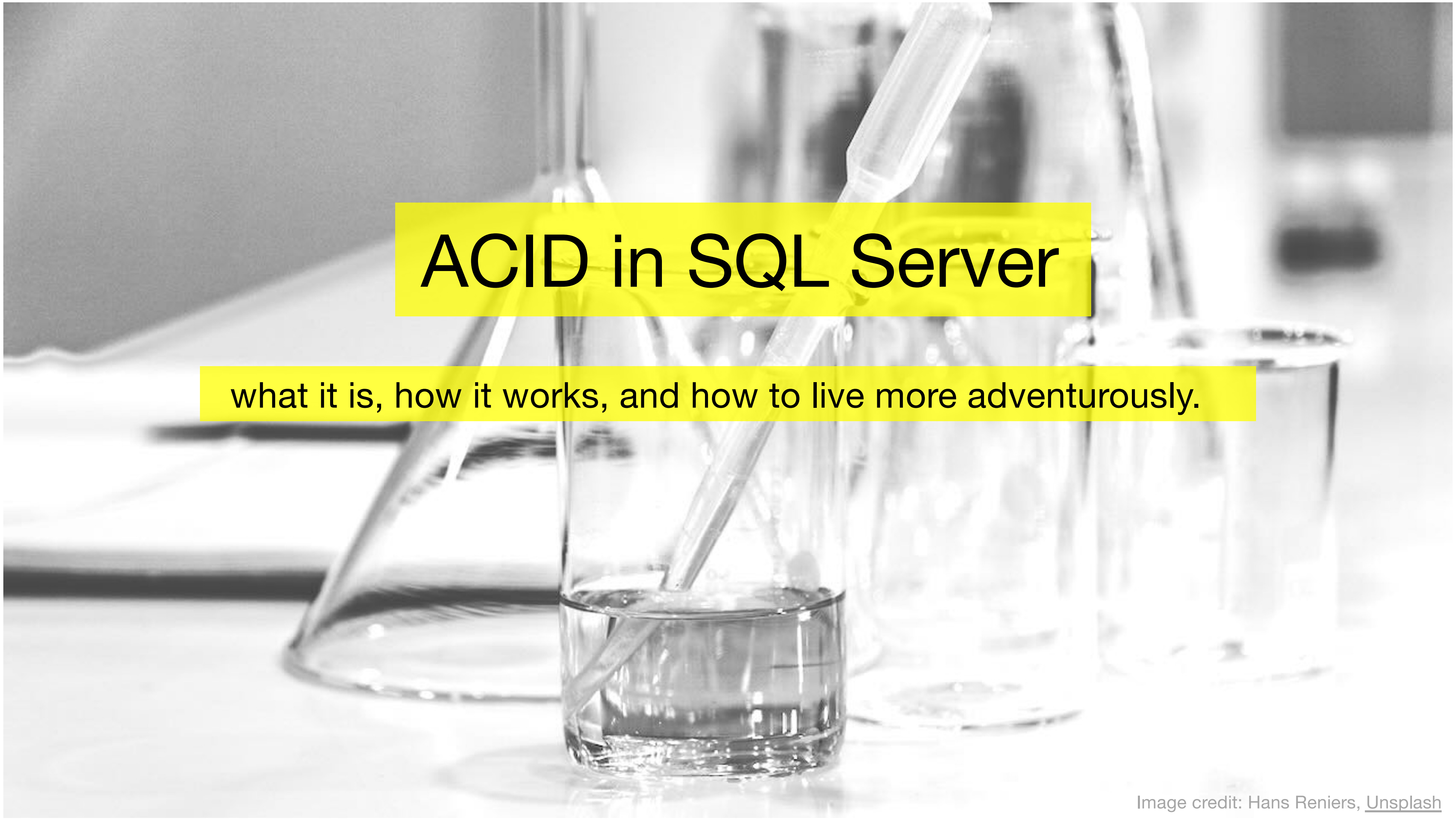


ACID in SQL Server

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

Image credit: Hans Reniers, Unsplash

Thank you to our sponsors

In this session

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

A	C	I	D
Atomic	Consistent	Isolated	Durable

A

Atomic

C

Consistent

I

Isolated

D

Durable

All or nothing.
The *business* logic.

A	C	I	D
Atomic	Consistent	Isolated	Durable

A single truth.
The *database* logic.

A	C	I	D
Atomic	Consistent	Isolated	Durable

No interference
when changing data.

A	C	I	D
Atomic	Consistent	Isolated	Durable 🟩

Keep all promises.

Who am I?

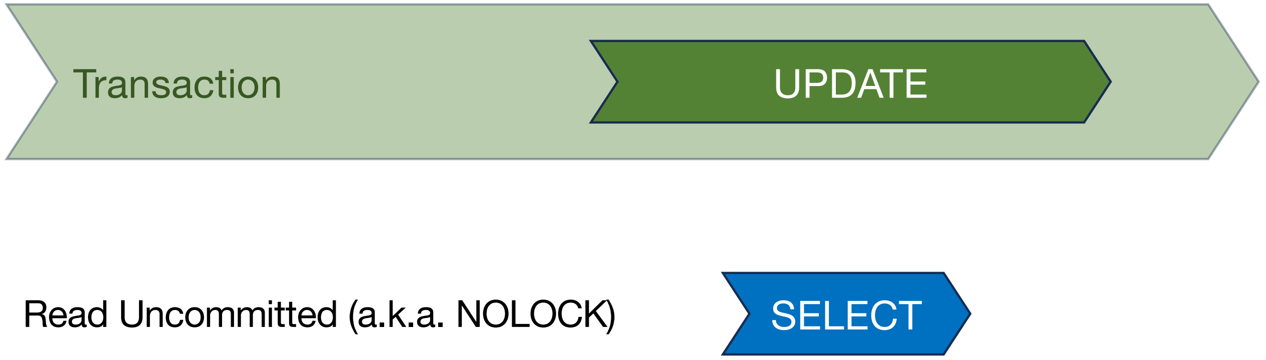
- SQL Server developer since 1997
- Consultant
- Organizer of Data Saturday Stockholm
- Data Platform MVP
- Dog person

@dhmacher on (almost) all the socials.

Isolation: Quirks and features of multi-user databases

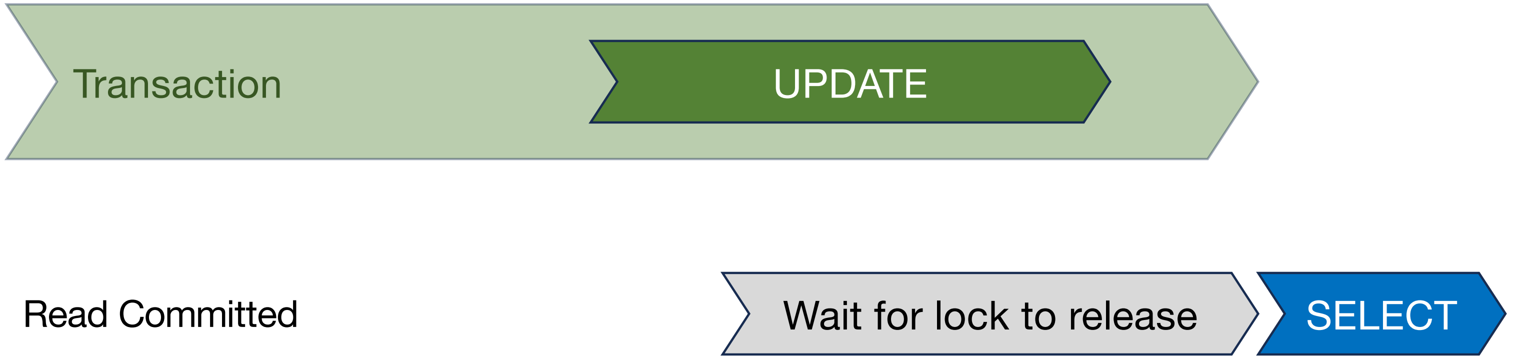
Dirty reads

- no Nobel literature prize for you.



Dirty reads

- no Nobel literature prize for you.



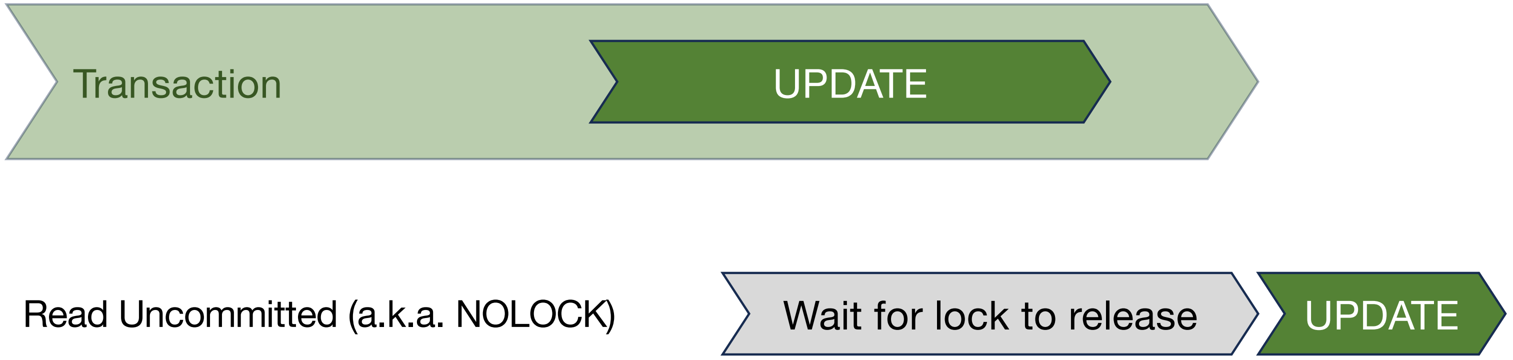
Dirty reads

- no Nobel literature prize for you.

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

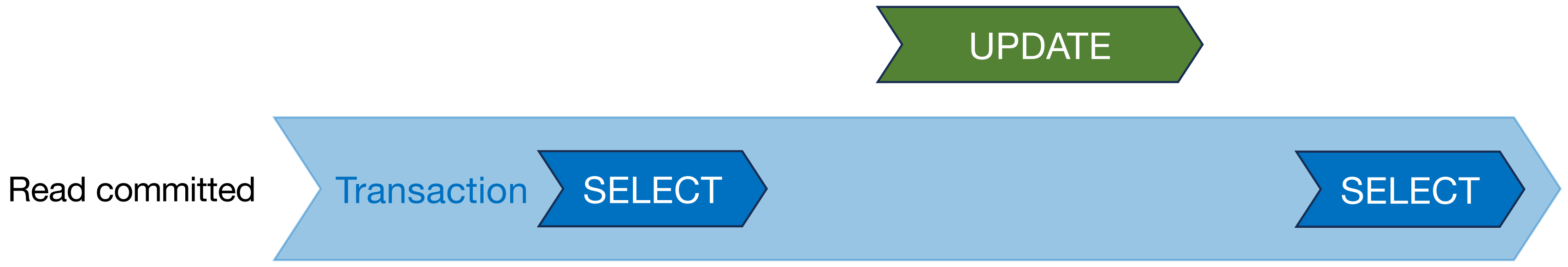
Dirty reads

- no Nobel literature prize for you.

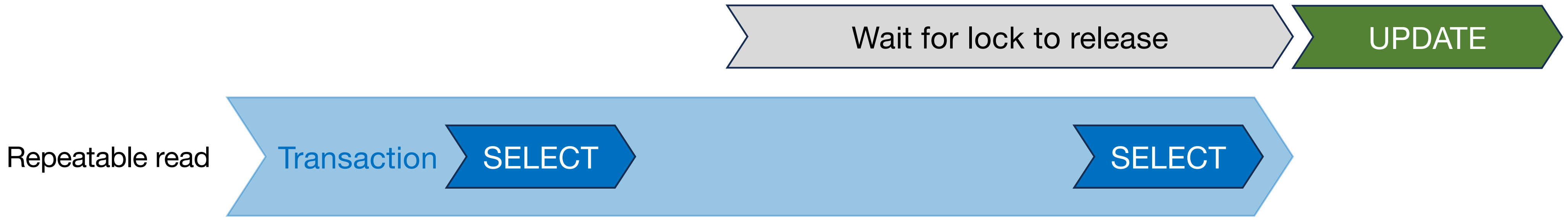


DEMO

Non-repeatable reads



Non-repeatable reads

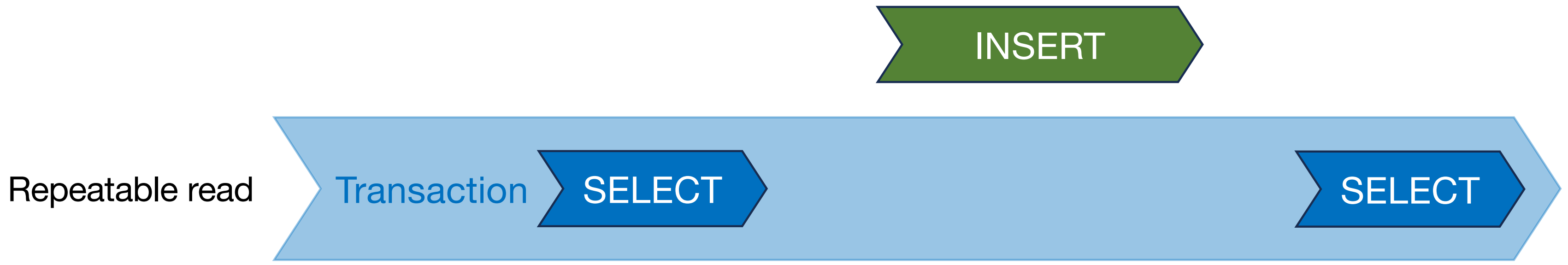


DEMO

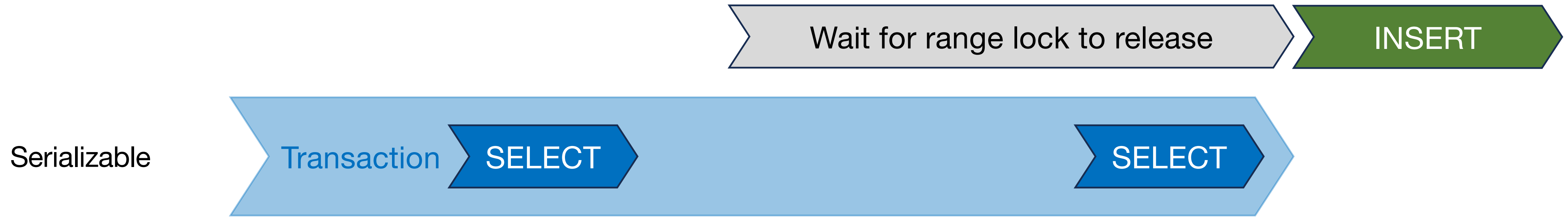
Phantom reads

Your database is gaslighting you.

Phantom reads



Phantom reads



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 →

What are isolation levels?

Read uncommitted

Read committed

Repeatable read

Serializable

Dirty reads

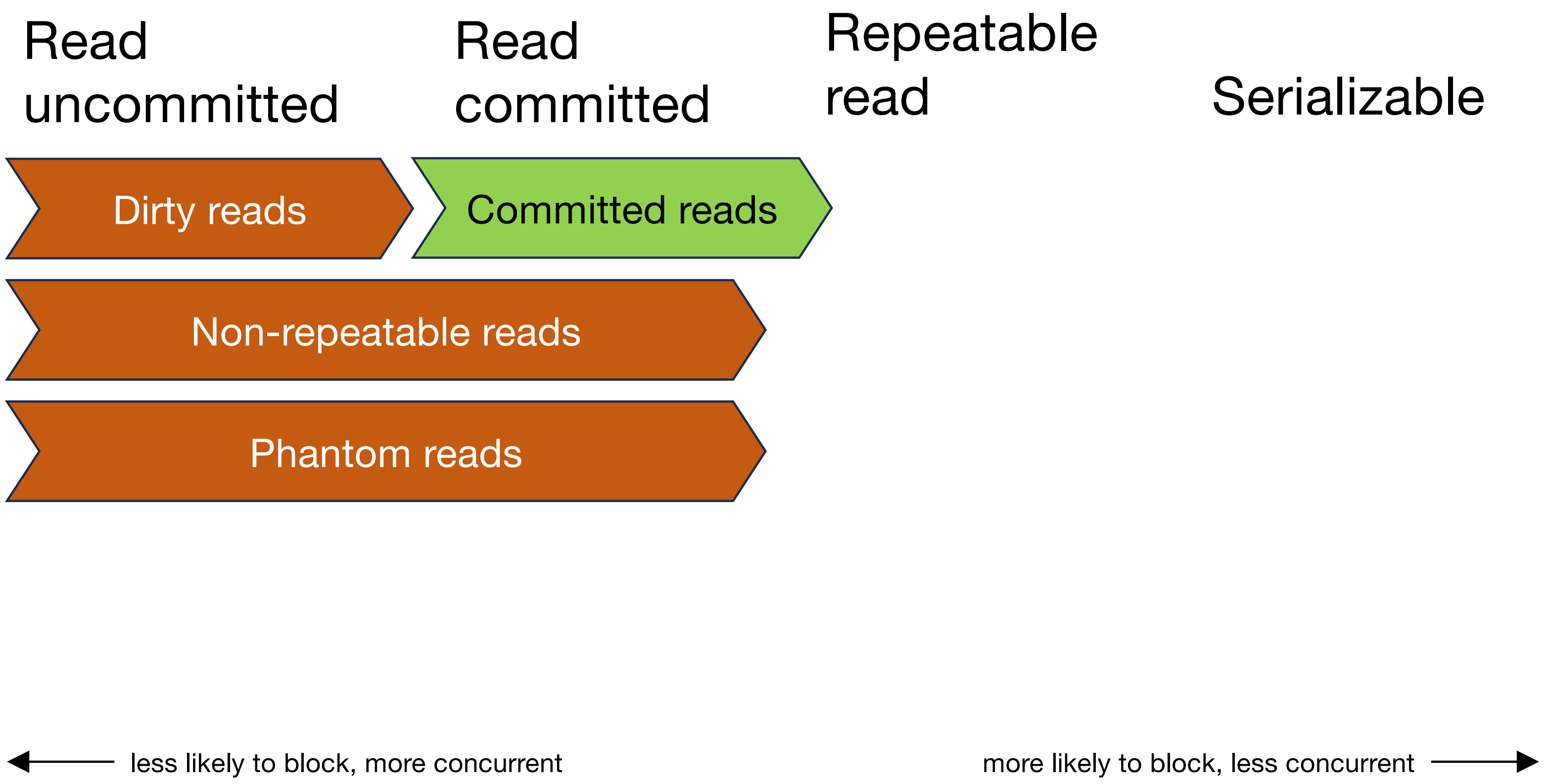
Non-repeatable

Phantom reads

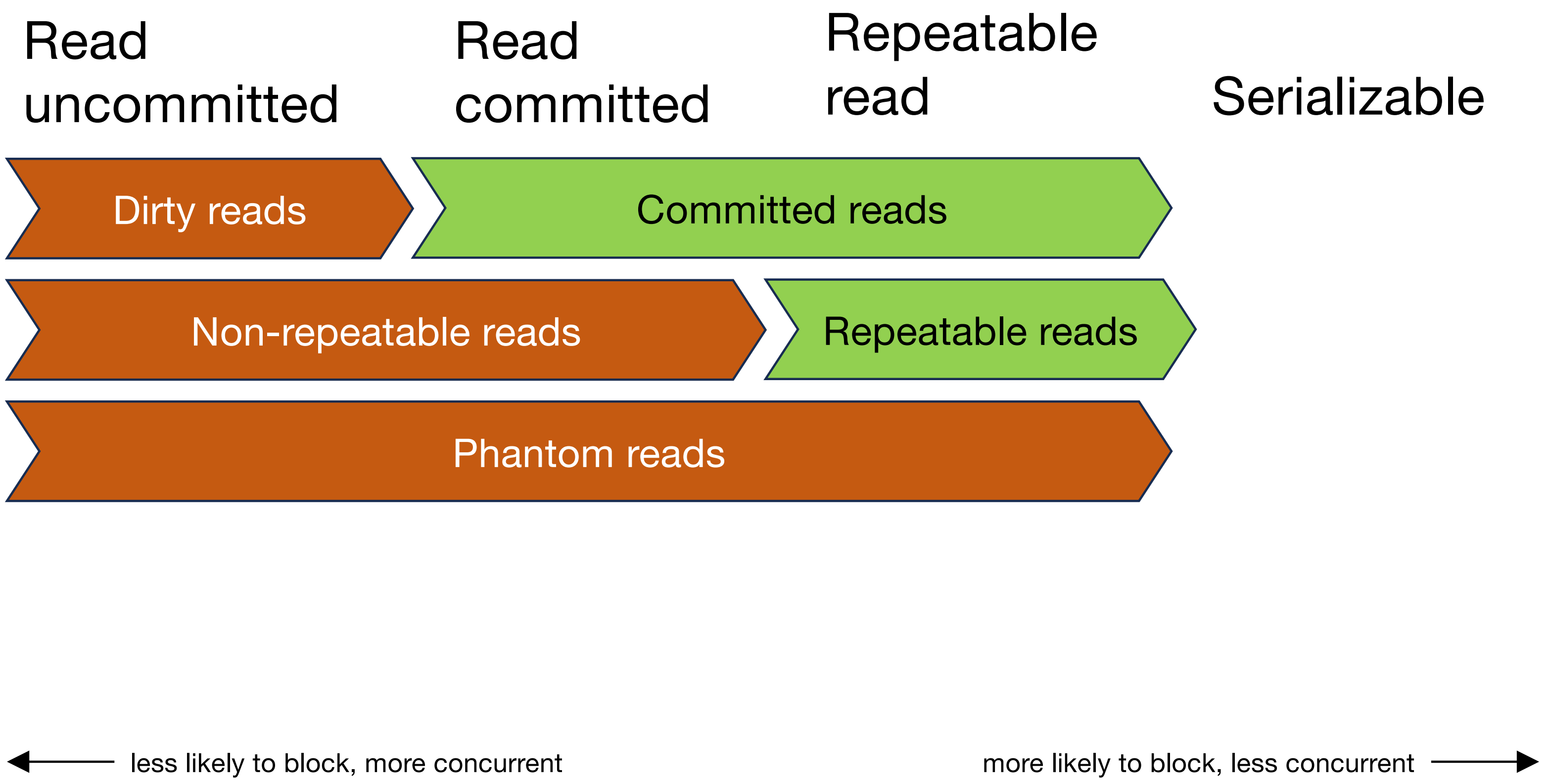
← less likely to block, more concurrent

more likely to block, less concurrent →

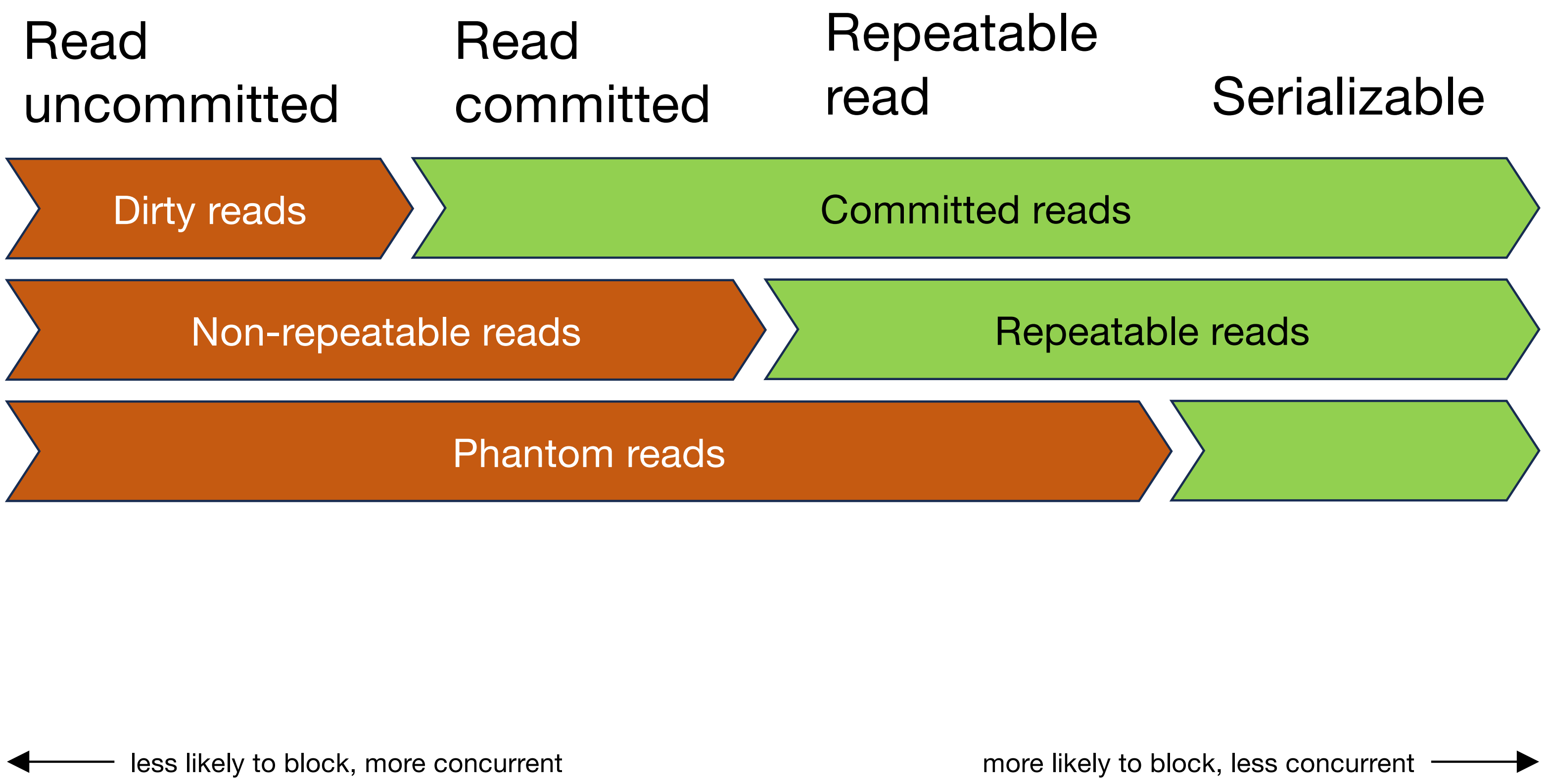
What are isolation levels?



What are isolation levels?



What are isolation levels?



So… Serialize all the things, then?

Less locking

- Less isolation
- Less predictable

More locking

- Better isolation
- More predictable

So… Serialize all the things, then?

Less locking

- Less isolation
- Less predictable
- Better concurrency

More locking

- Better isolation
- More predictable
- Lower concurrency

So… Serialize all the things, then?

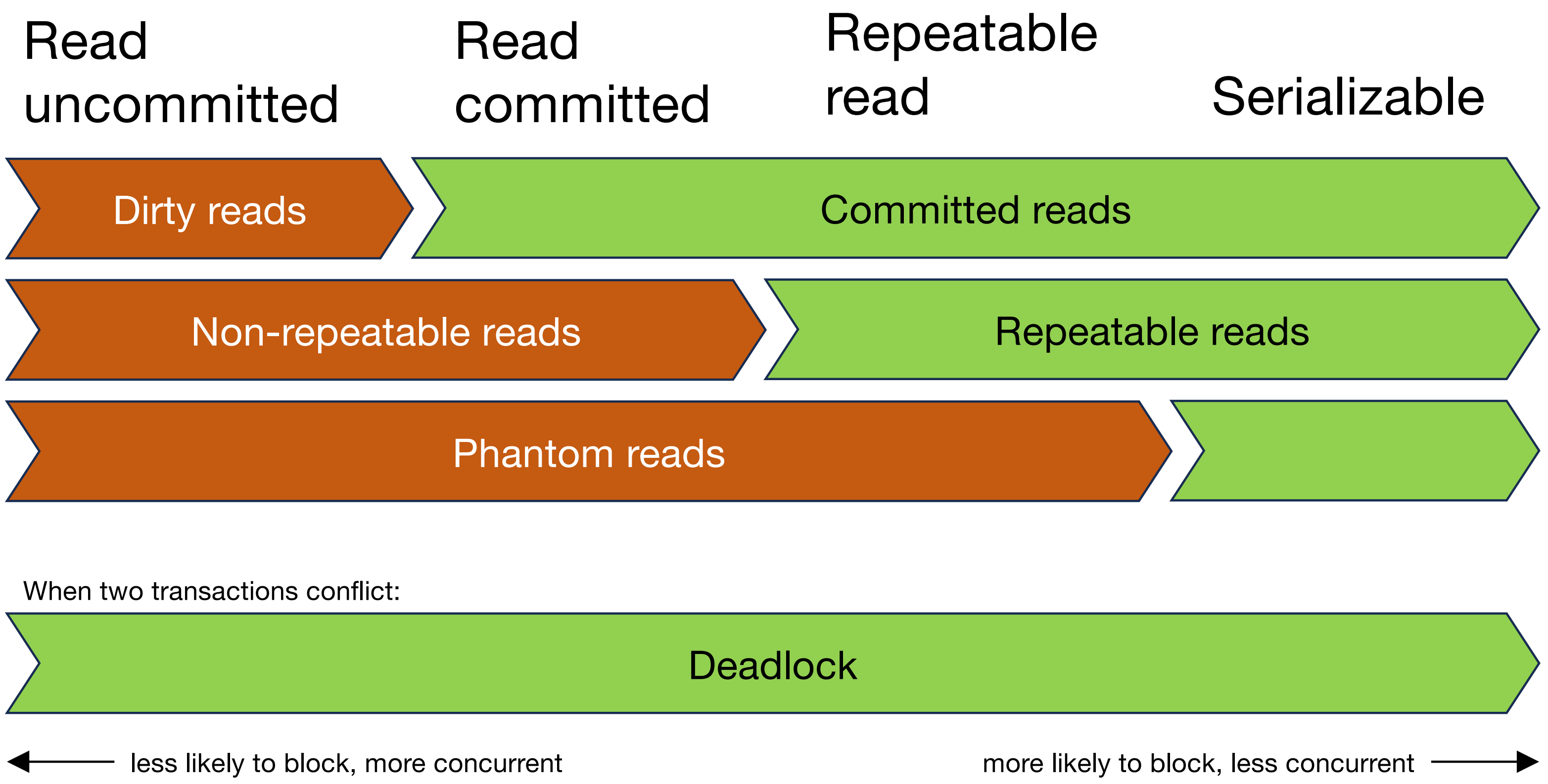
Less locking

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

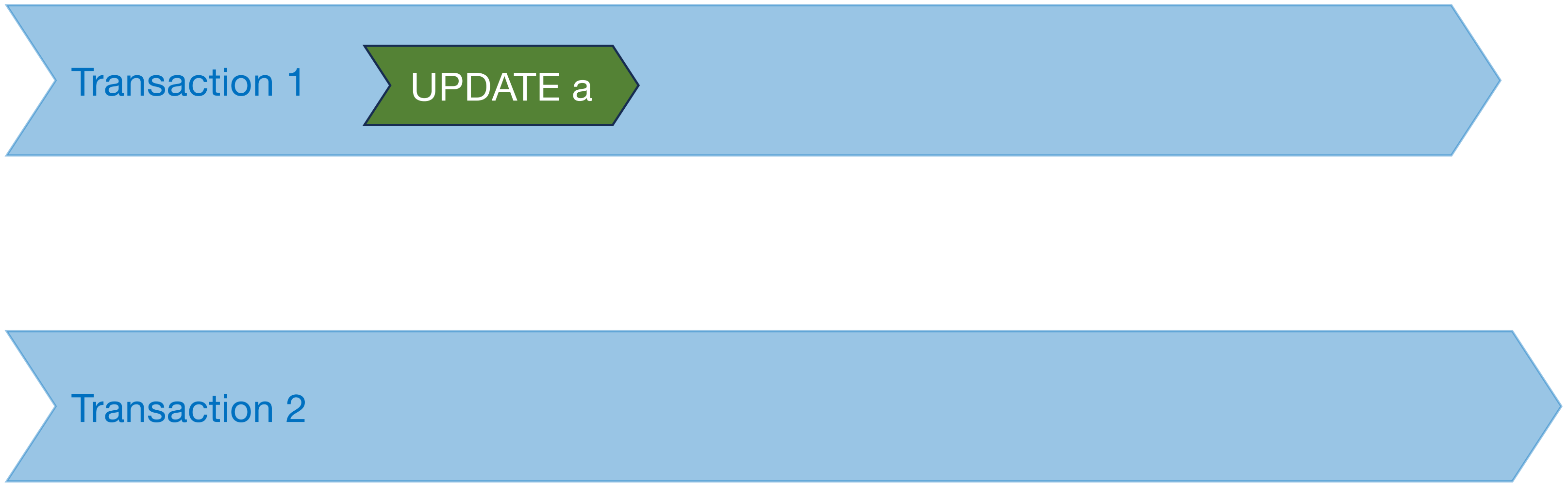
More locking

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

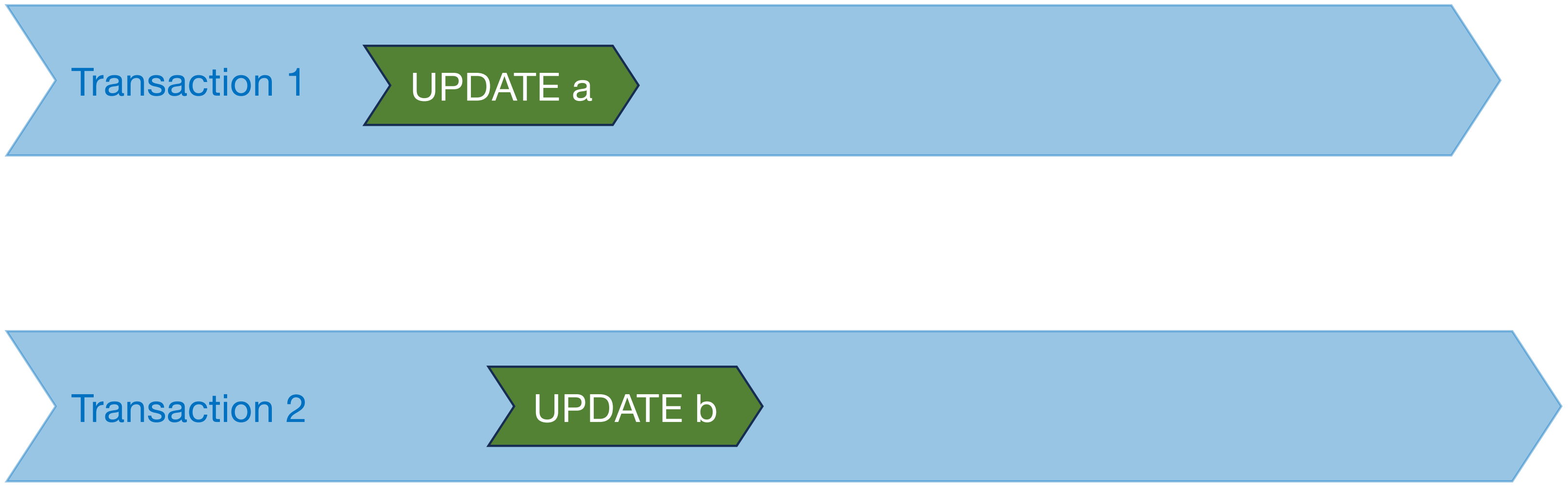
What are isolation levels?



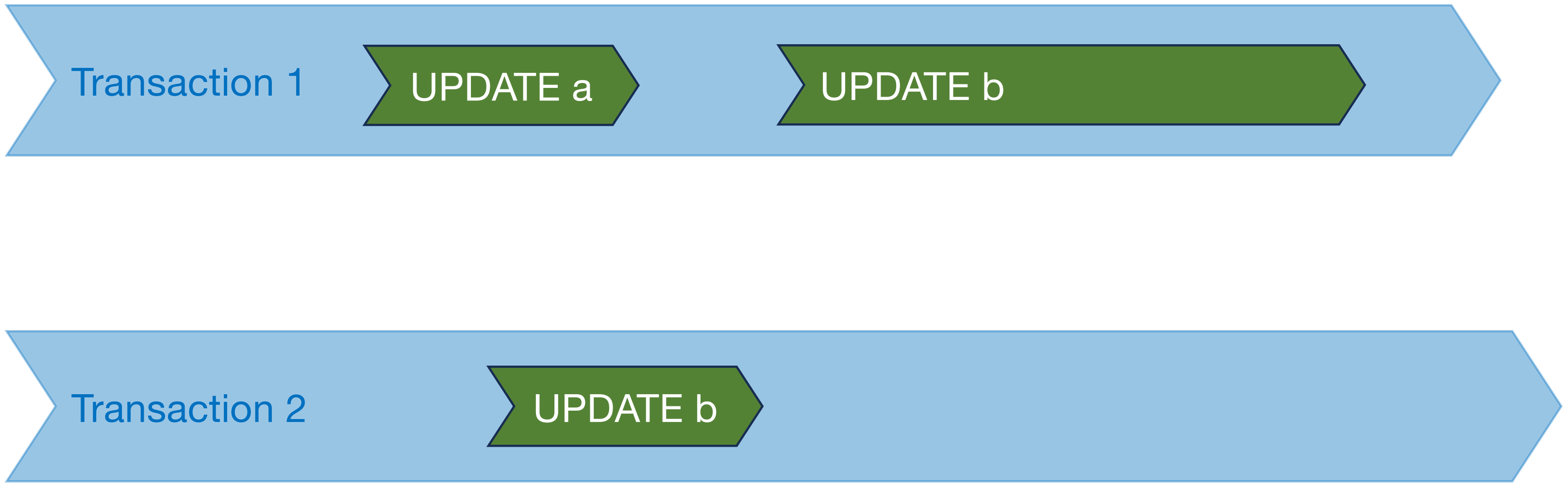
Deadlock



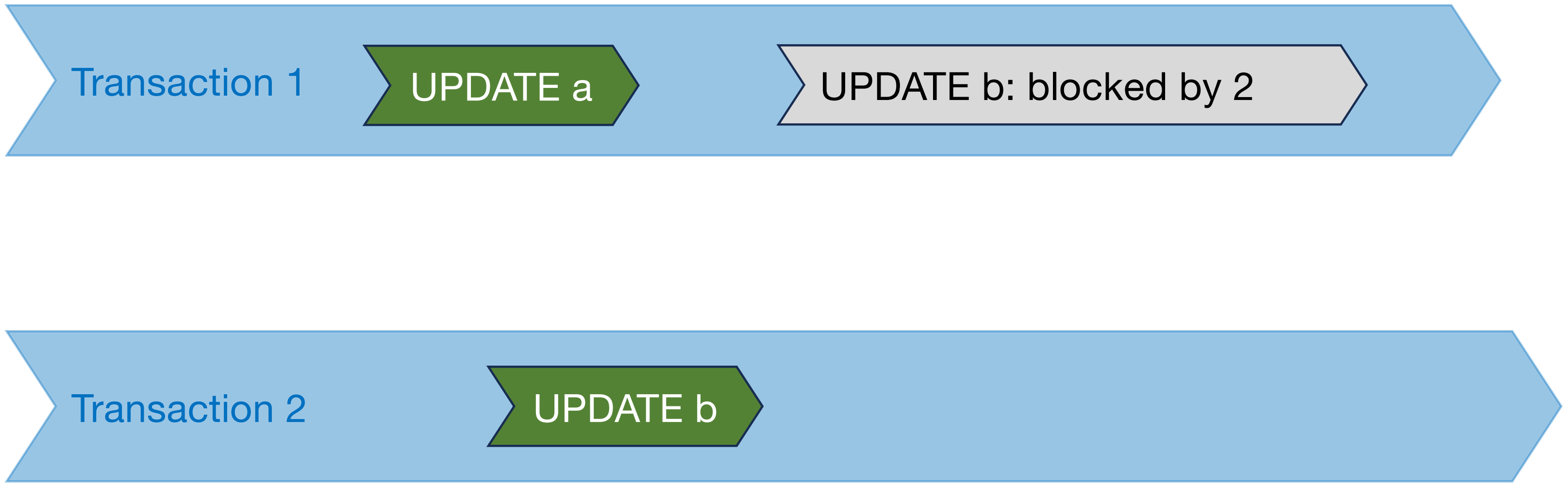
Deadlock



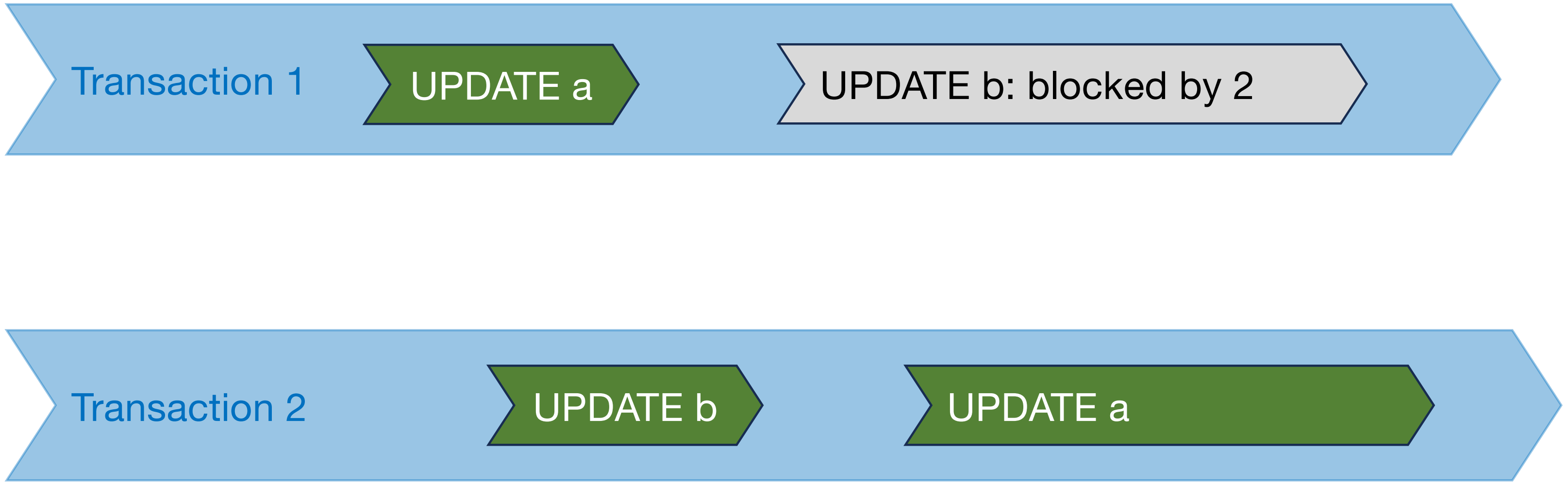
Deadlock



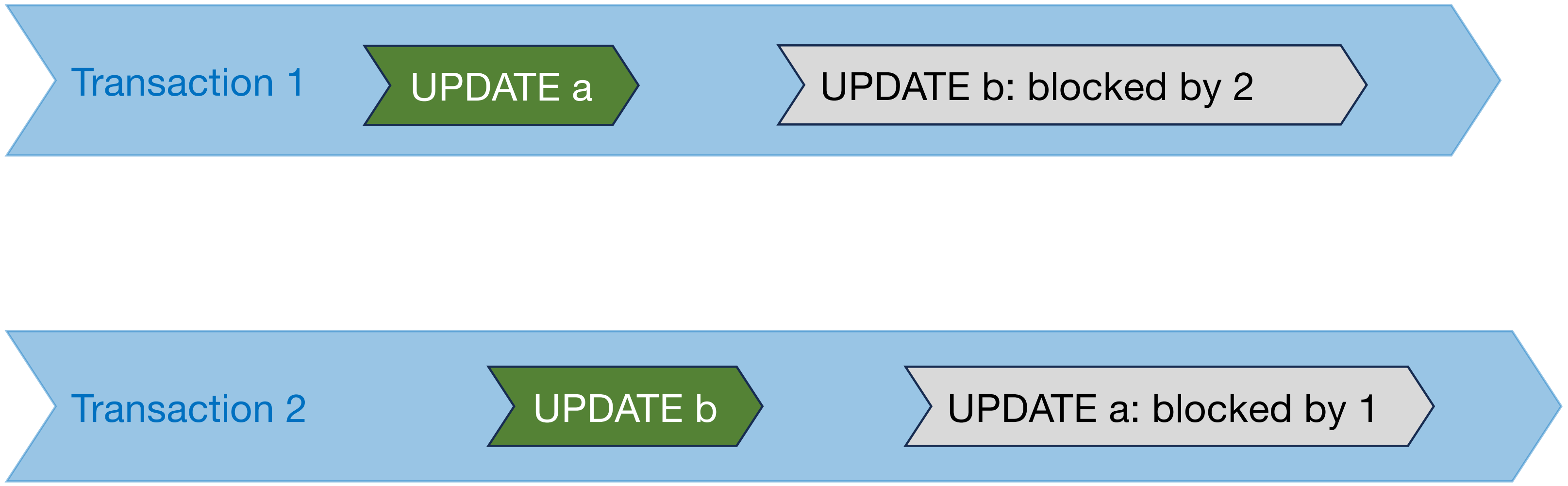
Deadlock


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

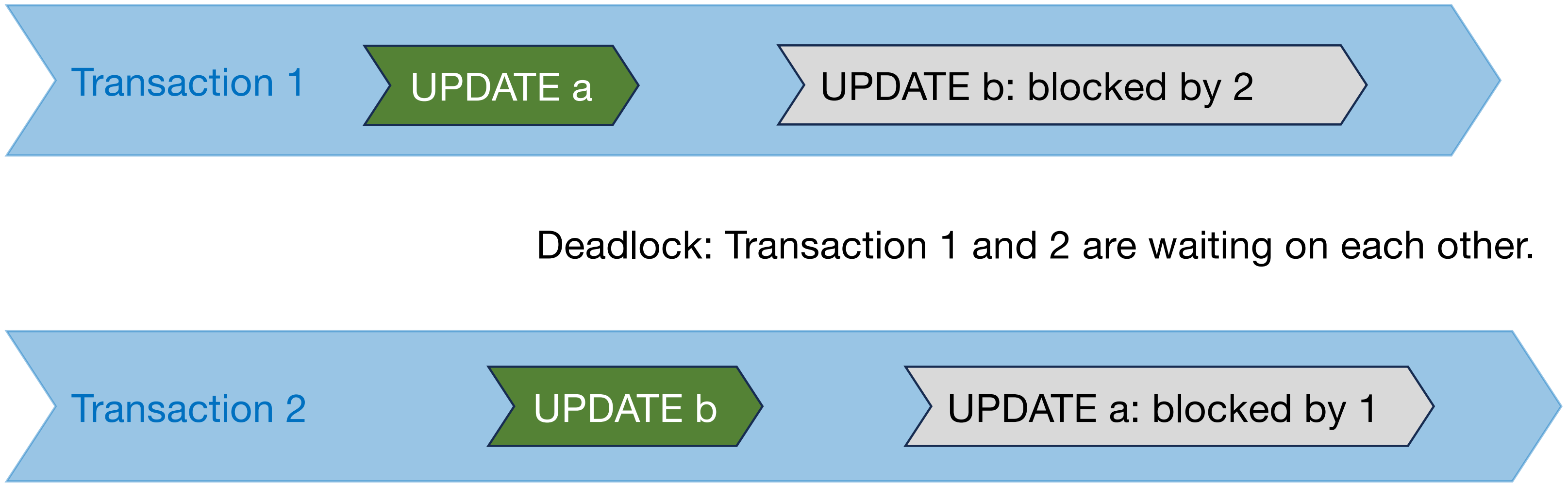
Deadlock


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

Deadlock

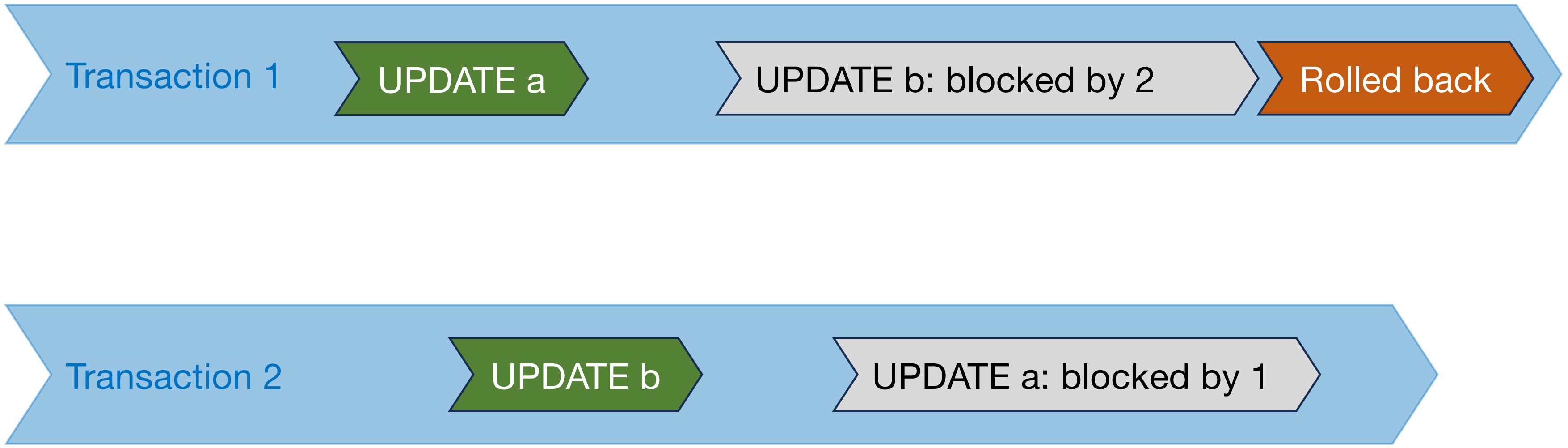

Transaction 1
UPDATE a
UPDATE b: blocked by 2
Transaction 2
UPDATE b
UPDATE a: blocked by 1

Deadlock

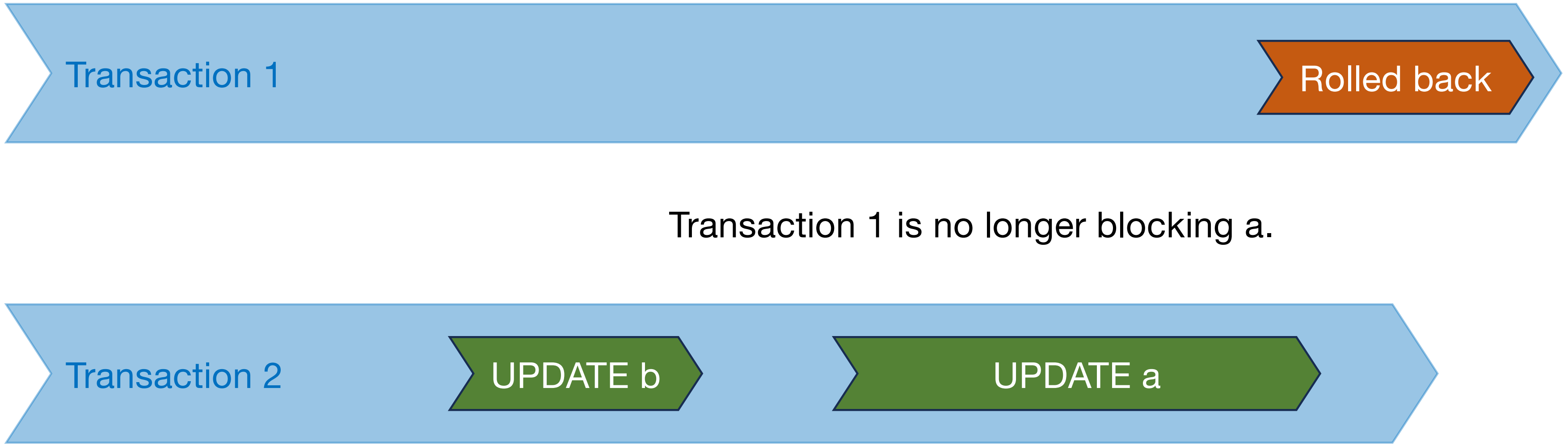

Transaction 1
UPDATE a
UPDATE b: blocked by 2
Deadlock: Transaction 1 and 2 are waiting on each other.
Transaction 2
UPDATE b
UPDATE a: blocked by 1

Deadlock

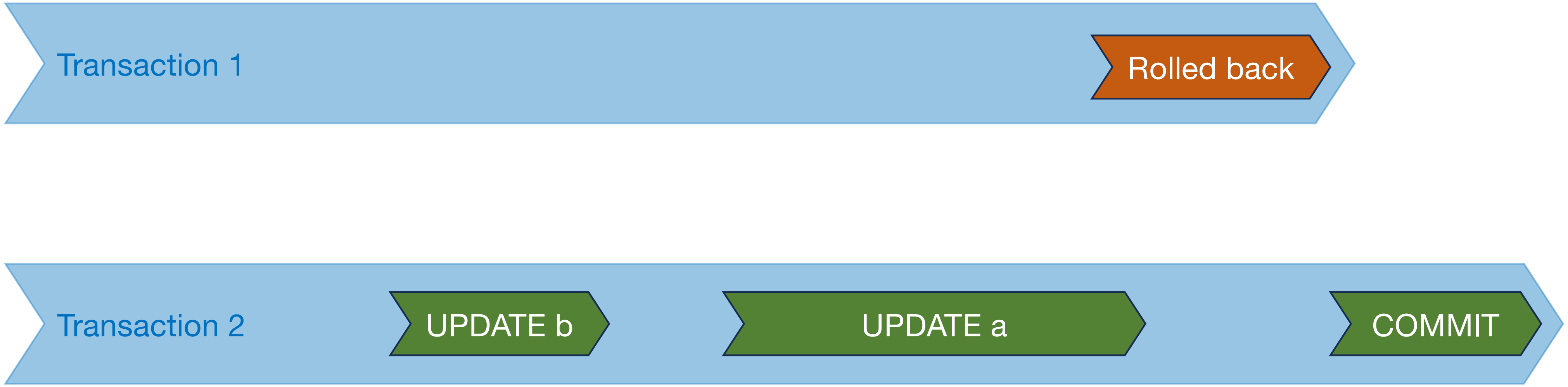
SQL Server will fail the transaction with the least amount of work to roll back:



Deadlock

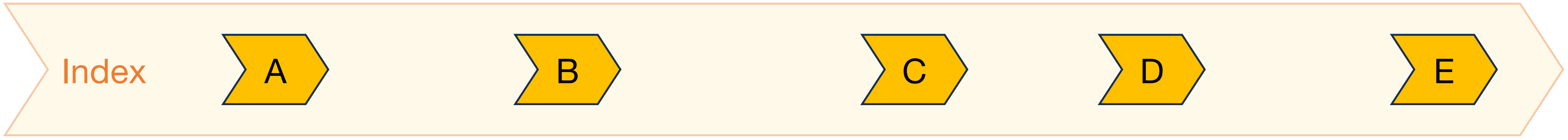

Transaction 1
Rolled back
Transaction 1 is no longer blocking a.
Transaction 2
UPDATE b
UPDATE a

Deadlock

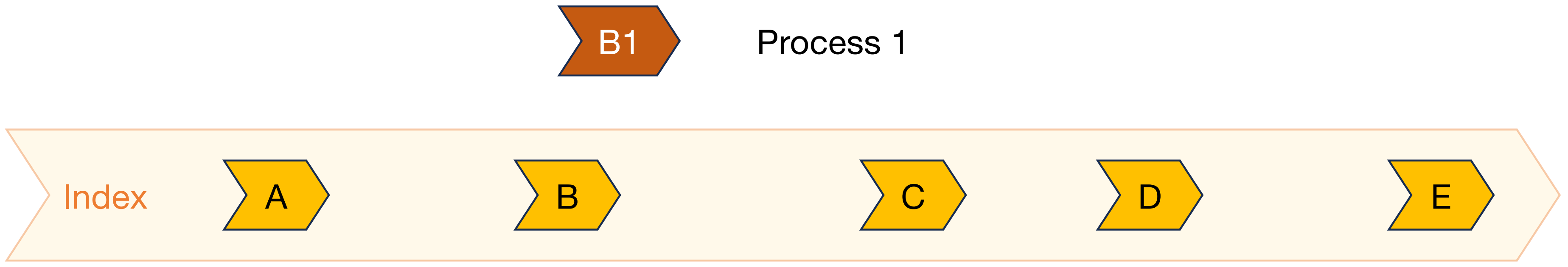

Transaction 1
Rolled back
Transaction 2
UPDATE b
UPDATE a
COMMIT

DEMO

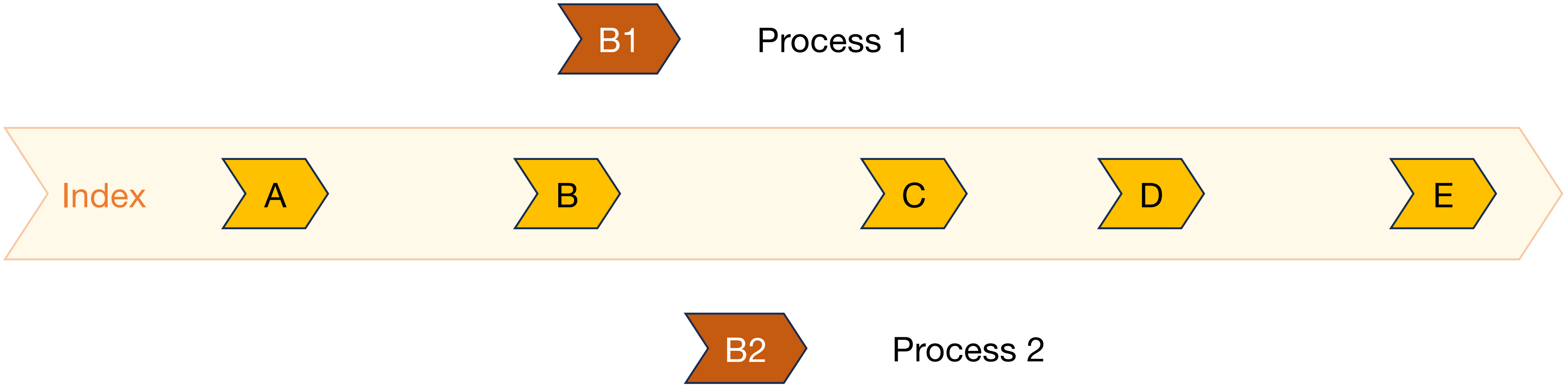
The serializable merge deadlock



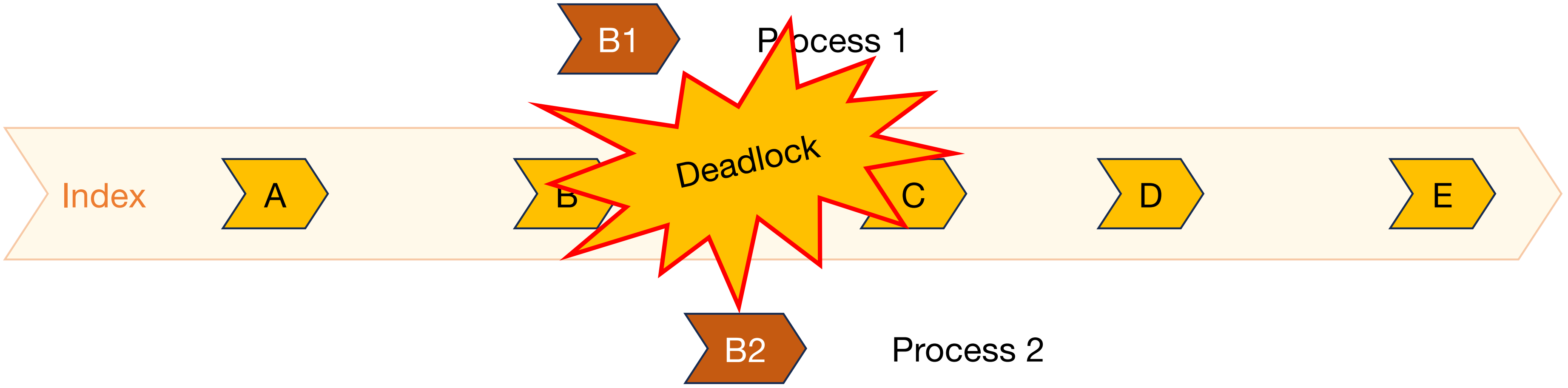
The serializable merge deadlock



The serializable merge deadlock

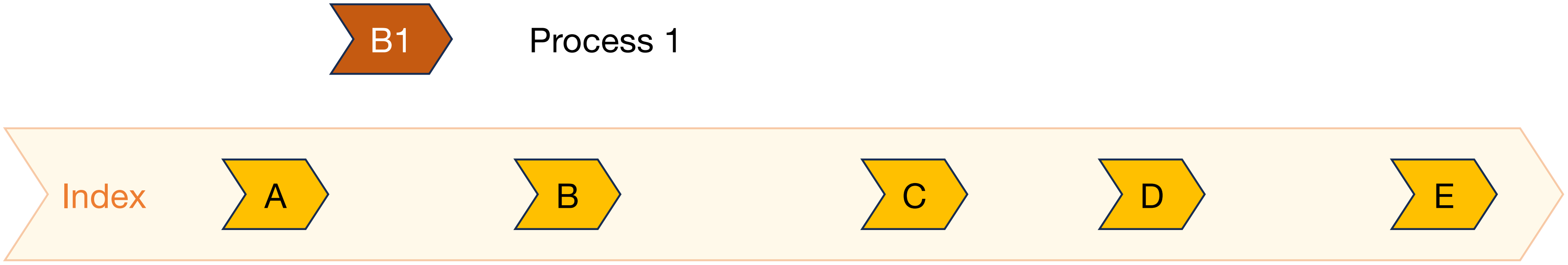


The serializable merge deadlock

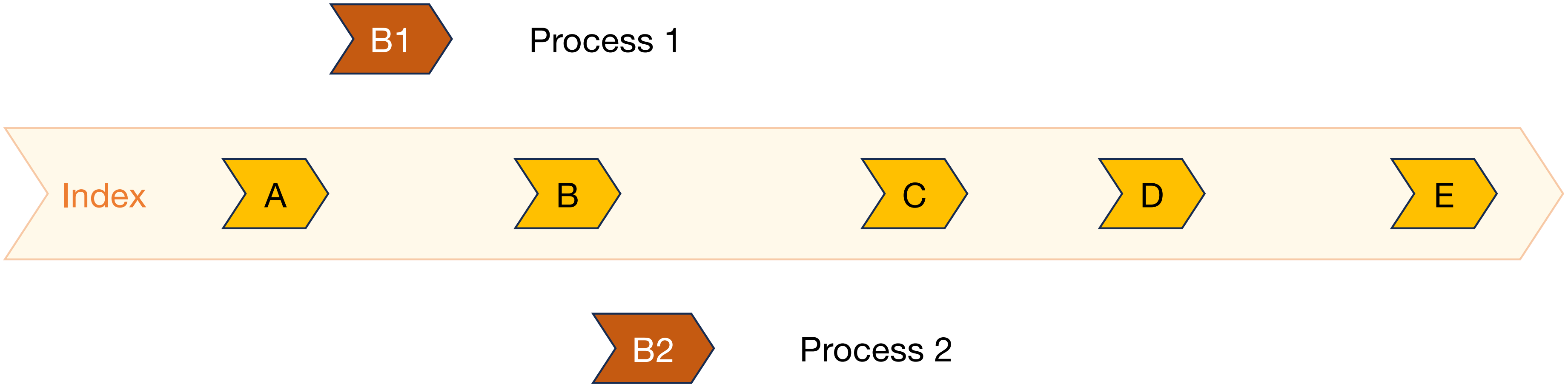


DEMO

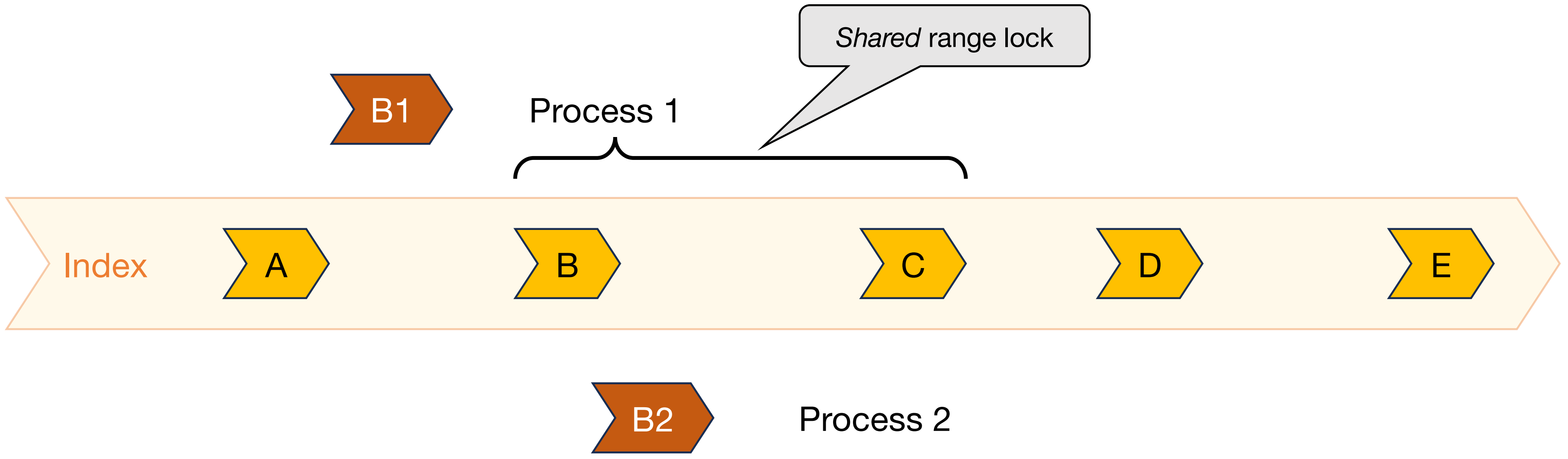
The serializable merge deadlock



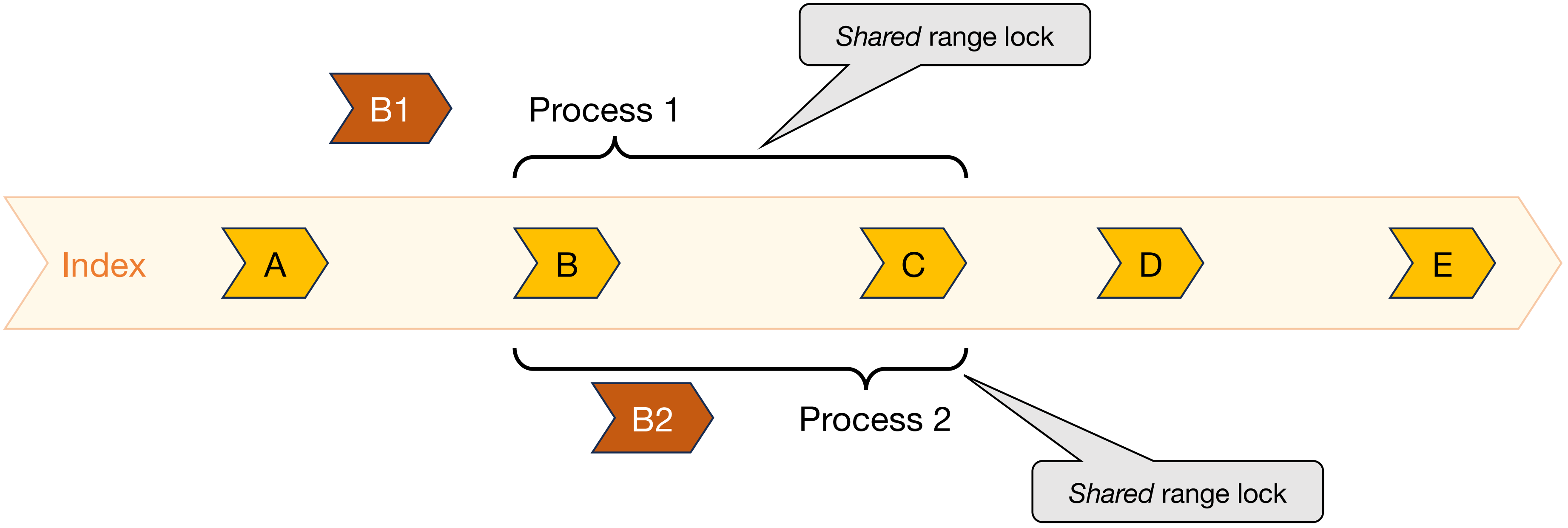
The serializable merge deadlock



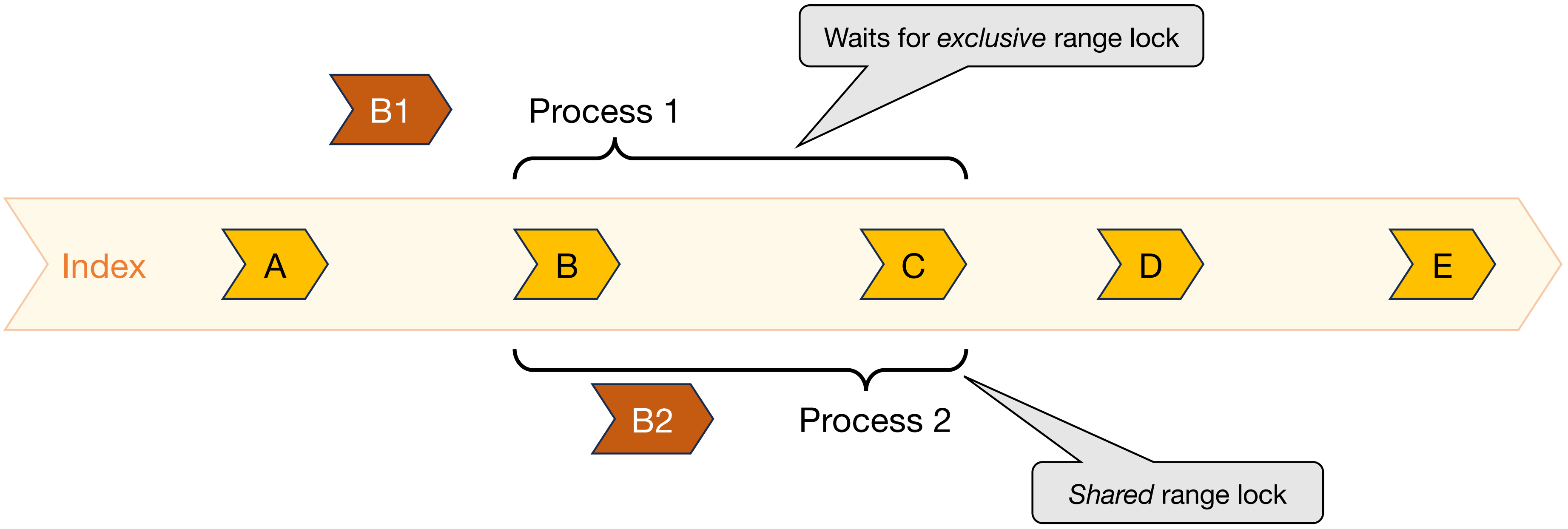
The serializable merge deadlock



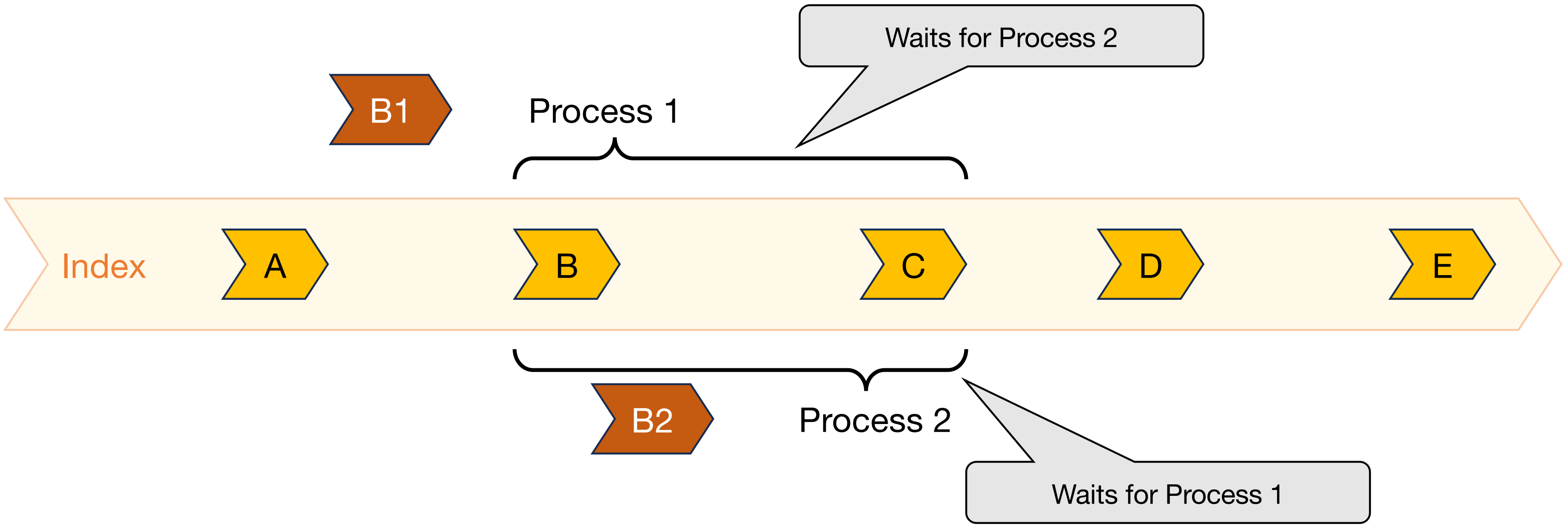
The serializable merge deadlock



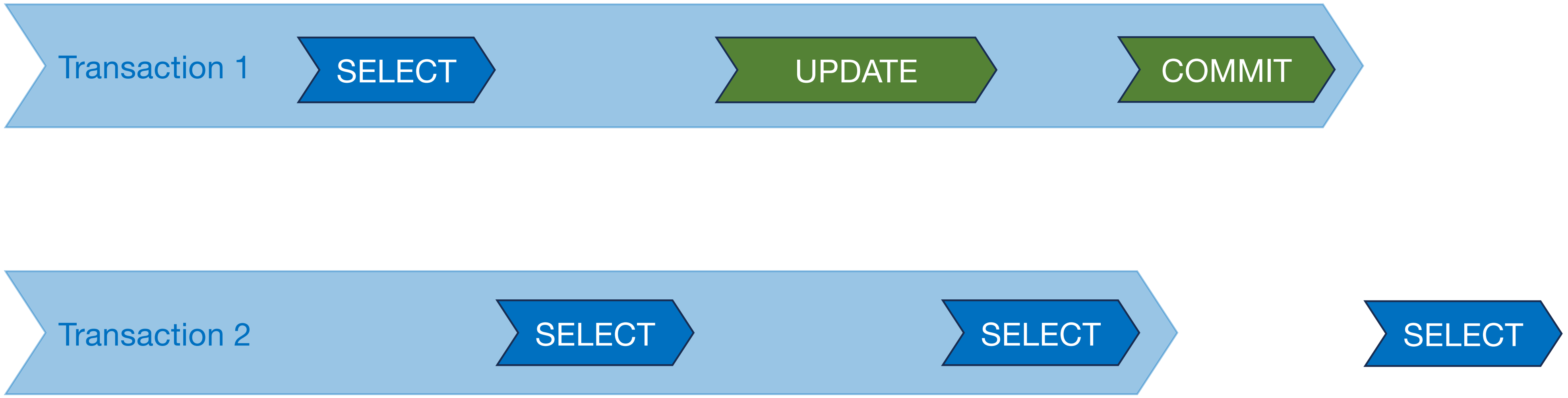
The serializable merge deadlock



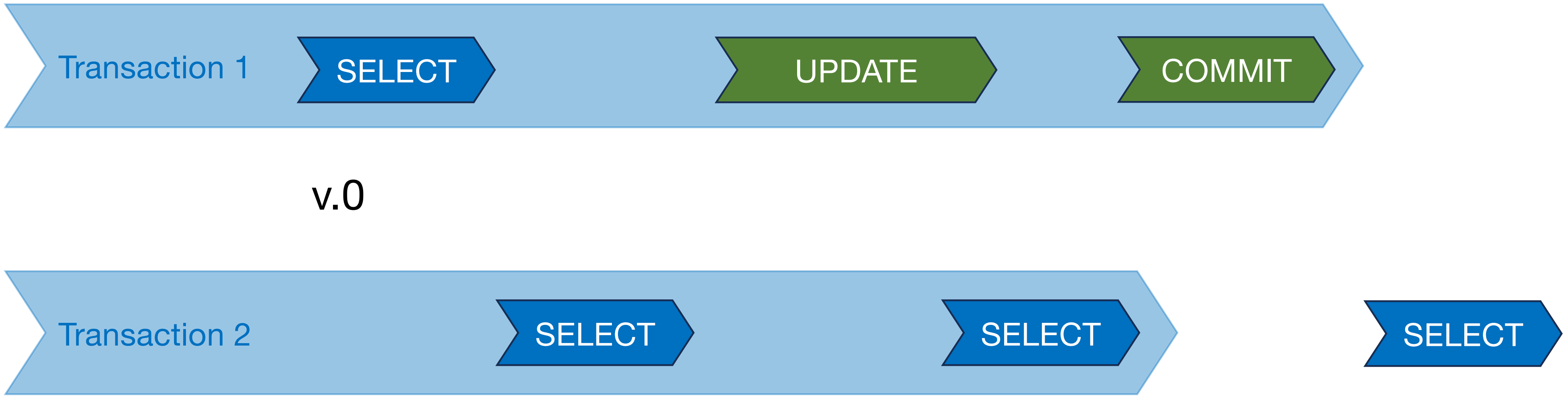
The serializable merge deadlock



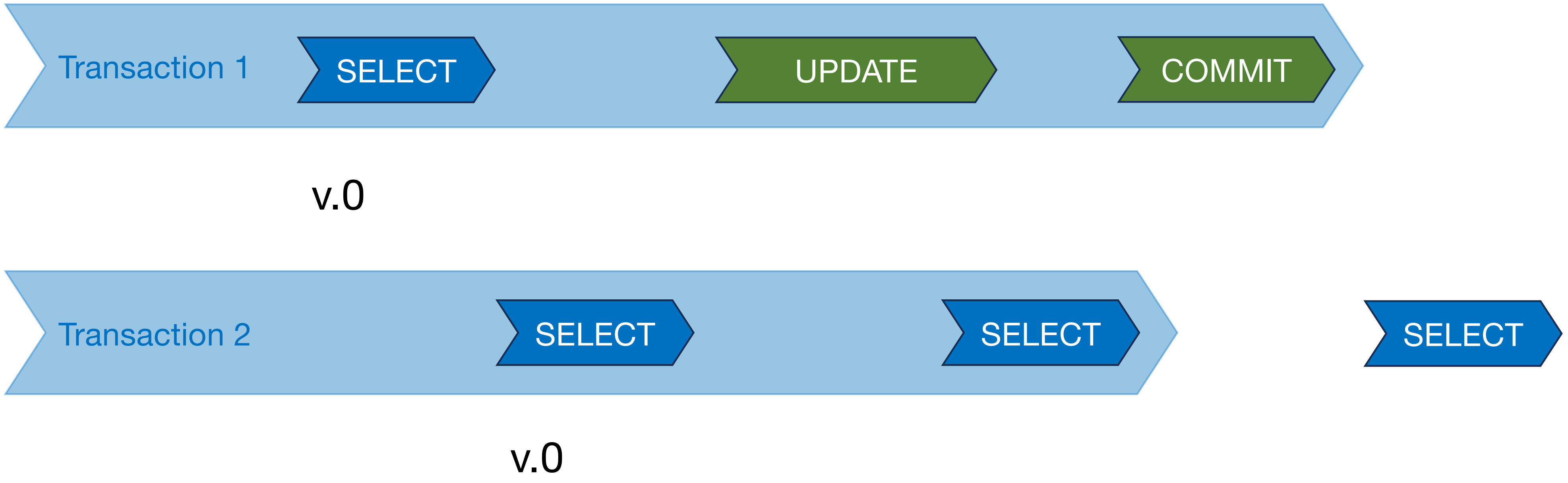
Snapshot isolation



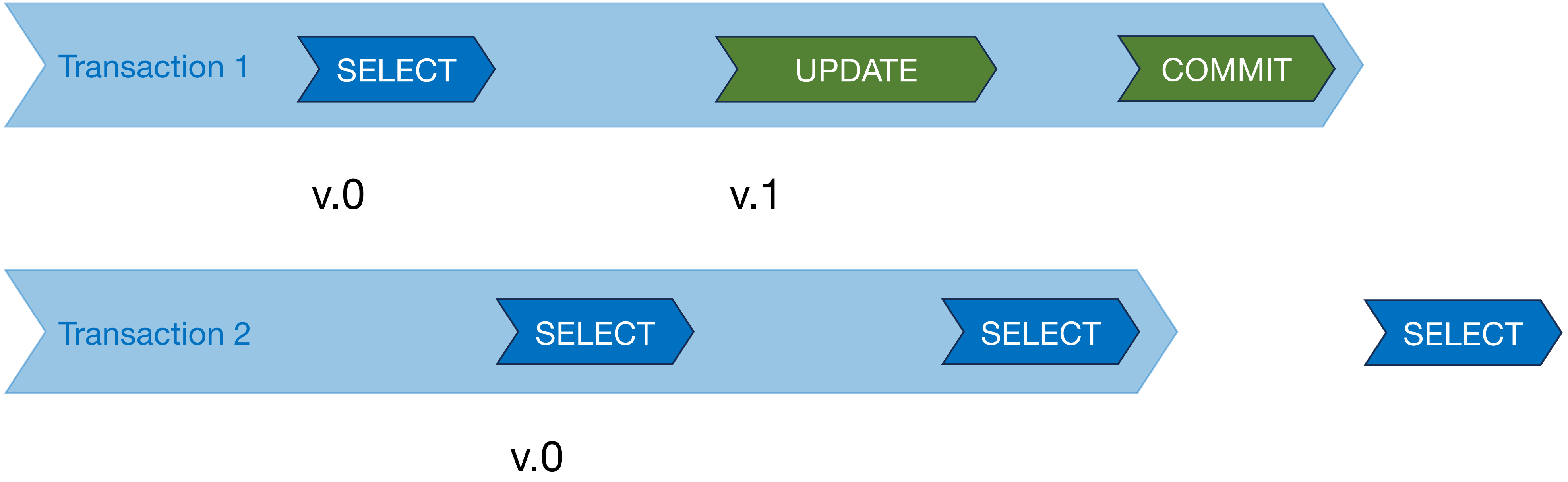
Snapshot isolation



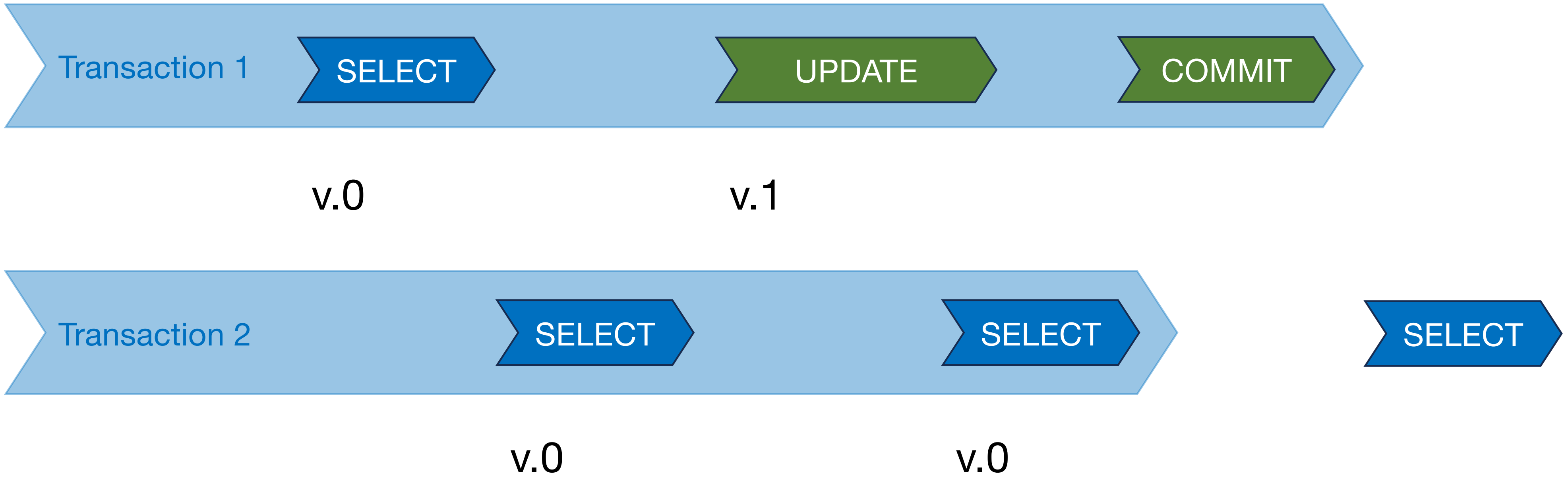
Snapshot isolation



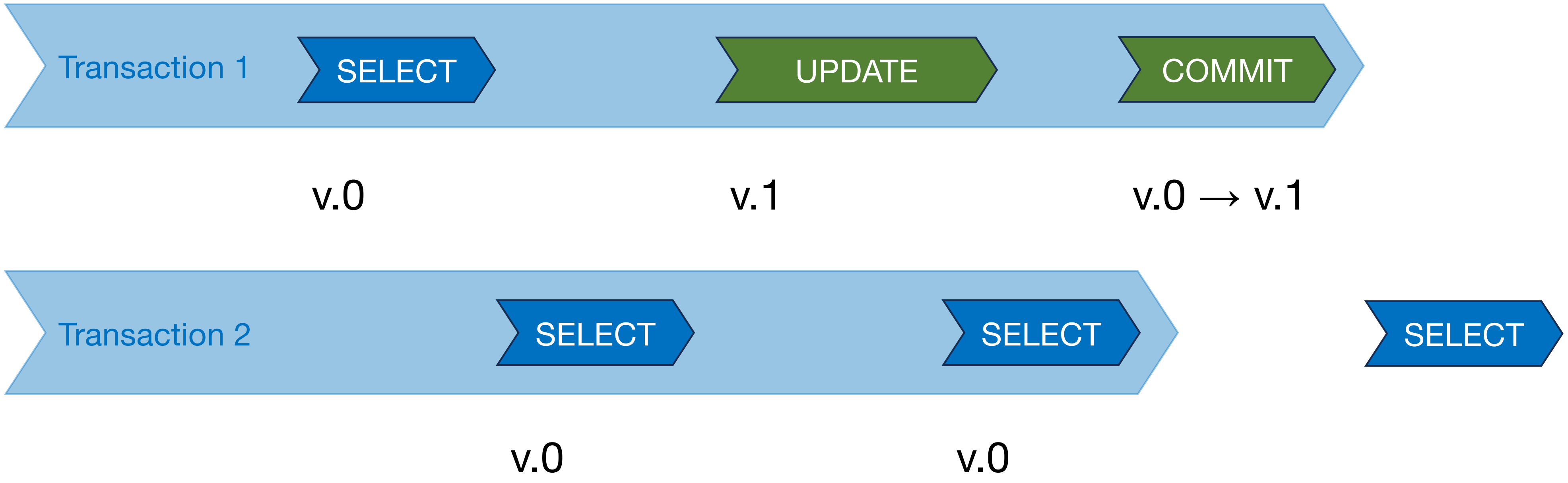
Snapshot isolation



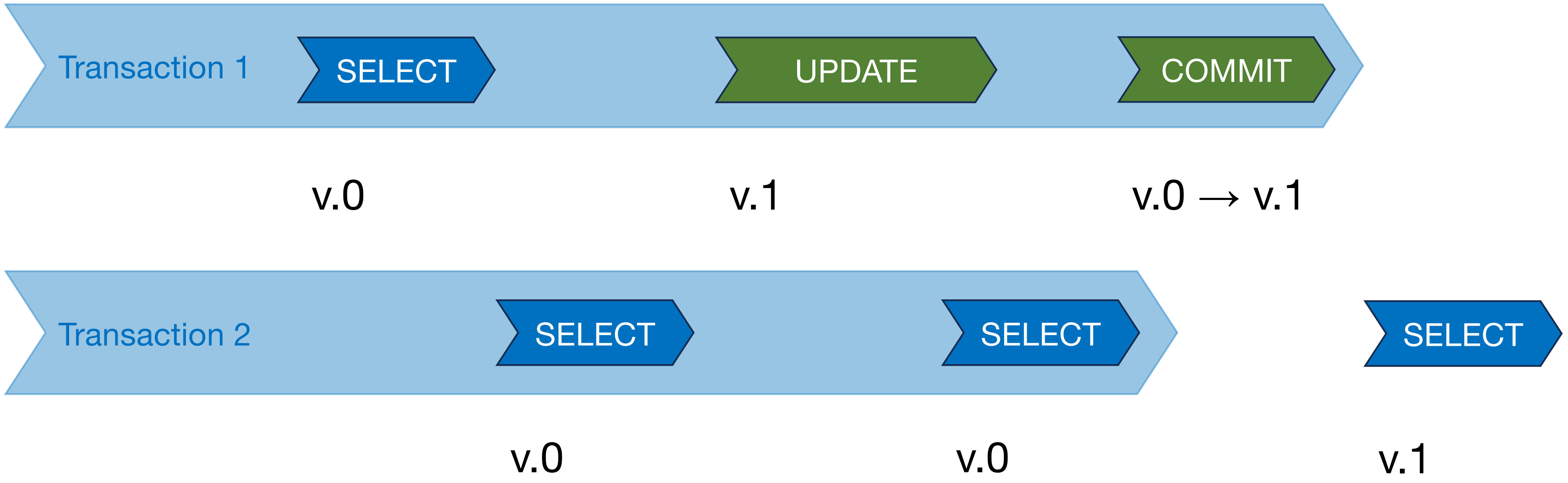
Snapshot isolation



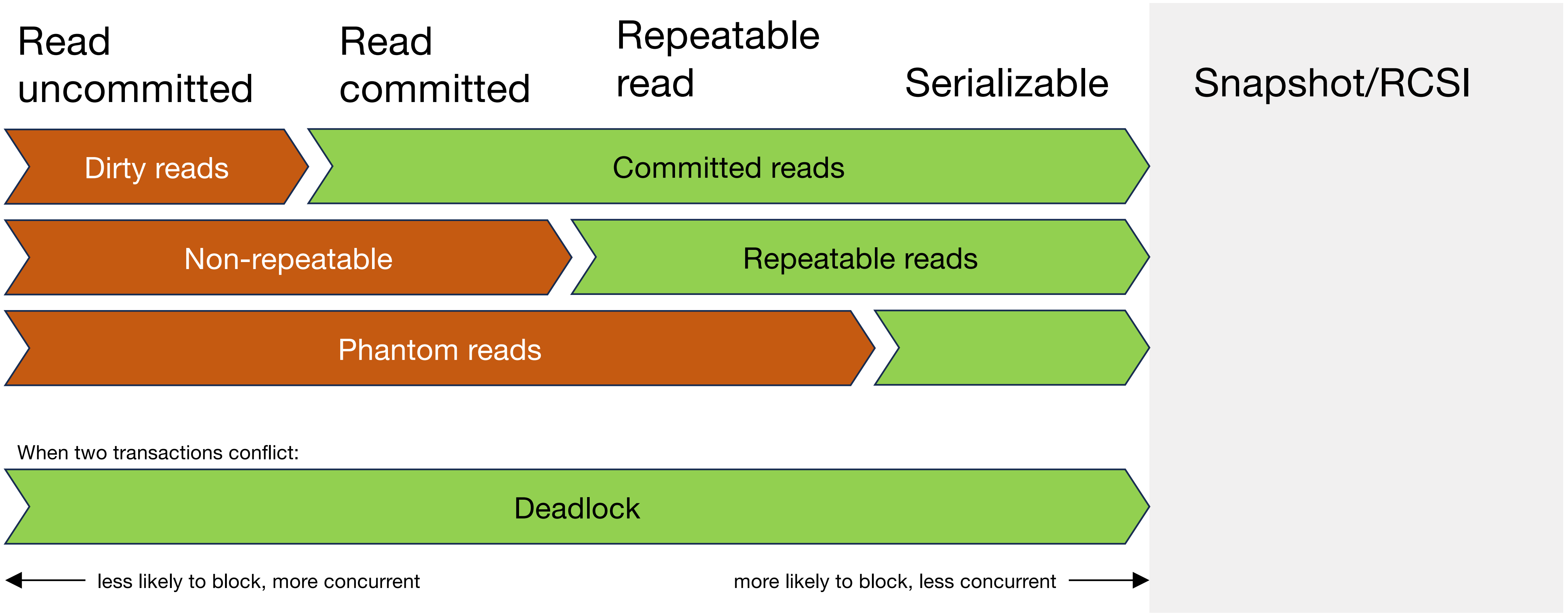
Snapshot isolation



Snapshot isolation

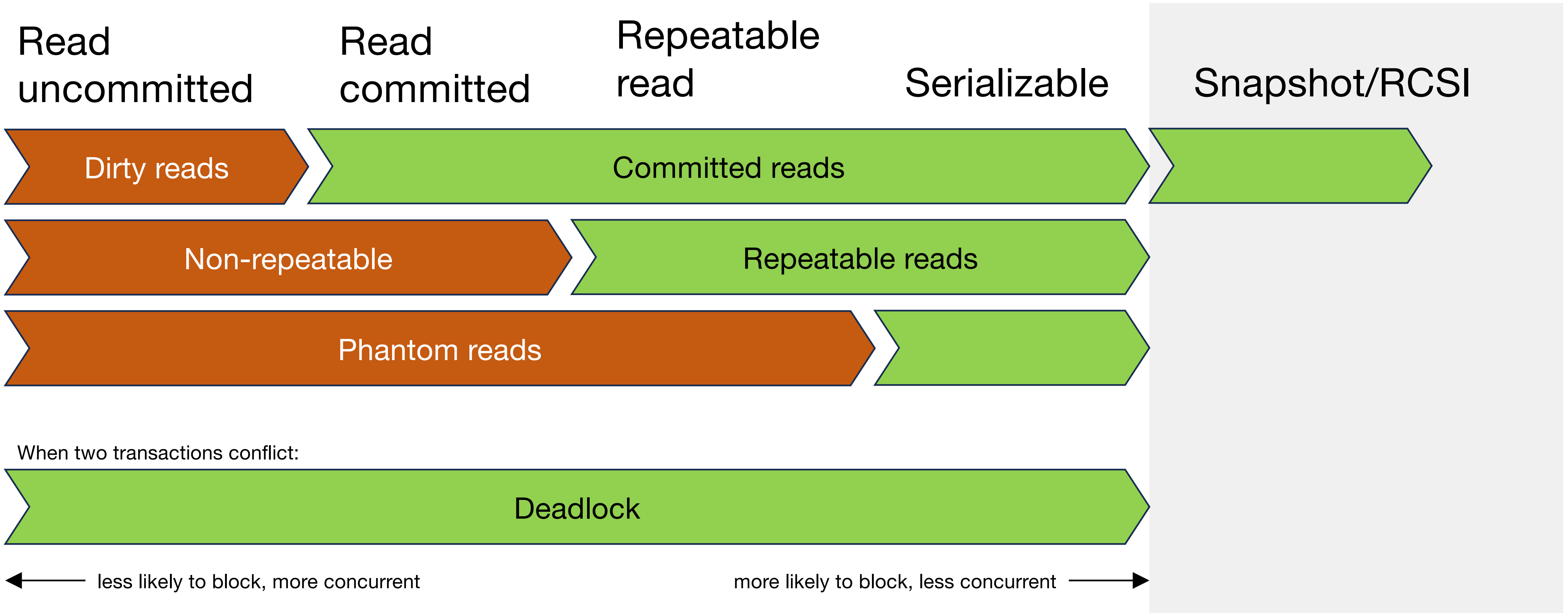


Snapshot

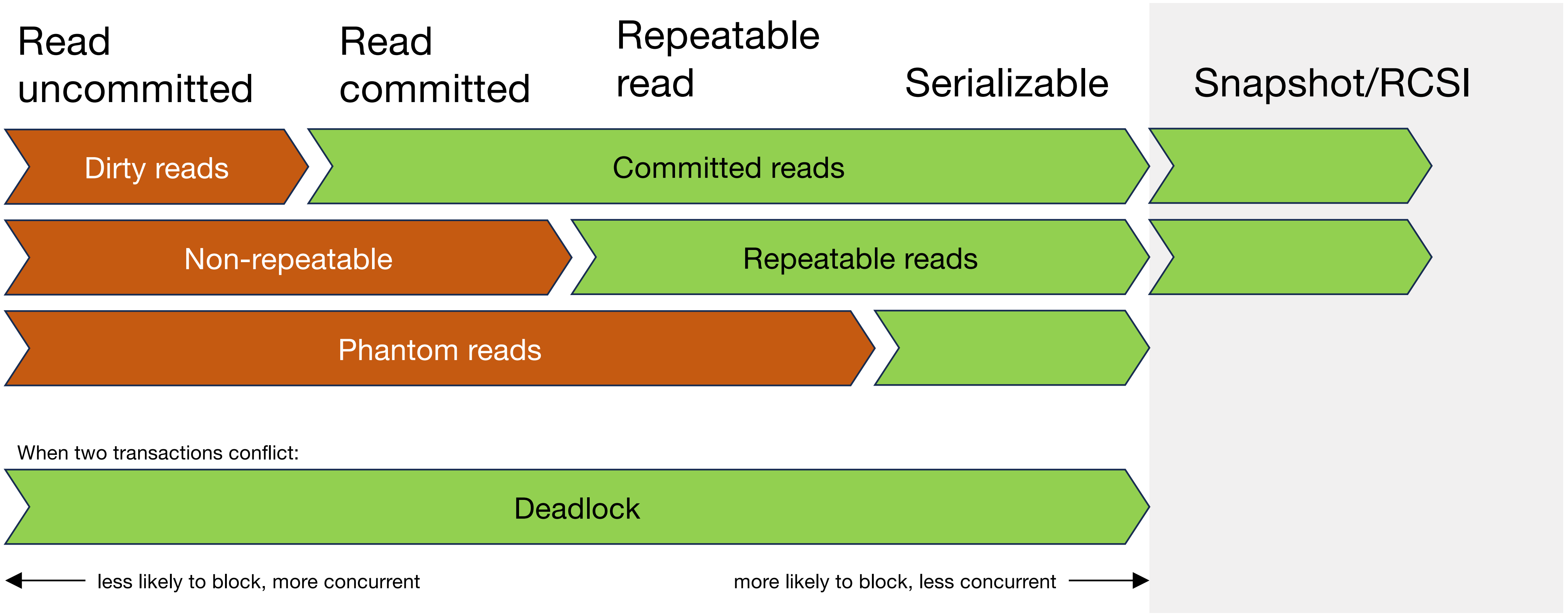


Read uncommitted
Read committed
Repeatable read
Serializable
Snapshot/RCSI
Dirty reads
Committed reads
Non-repeatable
Repeatable reads
Phantom reads
When two transactions conflict:
Deadlock
less likely to block, more concurrent
more likely to block, less concurrent

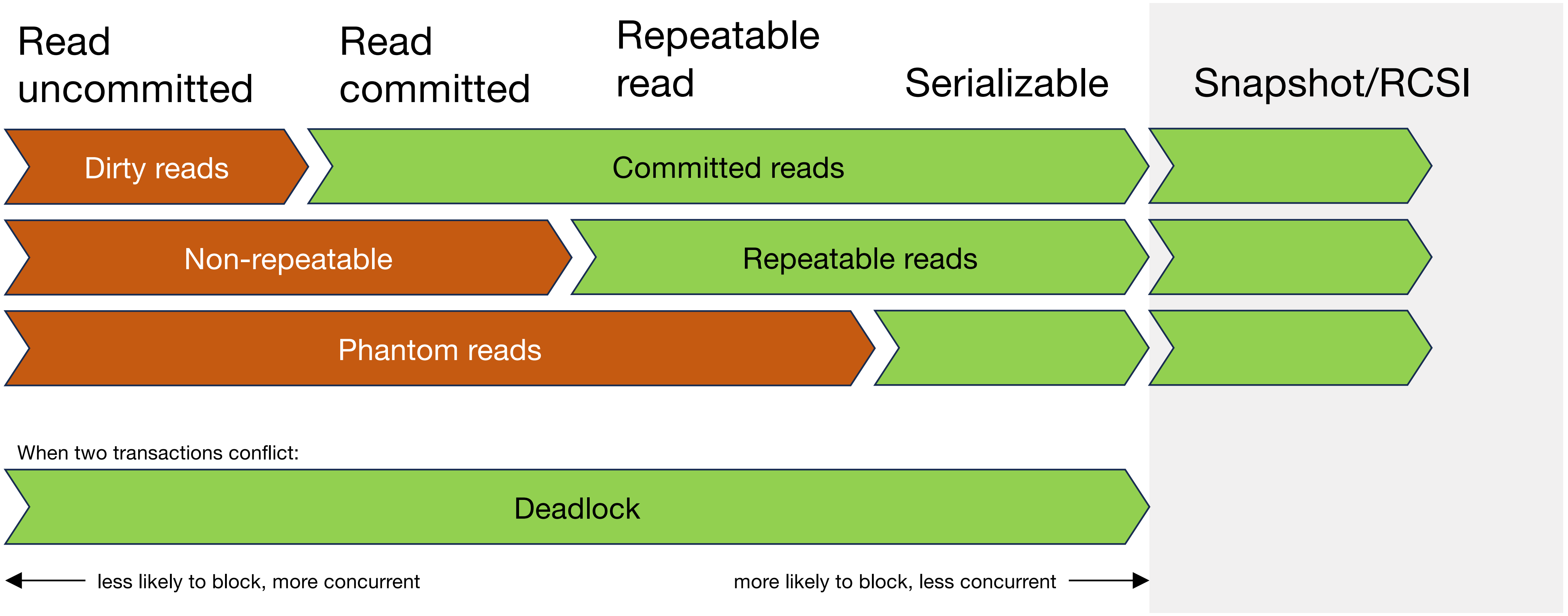
Snapshot


Read uncommitted
Read committed
Repeatable read
Serializable
Snapshot/RCSI
Dirty reads
Committed reads
Non-repeatable
Repeatable reads
Phantom reads
When two transactions conflict:
Deadlock
less likely to block, more concurrent
more likely to block, less concurrent

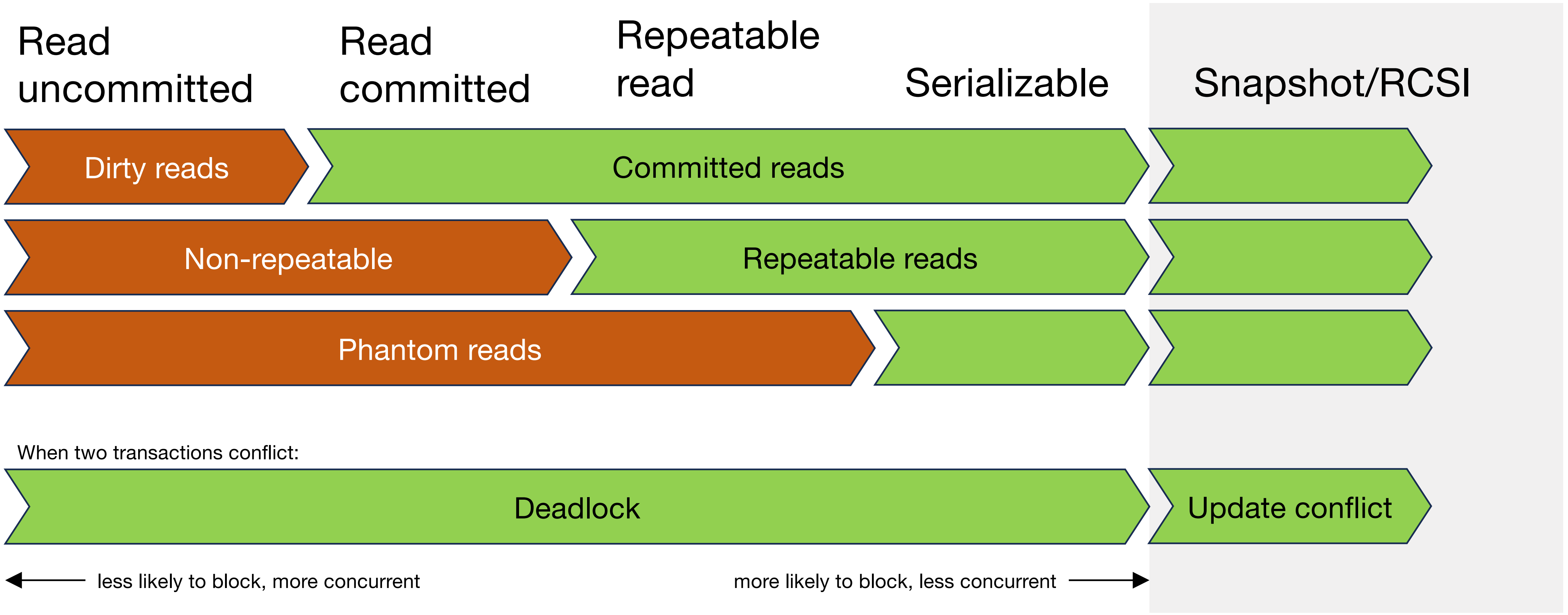
Snapshot


Read uncommitted
Read committed
Repeatable read
Serializable
Snapshot/RCSI
Dirty reads
Committed reads
Non-repeatable
Repeatable reads
Phantom reads
When two transactions conflict:
Deadlock
less likely to block, more concurrent
more likely to block, less concurrent

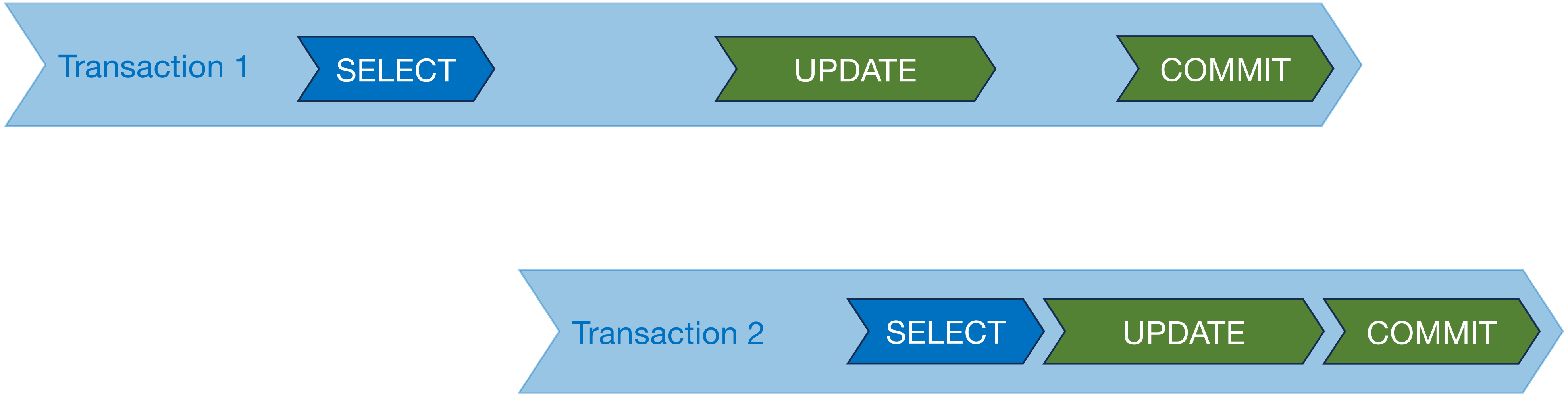
Snapshot


Read uncommitted
Read committed
Repeatable read
Serializable
Snapshot/RCSI
Dirty reads
Committed reads
Non-repeatable
Repeatable reads
Phantom reads
When two transactions conflict:
Deadlock
less likely to block, more concurrent
more likely to block, less concurrent

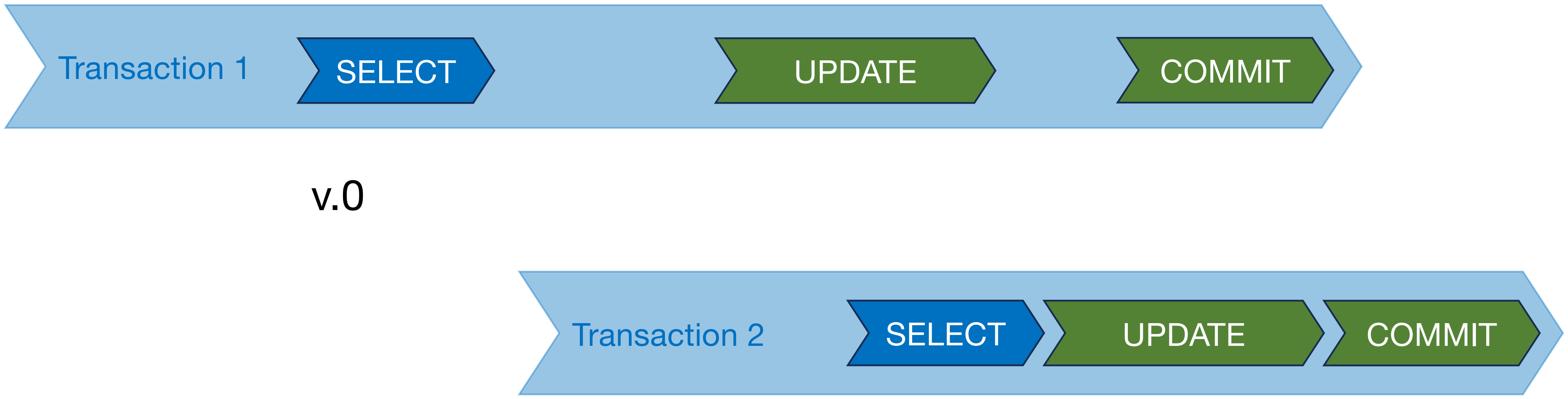
Snapshot


Read uncommitted
Read committed
Repeatable read
Serializable
Snapshot/RCSI
Dirty reads
Committed reads
Non-repeatable
Repeatable reads
Phantom reads
When two transactions conflict:
Deadlock
Update conflict
less likely to block, more concurrent
more likely to block, less concurrent

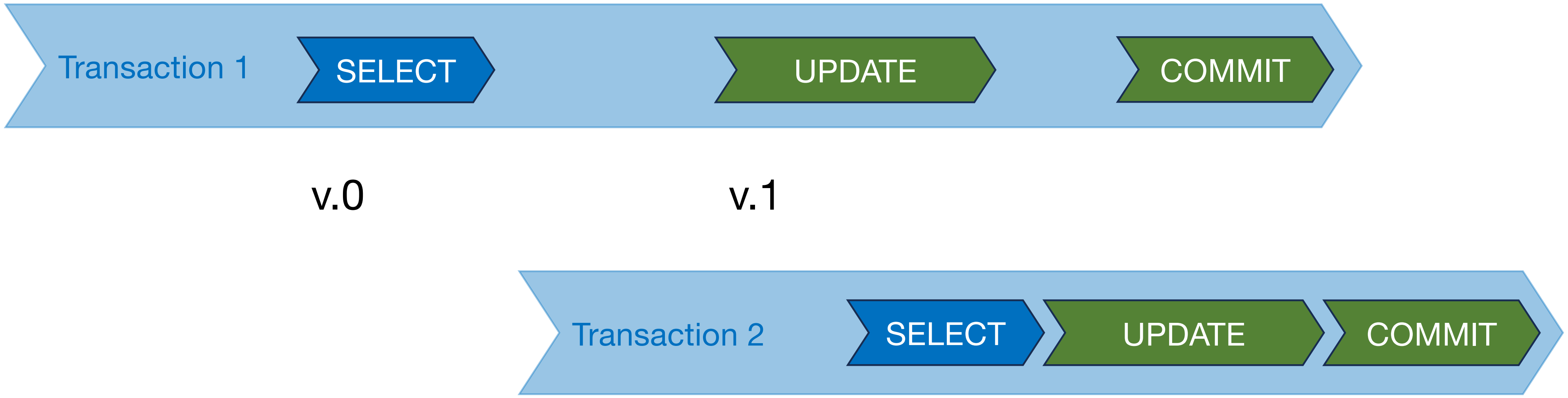
Snapshot isolation: Update conflict



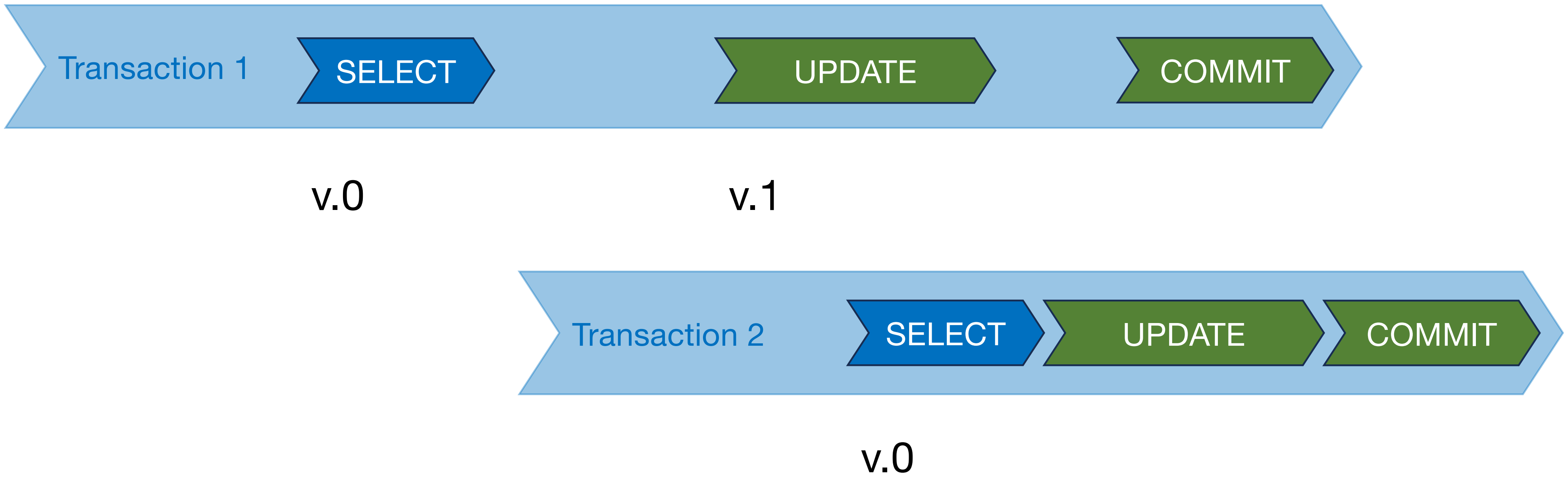
Snapshot isolation: Update conflict



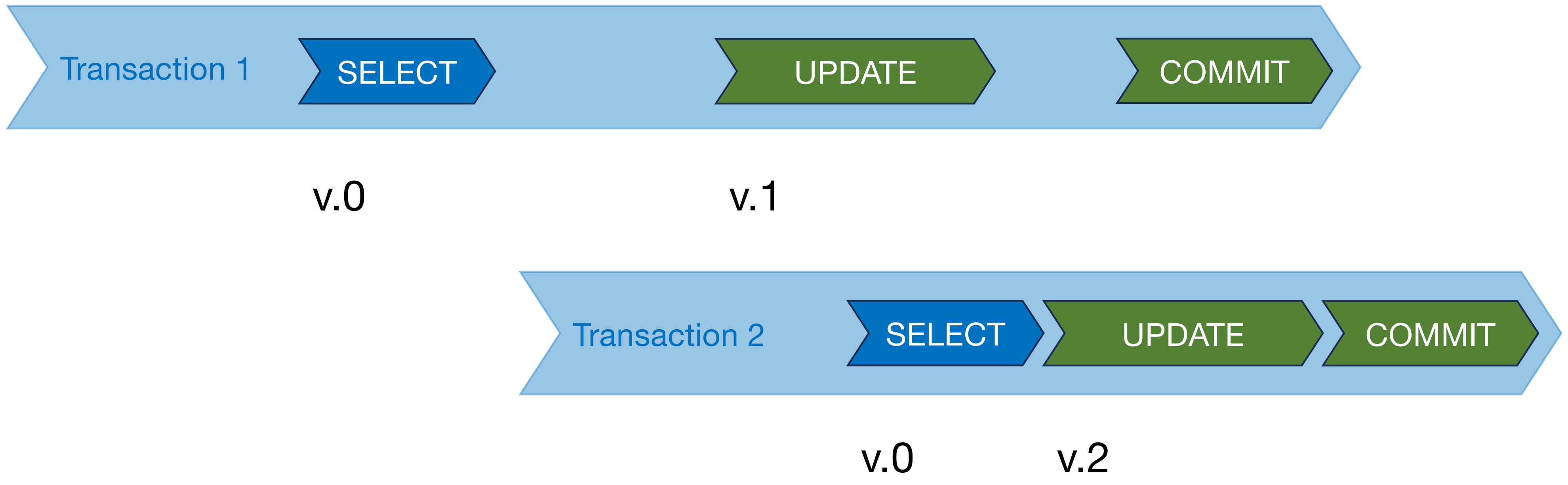
Snapshot isolation: Update conflict



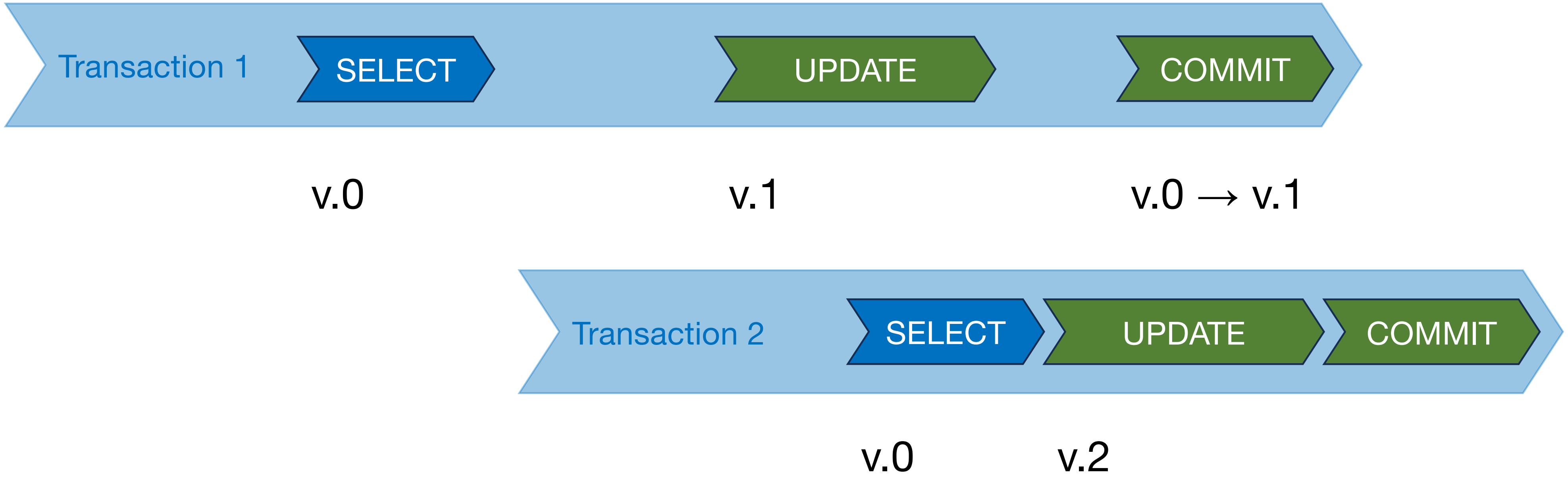
Snapshot isolation: Update conflict



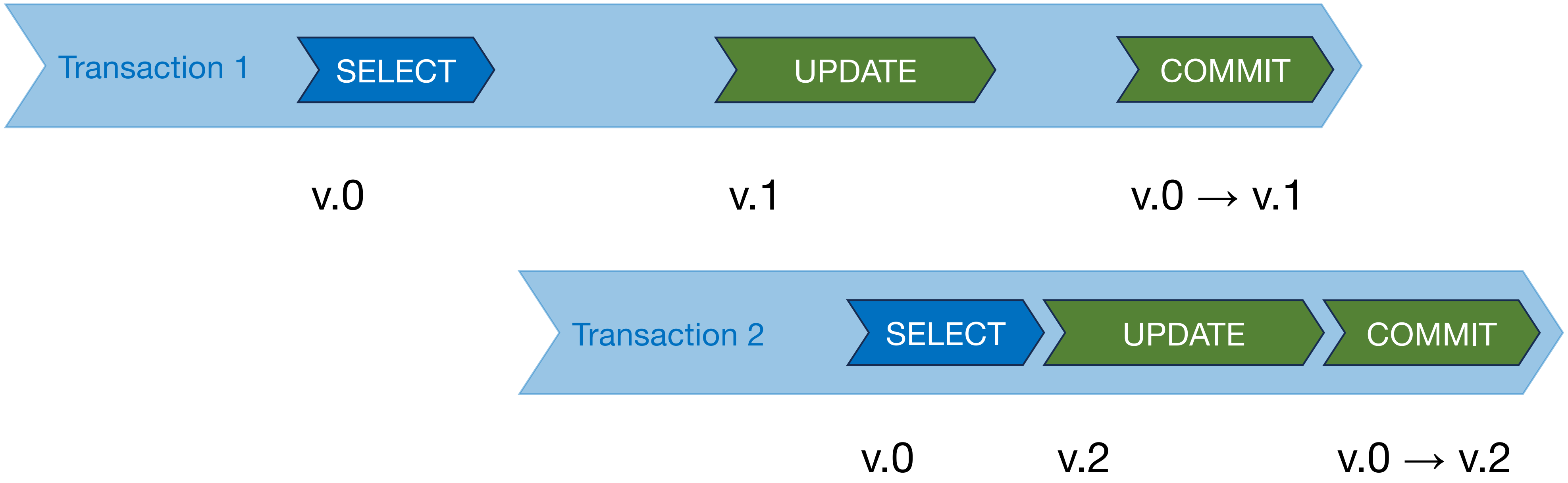
Snapshot isolation: Update conflict



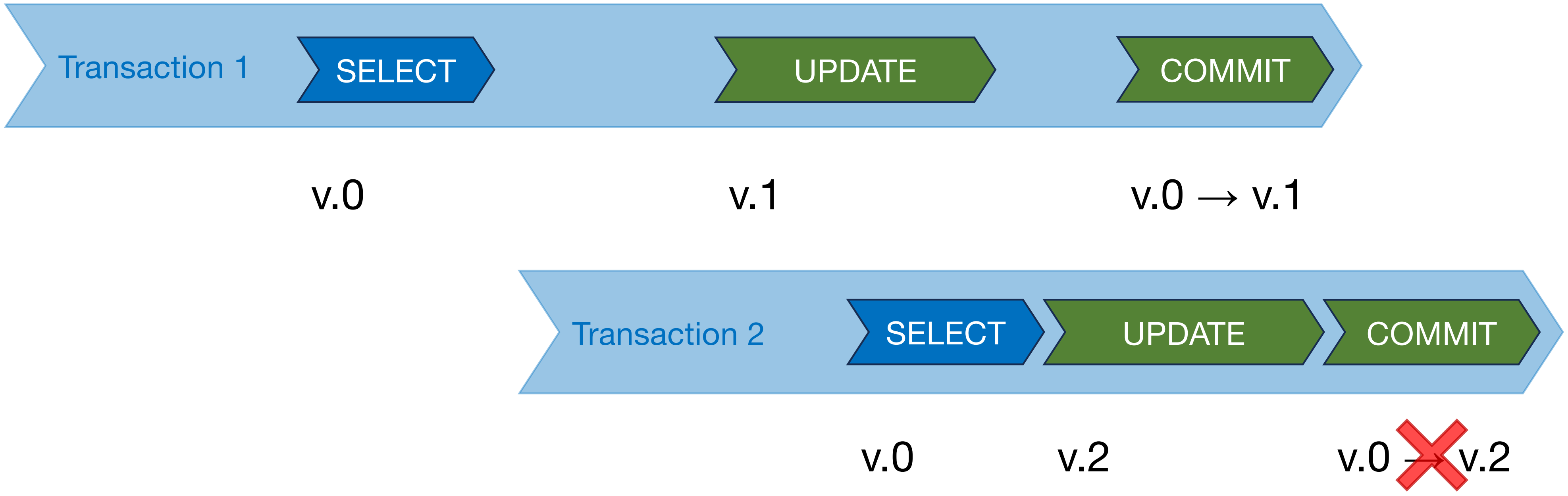
Snapshot isolation: Update conflict



Snapshot isolation: Update conflict



Snapshot isolation: Update conflict



DEMO

Read Committed Snapshot Isolation vs. Snapshot Isolation

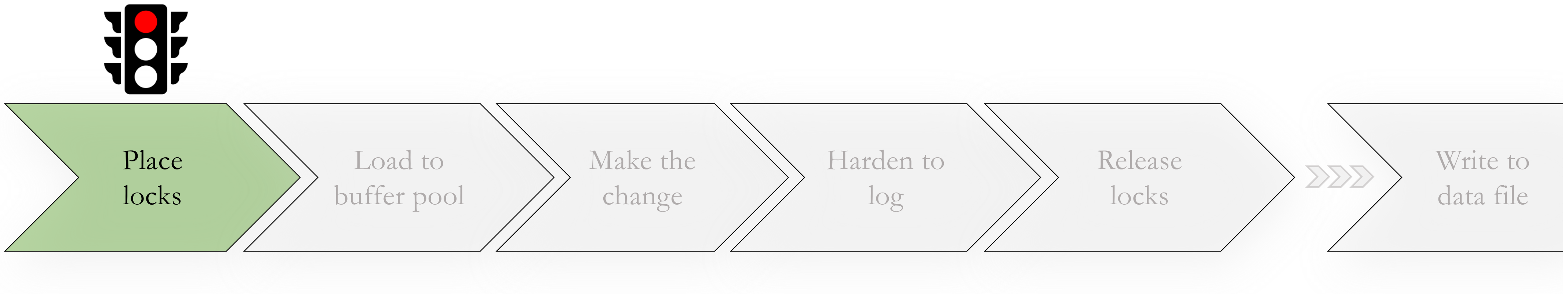
- Snapshot Isolation is set per *transaction*.
- Snapshot Isolation protects the *transaction*.
- Snapshot Isolation requires code change.

- Read Committed Snapshot Isolation is a *database* setting.
- Read Committed Snapshot Isolation protects the *statement*.

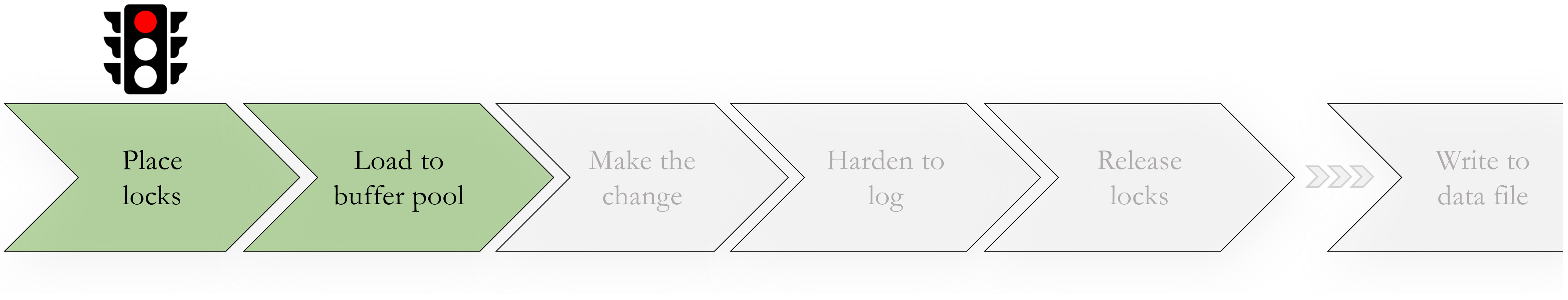
- Both require testing, because they behave differently.

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

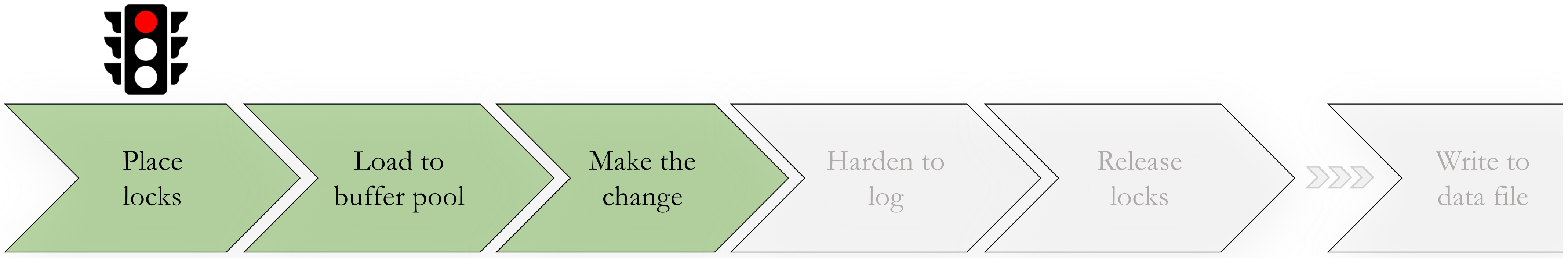
How the transaction log works



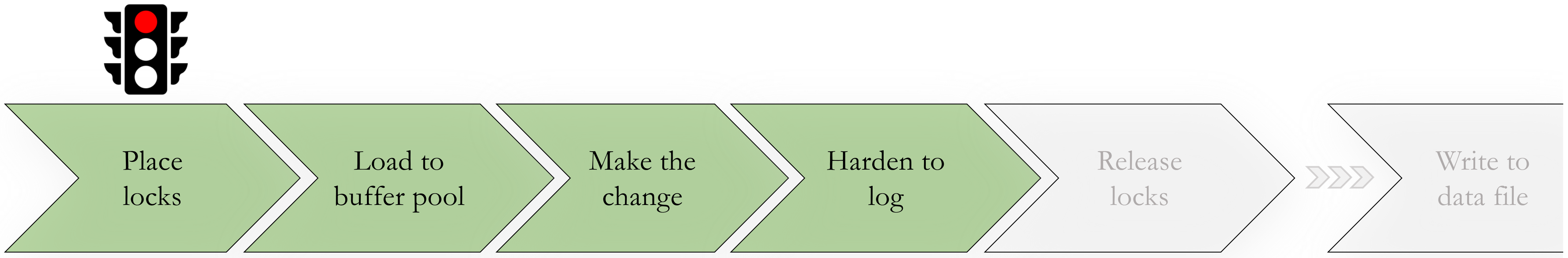
How the transaction log works



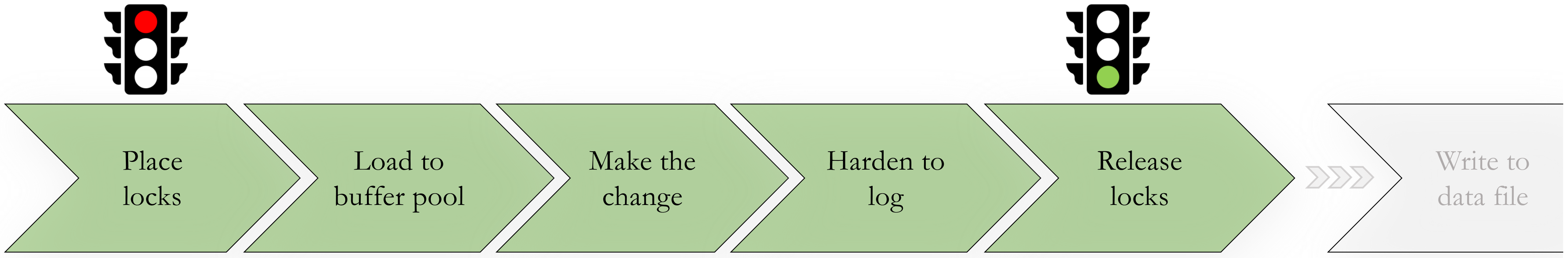
How the transaction log works



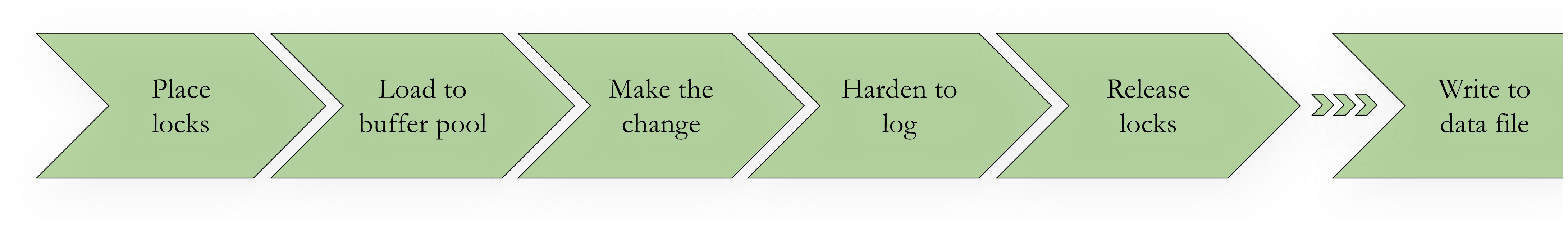
How the transaction log works



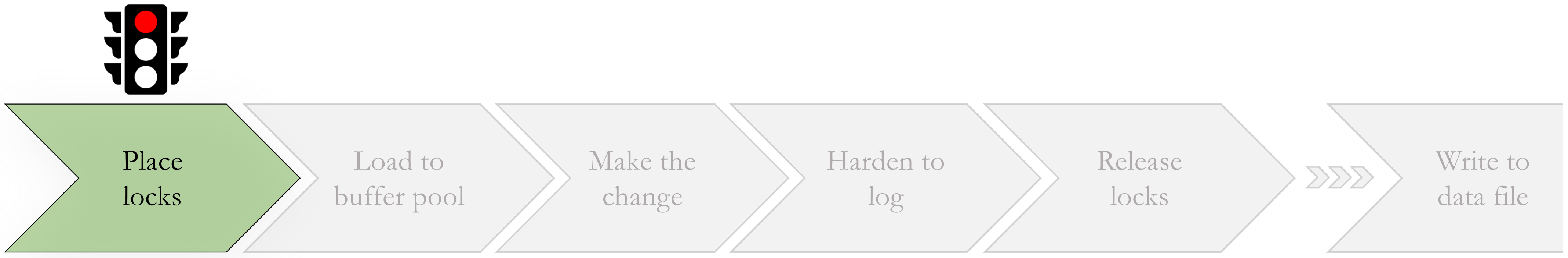
How the transaction log works



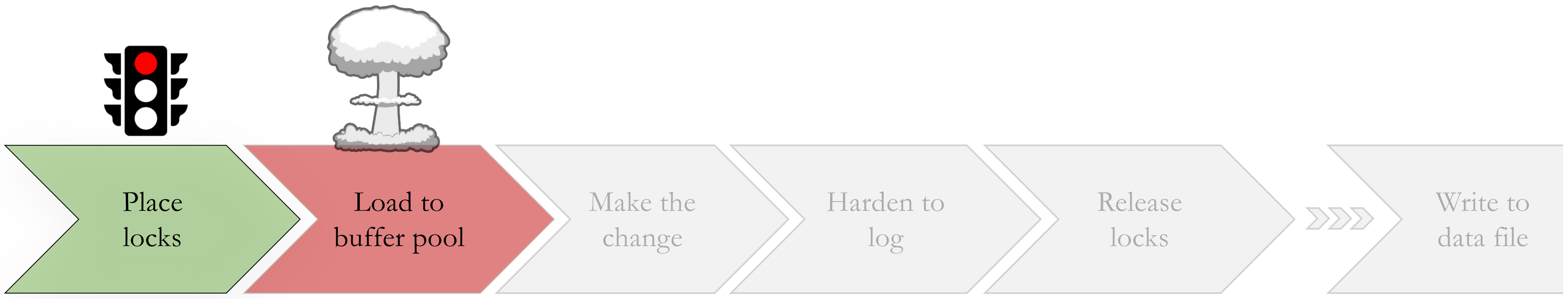
How the transaction log works



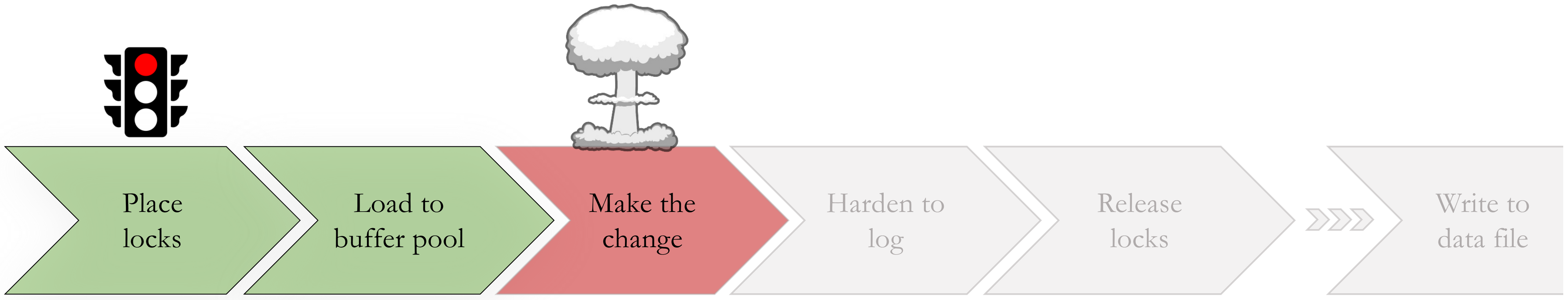
Durability


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

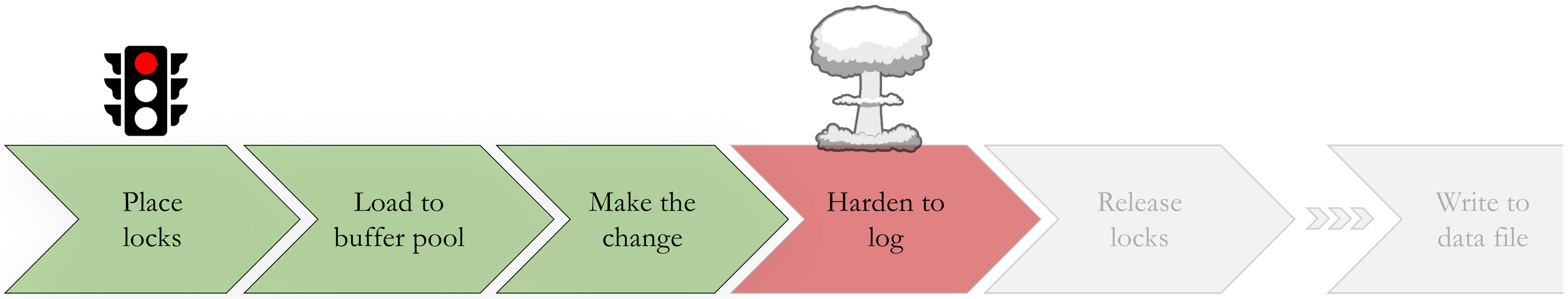
Durability


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

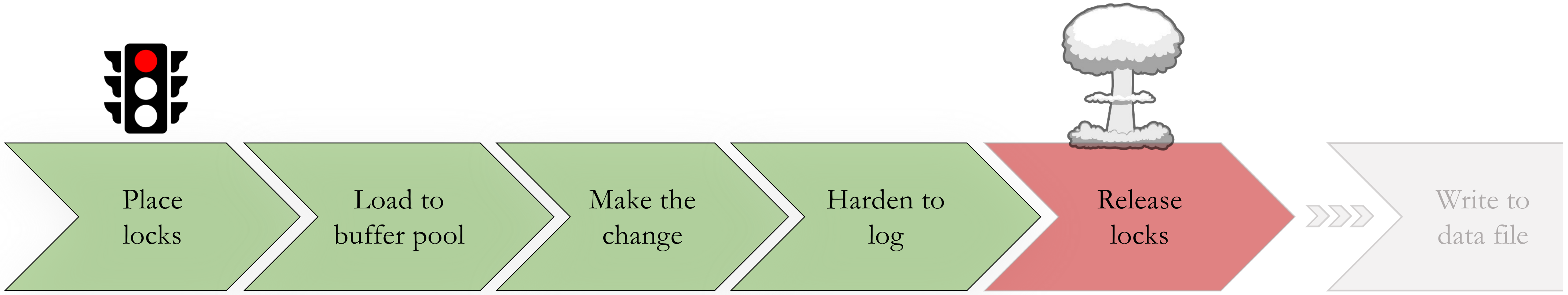
Durability


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

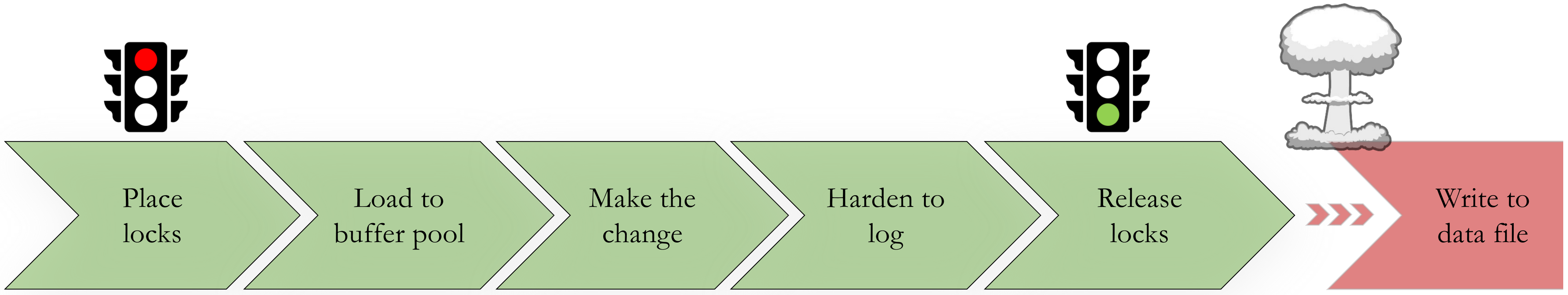
Durability


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

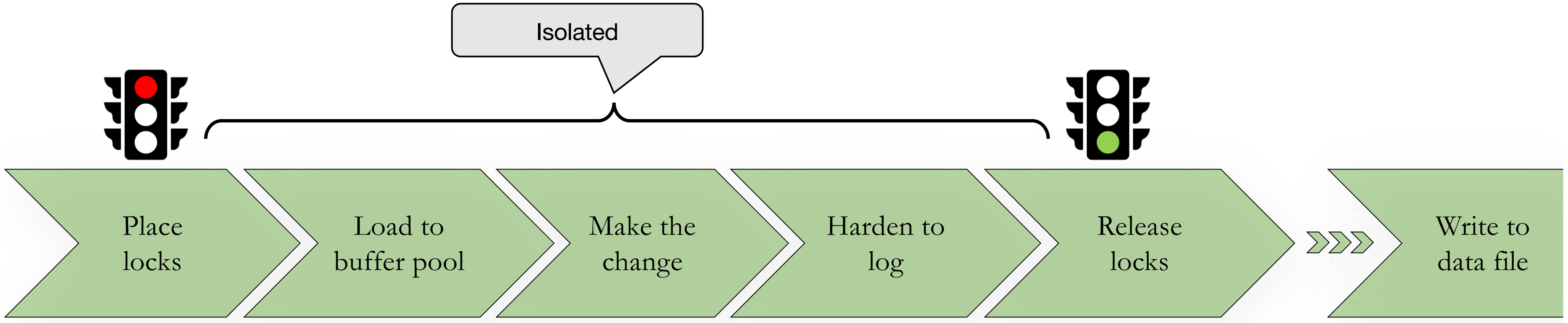
Durability


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

Durability

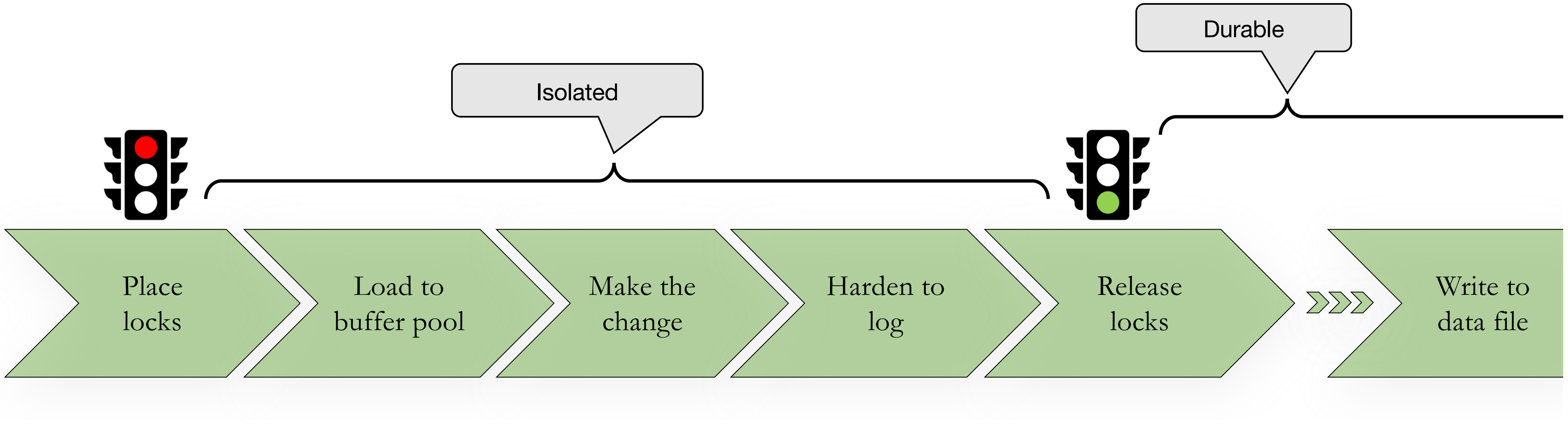

Place locks
Load to buffer pool
Make the change
Harden to log
Release locks
Write to data file

Durability

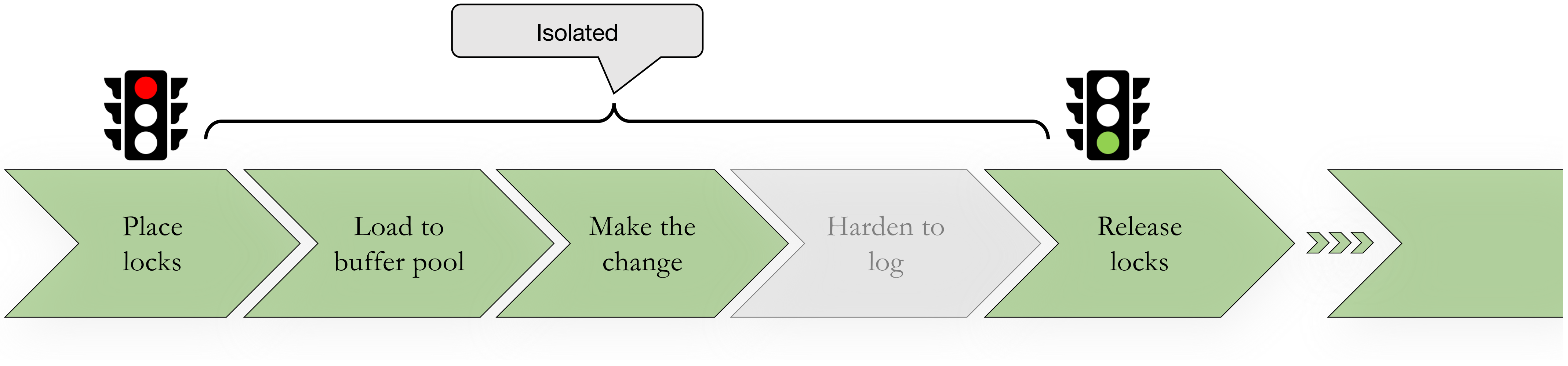


Isolated
Place locks
Load to buffer pool
Make the change
Harden to log
Release locks
Write to data file

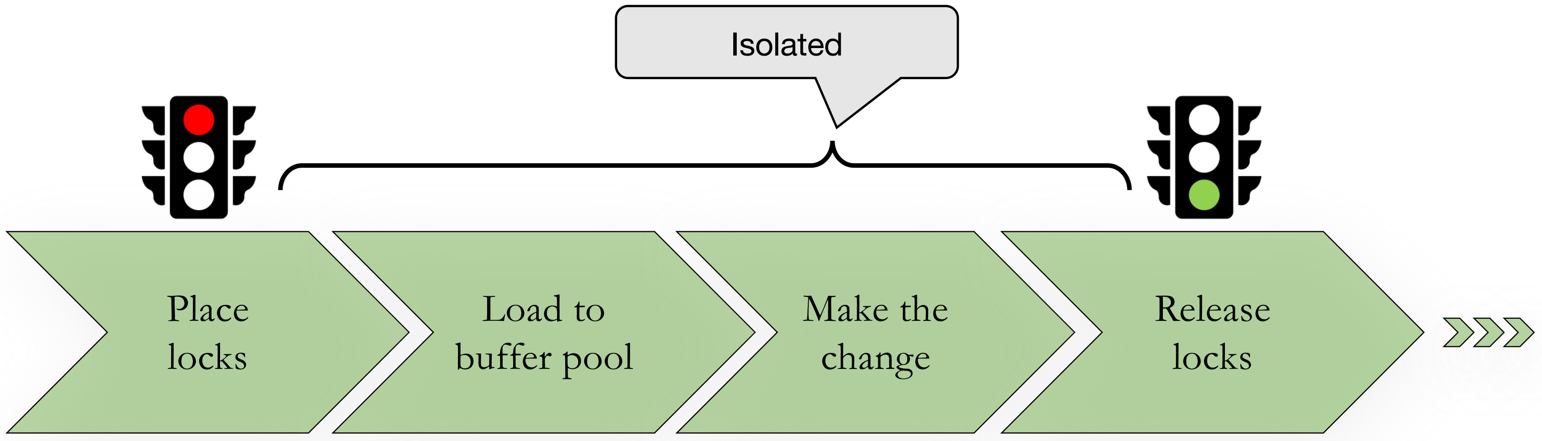
Durability


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

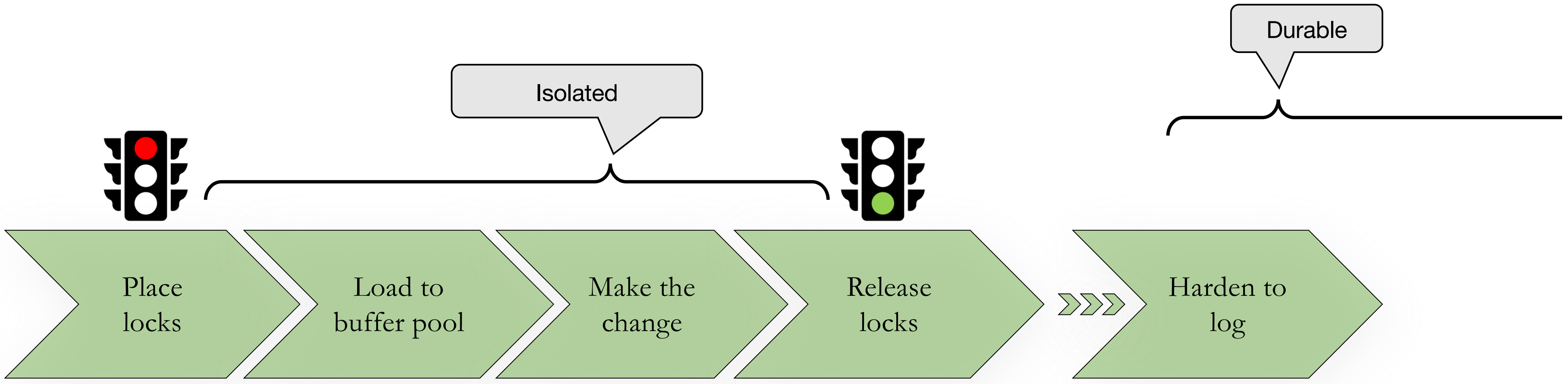
Delayed durability



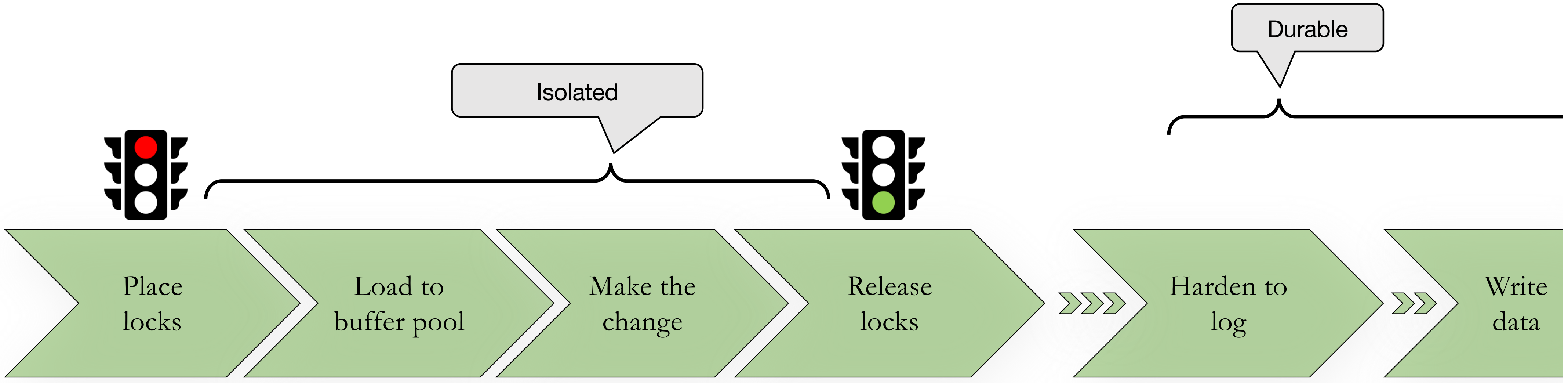
Delayed durability



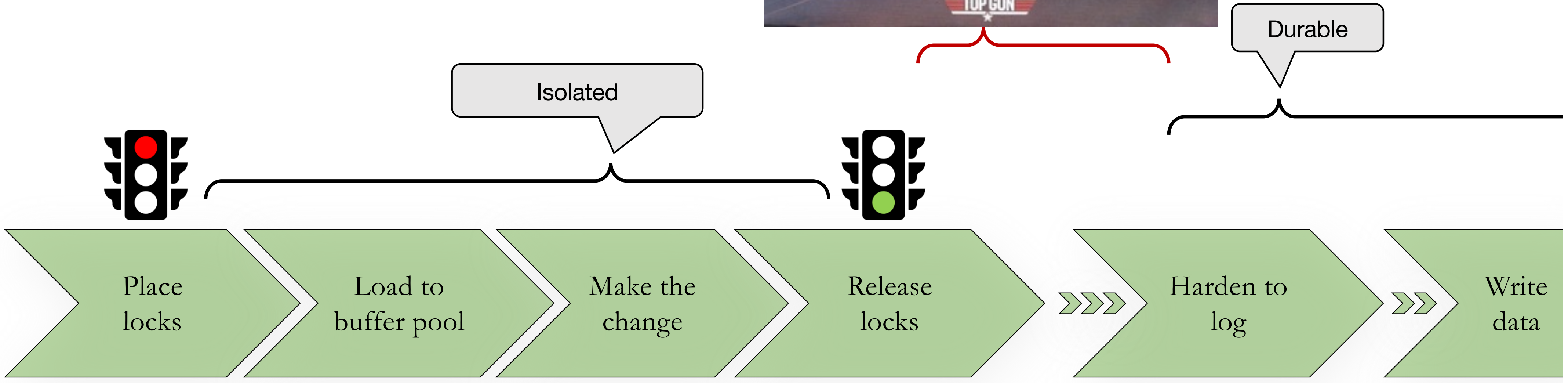
Delayed durability



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 uses a form of Delayed Durability under the hood

Key take-aways

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

Give me feedback. Please?

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