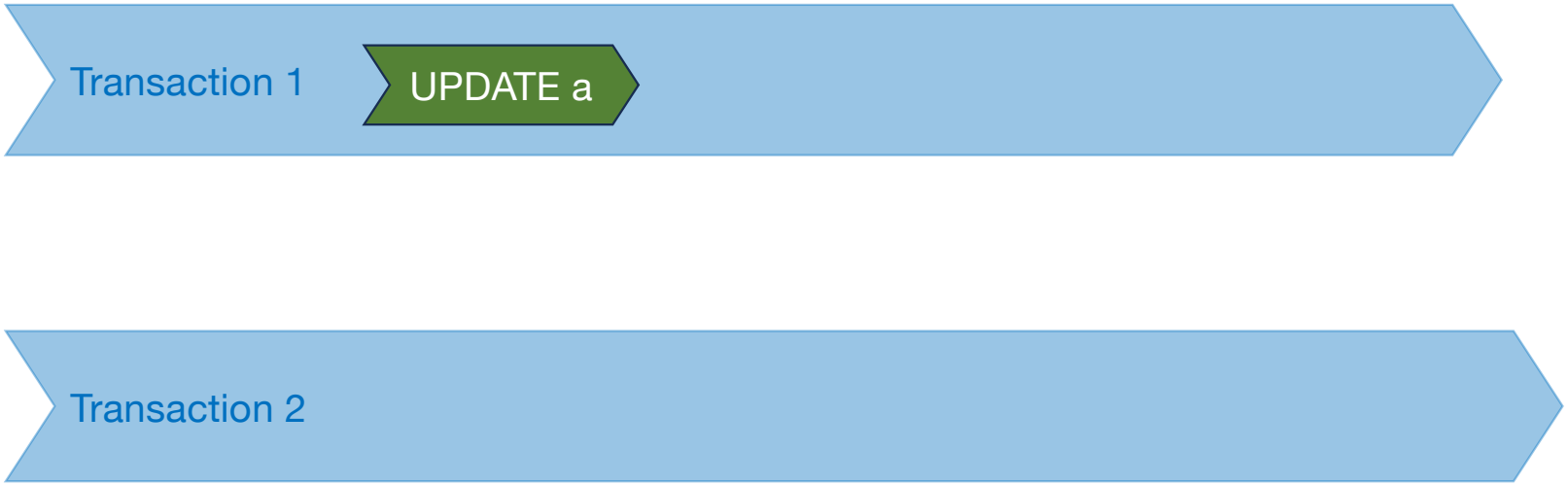
- Better isolation
- More predictable
- Lower concurrency
- More conflicts

What are isolation levels?

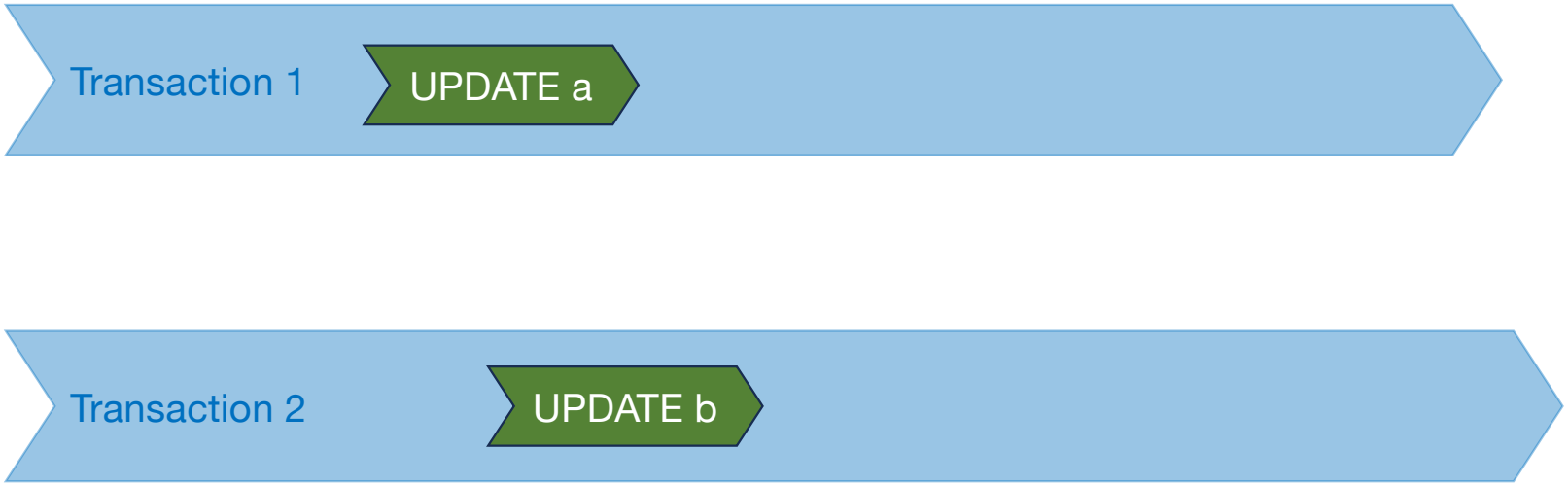


Deadlock

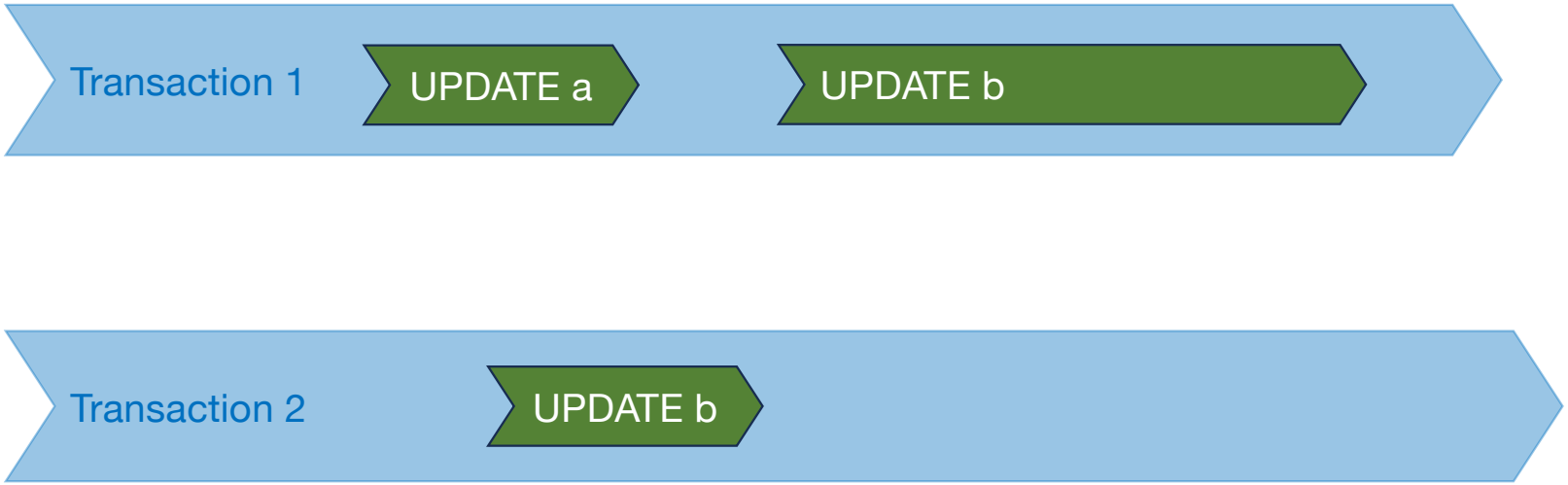
Deadlock



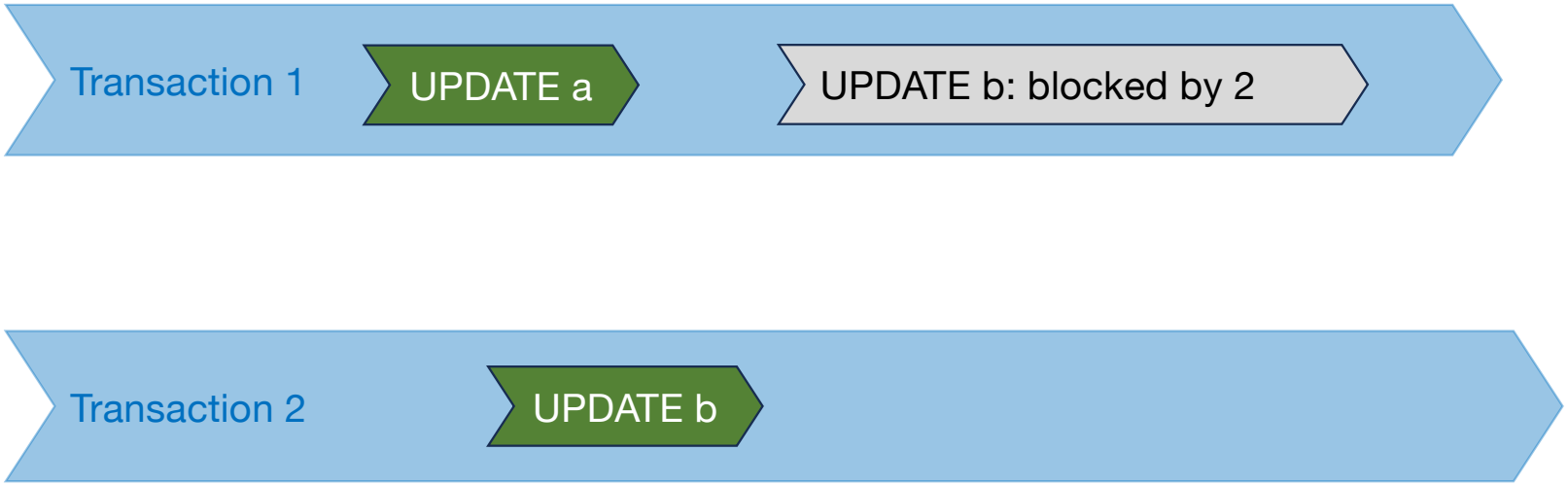
Deadlock



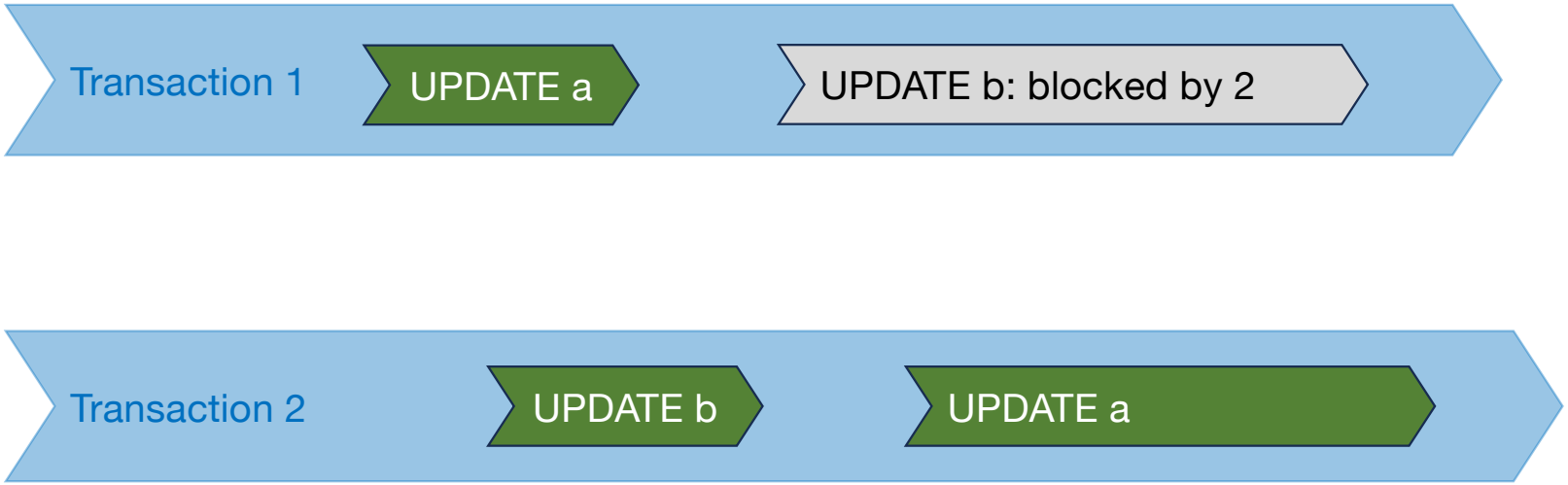
Deadlock



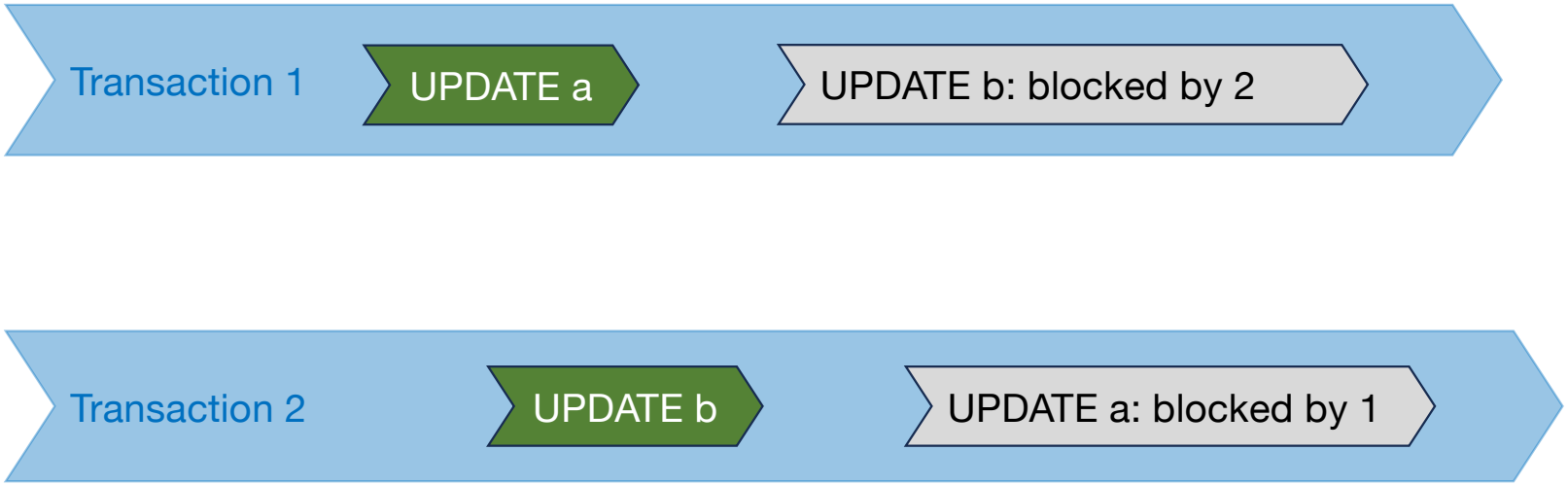
Deadlock


Transaction 1
UPDATE a
UPDATE b: blocked by 2
Transaction 2
UPDATE b

Deadlock


Transaction 1
UPDATE a
UPDATE b: blocked by 2
Transaction 2
UPDATE b
UPDATE a

Deadlock

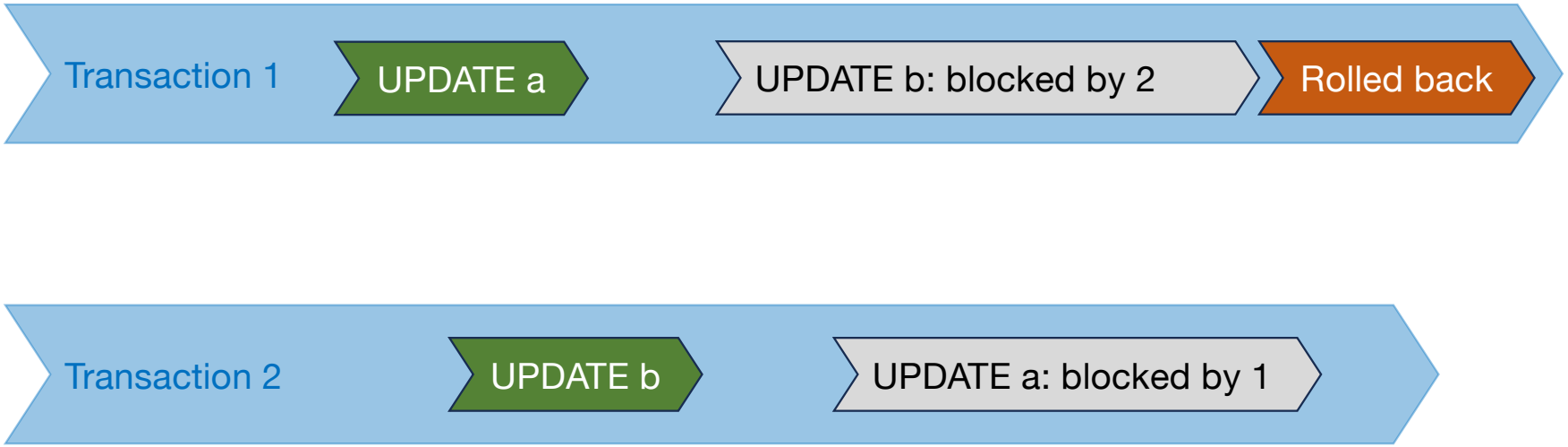

Transaction 1
UPDATE a
UPDATE b: blocked by 2
Transaction 2
UPDATE b
UPDATE a: blocked by 1

Deadlock

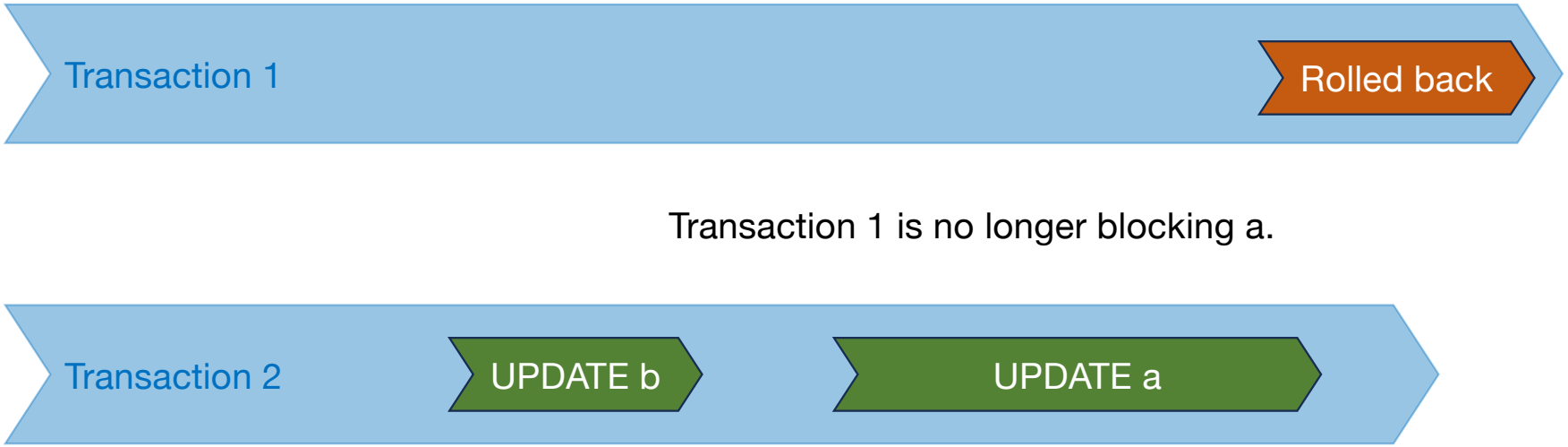
Deadlock: Transaction 1 and 2 are waiting on each other.

Deadlock

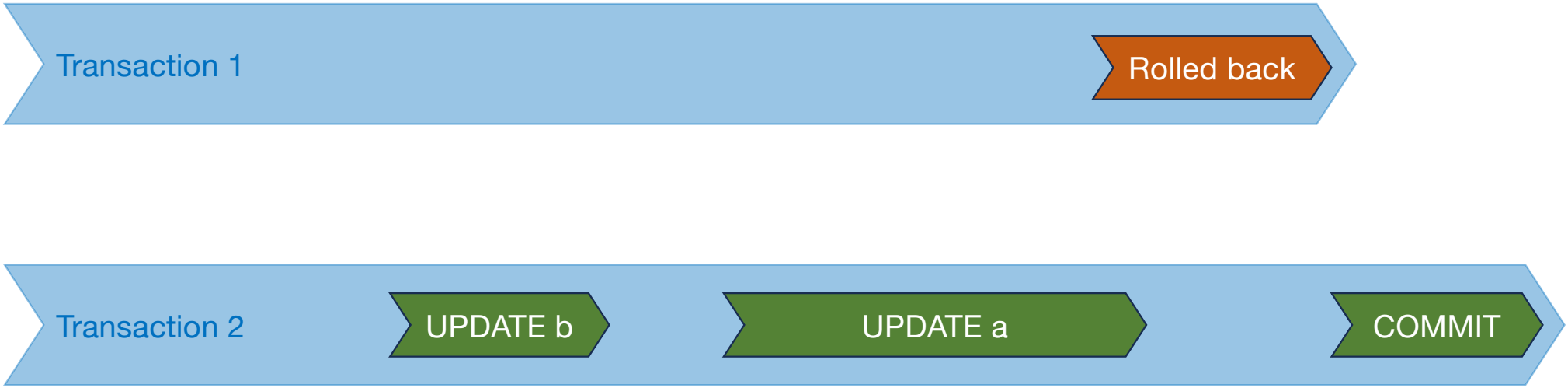
SQL Server will fail the transaction with the least amount of work to roll back:



Deadlock

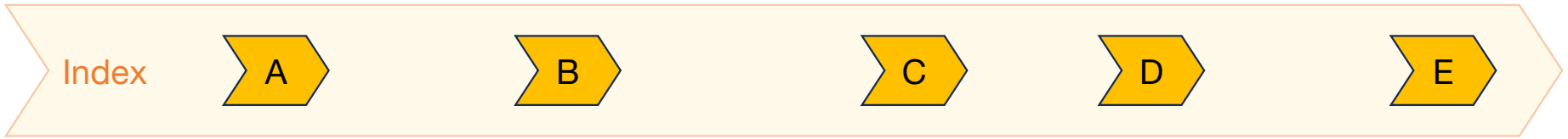

Transaction 1
Rolled back
Transaction 1 is no longer blocking a.
Transaction 2
UPDATE b
UPDATE a

Deadlock

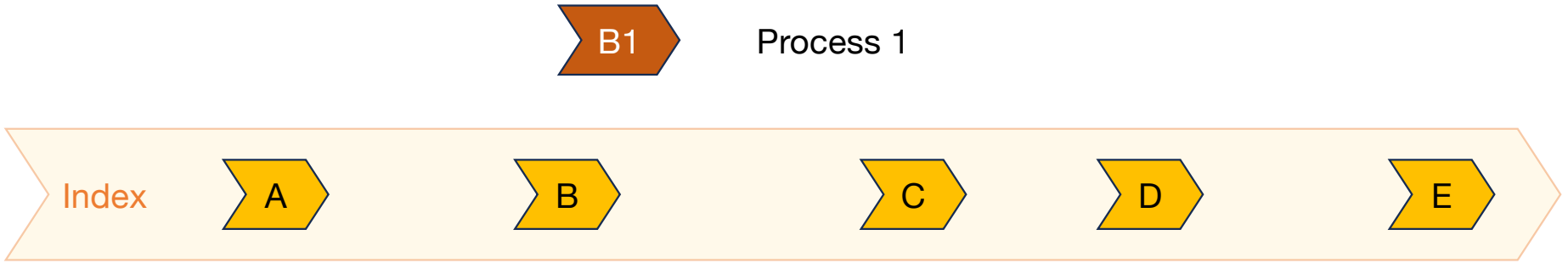

Transaction 1
Rolled back
Transaction 2
UPDATE b
UPDATE a
COMMIT

DEMO

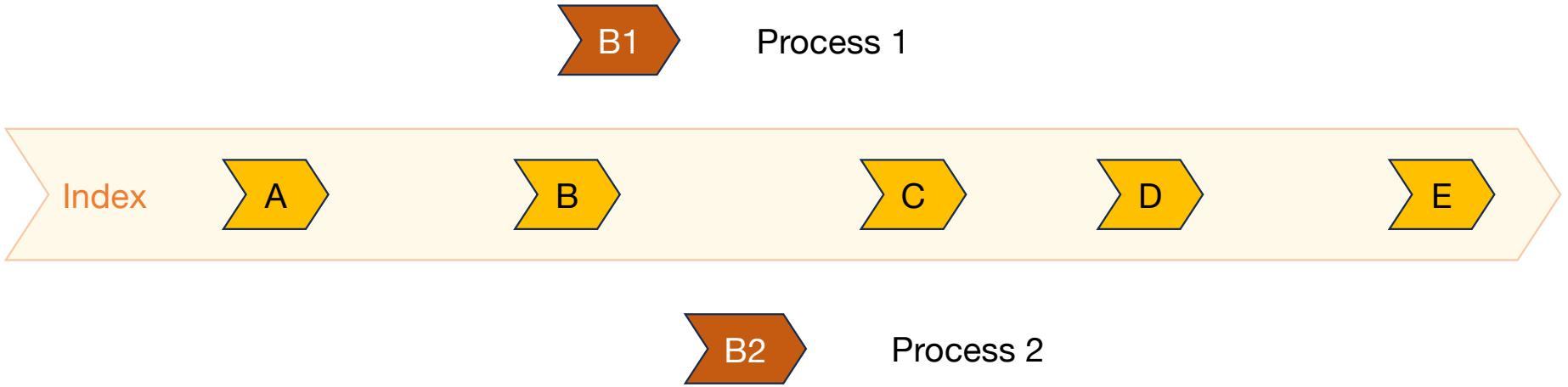
The serializable merge deadlock



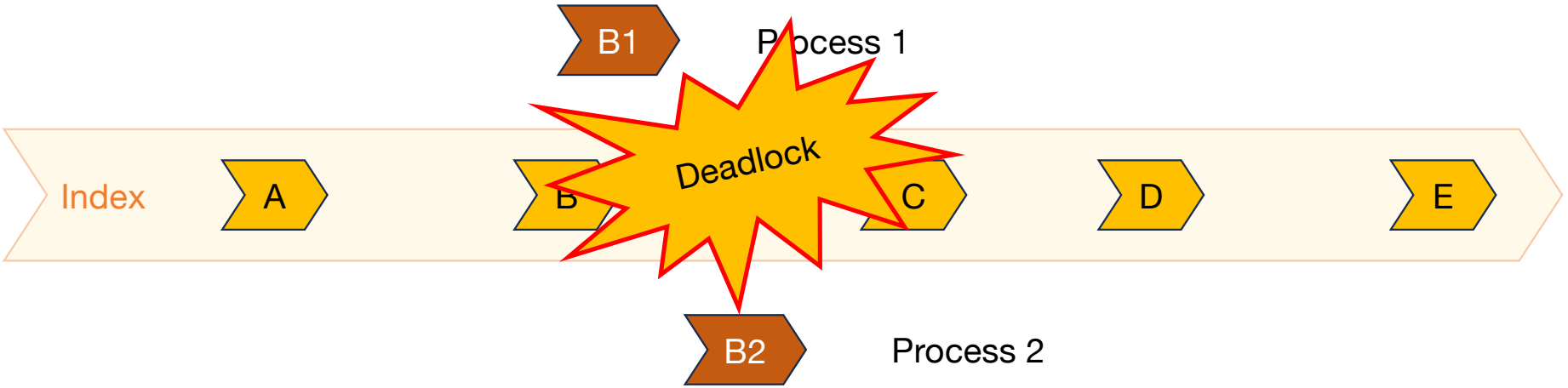
The serializable merge deadlock



The serializable merge deadlock

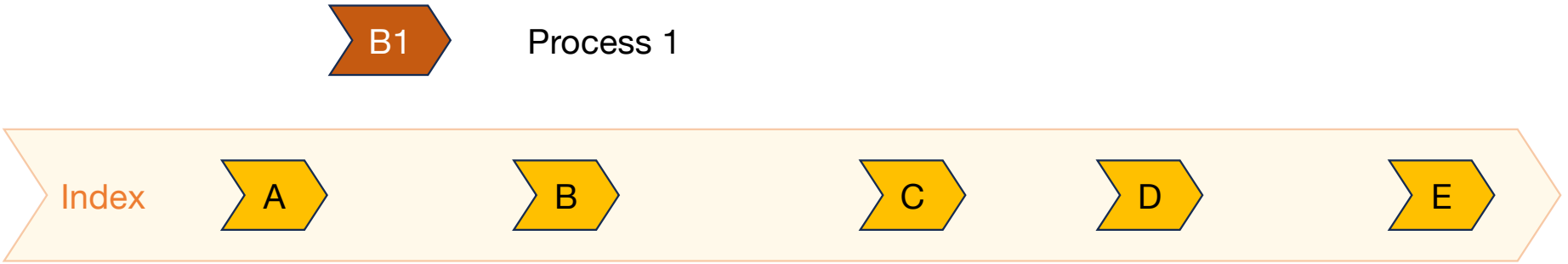


The serializable merge deadlock

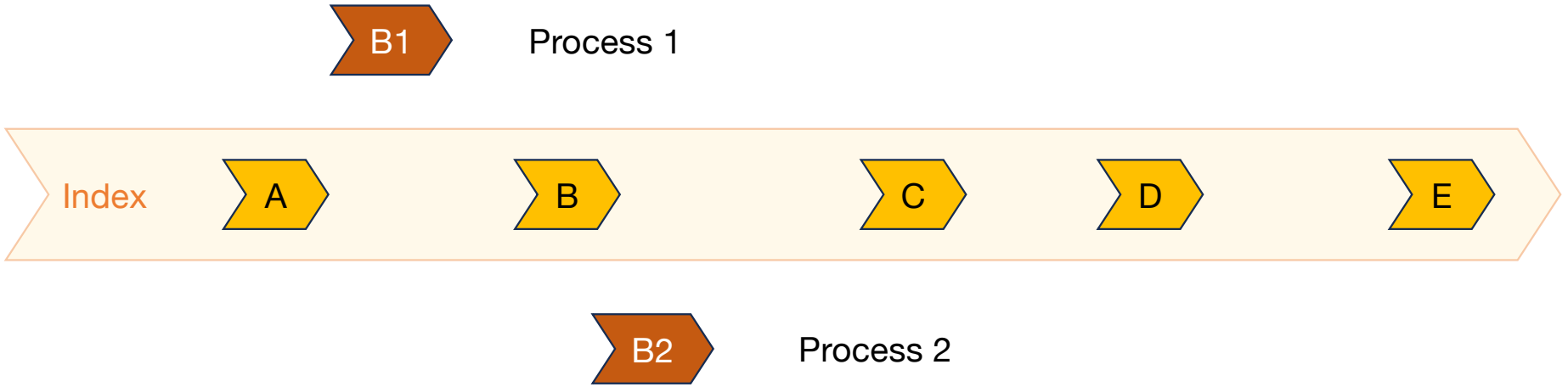


DEMO

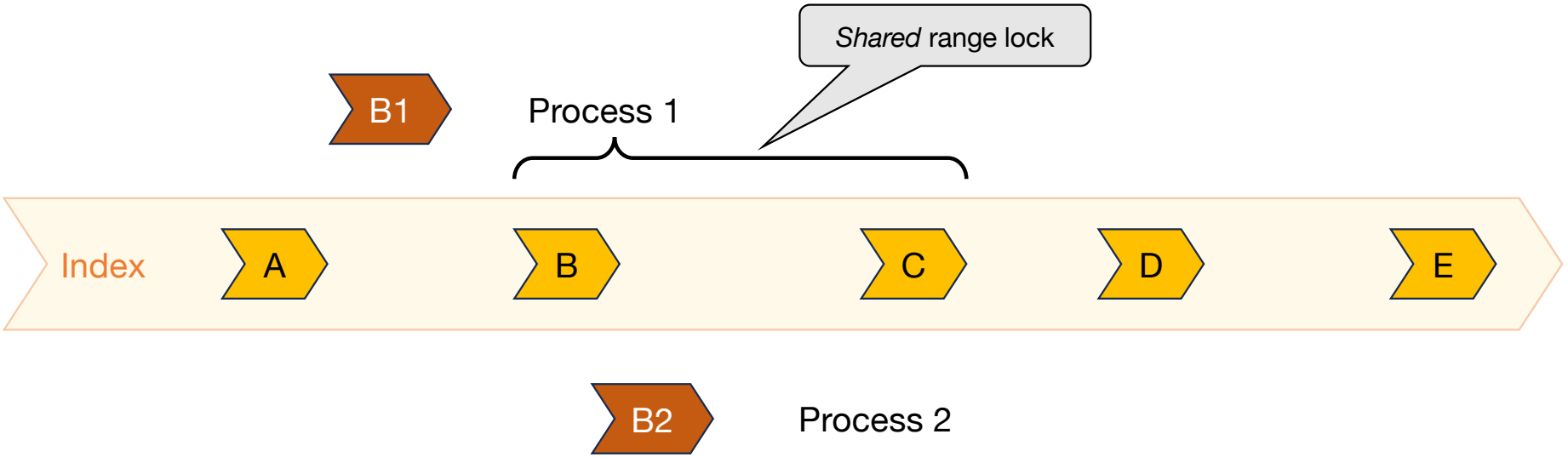
The serializable merge deadlock



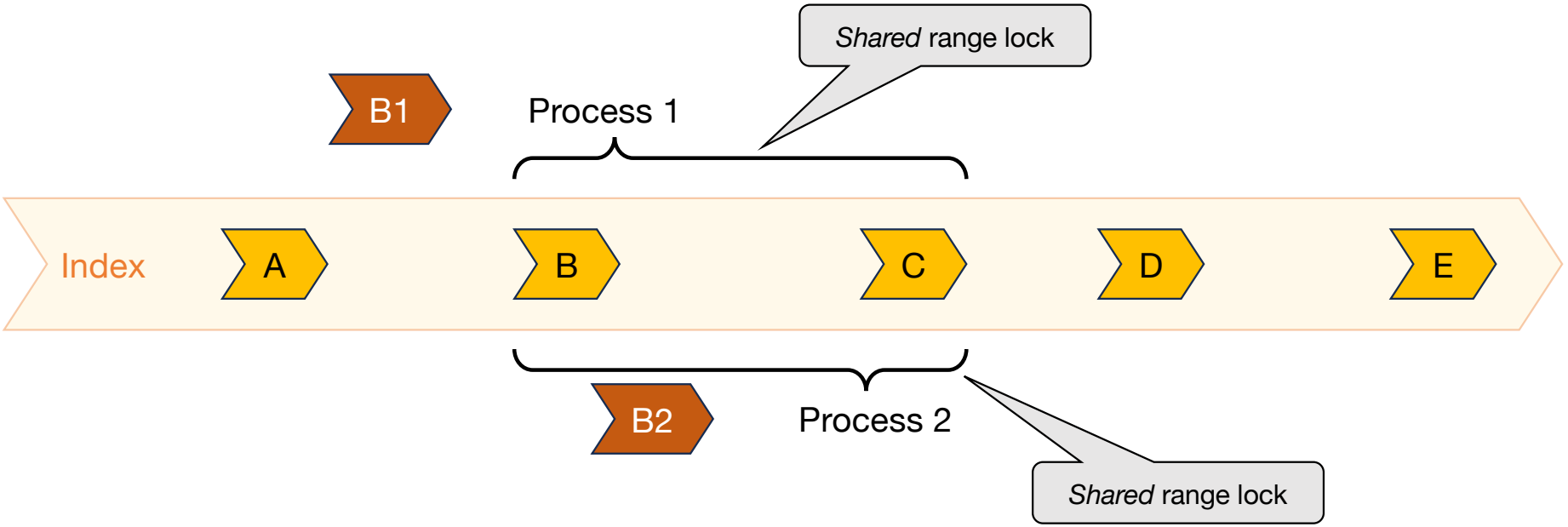
The serializable merge deadlock



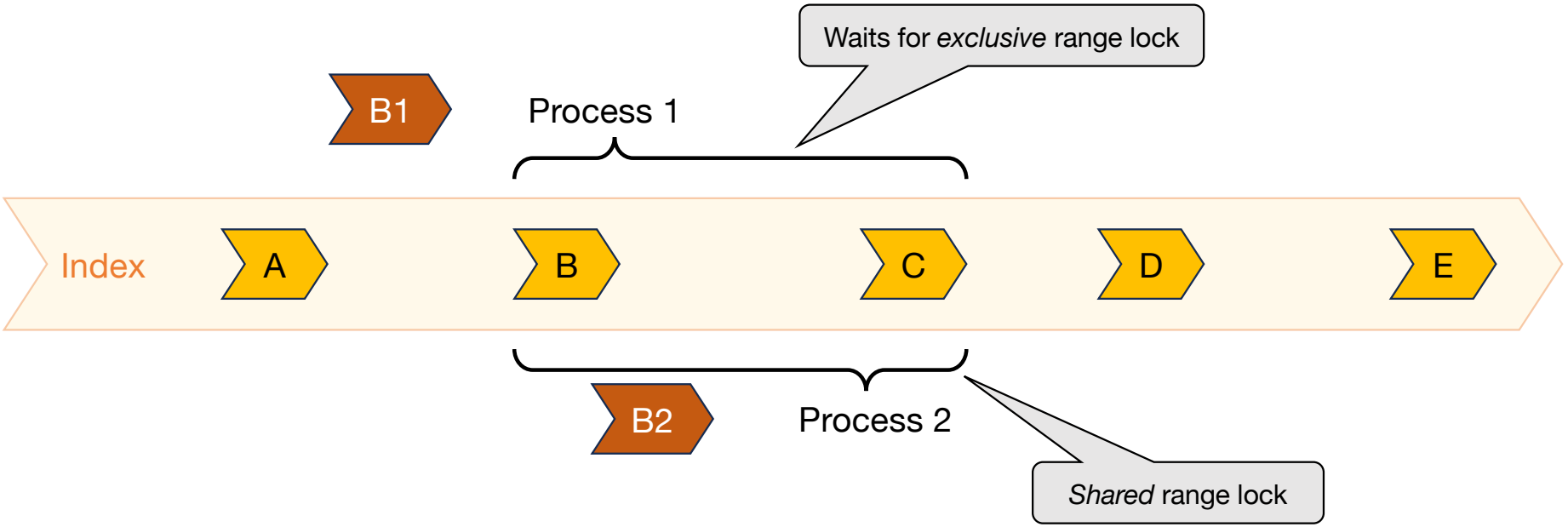
The serializable merge deadlock



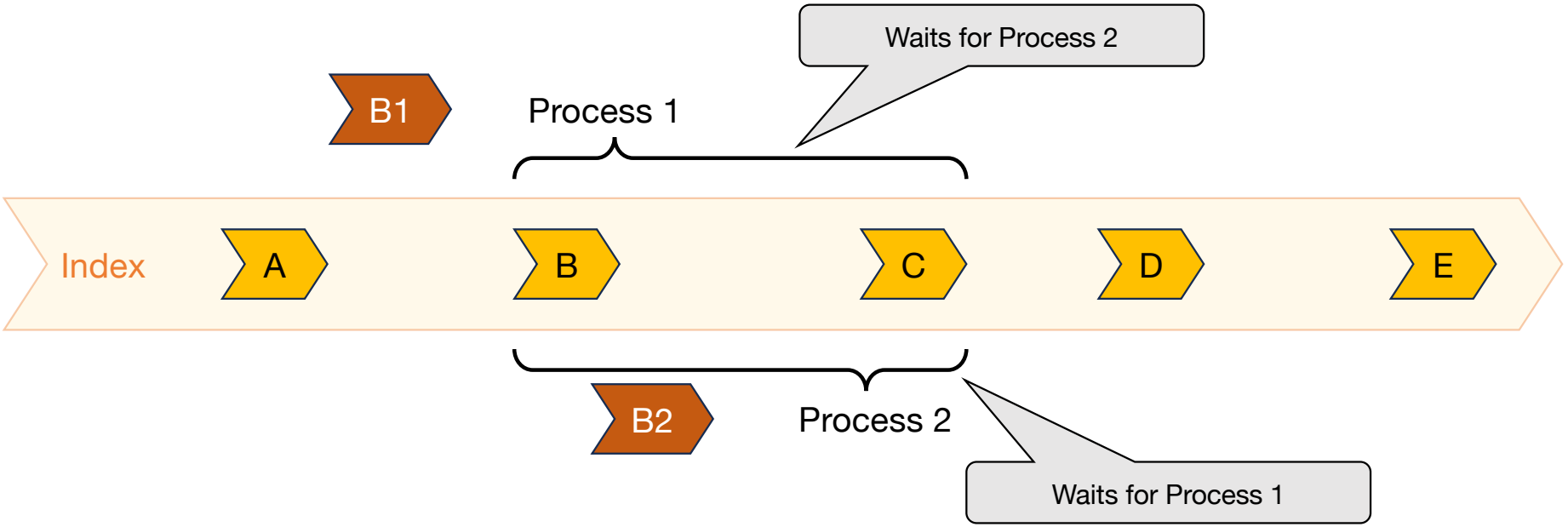
The serializable merge deadlock



The serializable merge deadlock

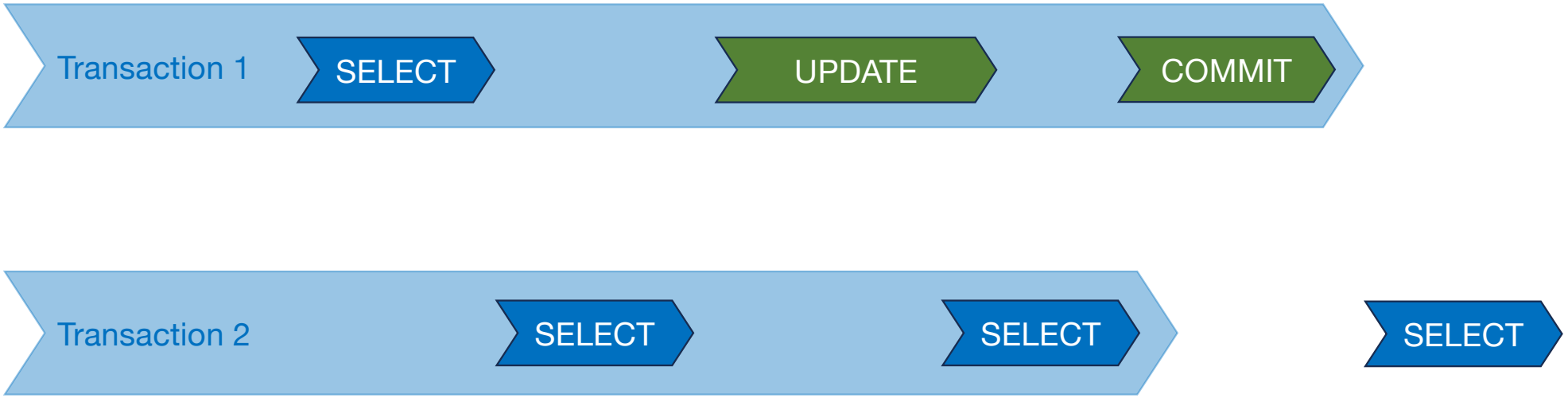


The serializable merge deadlock

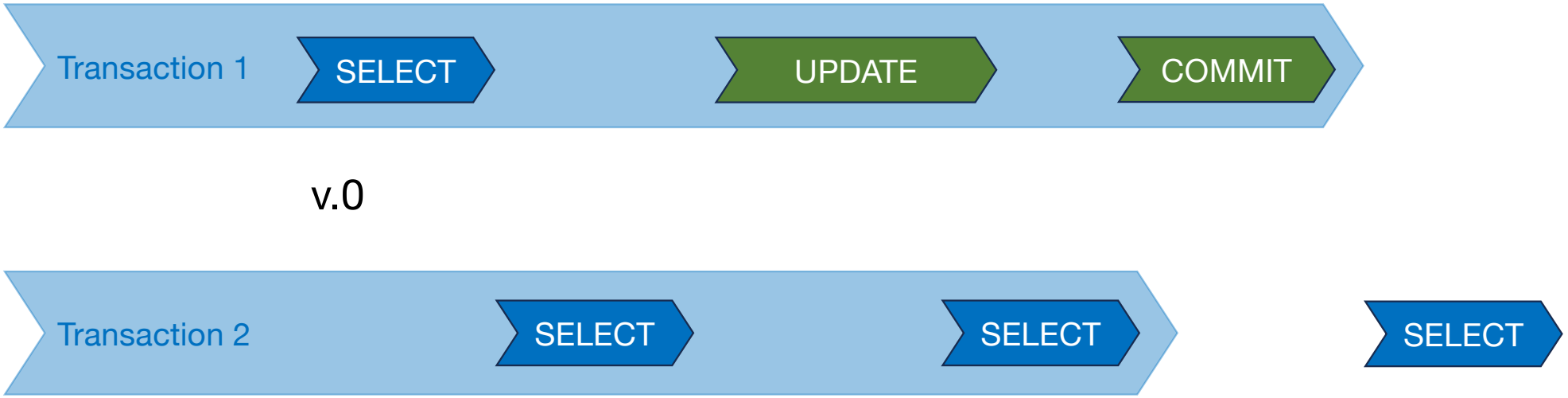


Snapshot isolation

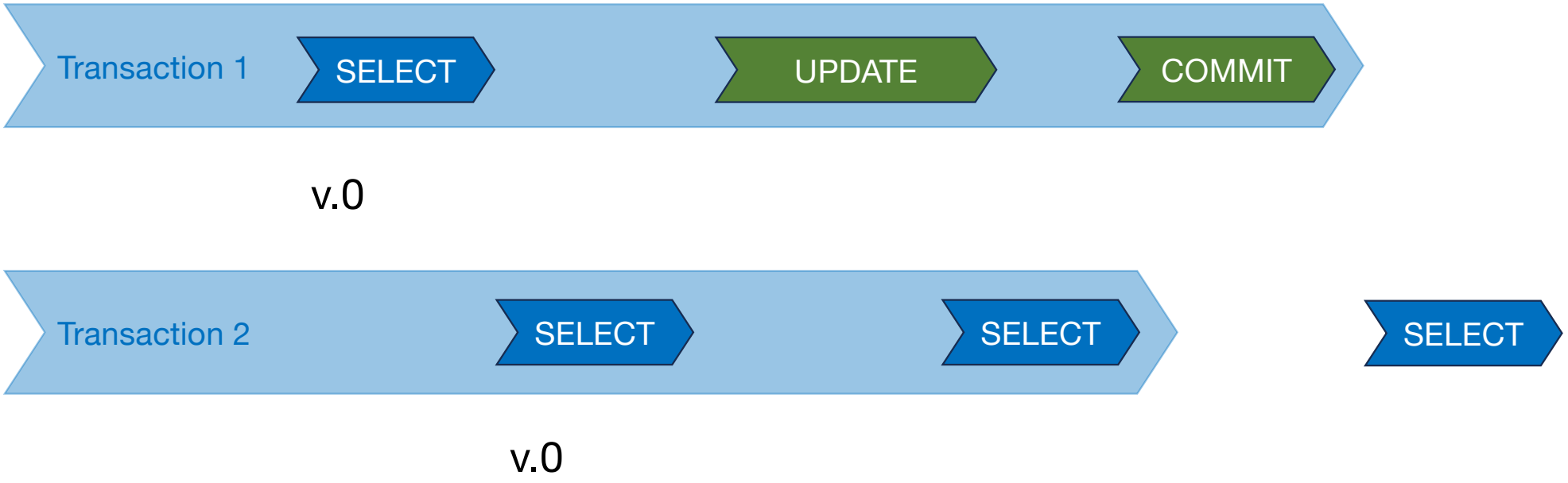
Snapshot isolation



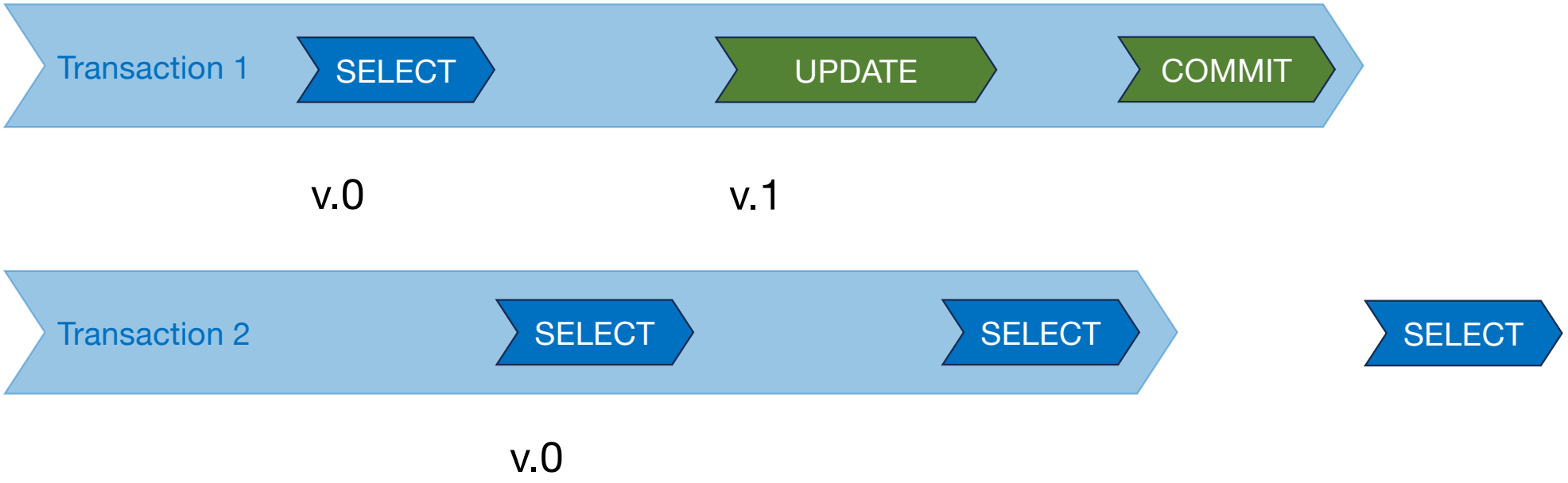
Snapshot isolation



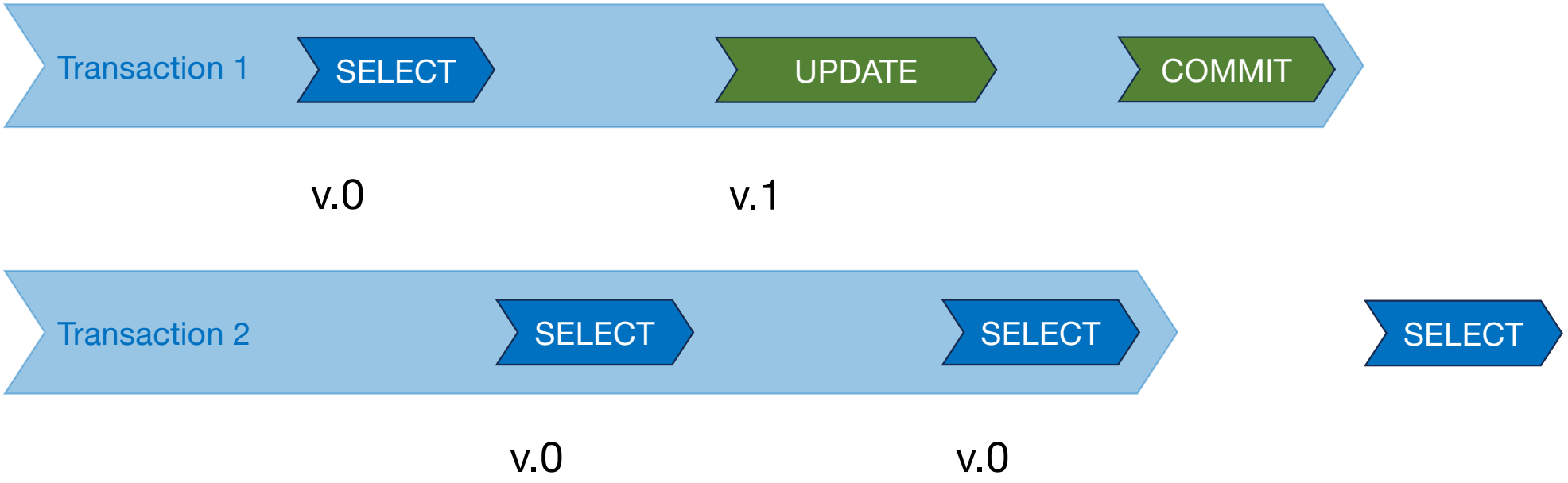
Snapshot isolation



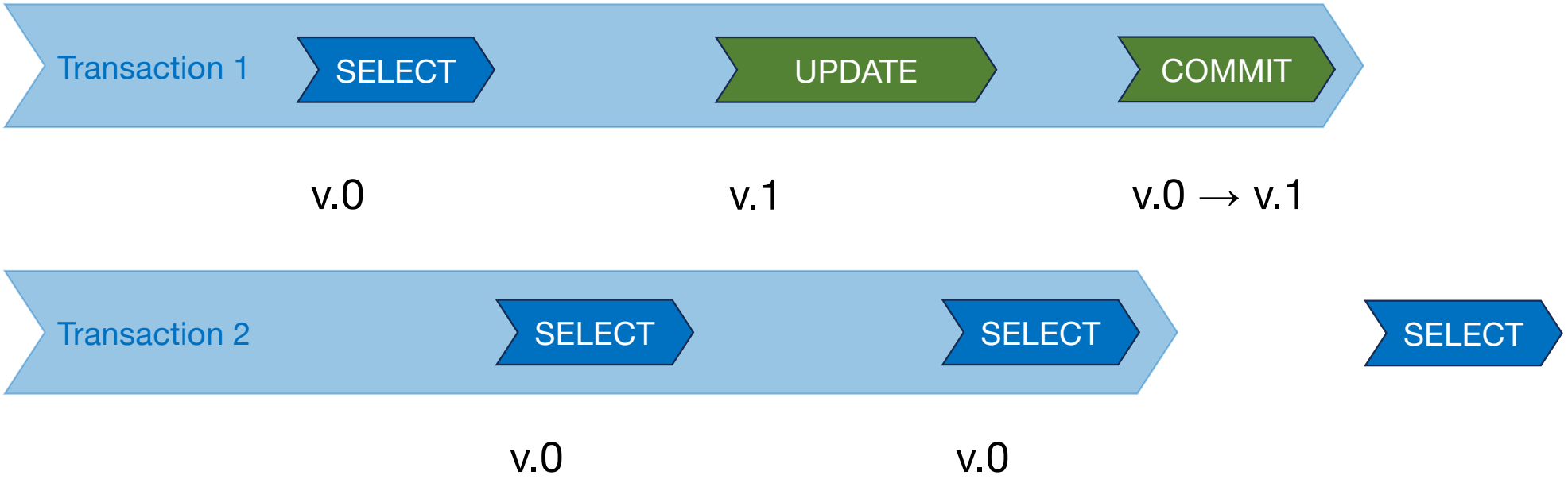
Snapshot isolation



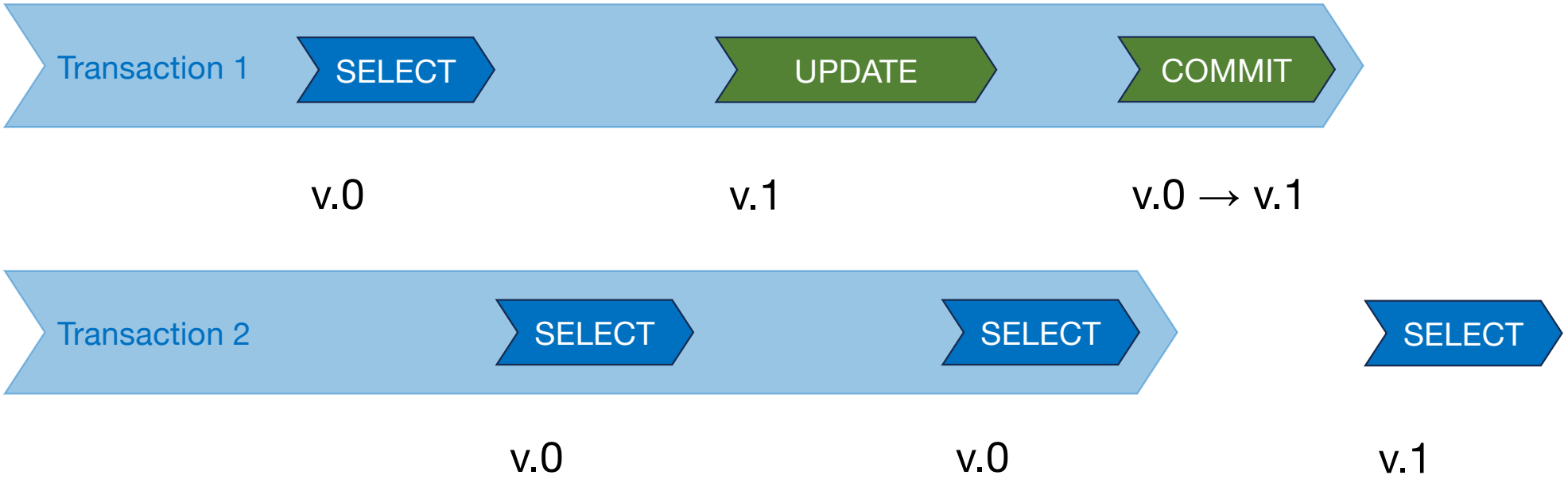
Snapshot isolation



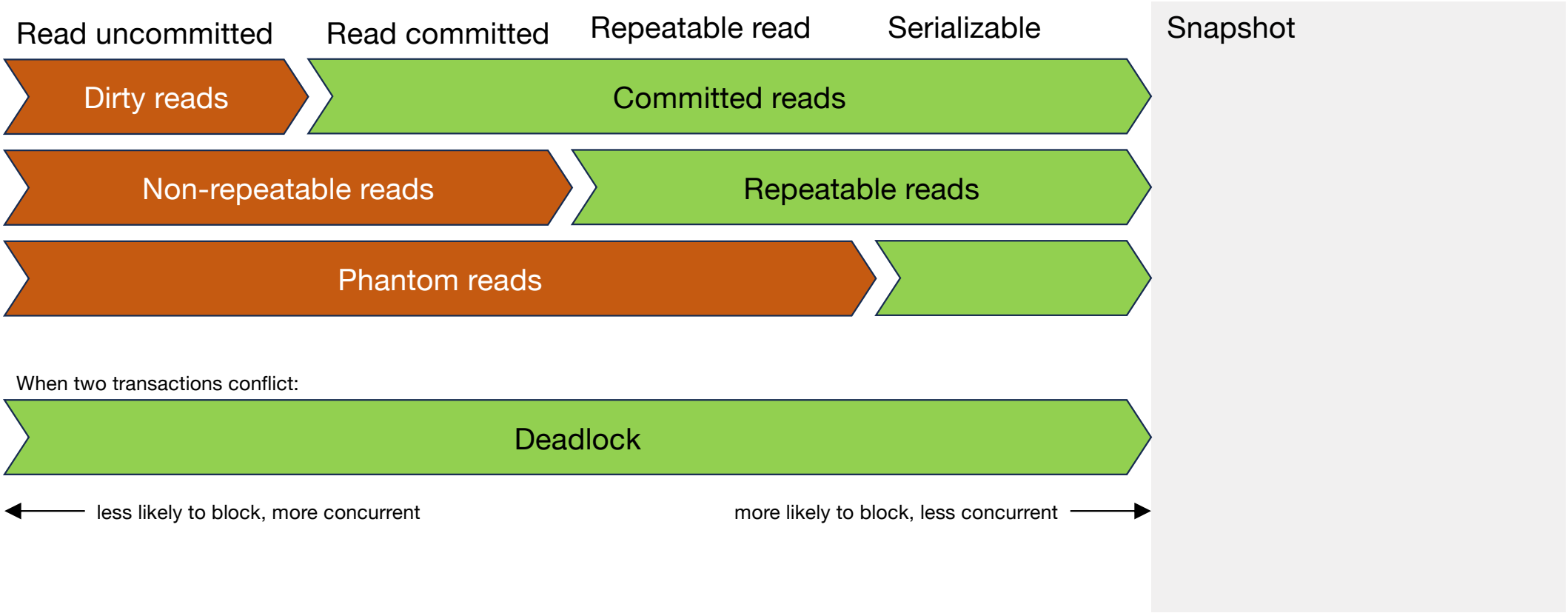
Snapshot isolation



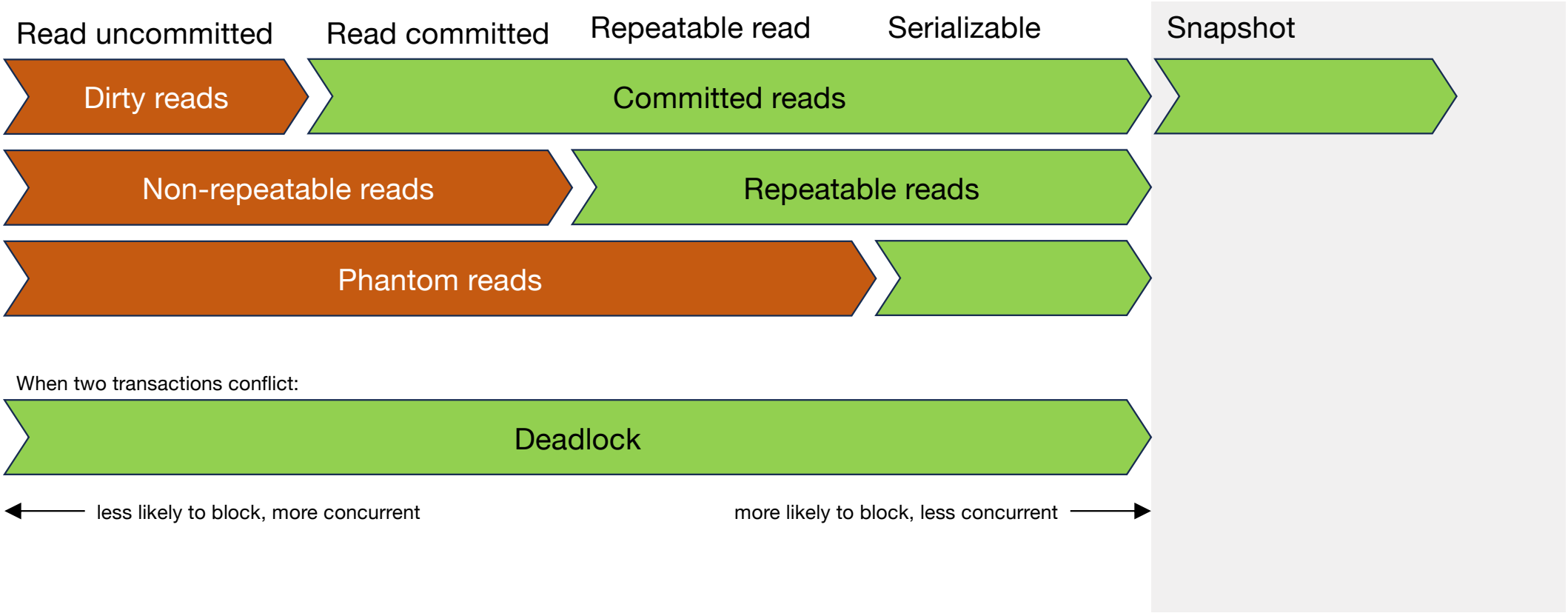
Snapshot isolation



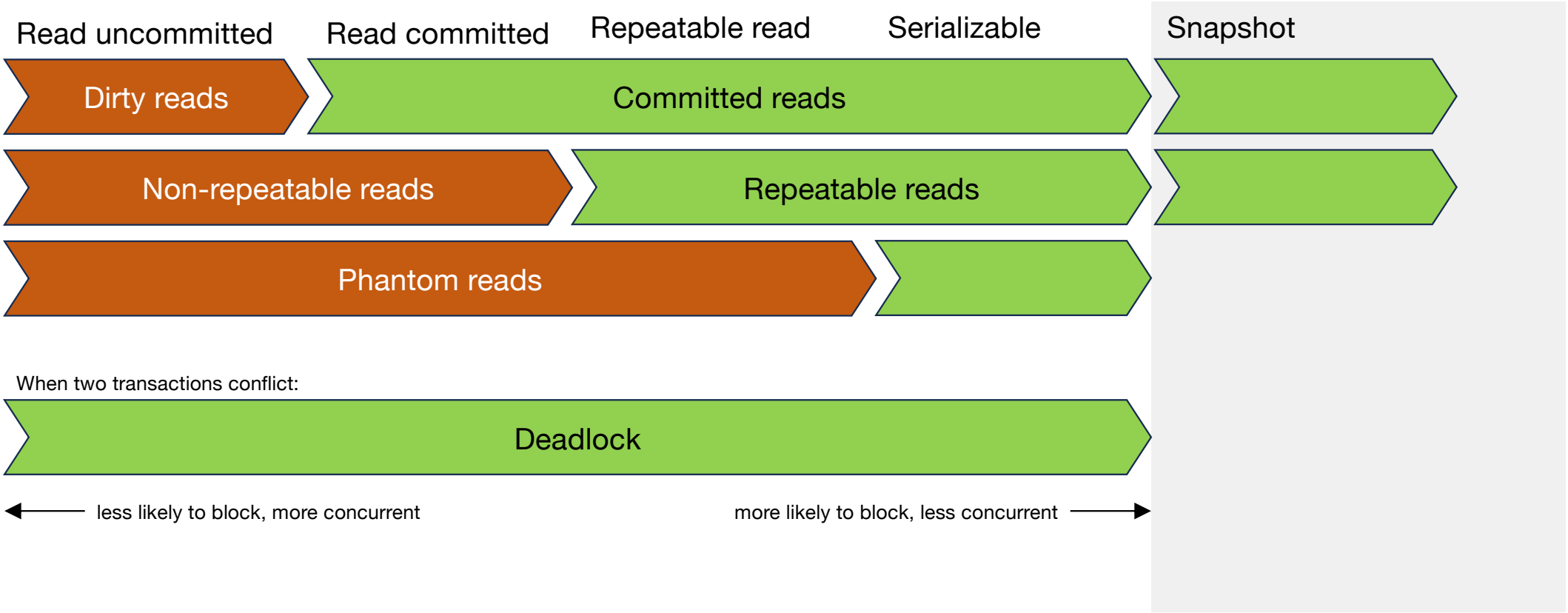
Snapshot isolation



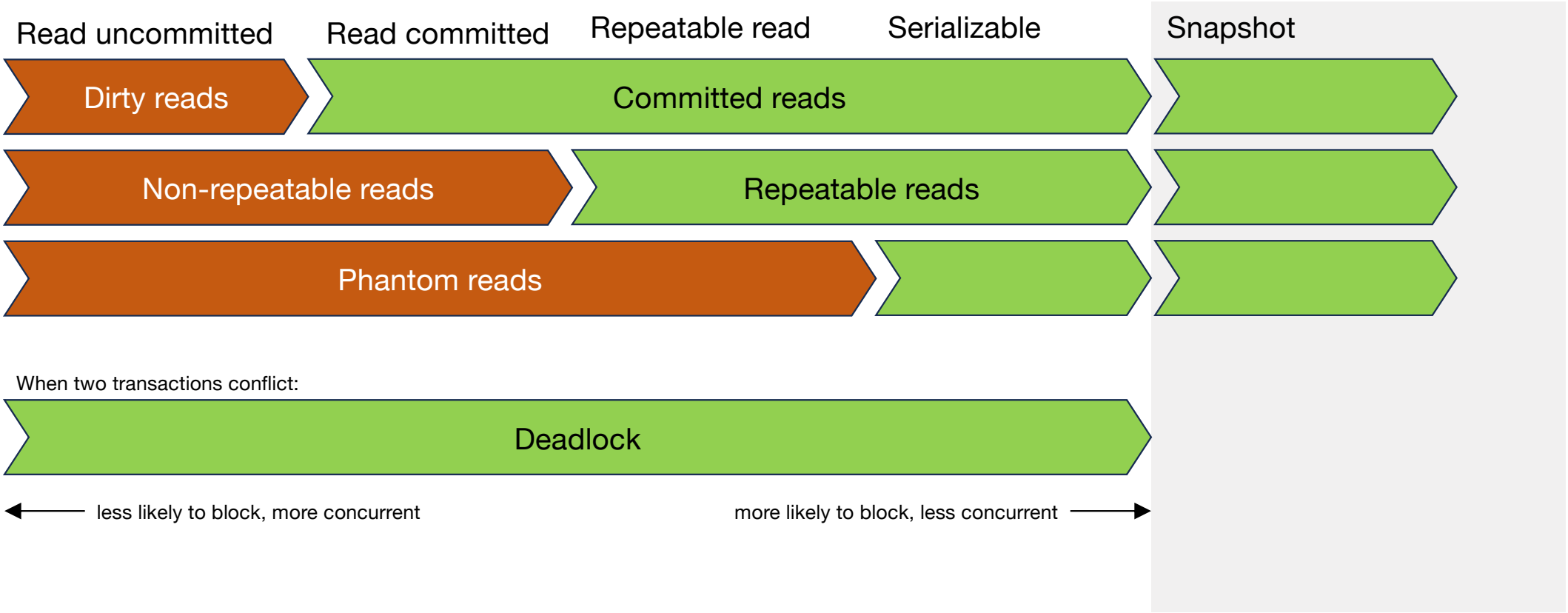
Snapshot isolation



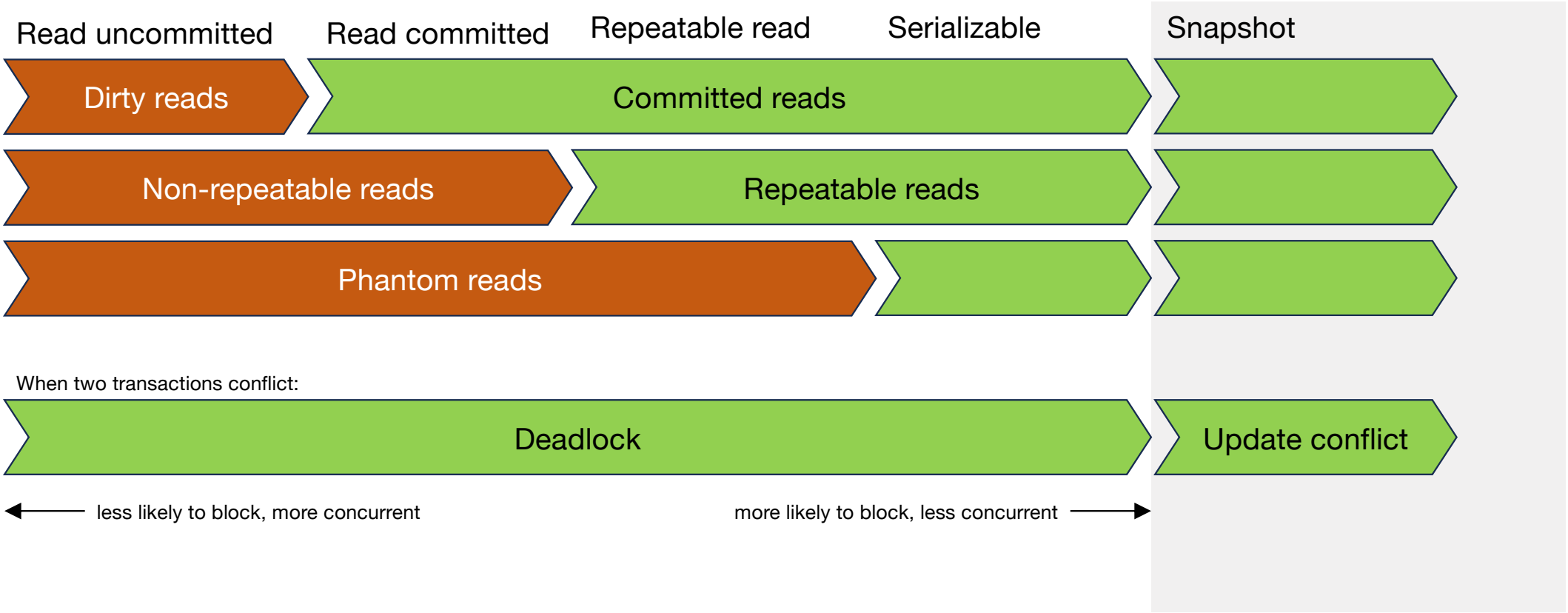
Snapshot isolation



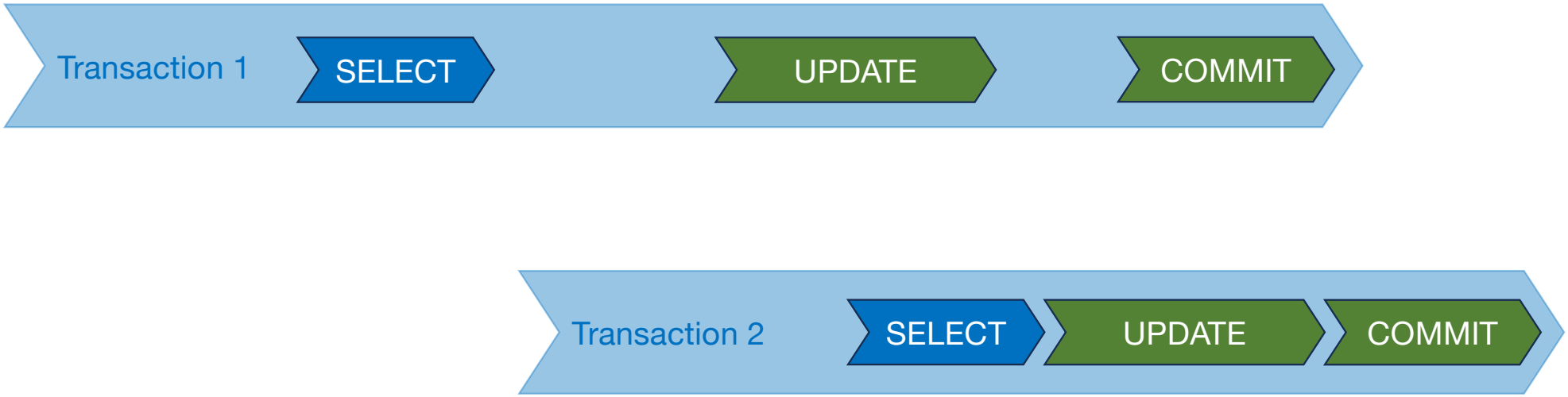
Snapshot isolation



Snapshot isolation

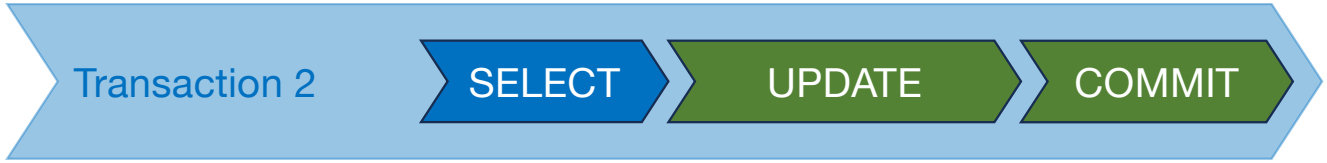


Snapshot isolation: Update conflict

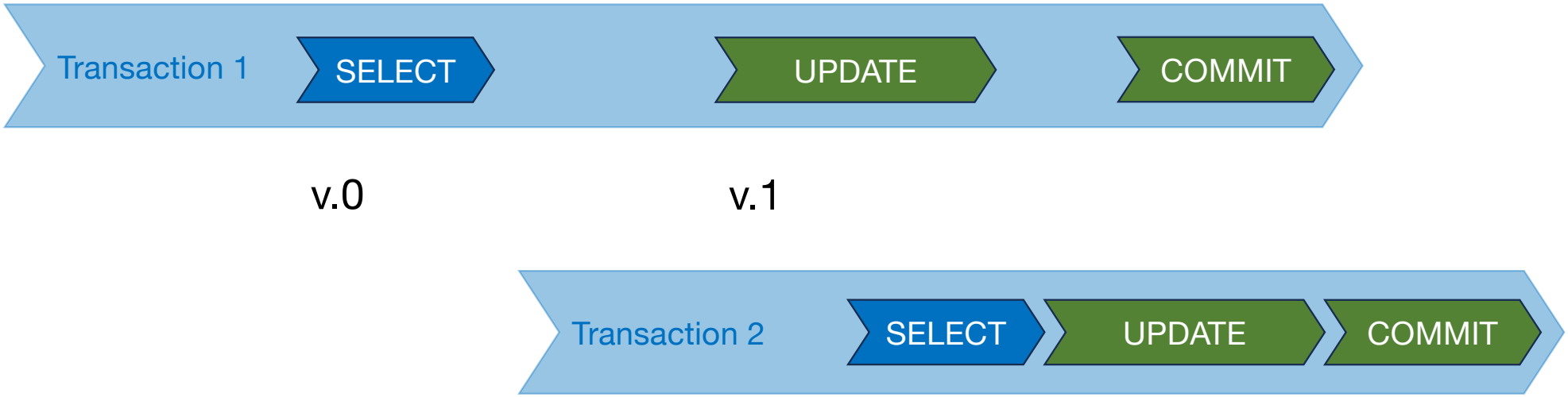


Snapshot isolation: Update conflict

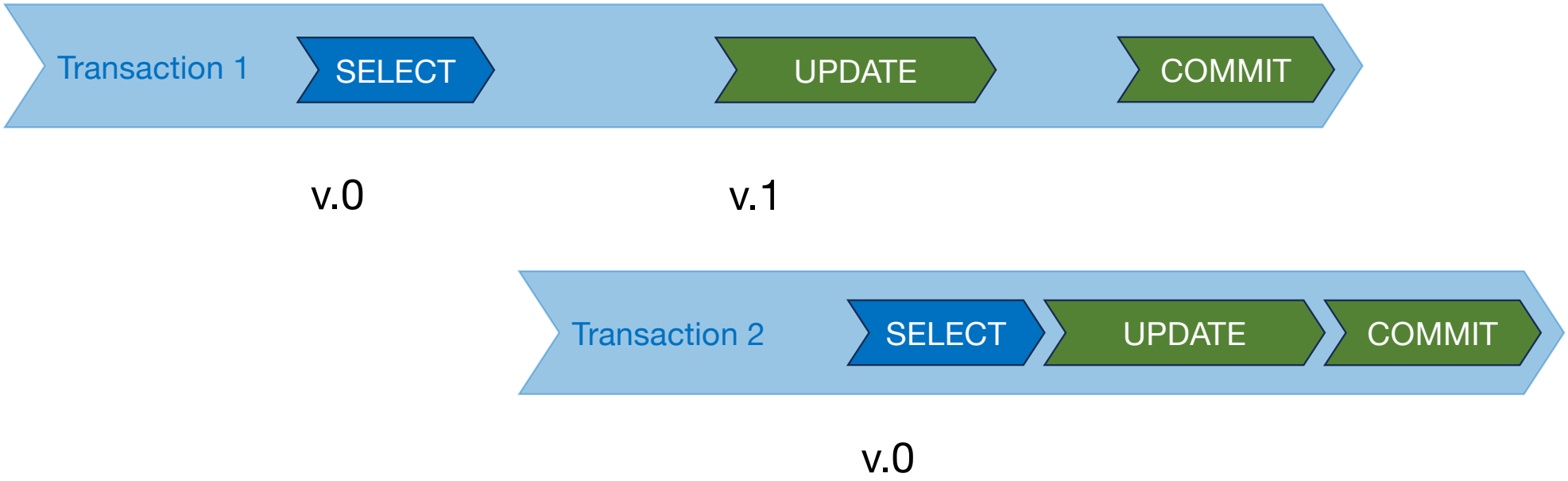
v.0



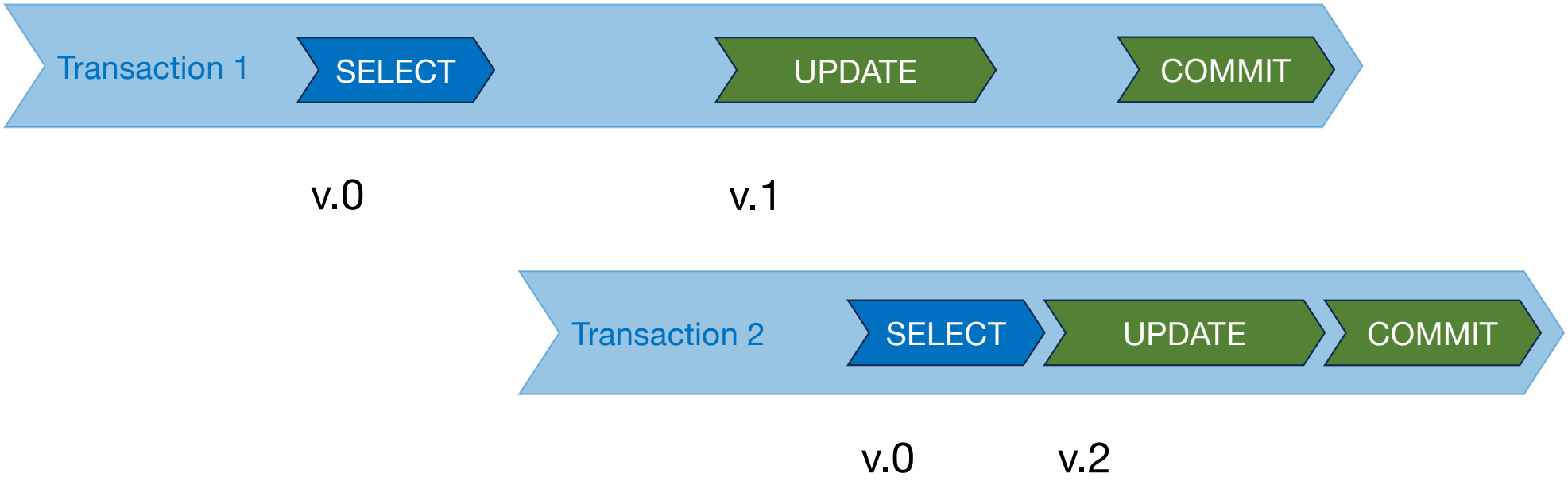
Snapshot isolation: Update conflict



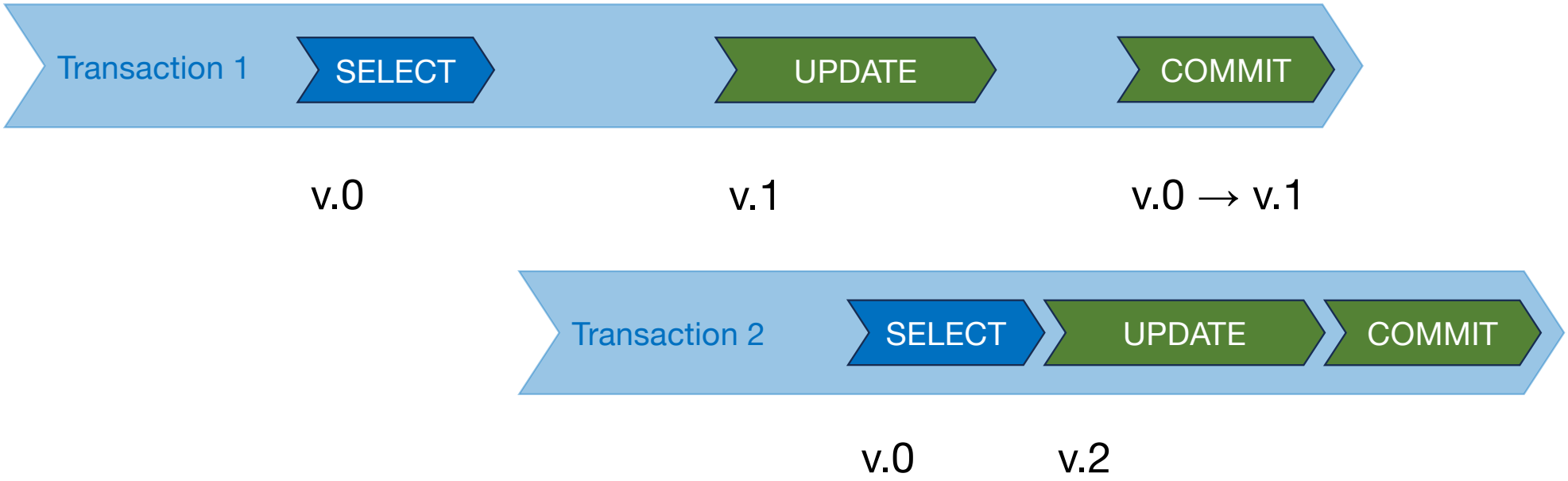
Snapshot isolation: Update conflict



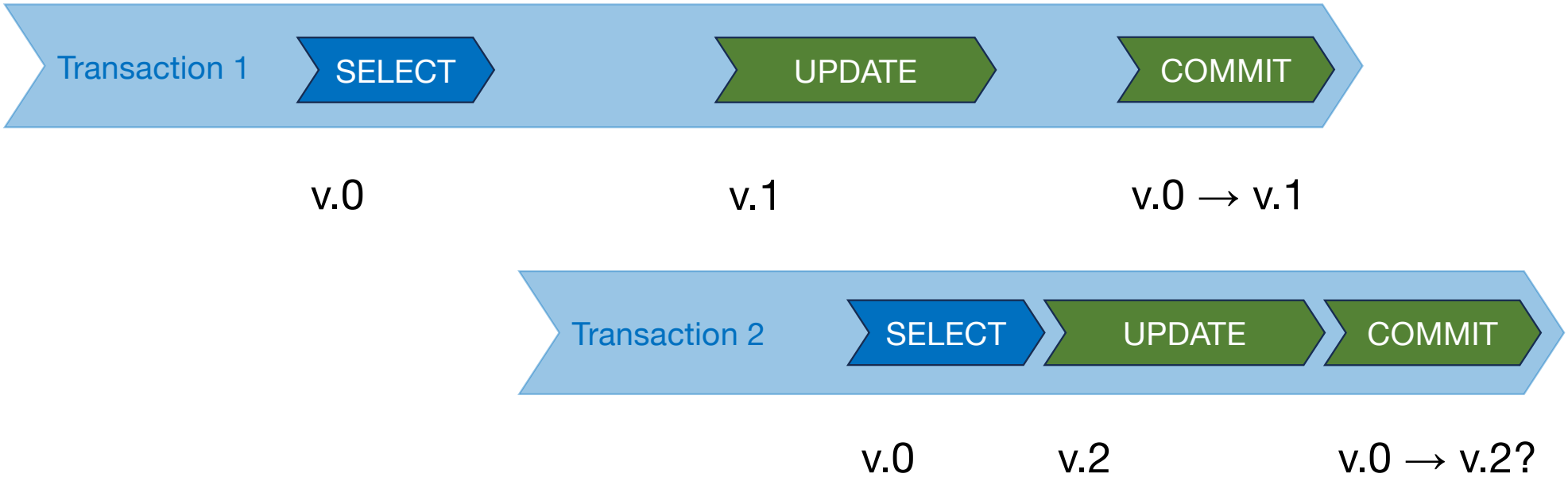
Snapshot isolation: Update conflict



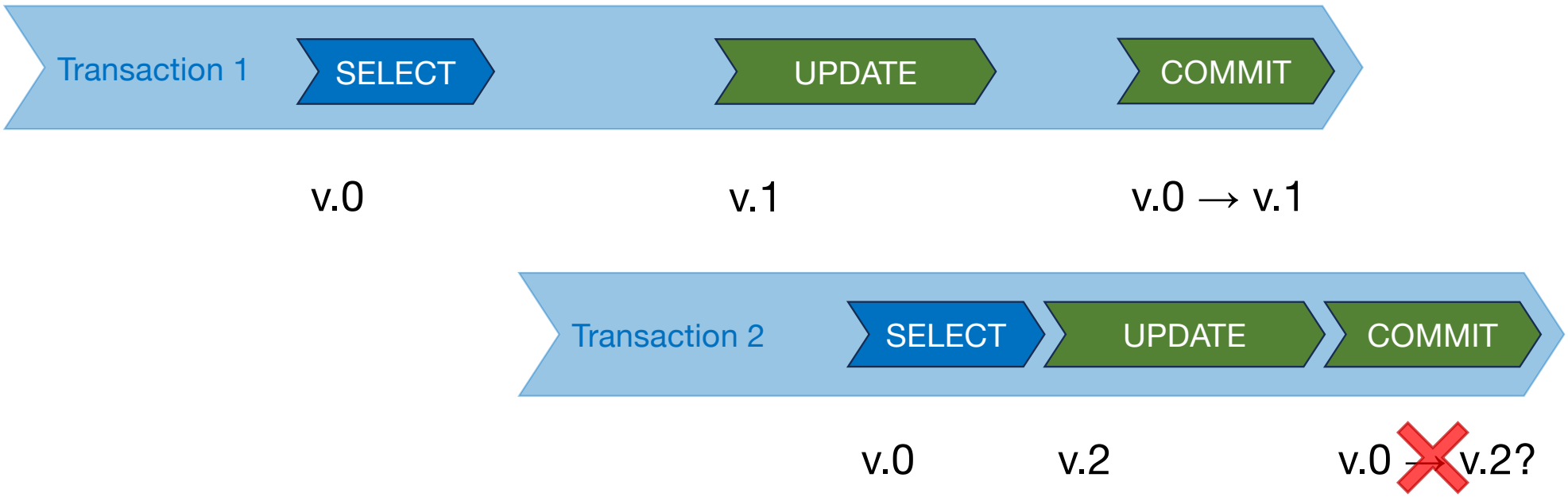
Snapshot isolation: Update conflict



Snapshot isolation: Update conflict



Snapshot isolation: Update conflict



DEMO

RCSI vs. Snapshot

- Snapshot Isolation is set per *transaction*.
- Snapshot Isolation protects the *transaction*.
- Snapshot Isolation requires code change.
- Conflicting writes will cause *update conflict*.

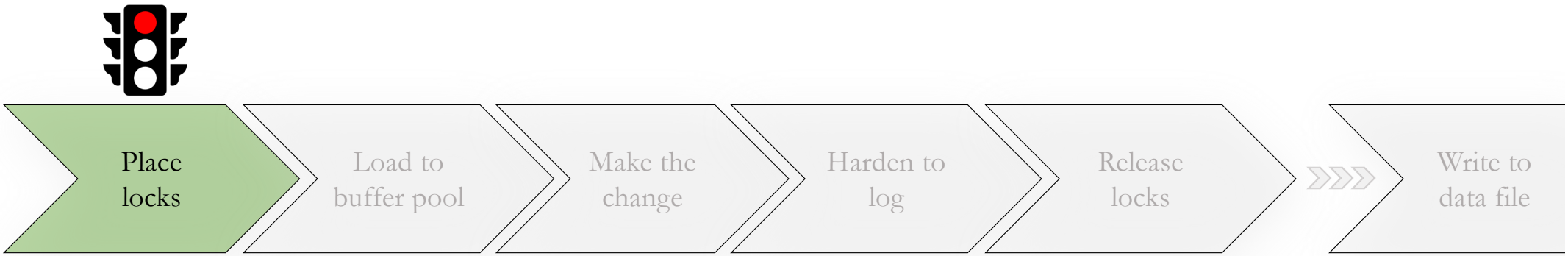
- Read Committed Snapshot Isolation is a *database* setting.
- Read Committed Snapshot Isolation protects the *statement*.
- Conflicting writes will cause *blocking*.

- Both require testing, because they behave differently.

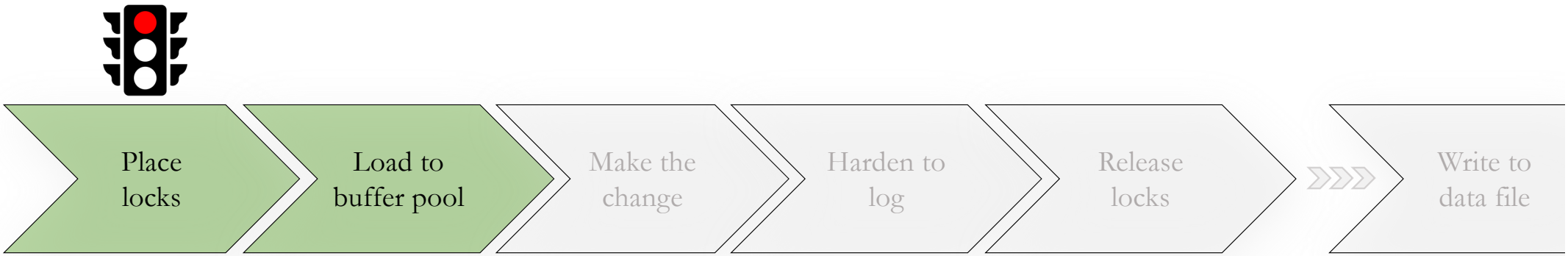
More details: https://brentozar.com/go/rcsi

How the transaction log works

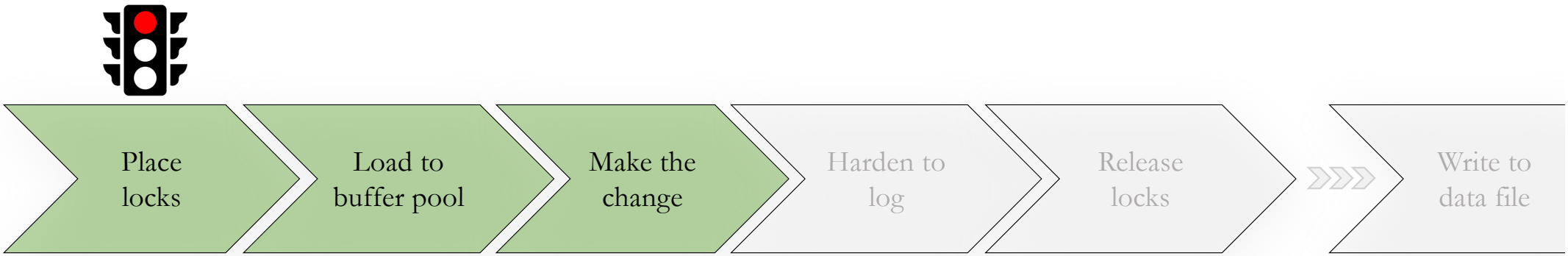
How the transaction log works



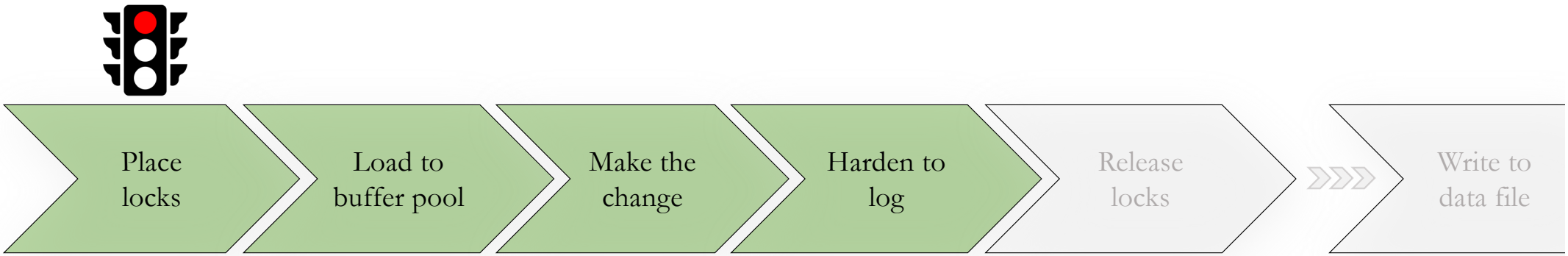
How the transaction log works



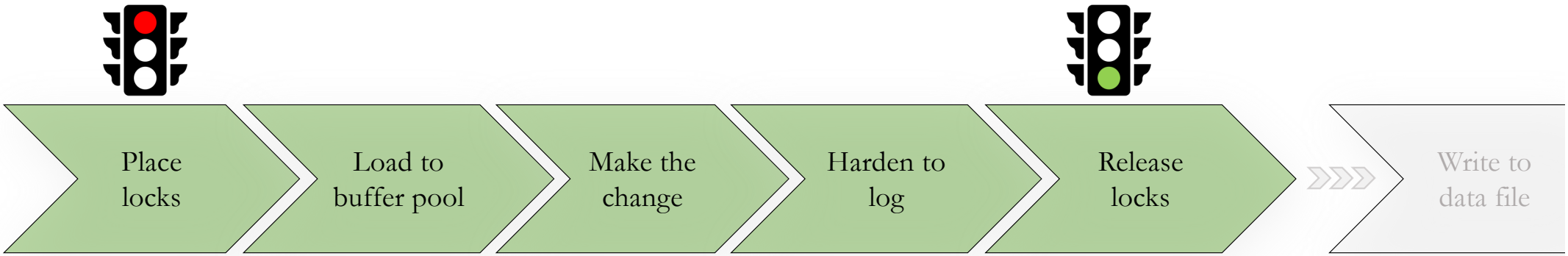
How the transaction log works



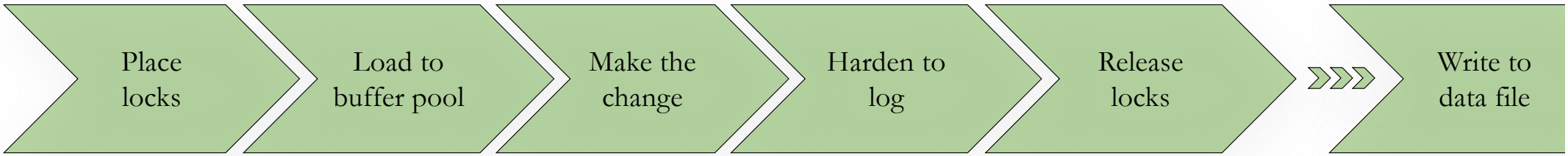
How the transaction log works



How the transaction log works



How the transaction log works



Durability

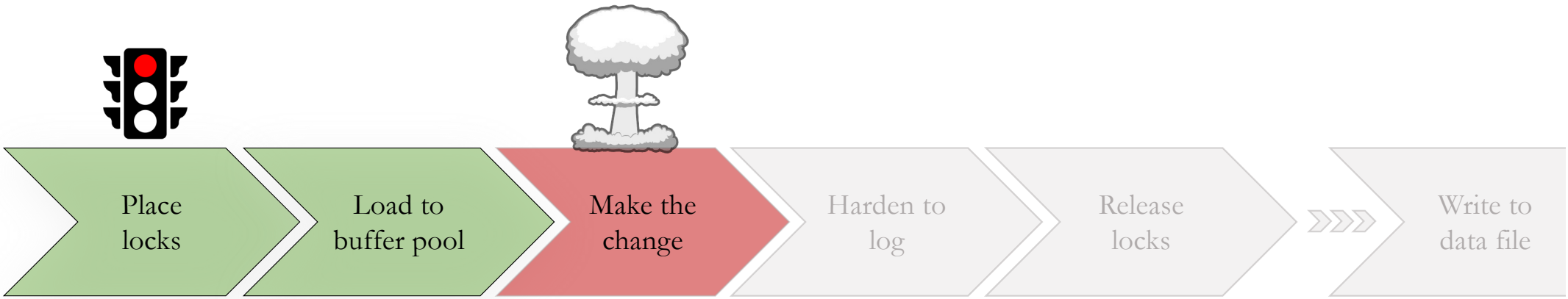
Durability


Place locks
Load to buffer pool
Make the change
Harden to log
Release locks
Write to data file

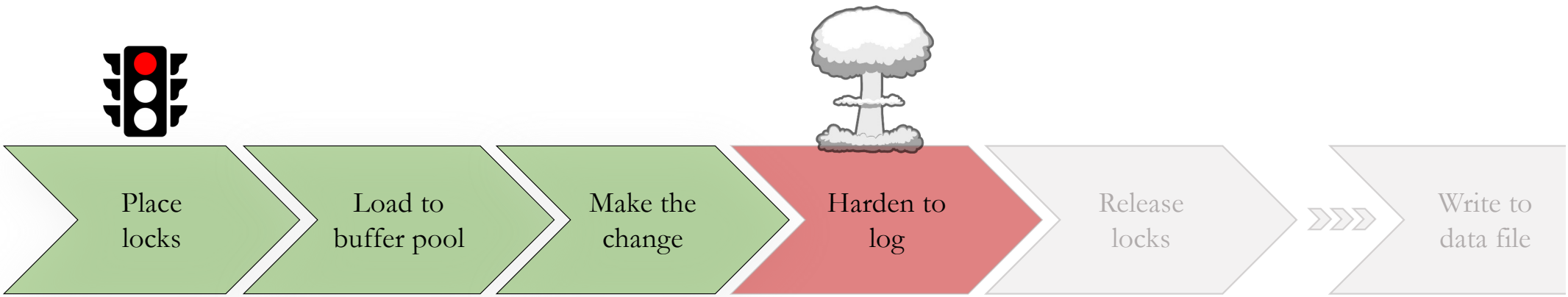
Durability


Place locks
Load to buffer pool
Make the change
Harden to log
Release locks
Write to data file

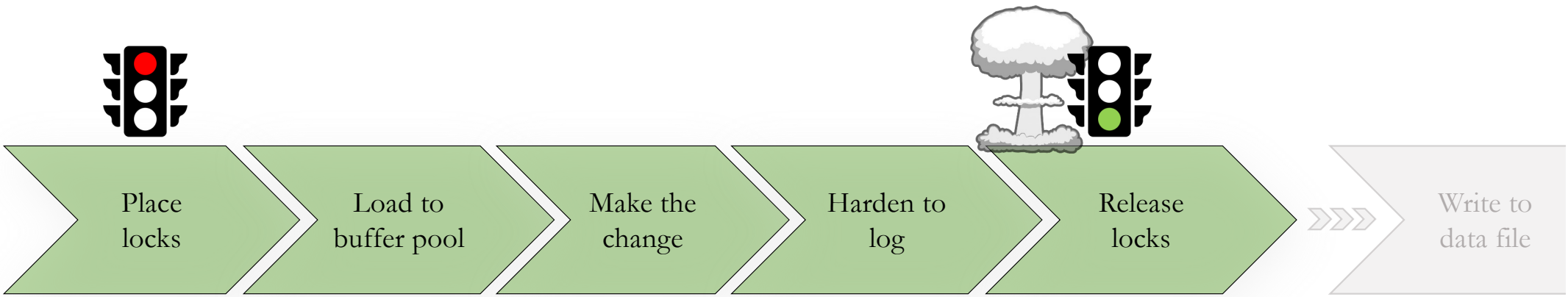
Durability


Place locks
Load to buffer pool
Make the change
Harden to log
Release locks
Write to data file

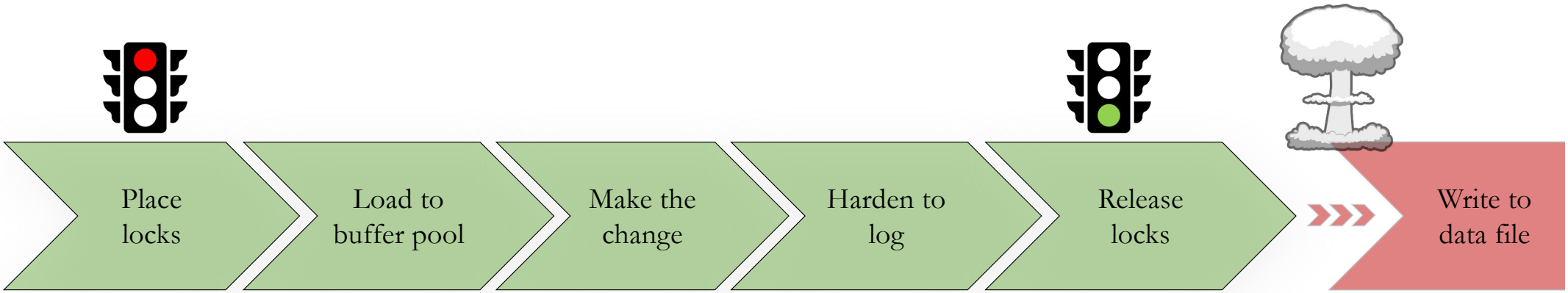
Durability


Place locks
Load to buffer pool
Make the change
Harden to log
Release locks
Write to data file

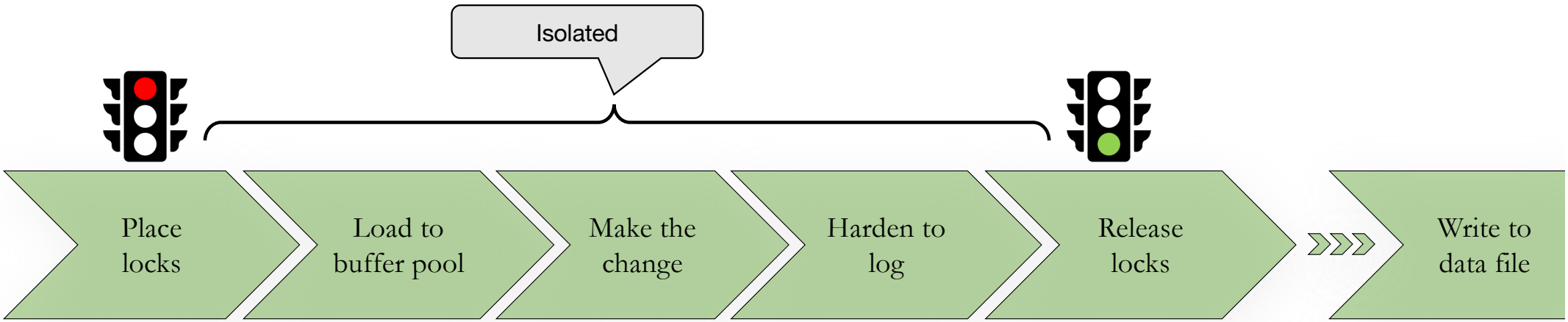
Durability


Place locks
Load to buffer pool
Make the change
Harden to log
Release locks
Write to data file

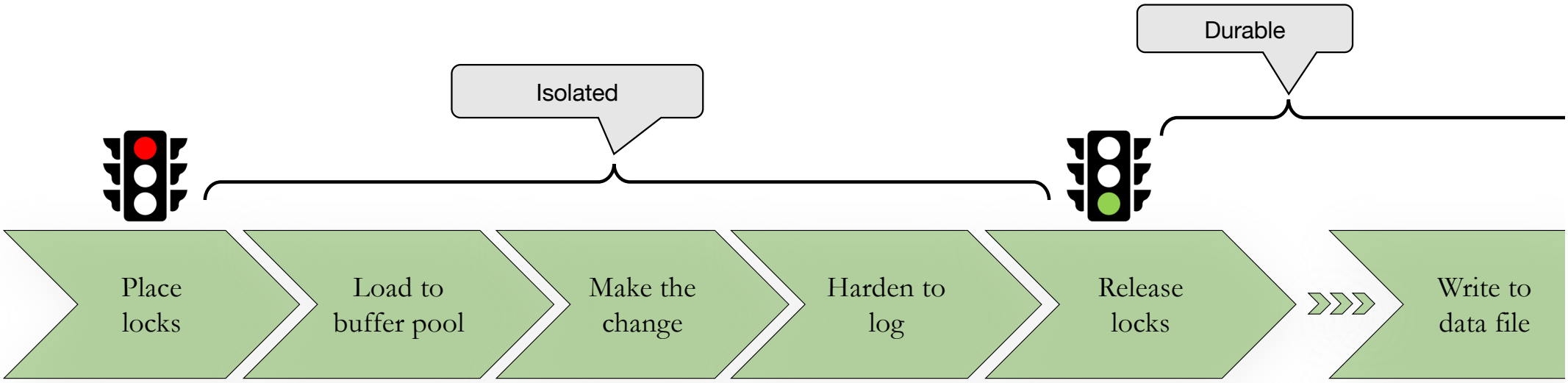
Durability


Place locks
Load to buffer pool
Make the change
Harden to log
Release locks
Write to data file

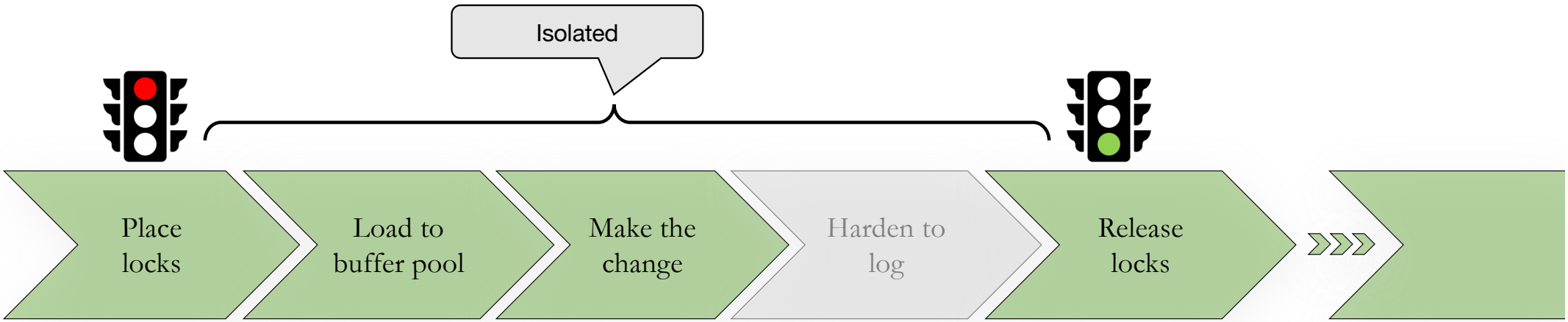
Durability


Isolated
Place locks
Load to buffer pool
Make the change
Harden to log
Release locks
Write to data file

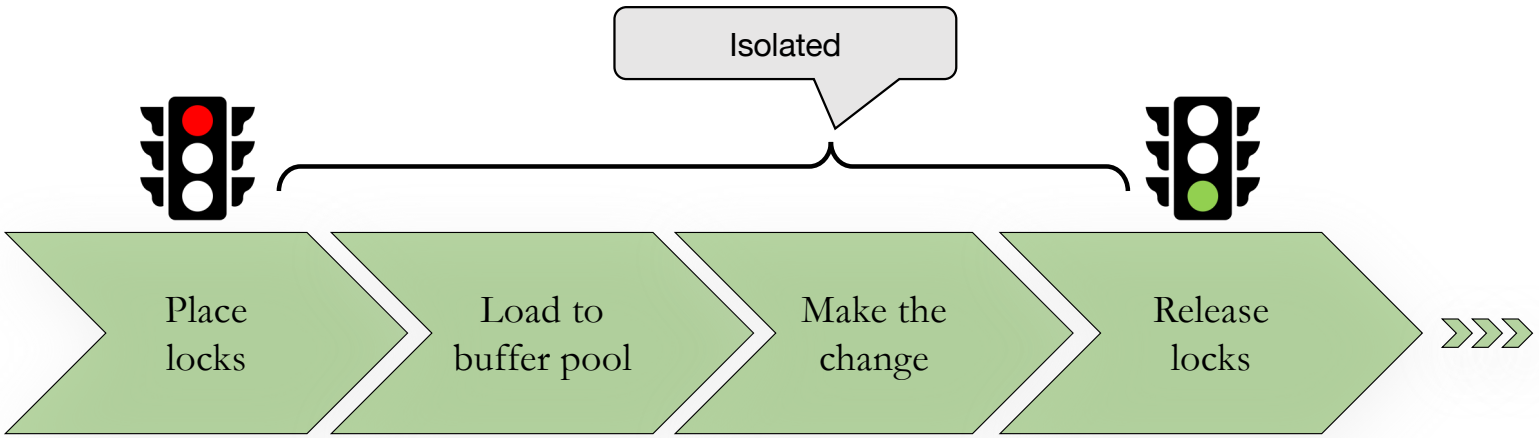
Durability


Durable
Isolated
Place locks
Load to buffer pool
Make the change
Harden to log
Release locks
Write to data file

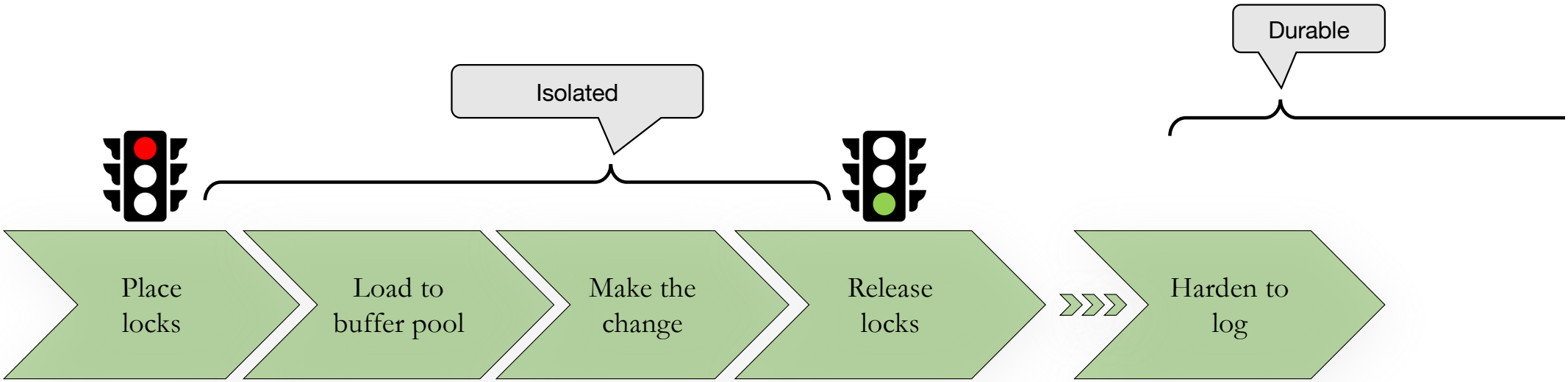
Delayed durability



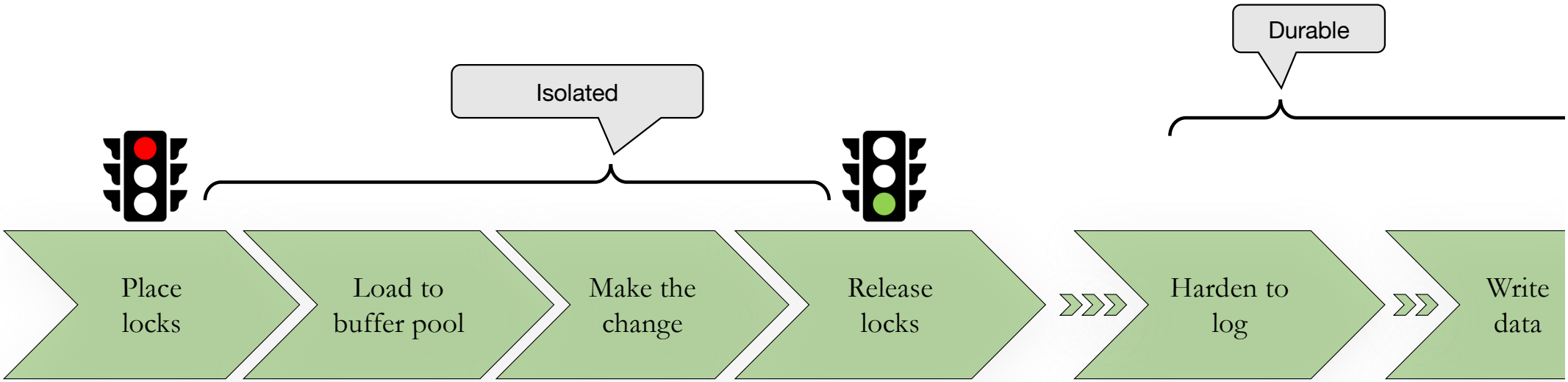
Delayed durability



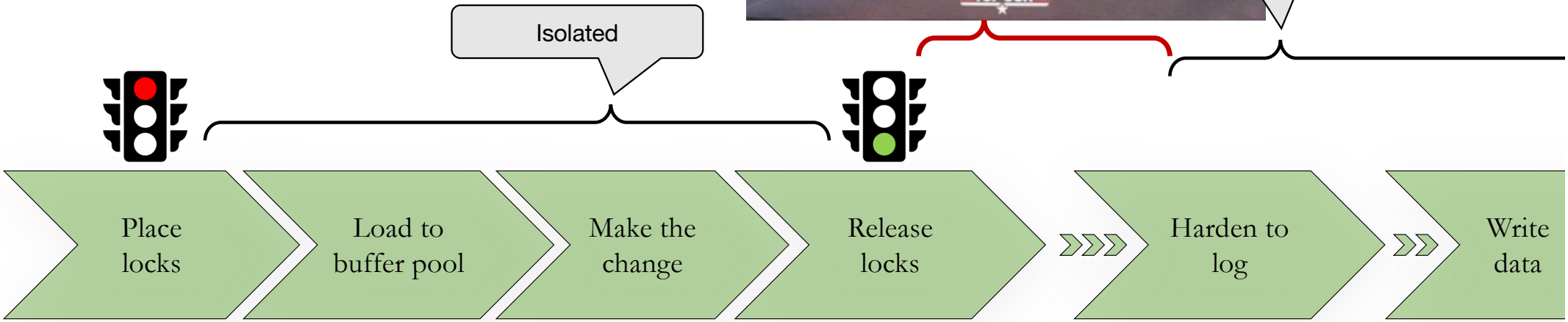
Delayed durability



Delayed durability



Delayed durability



Delayed durability

- Writes are batched into the transaction log
- Significantly reduced latency for tiny transactions
- Great if you don’t mind losing the data – like a DW or a staging environment
- tempdb already uses a form of Delayed Durability under the hood

Takeaways and questions

- I was going to do this slide later, but here we are.

Please give me feedback

Oh, and the slides and scripts: github.com/sqlsunday/presentations