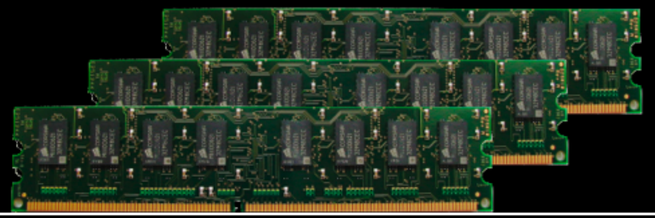
DBMS

Query Optimizer

Indexer

Store

stable storage



DBMS

Query Optimizer

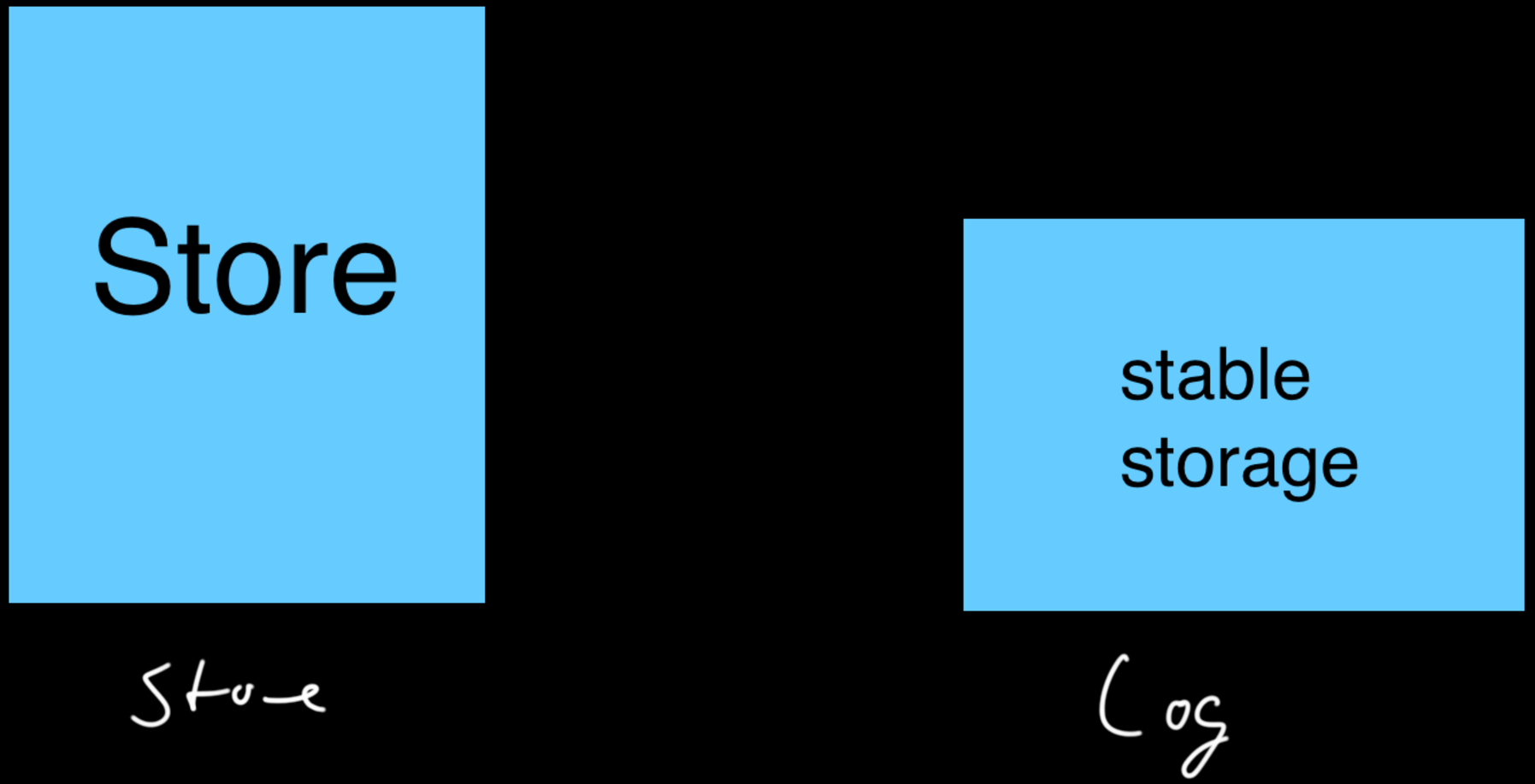
Indexer

Store

stable
storage


Store
stable
storage

Logging! (see video 14.163)



Publishing a Book

1st edition

changes:

Publishing a Book

1.1st edition

store

changes:

page 23:
“datbase“ → “database“

log

Publishing a Book

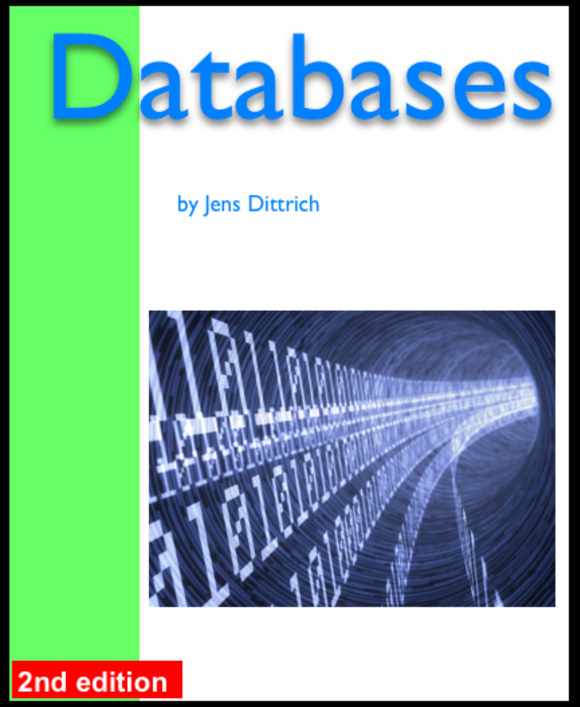
1.2nd edition

changes:

page 23:
“datbase“ → “database“
page 345:
“idex“ → “index“

Publishing a Book

2nd edition

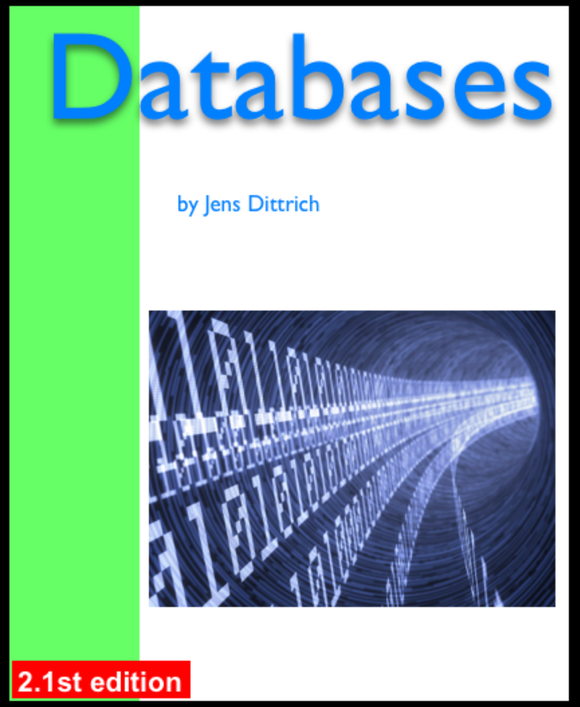


changes:

page 23:
“datbase“ → “database“
page 345:
“idex“ → “index“
page 77:
“idex“ → “index“

Publishing a Book

2.1st edition

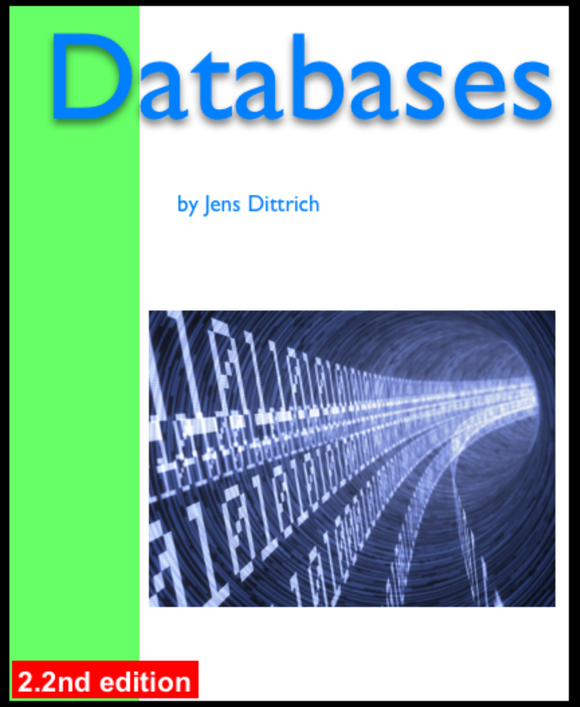


changes:

page 23:
“datbase“ → “database“
page 345:
“idex“ → “index“
page 77:
“idex“ → “index“
page 75:
“kamera“ → “camera“

Publishing a Book

2.2nd edition



changes:

page 23:
“datbase“ → “database“
page 345:
“idex“ → “index“
page 77:
“idex“ → “index“
page 75:
“kamera“ → “camera“
page 143:
“big date“ → “big data“

Publishing a Book

3rd edition

changes:

page 23:
“datbase“ → “database“
page 345:
“idex“ → “index“
page 77:
“idex“ → “index“
page 75:
“kamera“ → “camera“
page 143:
“big date“ → “big data“
new chapter on “tools“

Publishing a Book

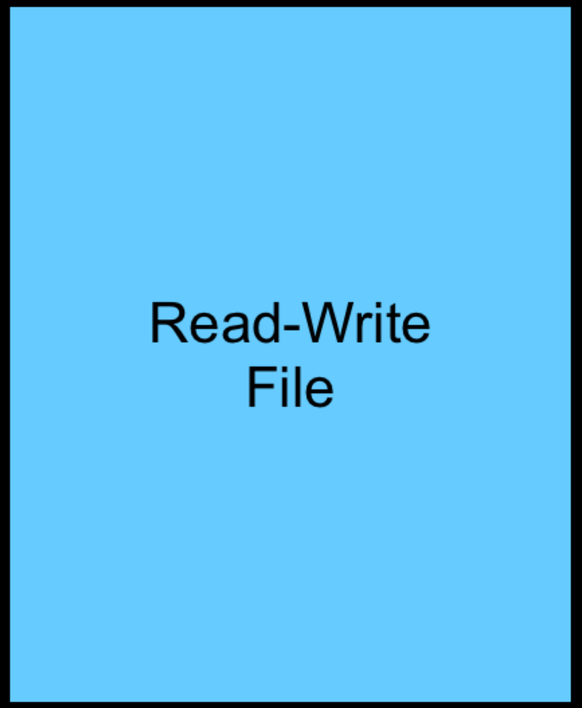
3.1st edition

changes:

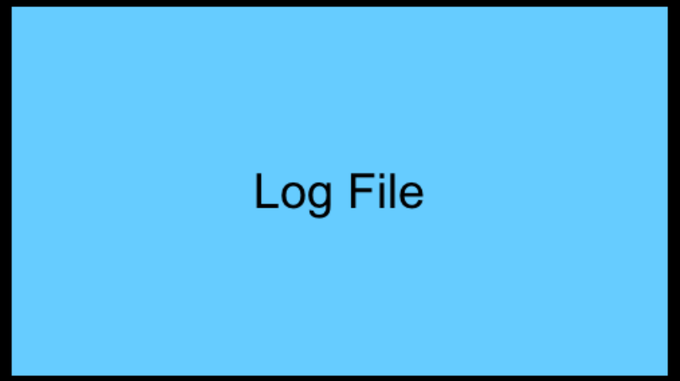
page 23:
“datbase“ → “database“
page 345:
“idex“ → “index“
page 77:
“idex“ → “index“
page 75:
“kamera“ → “camera“
page 143:
“big date“ → “big data“
new chapter on “tools“
page 55:
“profi“ → “profile“

Logging

current edition



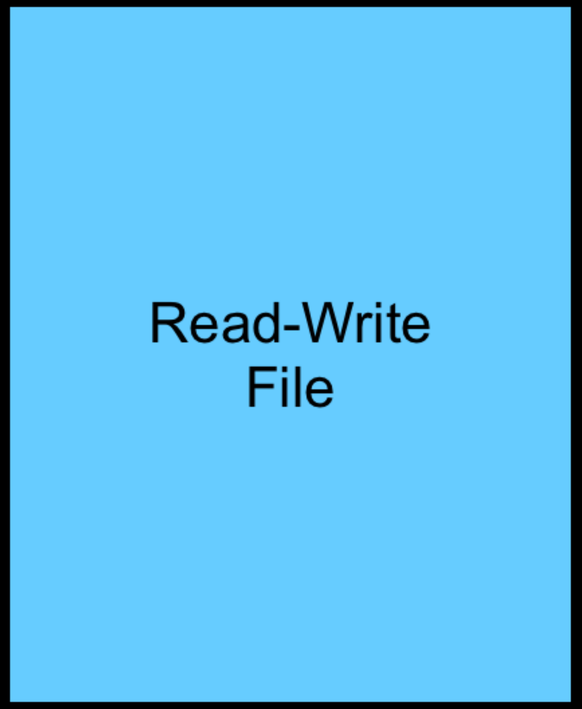
changes:



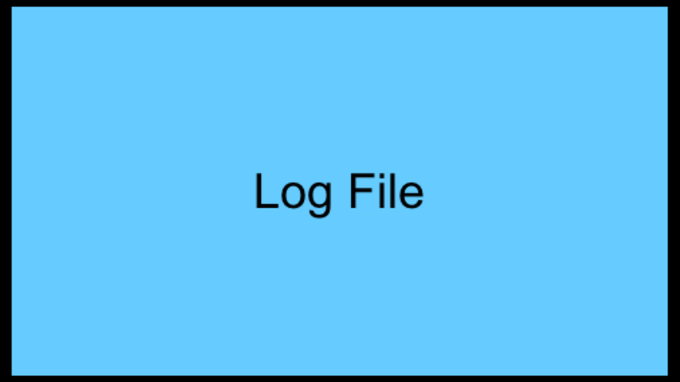
= yet another instance of: The Data Redundancy Pattern and The All Levels are Equal Pattern

Write-Ahead Logging (WAL)

current edition



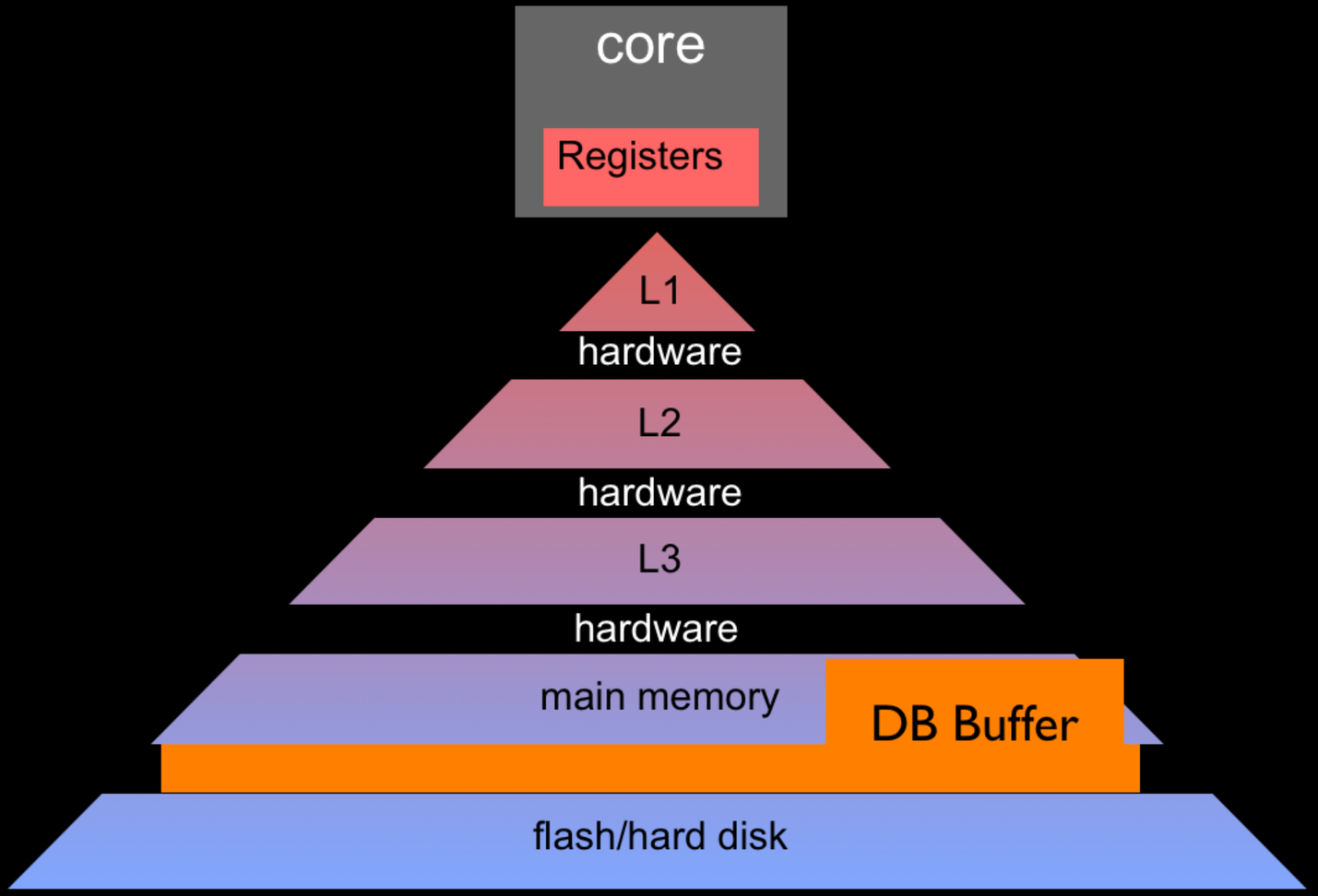
changes:



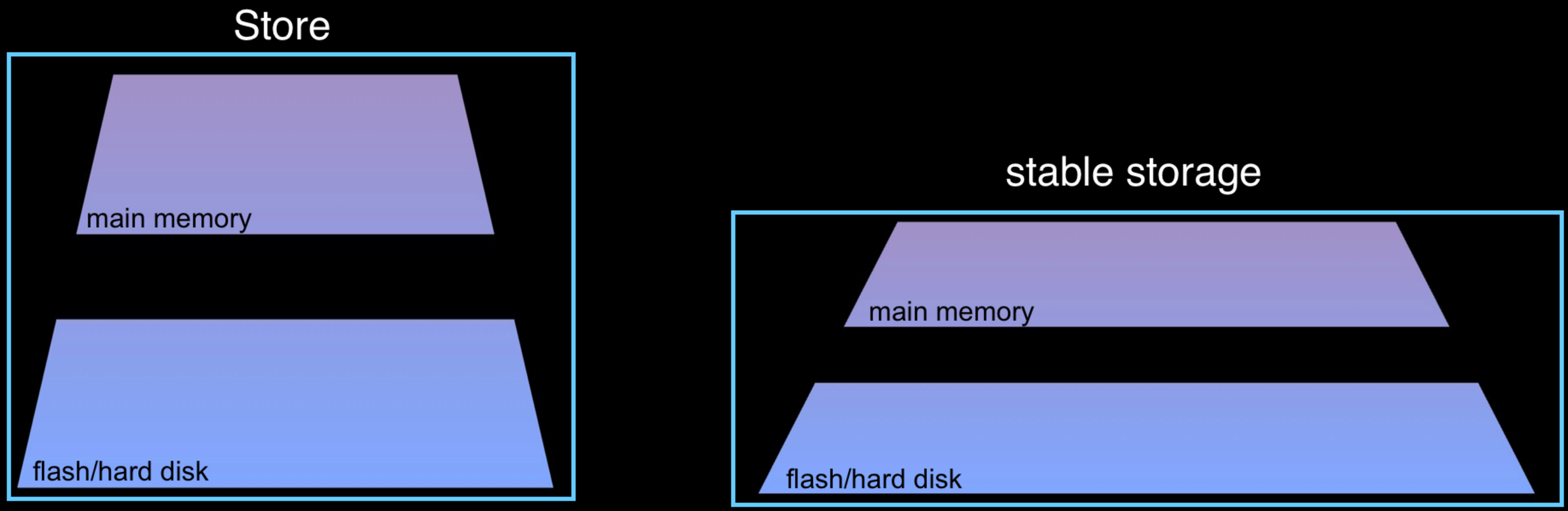
2nd: apply change to store

1st: append to log (and flush)

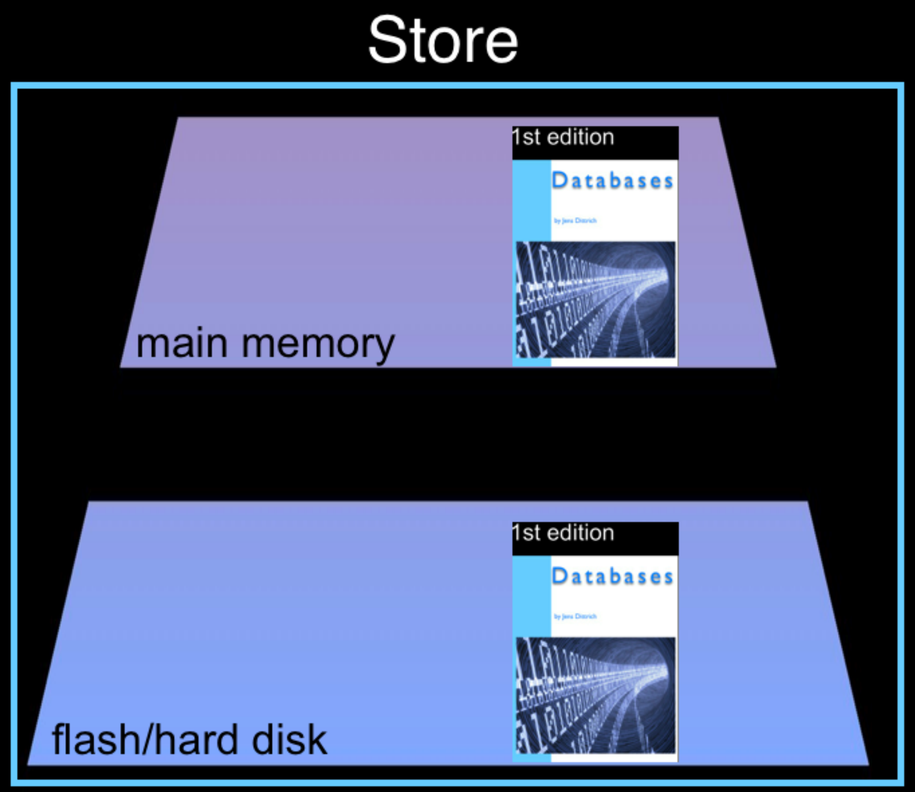
Where is the Database Buffer? (see video 14.142)

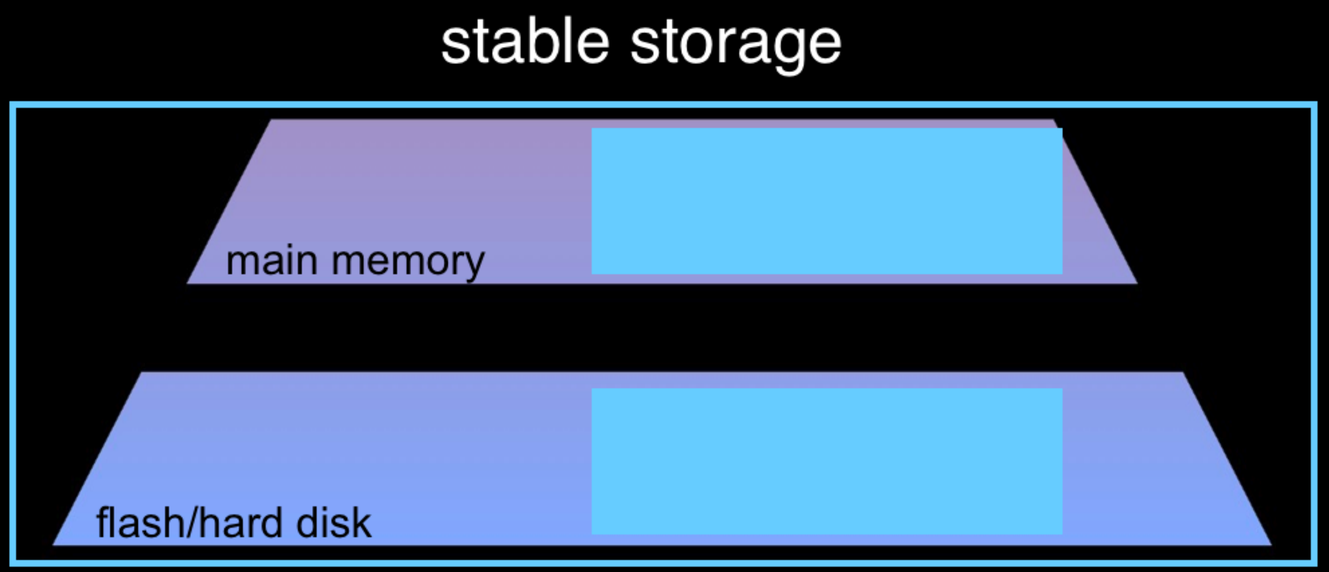


WAL with Multiple Storage Layers

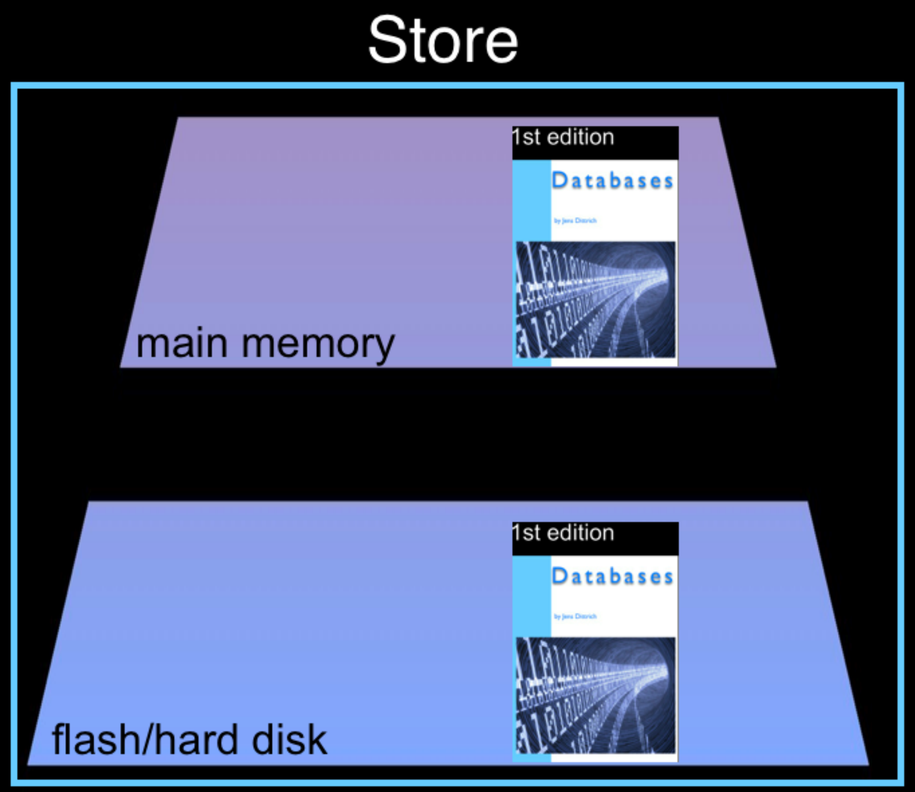


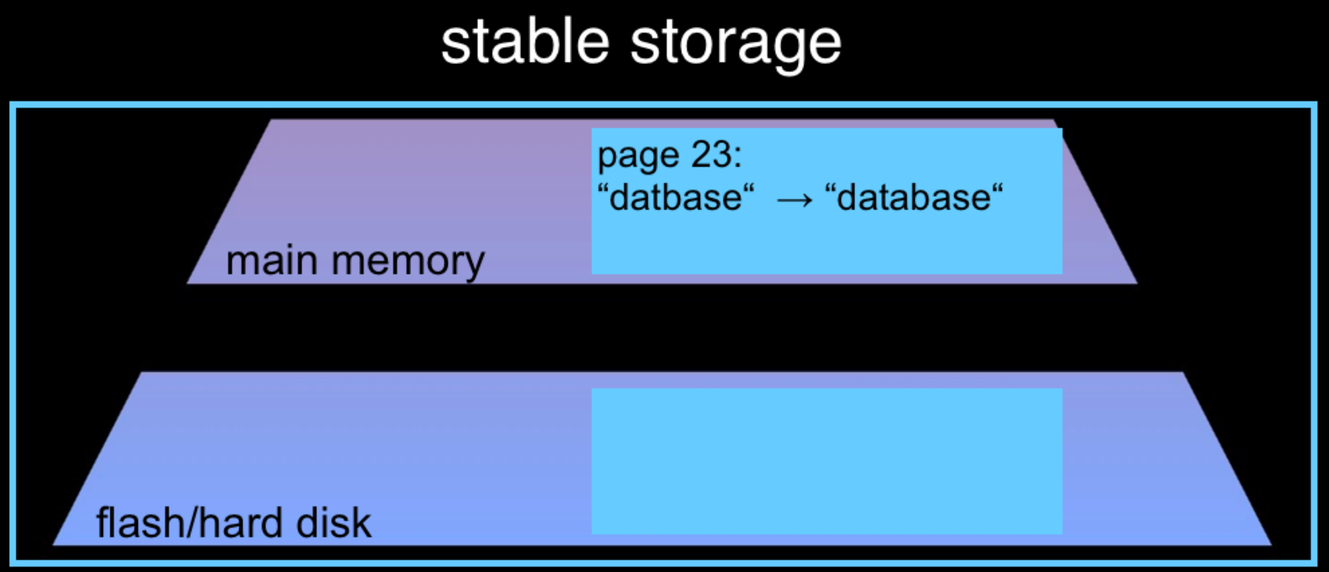
Starting: No Changes yet



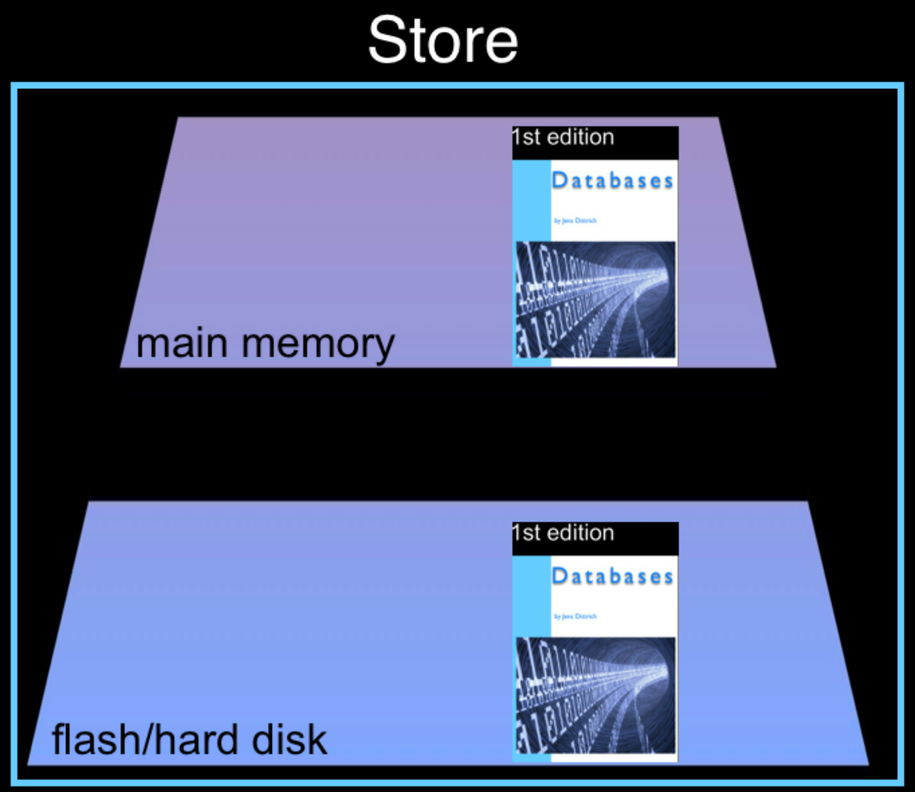


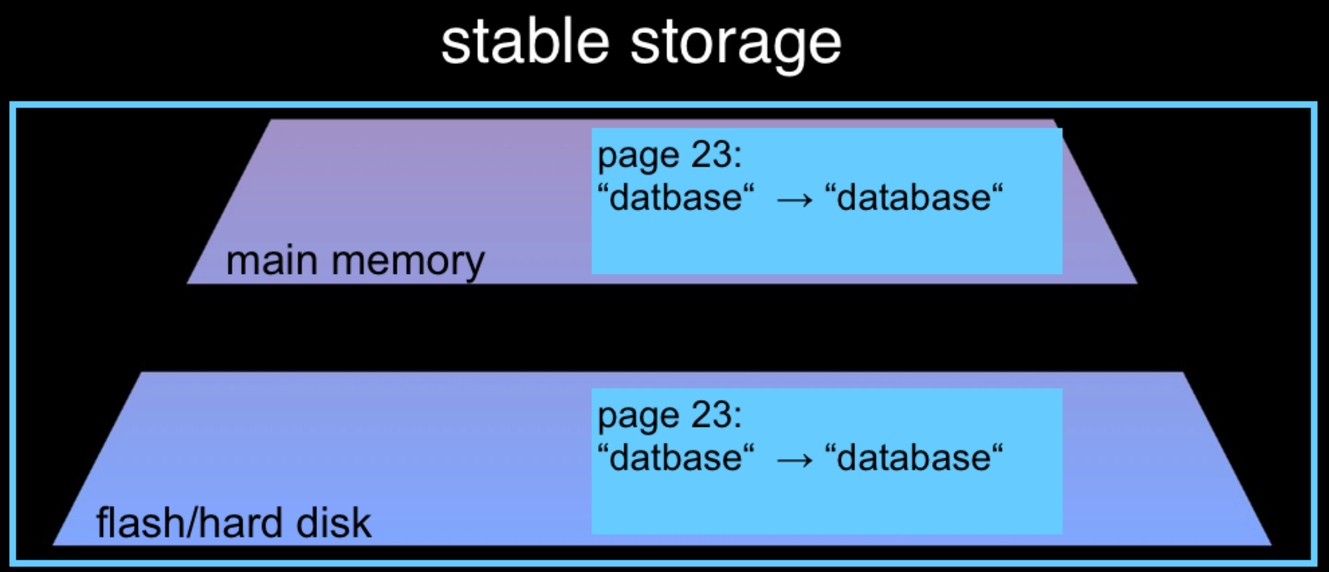
There was a change, let‘s log it.





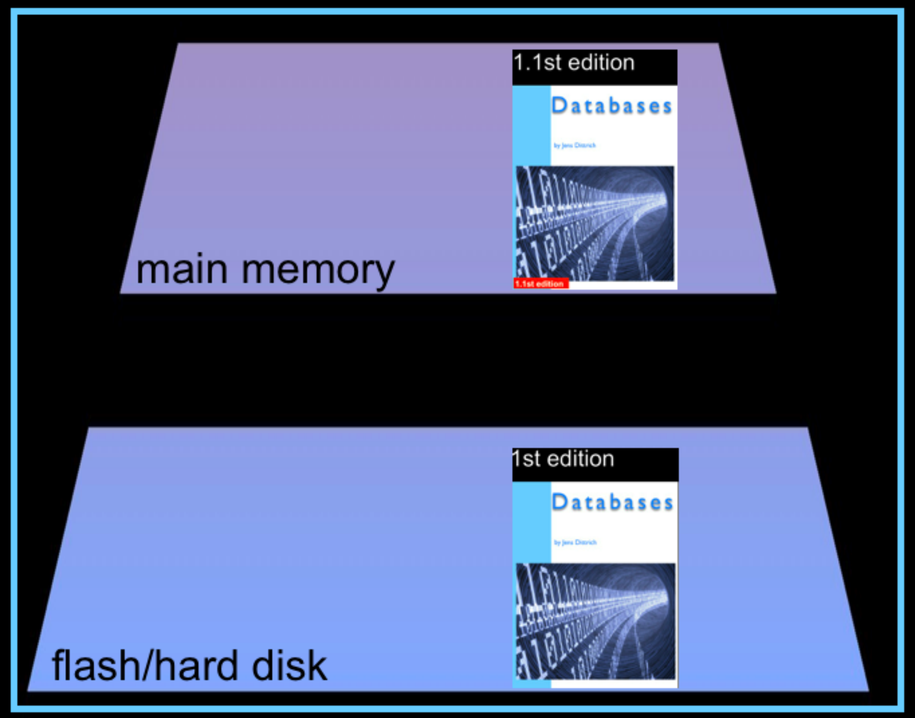
Force the log entry to log disk.



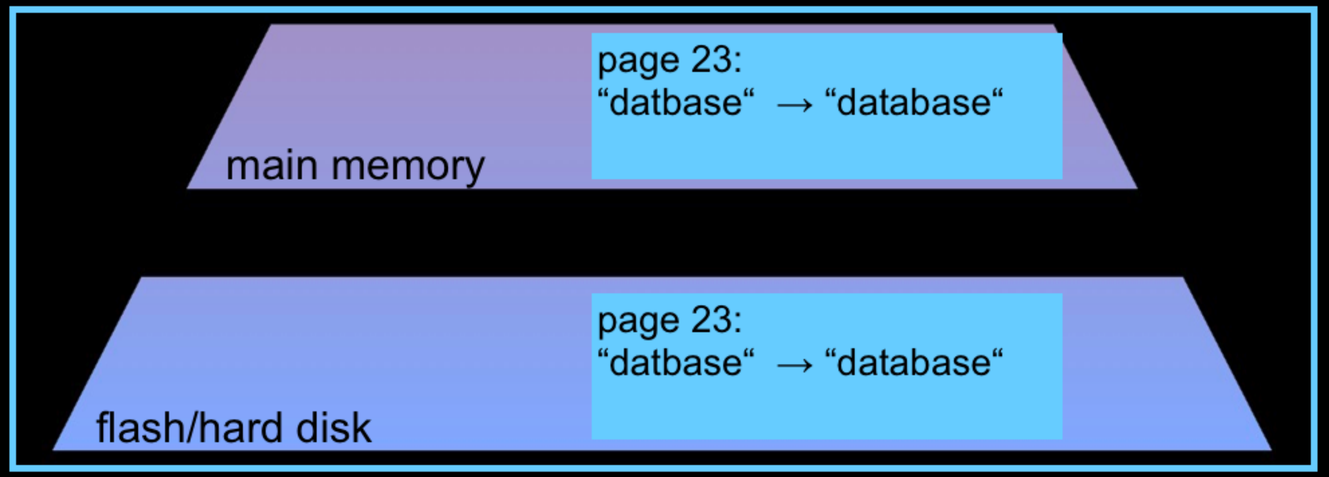


Eventually create a new Edition 1.1

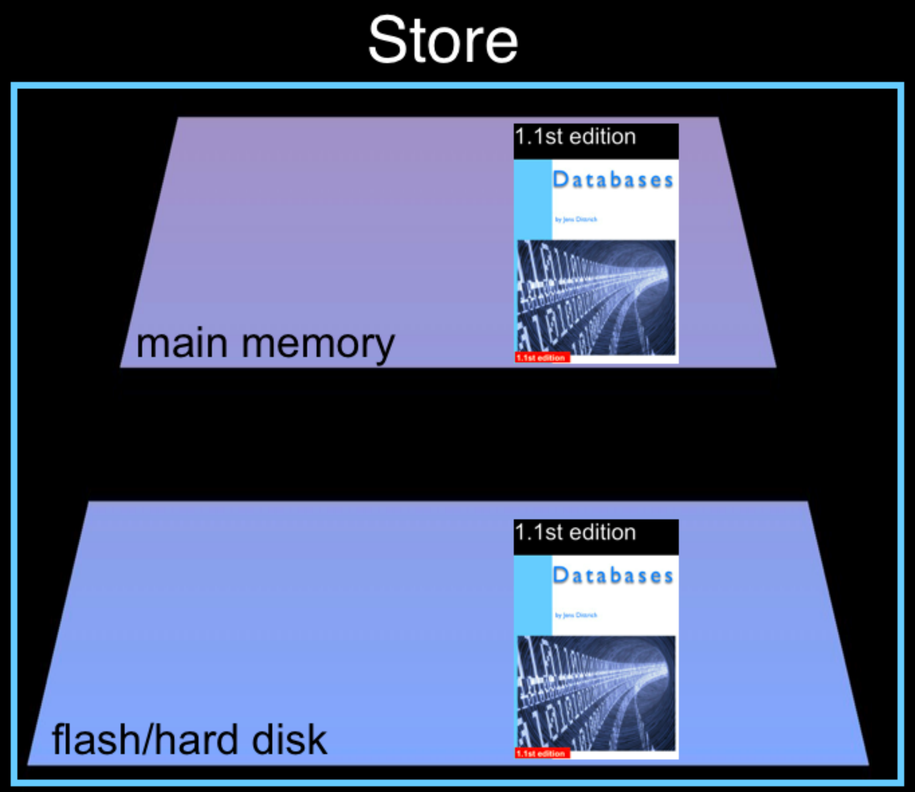
Store

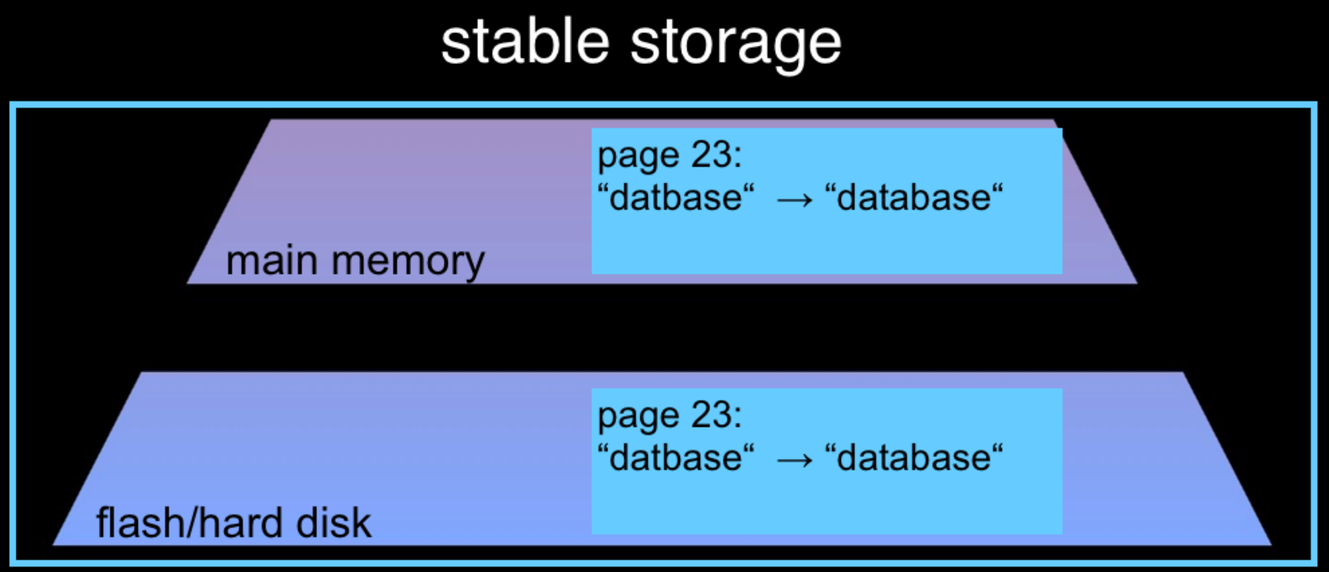


stable storage



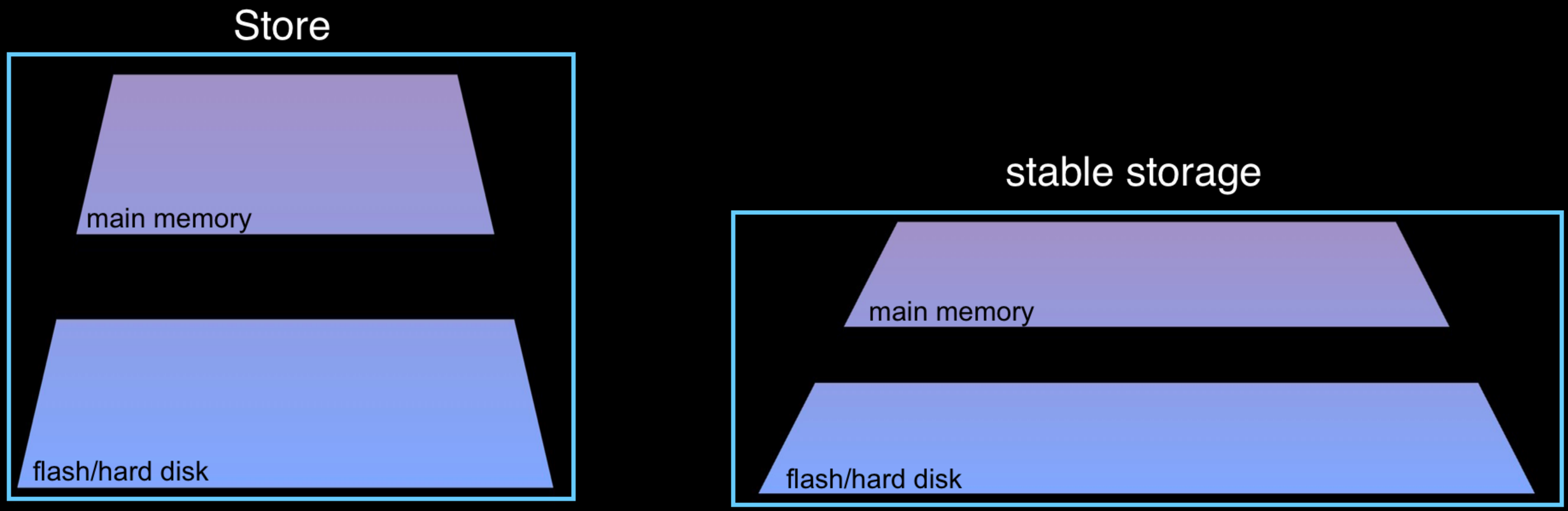
Eventually write Edition 1.1 to Disk



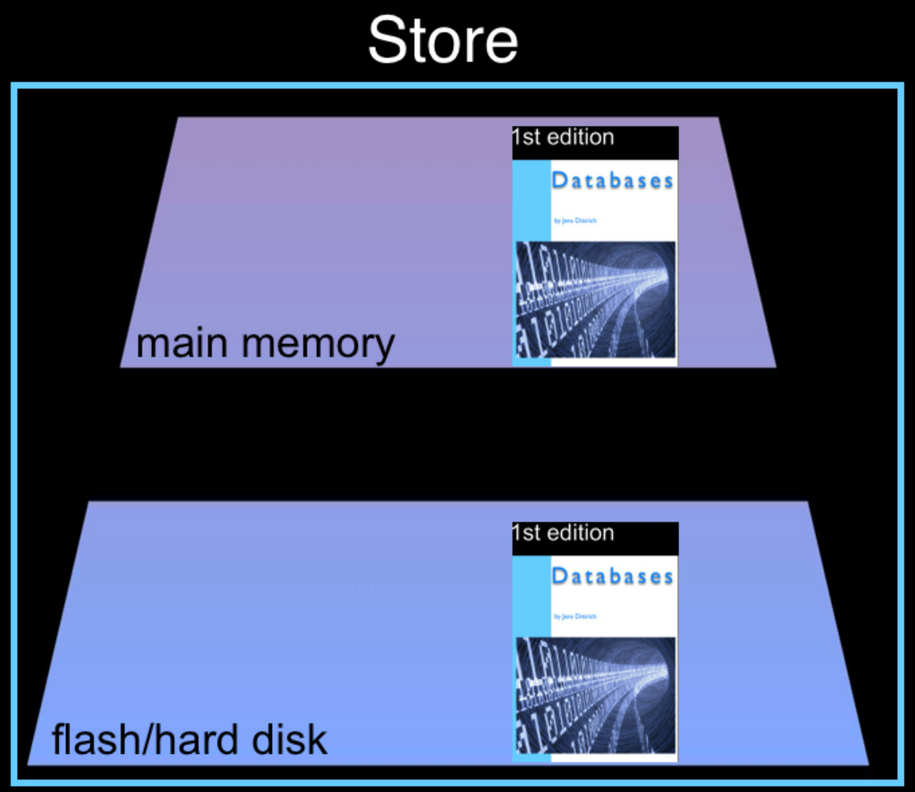


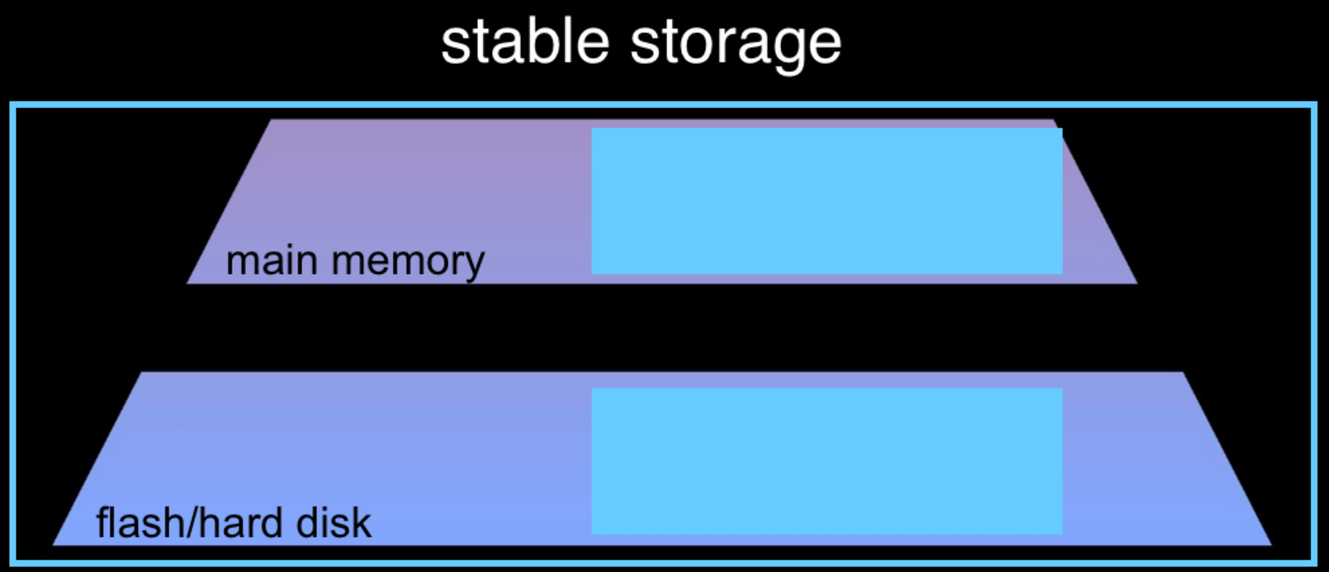
OR:

WAL with Multiple Storage Layers

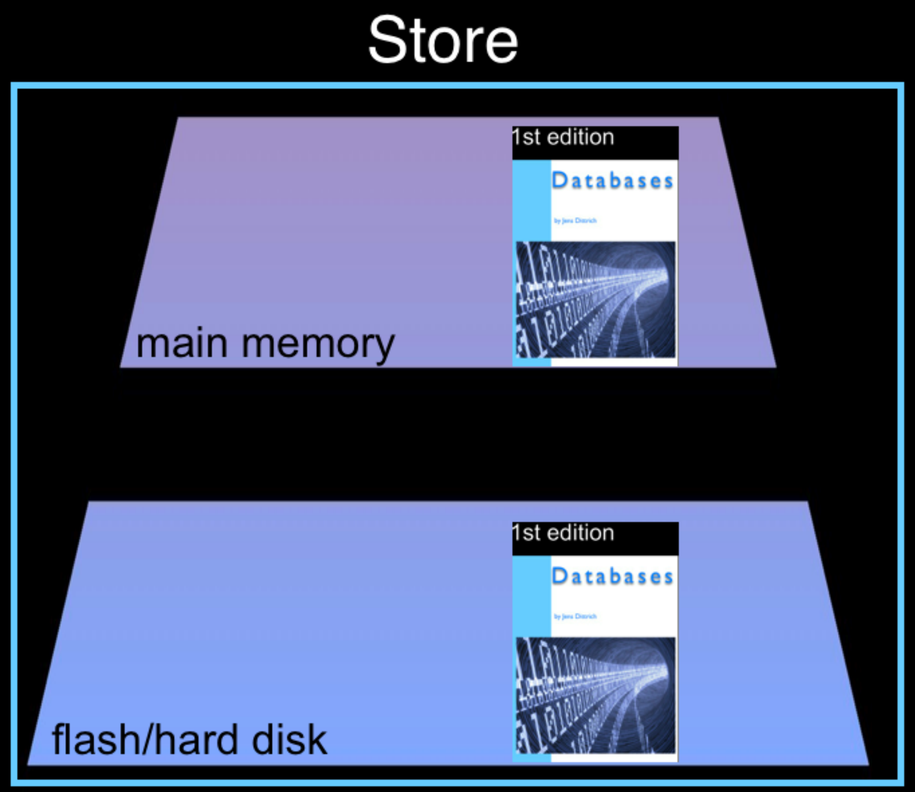


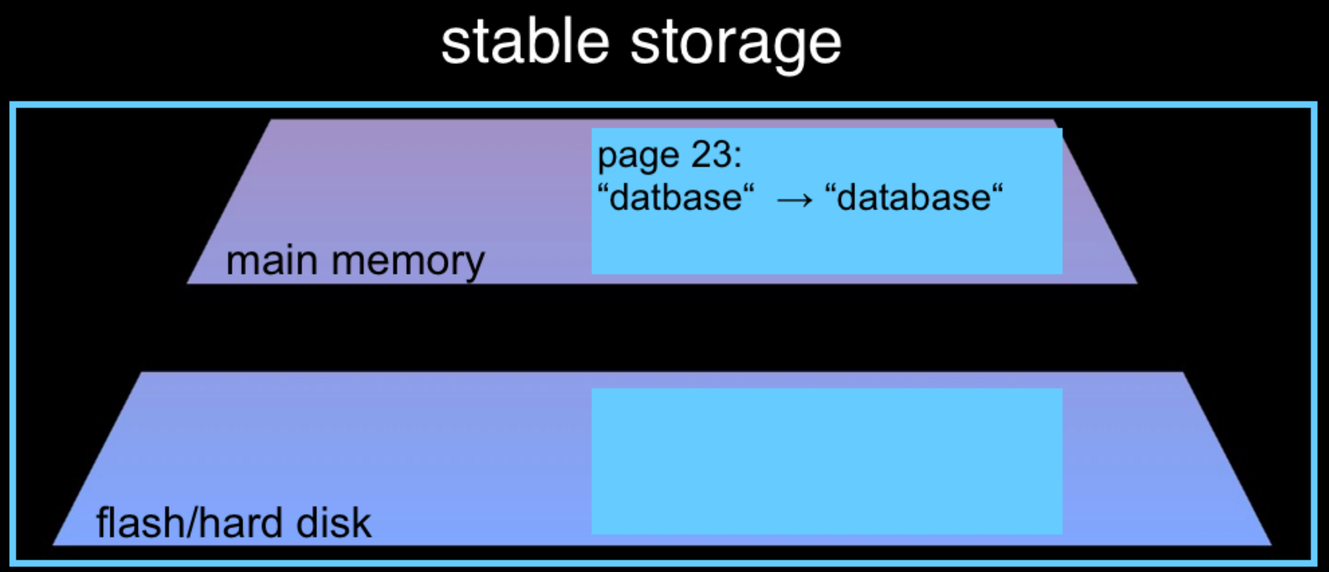
Starting: No Changes yet





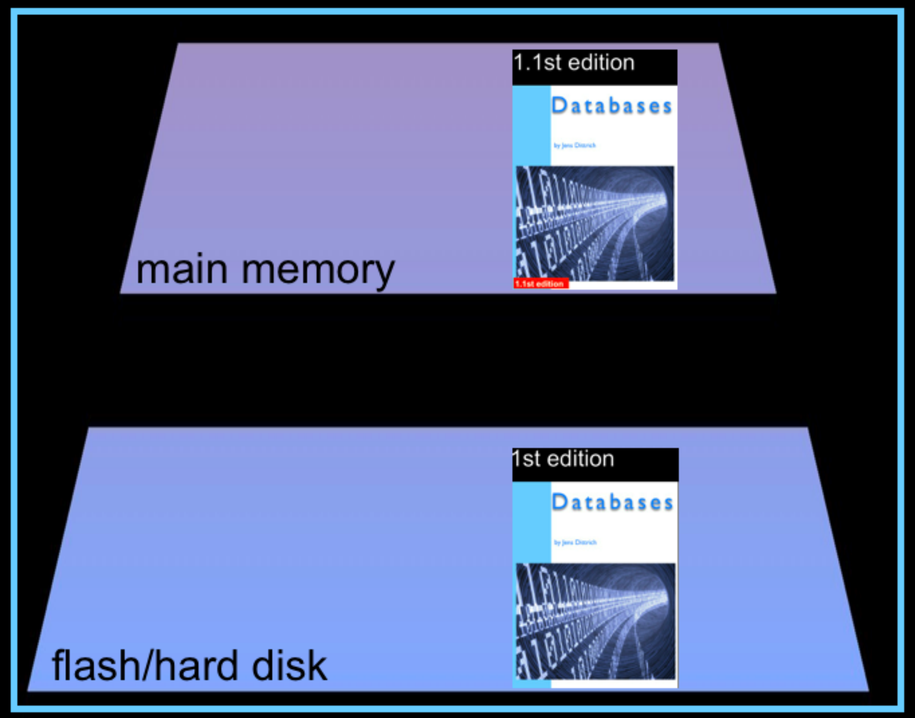
There was a change, let‘s log it.



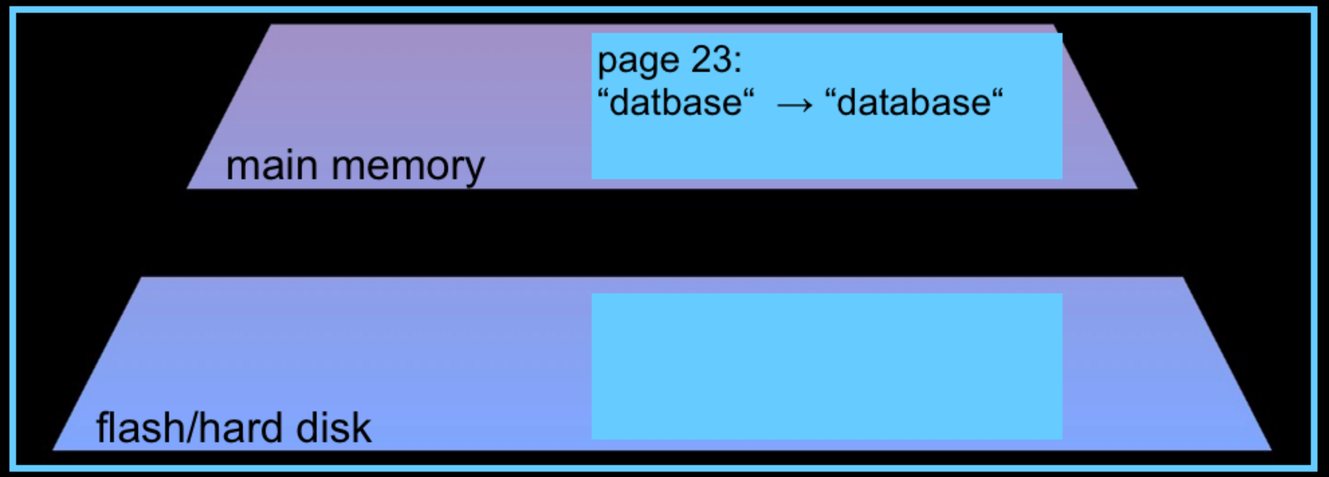


But, First: create a new Edition 1.1 in Main Memory

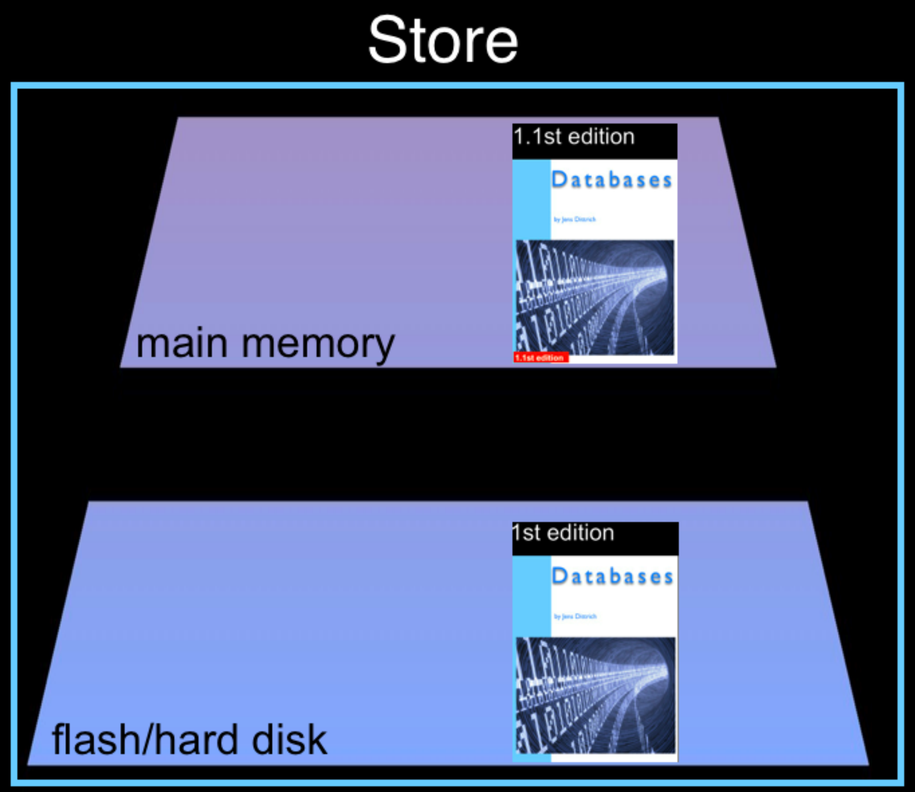
Store

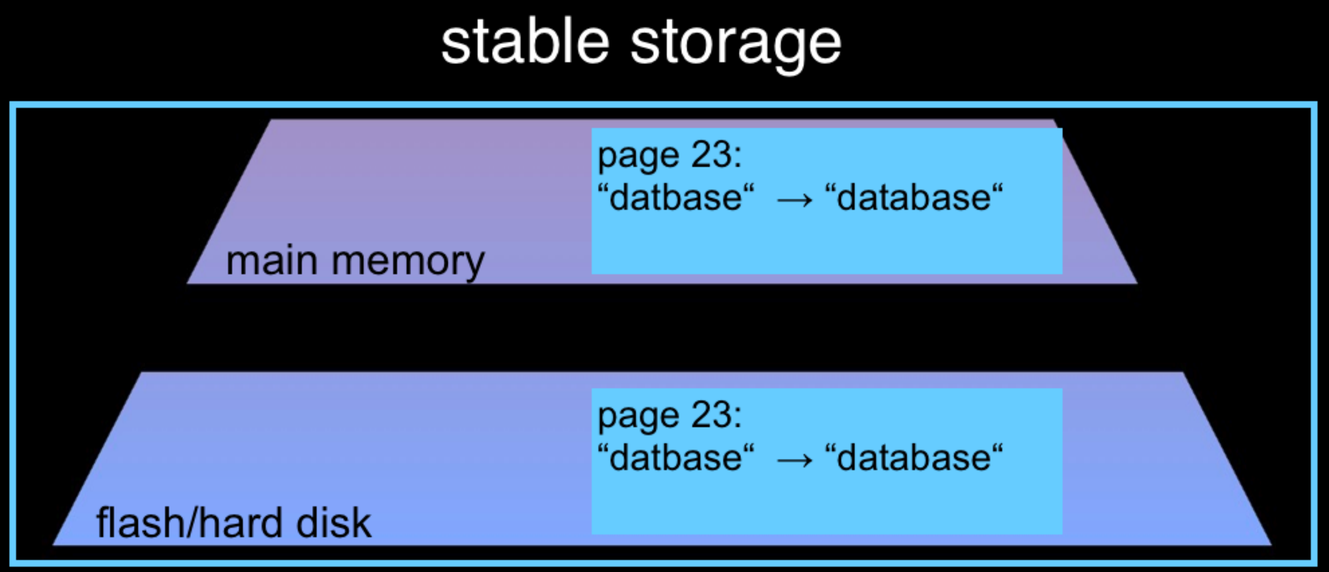


stable storage

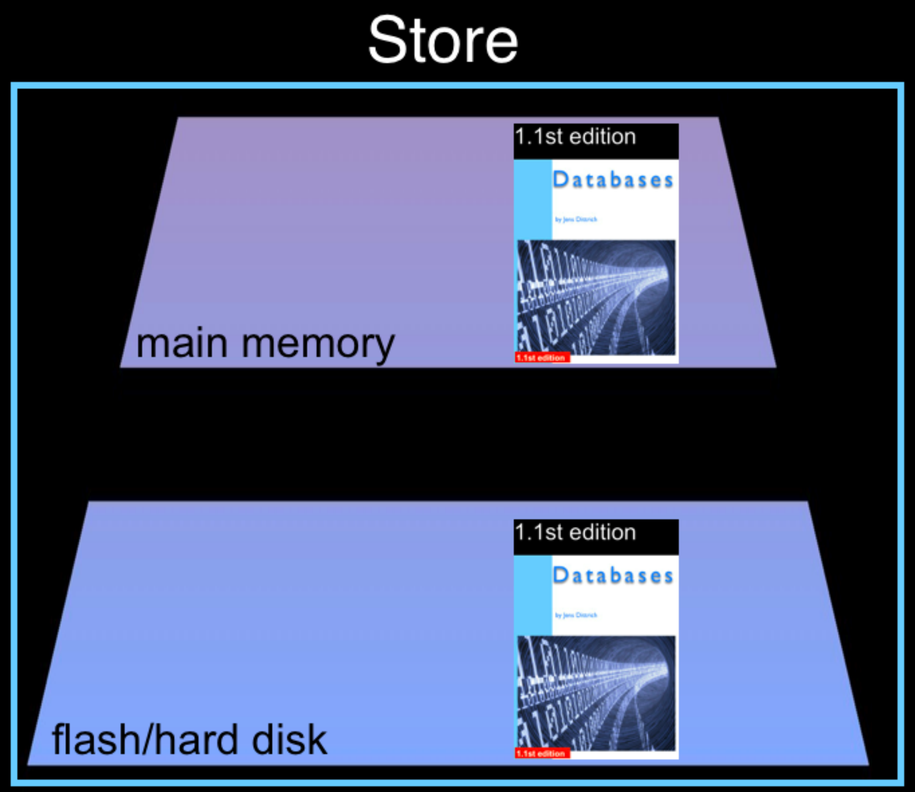


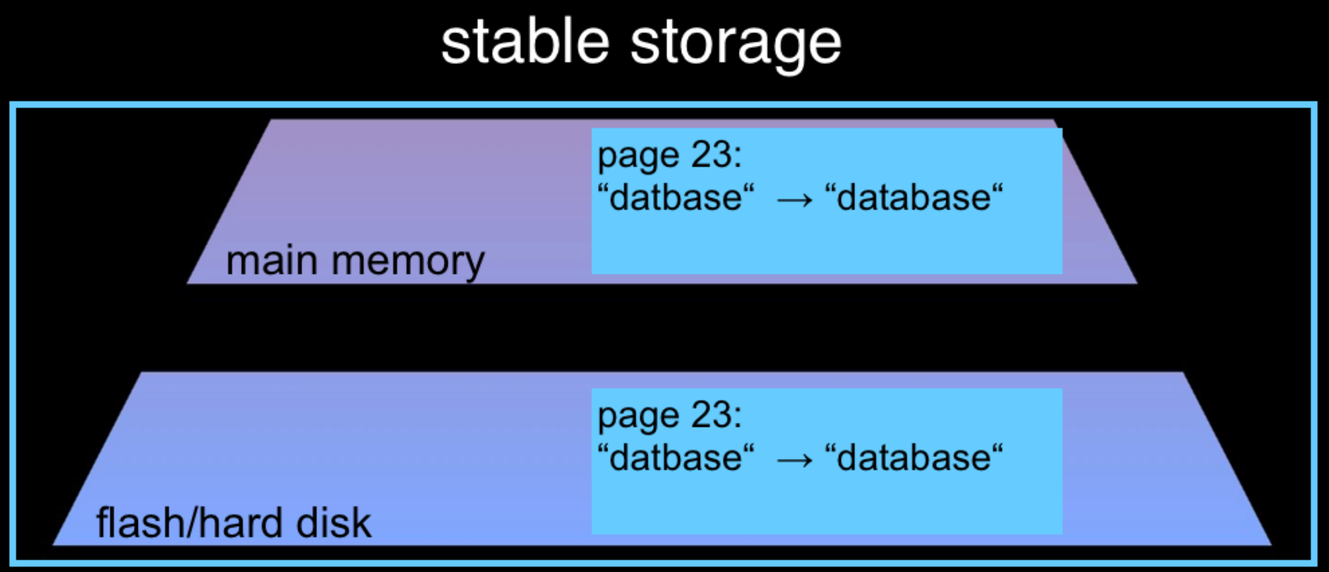
Then: Force the log entry to disk.





Then write Edition 1.1 to Disk

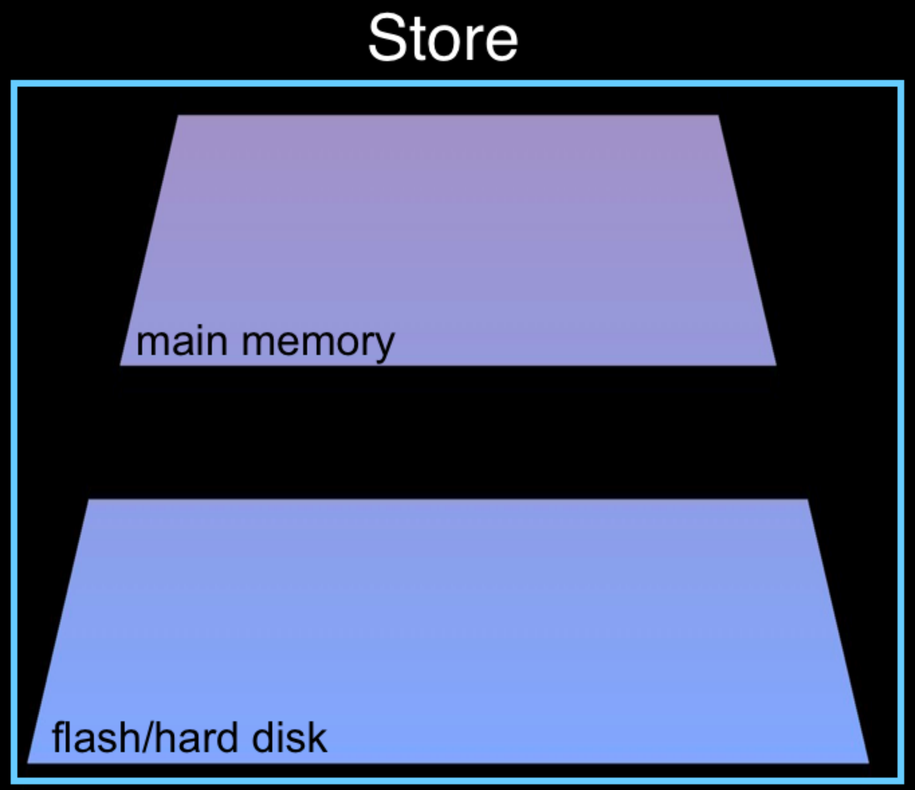


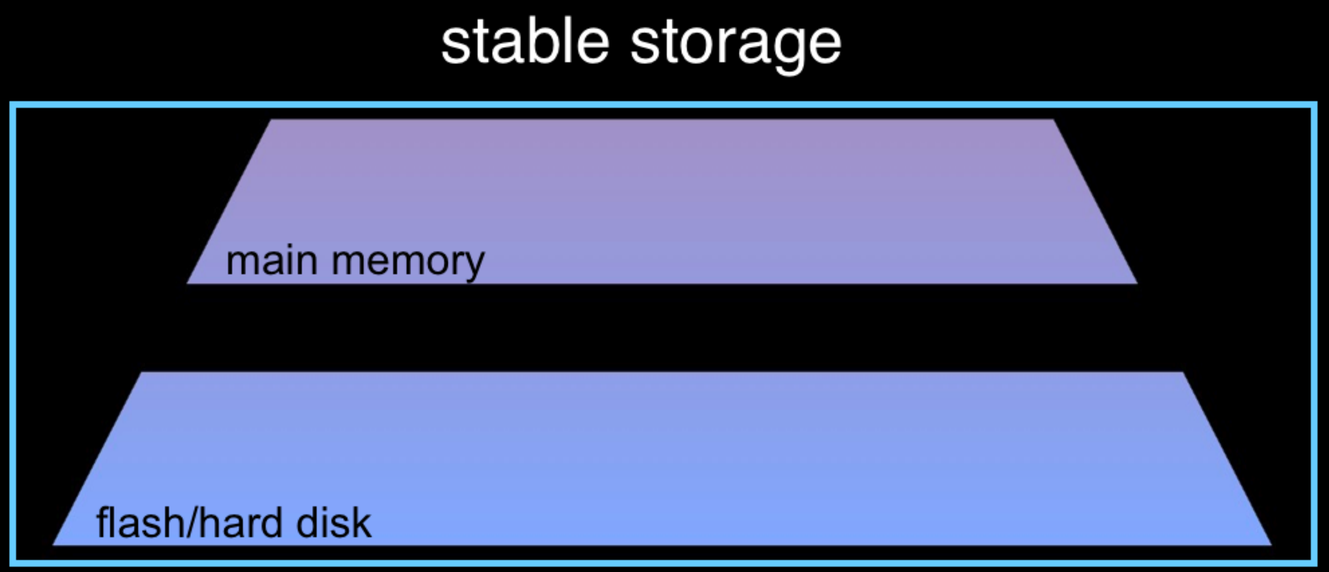


OR:

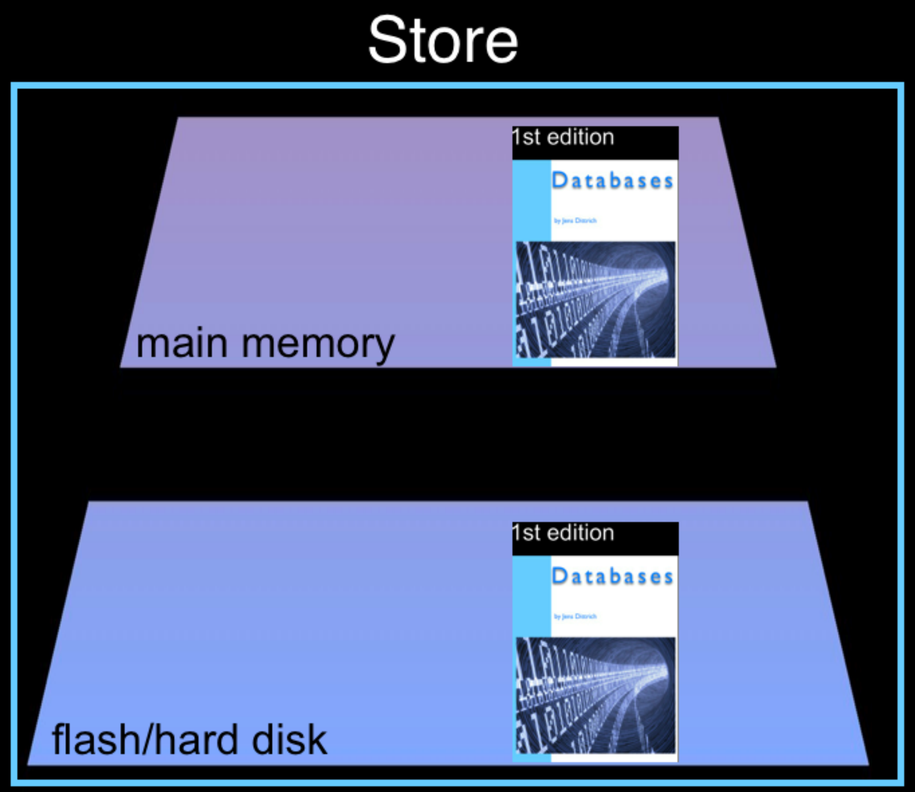
(This would be Non-WAL)

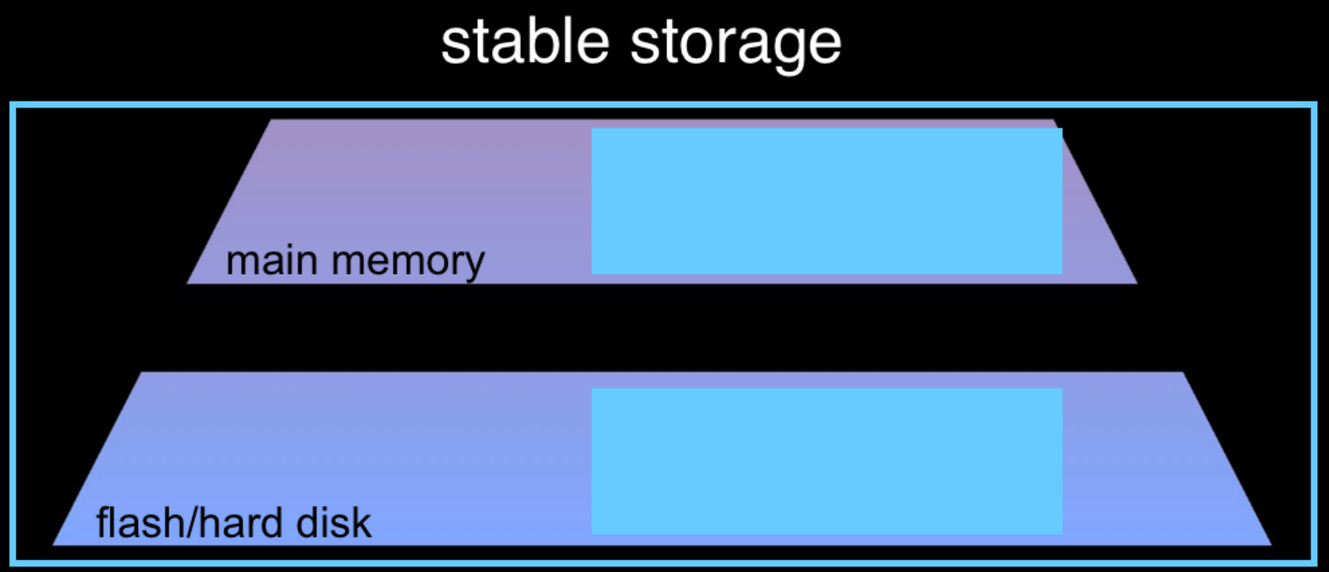
Non-WAL with Multiple Storage Layers



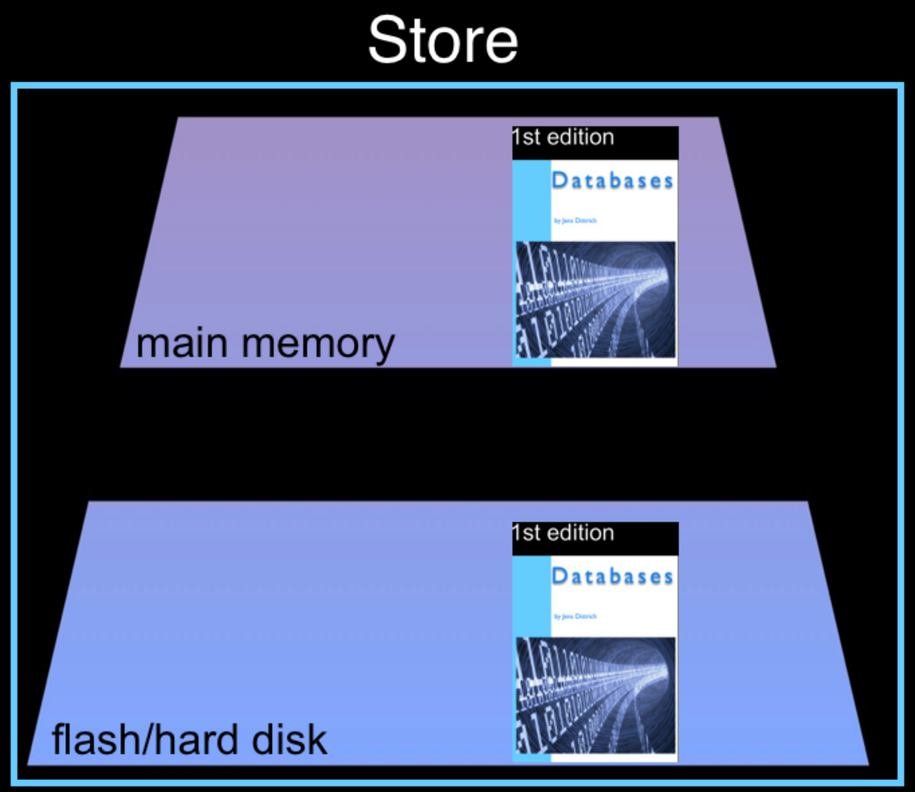


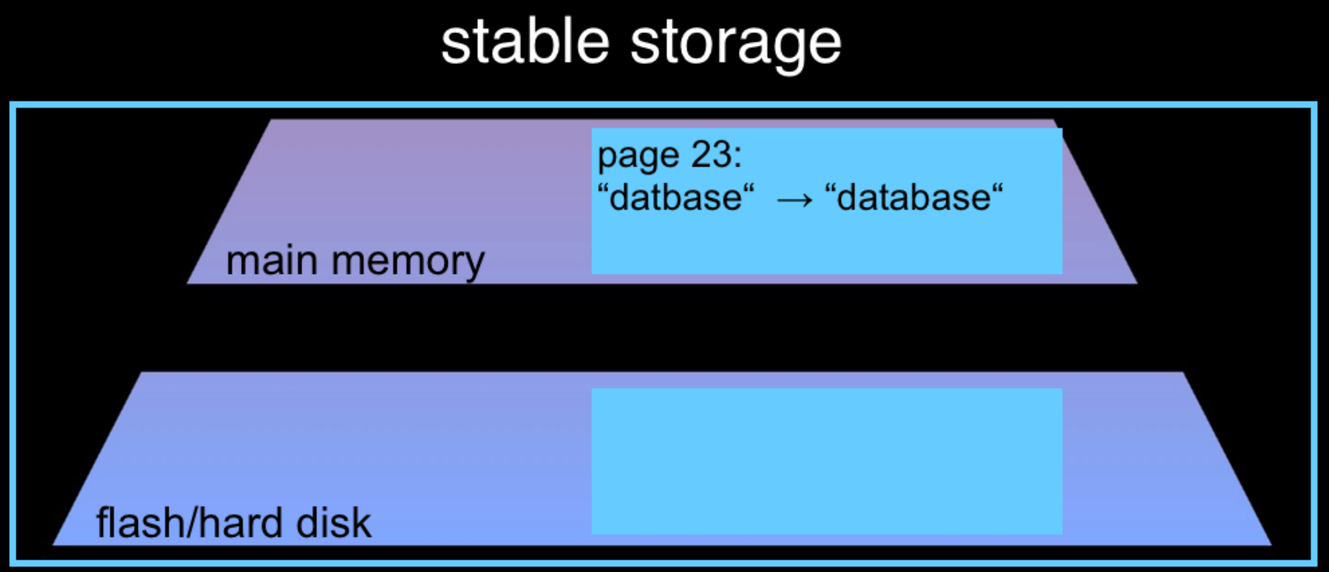
Starting: No Changes yet



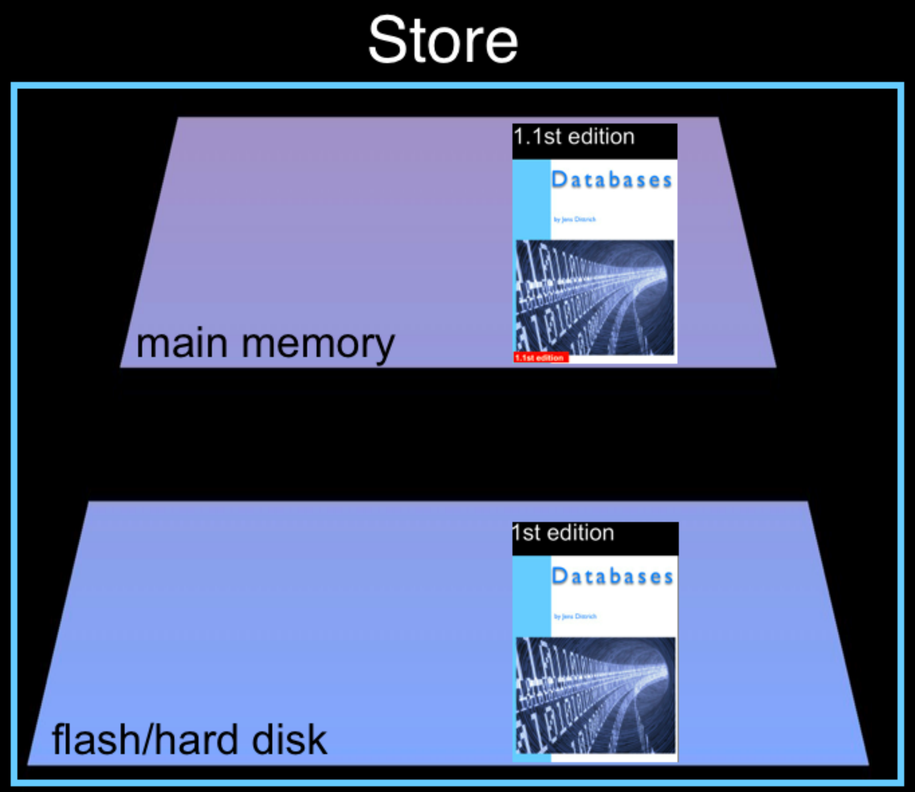


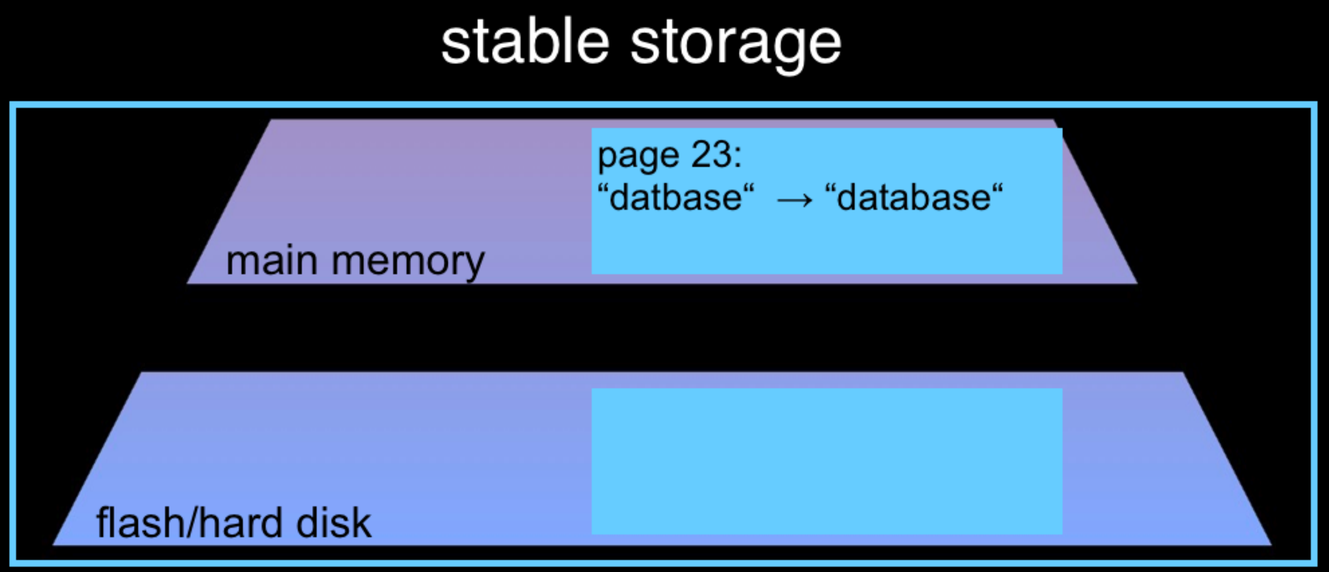
There was a change, let‘s log it.



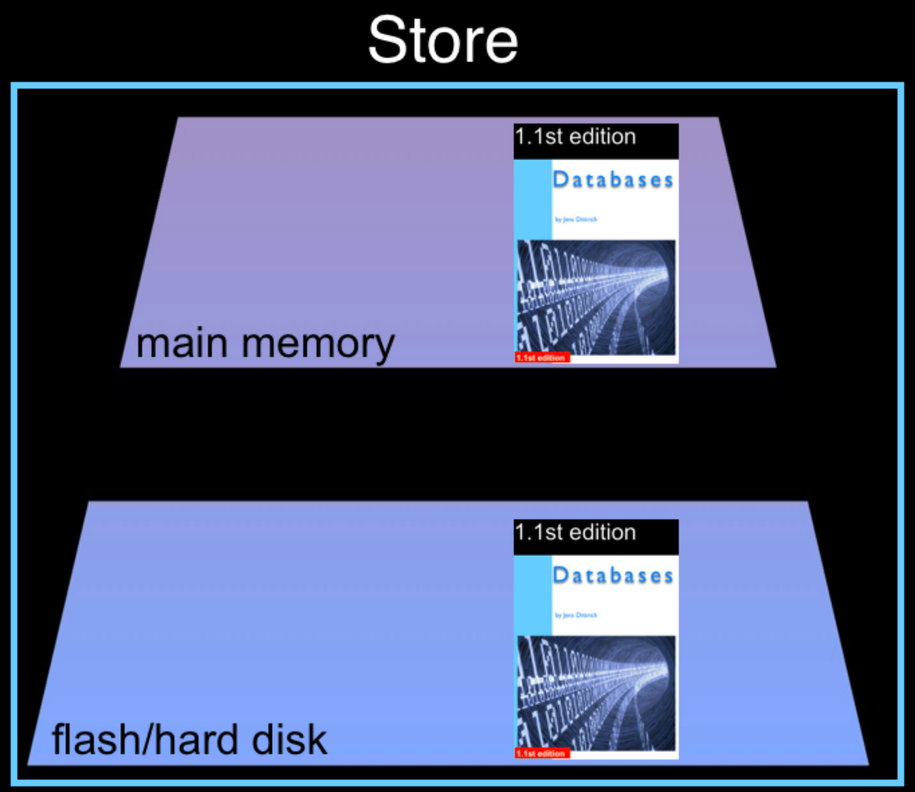


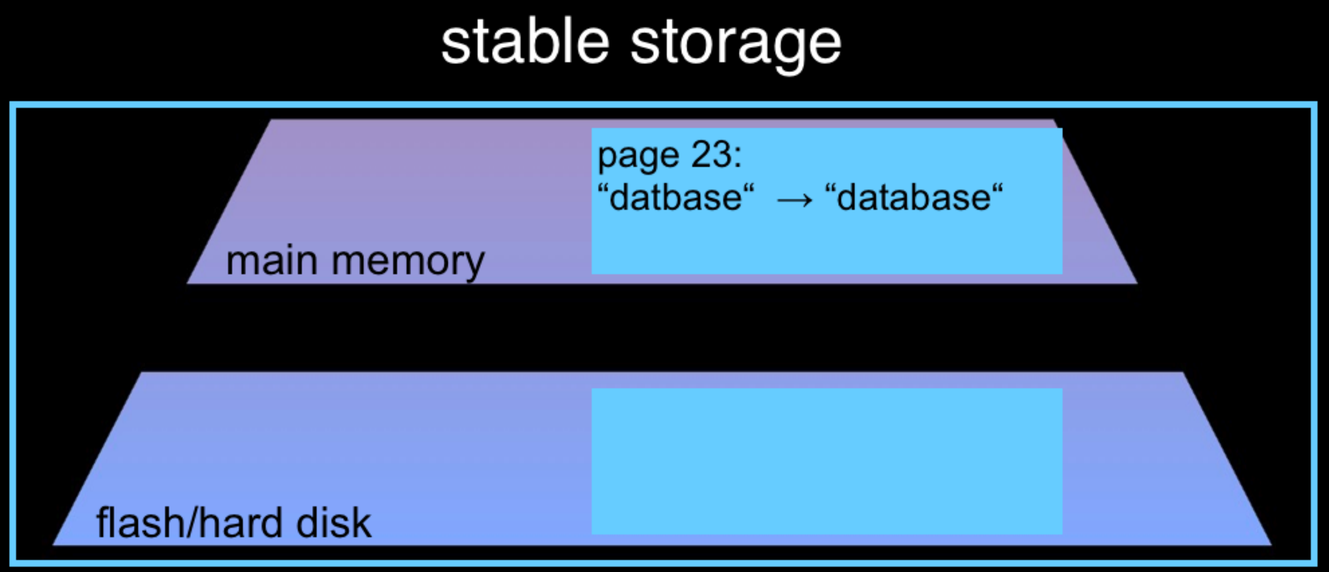
First: create a new Edition 1.1 in Main Memory



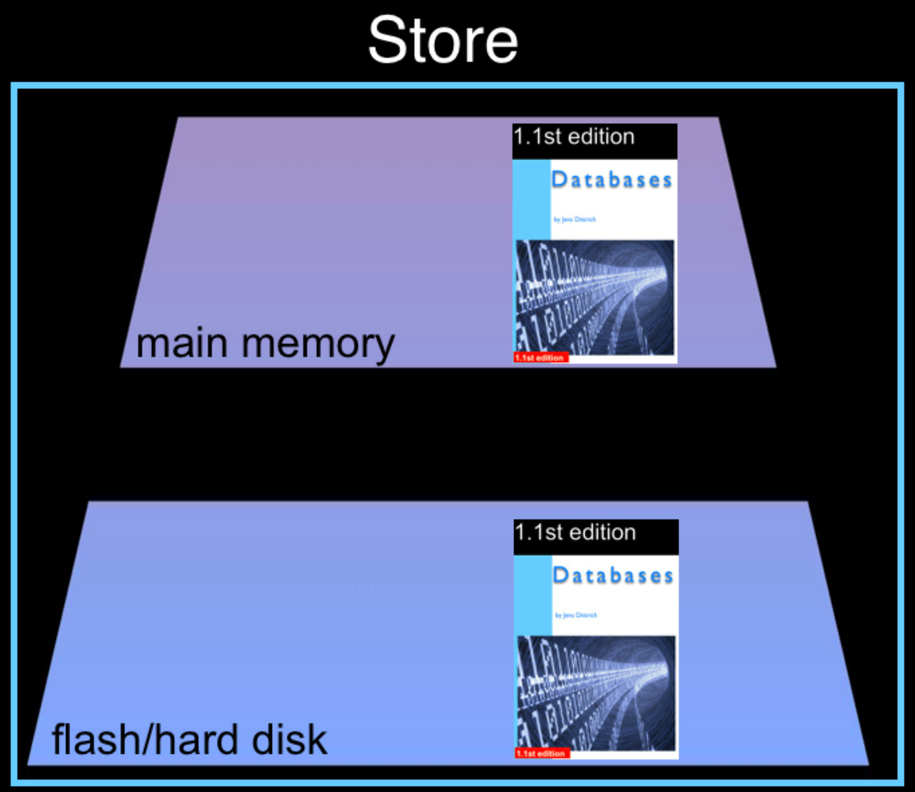


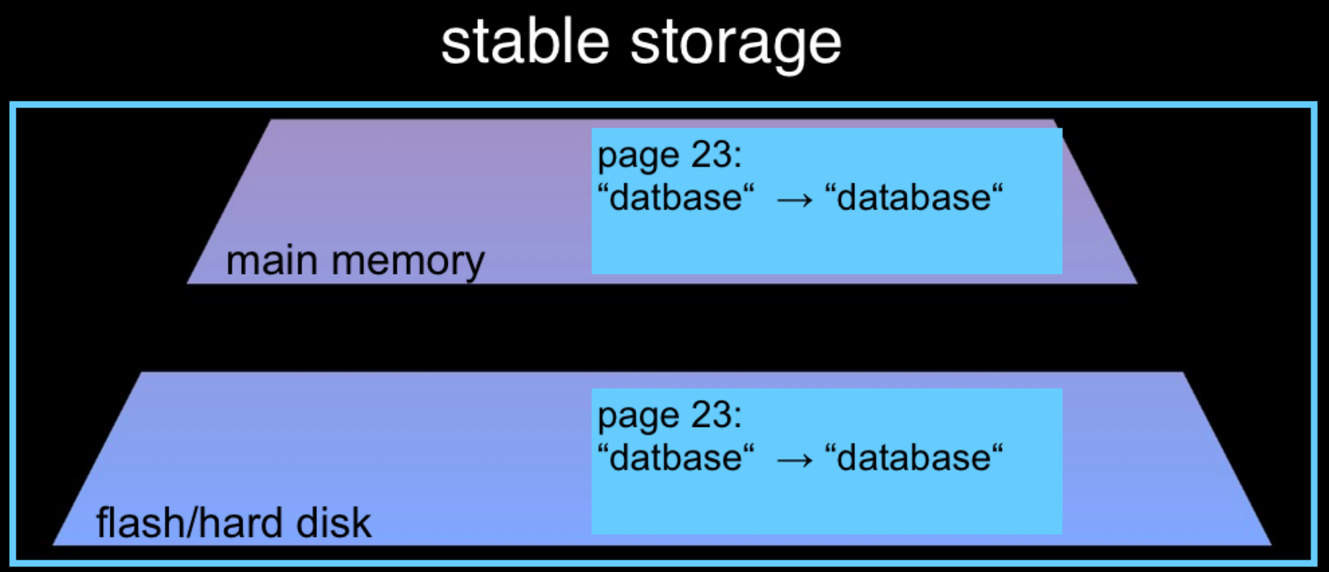
Then write Edition 1.1 to Disk





Then: Force the log entry to disk.





WAL Principle

when committing a transaction:

first: force log entry to log disk / stable storage

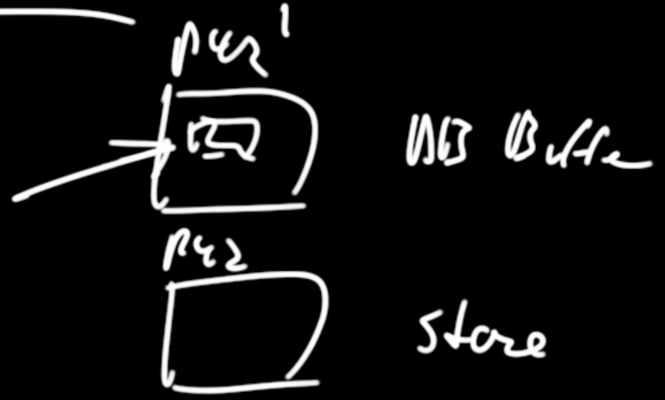
then: write changed page to disk store

when writing back **any dirty page** to the disk store: evict

first: force all corresponding log entries to log disk

then: write changed page to disk store

overall:
1.: flush log entries
2.: flush pages of db store



Credits and Copyrights