


交易

朱克剛

交易

一連串的異動指令屬於同一件事情，若最後成功，全部異動指令都成功，若失敗，全部異動指令都恢復異動前狀態



交易是在client端運作，非server端

透過建立session來建立交易，所有的異動指令都放在該session中，最後可選擇交易成功或交易失敗 

交易時間越短越好，超過60秒算交易失敗，可以改

```
rs0 [direct: primary] test> db.adminCommand({  
  setParameter: 1,  
  transactionLifetimeLimitSeconds: 120  
})
```

基本架構

```
session = client.start_session()  
# 交易開始  
session.start_transaction()   
db.test.insert_one({ 'name': 'David' }, session=session); 
```

```
session.commit_transaction() 
```

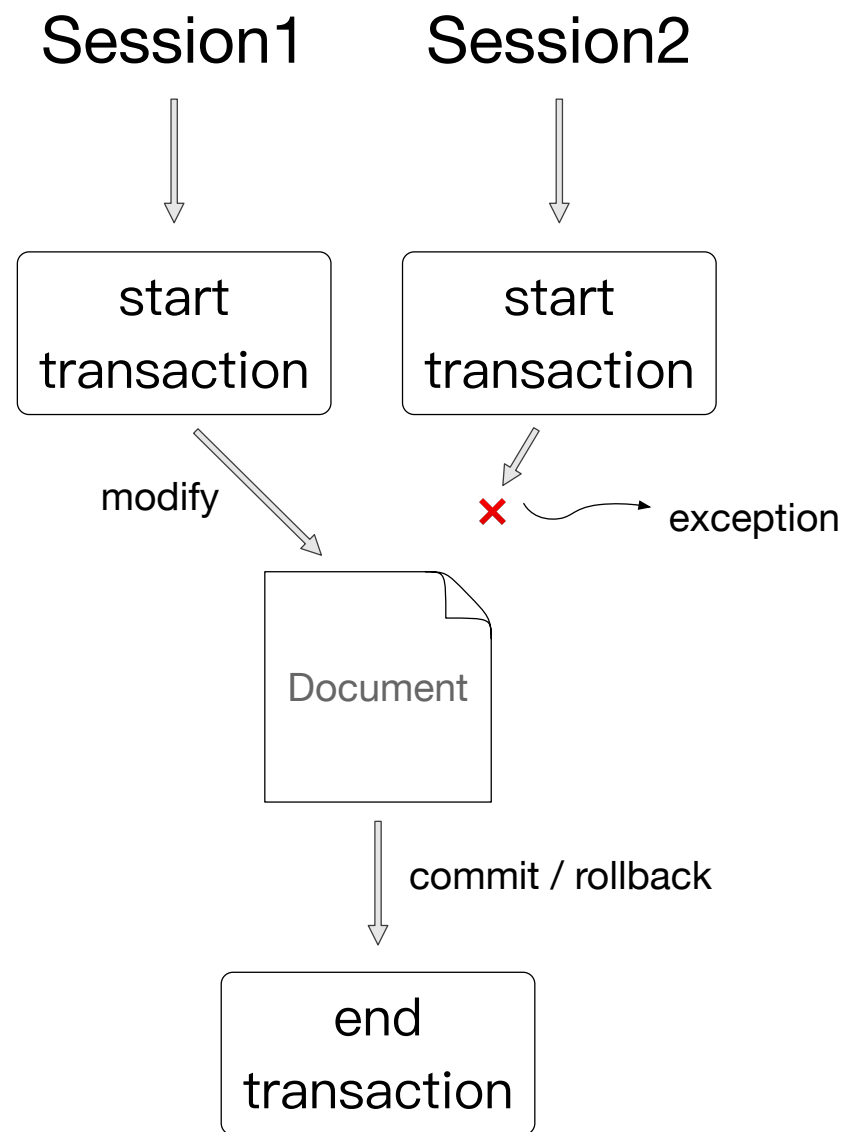
or

```
session.abort_transaction() 
```

寫入衝突 交易與交易

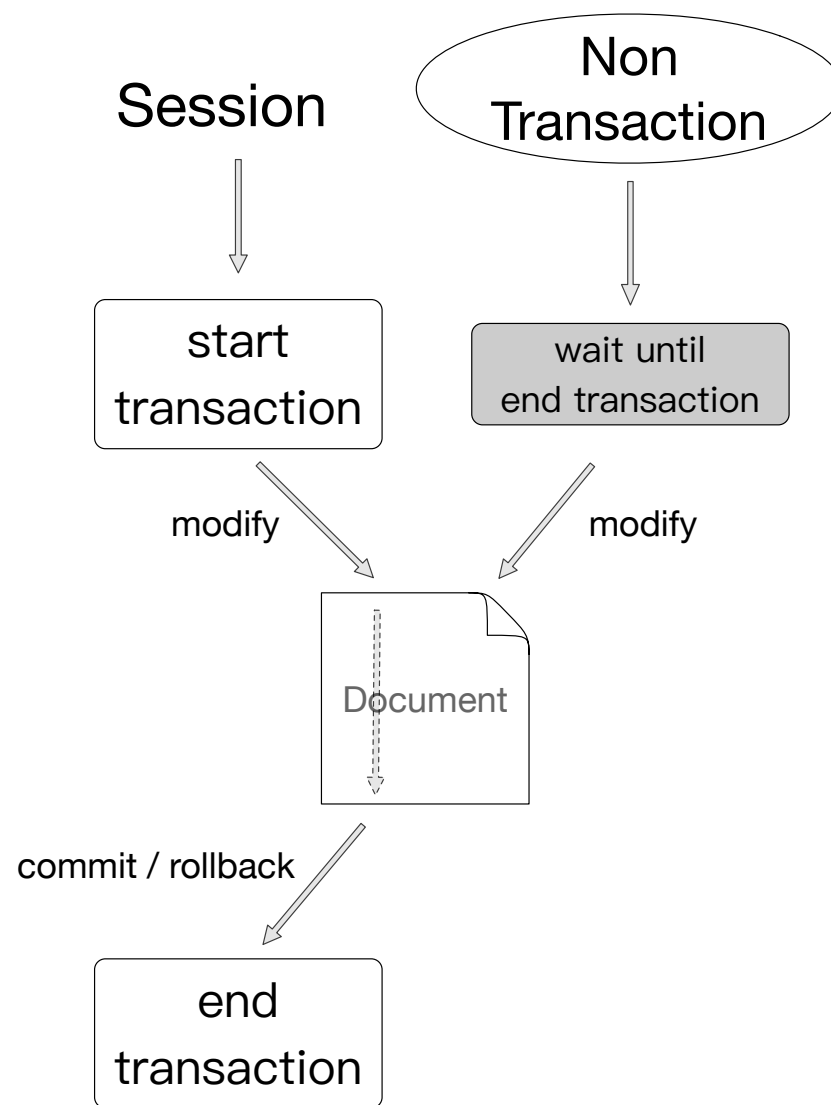
何謂寫入衝突：兩個異動程序同時要異動同一筆資料

對於同一筆資料，**session1** 還沒異動完畢時，**session2** 會丟出錯誤



寫入衝突 交易與非交易

對於同一筆資料，當交易還沒完成時，
非交易異動會進入等待



鎖

種類	說明
S	共享鎖。針對讀取資料的指令，例如執行find()或aggregate()時，會對資料加上S鎖。
X	獨佔鎖（也稱排他鎖）。針對異動資料指令，例如修改資料時，會對該資料加上X鎖，此時這筆資料不可被其他人讀寫。
IS	意圖共享鎖，作用在Global、Database與Collection上，表示所屬的資料表中已有文件設定了S鎖。
IX	意圖獨佔鎖，作用在Global、Database與Collection上，表示所屬的資料表中已有文件設定了X鎖。

上鎖狀態


每筆文件同一時間只能有一種鎖，例如 **s** 鎖未解除時就不能再上 **x** 鎖
查看目前上鎖狀態指令如下

```
rs0 [direct: primary] test> db.adminCommand( { lockInfo: 1 } )
```




討論

A資料異動中尚未完成，另一程序要修改A資料是否可行？

- 此時上鎖狀態？ 

A資料異動中尚未完成，另一程序要修改B資料是否可行？


- 此時上鎖狀態？ 

資料庫中有兩筆資料都在異動中尚未完成，何時會使用到 IX 鎖？ 

超賣問題

總共只有10樣東西可賣，但最後發現賣超過數量了

原因在於「取出數量」、「檢查數量」、「決定賣出」為三個獨立程序，在多工環境下產生的必然現象

解決方式：將這三個程序合併成一個程序 



因果一致性

客戶端一定要能讀到自己寫入的資料

單調讀

單調寫

寫跟隨讀



寫入關注 讀取關注

寫入關注預設值

- { w: "majority" }

讀取關注預設值

- { level: "local" }

右邊程式碼造成的問題是，可能讀不到自己寫入的資料

```
import pymongo
from pymongo.write_concern import WriteConcern
from pymongo.read_concern import ReadConcern

hosts = ['localhost:20000', 'localhost:20001', 'localhost:20002']
client = pymongo.MongoClient(hosts, readPreference='secondary')
db = client.test
```

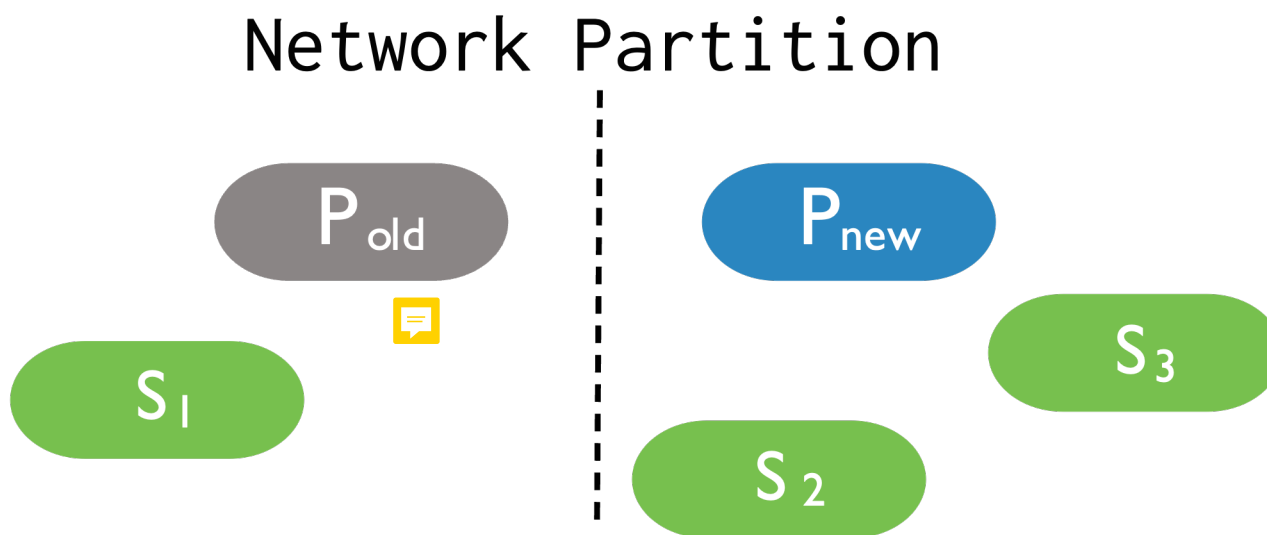
Primary 寫資料

```
db.test.drop()
db.test.with_options(
    write_concern=WriteConcern(w='majority', j=True)
).insert_one({'name': 'aaa'})
```

Secondary 讀資料

```
doc = db.test.with_options(
    read_concern=ReadConcern(level='local')
).find_one()
print(doc)
```

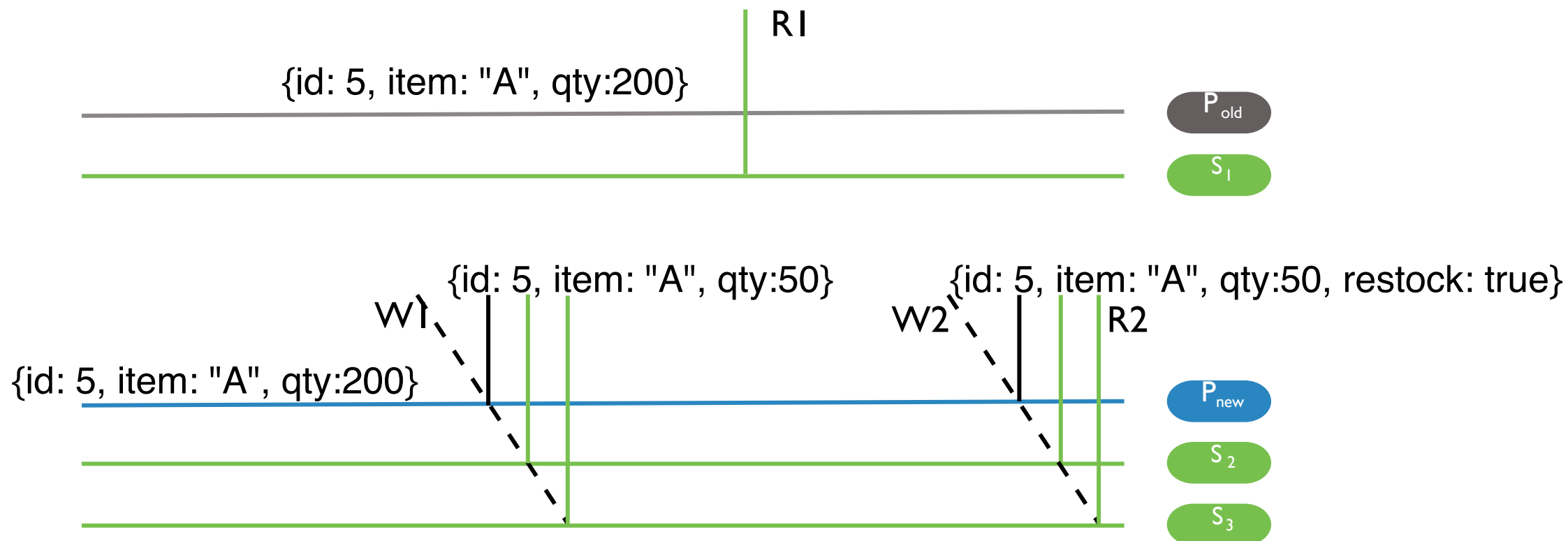
若網路問題導致複寫集一分為二



問題

- ✗ 讀到自己寫入的資料
- ✗ 單調讀
- ✓ 單調寫
- ✗ 寫跟隨讀

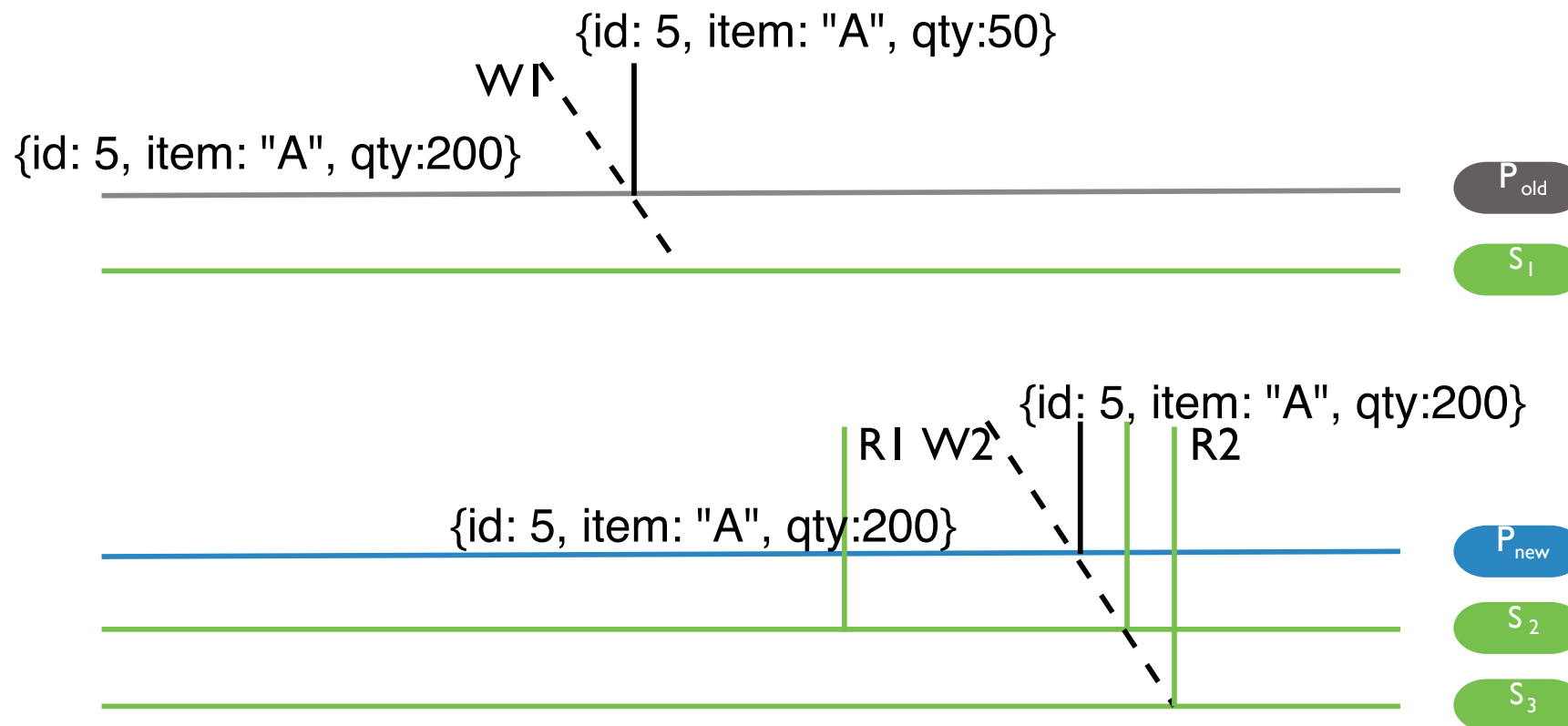
Read Concern "local" , Write Concern "majority"



問題

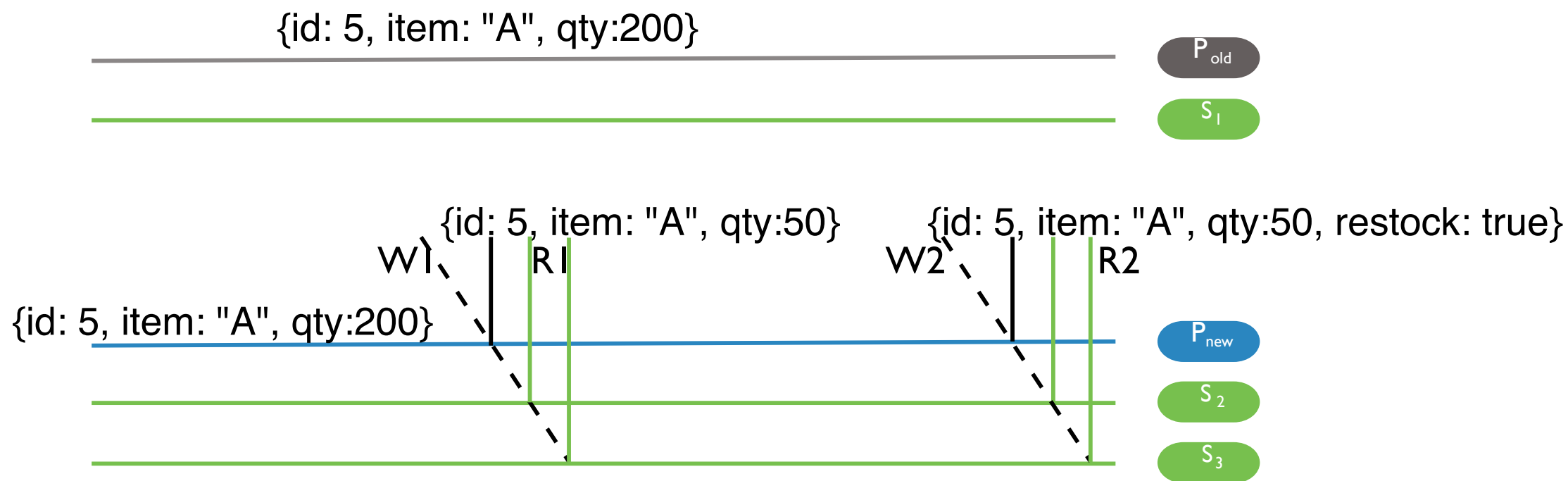
- ✗ 讀到自己寫入的資料
- ✓ 單調讀
- ✗ 單調寫
- ✓ 寫跟隨讀

Read Concern "majority", Write concern {w: 1}



如何保證因果一致

Read Concern "majority" , Write Concern "majority"



延遲狀況

查看 secondary 落後 primary 多久的指令

```
test> db.printSecondaryReplicationInfo()
```