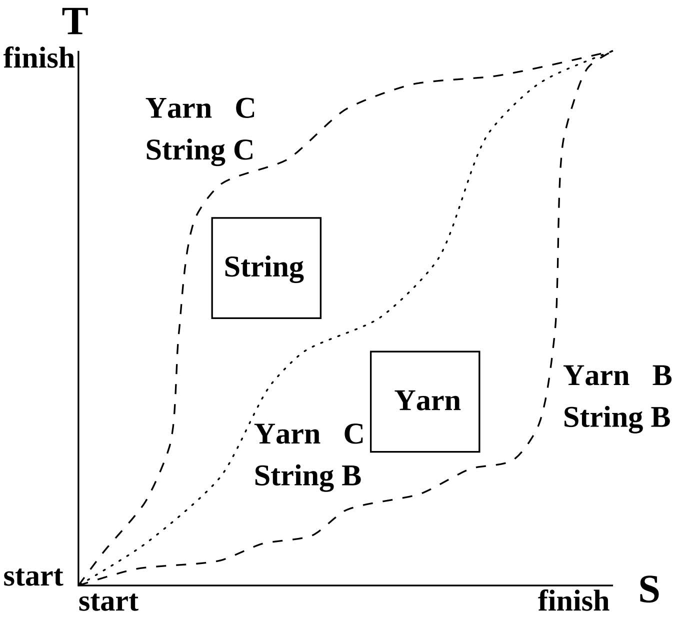
The author gratefully acknowledges support from the taxpayers of Québec and of Canada who have paid his salary and research grants while this work was developed at McGill University, and from his students (who built the implementations and investigated the data structures and algorithms) and their funding agencies.

Concurrency: Sharing Volatile Files

Item	*Type*
`Yarn`	`A`
`String`	`A`
`Ball`	`B`
`Sandal`	`C`



Transaction S:
Change type `A` to `B`

Transaction T:
Change type `A` to `C`

Acceptable outcomes (serial or *serializable*):

`Yarn`	`B`	`Yarn`	`C`
`String`	`B`	`String`	`C`

Unacceptable outcomes (unserializable):

`Yarn`	`B`	`Yarn`	`C`
`String`	`C`	`String`	`B`

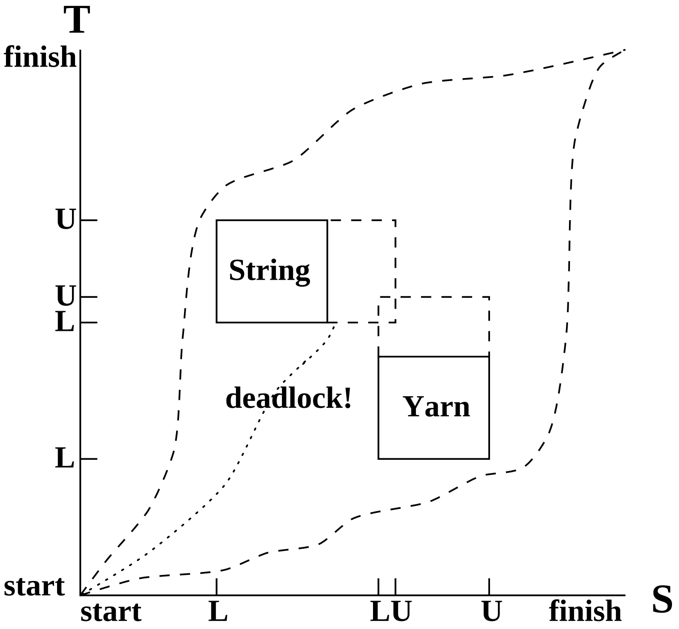
(For instance, there might be a *consistency constraint* that *Yarn* and *String* have the same type.)

Concurrency: Two-Phase Locking

Never lock after starting to unlock!

(Phase 1: lock. Phase 2: unlock)

Apply rule to each transaction *independently* of other transactions.



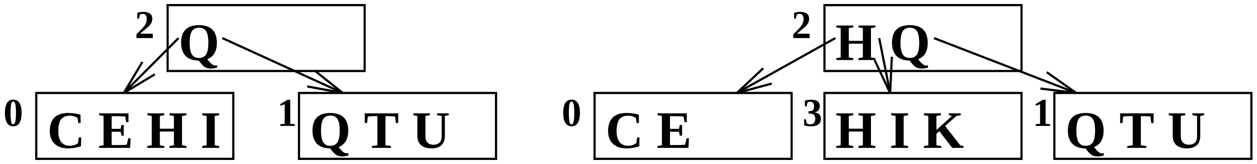
Deadlock is possible.

T. H. Merrett

Concurrency and File Structures

B$^+$-trees

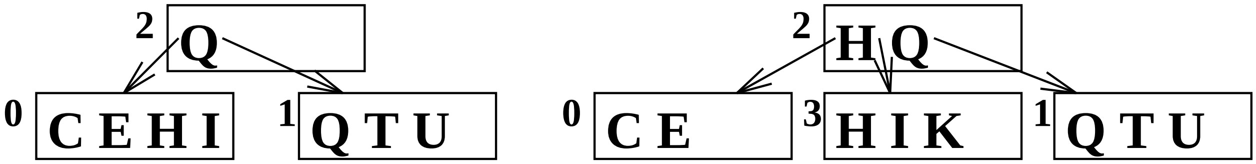
Insert `K`, concurrently read `I`



1. 2PL

insert	*read*
lock 2	
lock 0	
write 0	
unlock 0	
write 3	
write 2	
unlock 2	
	lock 2
	lock 3
	unlock 3
	unlock 2

Concurrency and B-trees



2. “Lock Conversion” variant of “Lock Coupling”

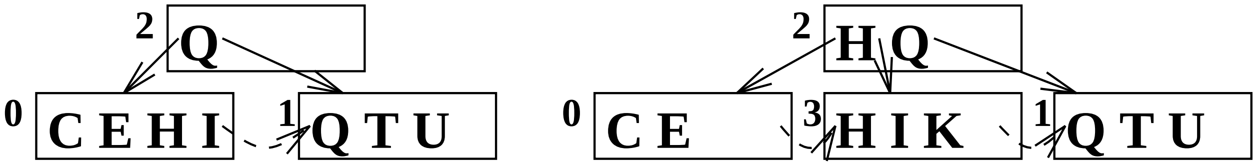
insert	*read*
rLock 2	rLock 2
	rLock 0
	unlock 0
	unlock 2
wLock 0	
wLock 2	
write 0	
unlock 0	
write 2	
write 3	
unlock 2	

Lock coupling: don’t unlock node before locking children!

Serializable, non-2PL: Bernstein, Hadzilacos & Goodman, 1987.

Lock conversion: change rLock to wLock if needed!

Concurrency and B-trees



3. Using Links

Hold at most 1 lock (no deadlock)
and modify search procedure.

insert	*read*
rLock 2	rLock 2
unlock 2	unlock 2 write 3
wLock 0	
write 0	
unlock 0	
	rLock 0
wLock 2	rLock 3
write 2	
unlock 2	

Concurrency: Other Dynamic Files

1. Order-Preserving

- Can all use links in similar ways.
- E.g., tries (next).
- E.g., dynamic multipaging: need d links.

2. Linear Hashing

- If no overflows, only 1 block involved.
- If overflow chain order-preserving, has links already
- Normally overflow chain adds to end: no problem.
- Current-split pointer may have changed: search semantics must check addresses given by both hash functions.

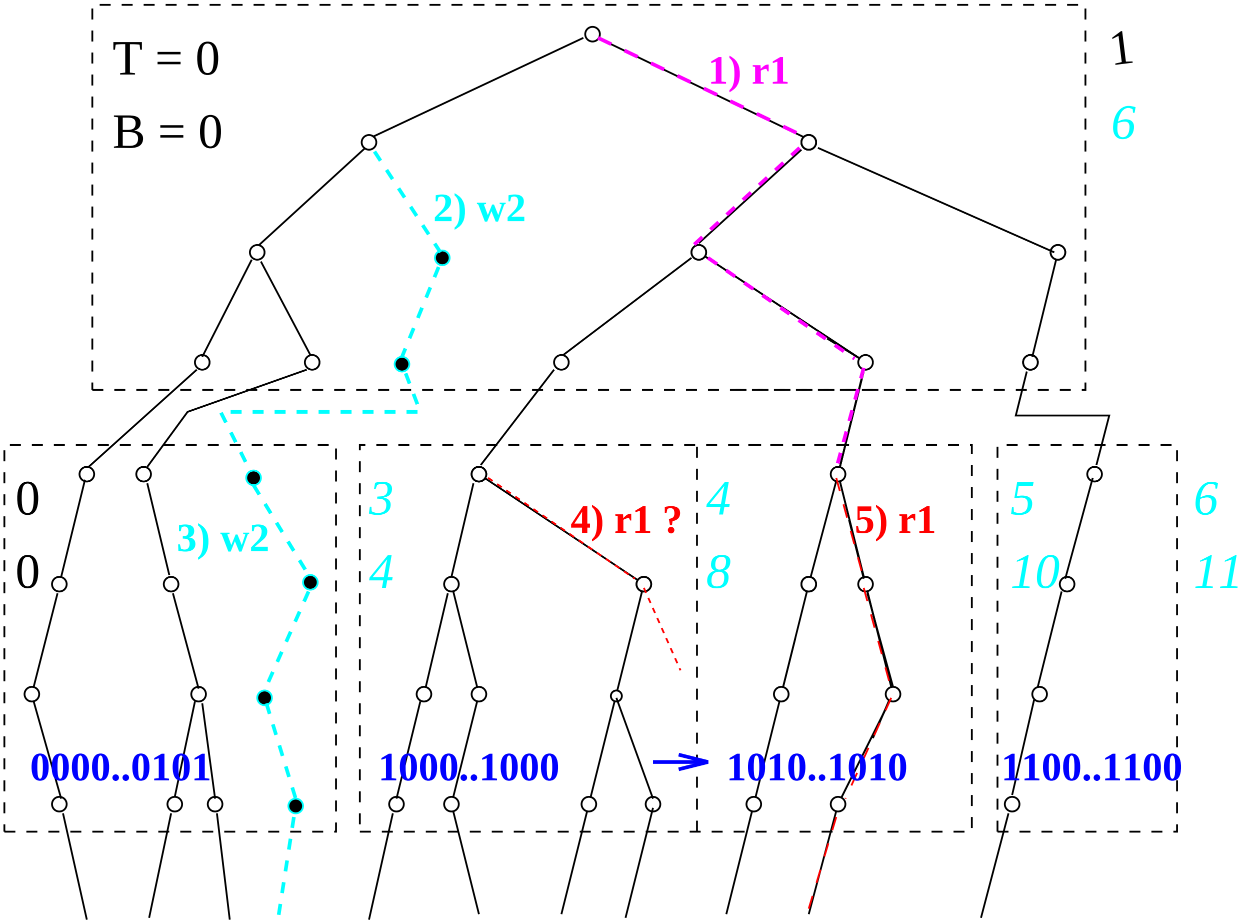
Concurrency: Tries

Counts stored separately for dynamic tries:
must lock at least the trailing counts of a level.
This reduces concurrency.

E.g. read 10101100 (transaction 1)
and concurrently
write 01011010 (transaction 2):

r1 1010	#4
w2 0101	#4 → #5
w2 1010	
r1 1100	not found!

- B-trees can support read/write concurrency by advancing the read page if a write happens to have messed it up.
- Tries do not have enough redundancy to check for a mess.
- So add redundancy: on each page, store the lowest and highest prefix from parent to this page.



- Two writes can lock each other out of the whole trie.
- B-trees have the same problem because root may be split.