



YATA: COLLABORATIVE DOCS AND HOW TO MAKE THEM FAST

INTRODUCTION

Bartosz Sypytkowski

- @Horusiath
- b.sypytkowski@gmail.com
- bartoszsypytkowski.com



AGENDA

- When do we need CRDTs
- YATA: conflict resolution for arrays and maps
- Update merge & split
- Optimizations

YJS.DEV

PROBLEM STATEMENT

COLLABORATIVE TEXT EDITOR

COLLABORATIVE TEXT EDITOR

Alice



Bob



Carol



COLLABORATIVE TEXT EDITOR

Alice



Bob



`insert(1, 'b')`

Carol



`insert(1, 'c')`

COLLABORATIVE TEXT EDITOR

Alice



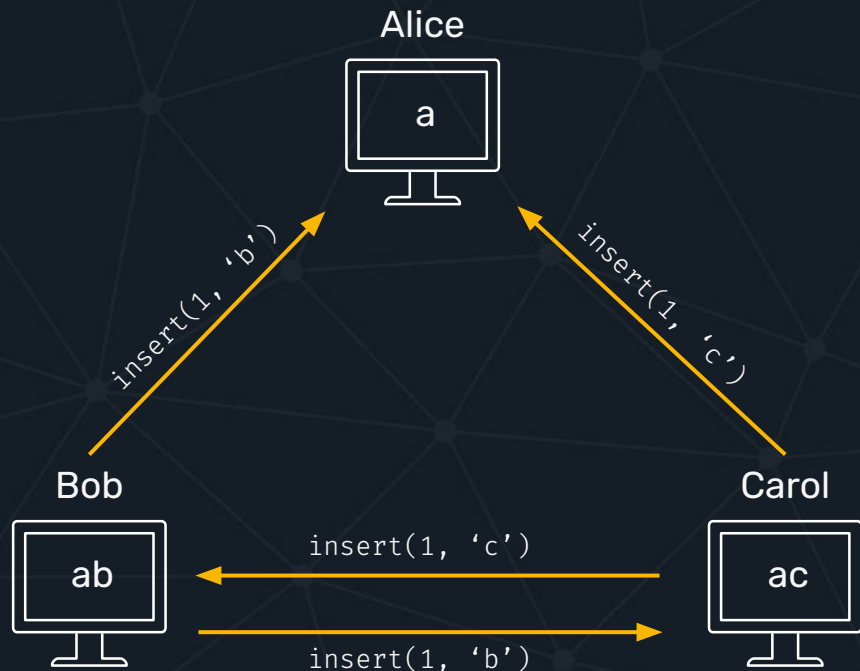
Bob



Carol



COLLABORATIVE TEXT EDITOR



COLLABORATIVE TEXT EDITOR

Alice



Bob



Carol



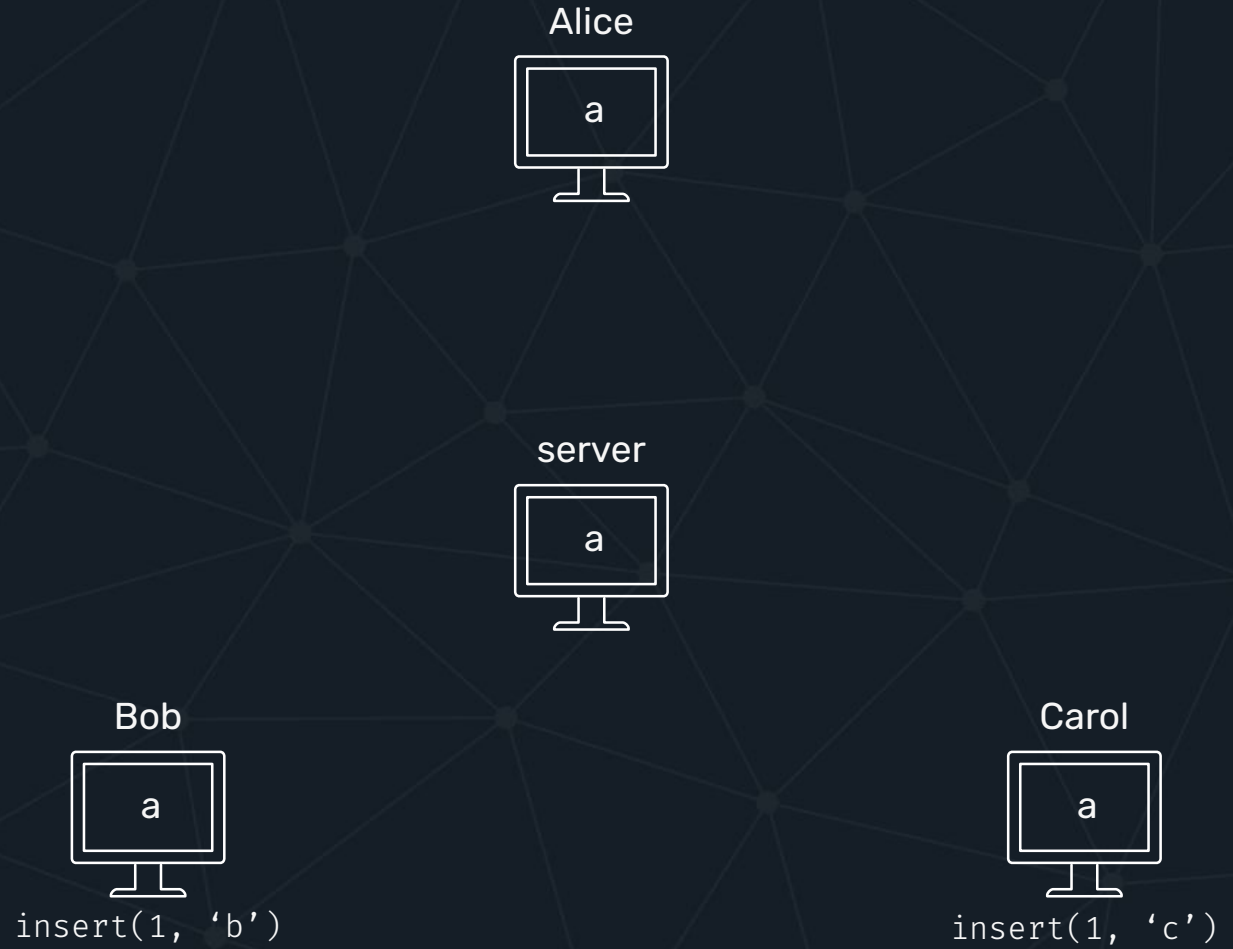


MAYBE JUST SYNCHRONIZE VIA CENTRAL SERVER?

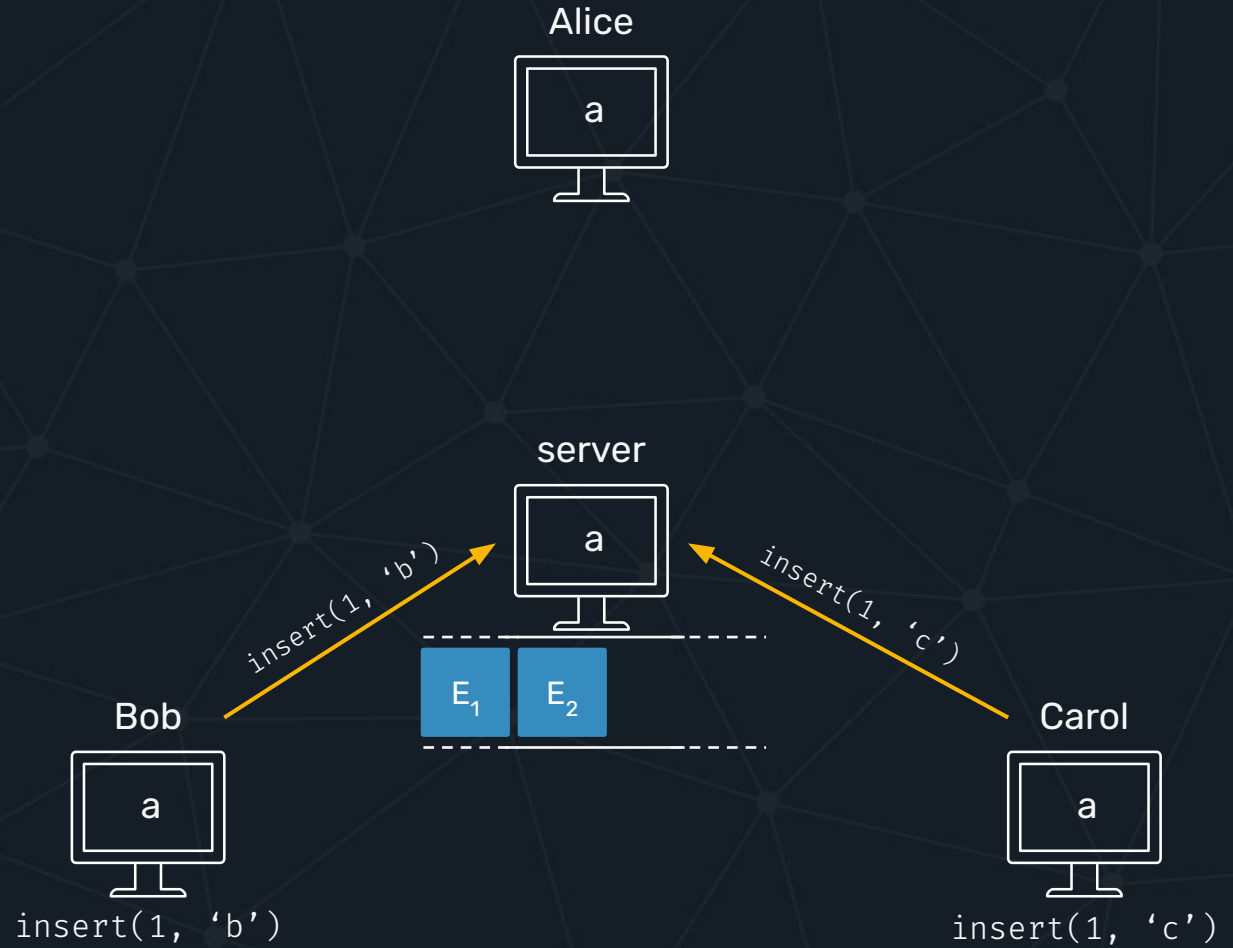
SERVER-DRIVEN TEXT EDITOR



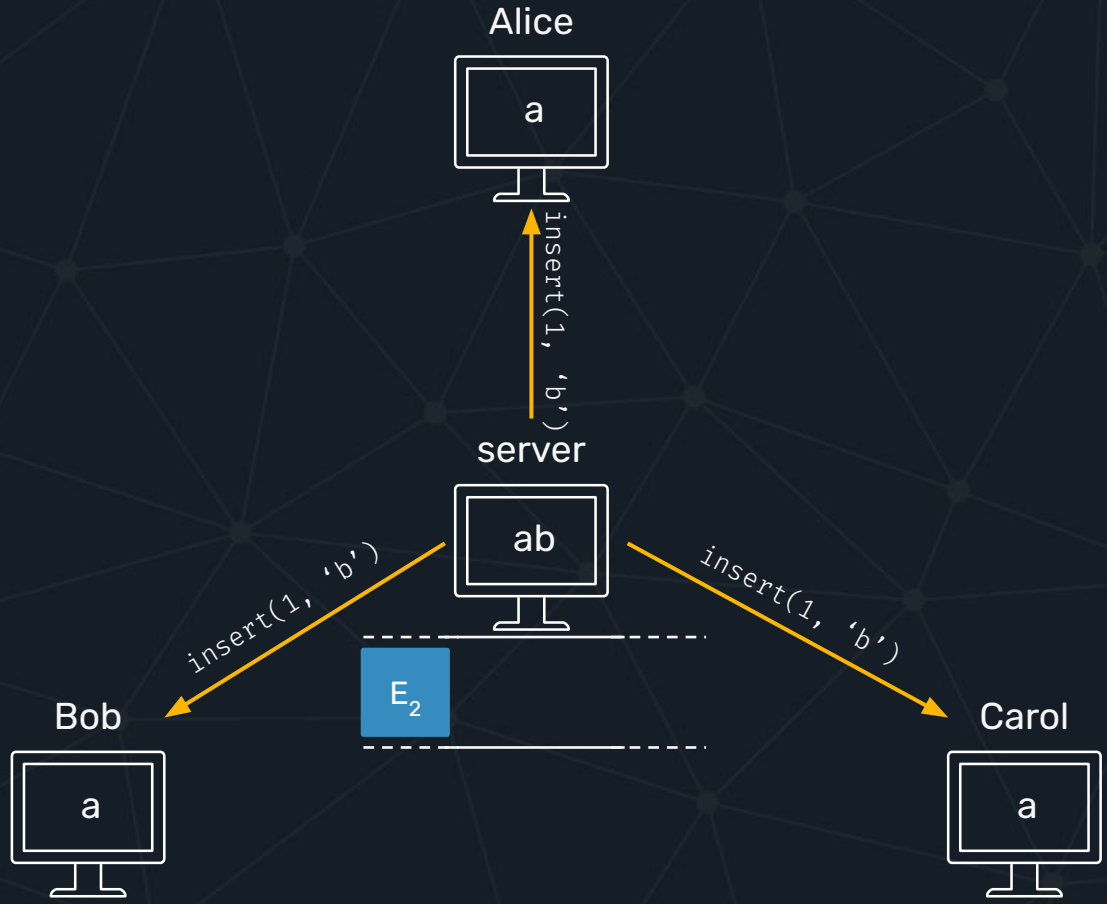
SERVER-DRIVEN TEXT EDITOR



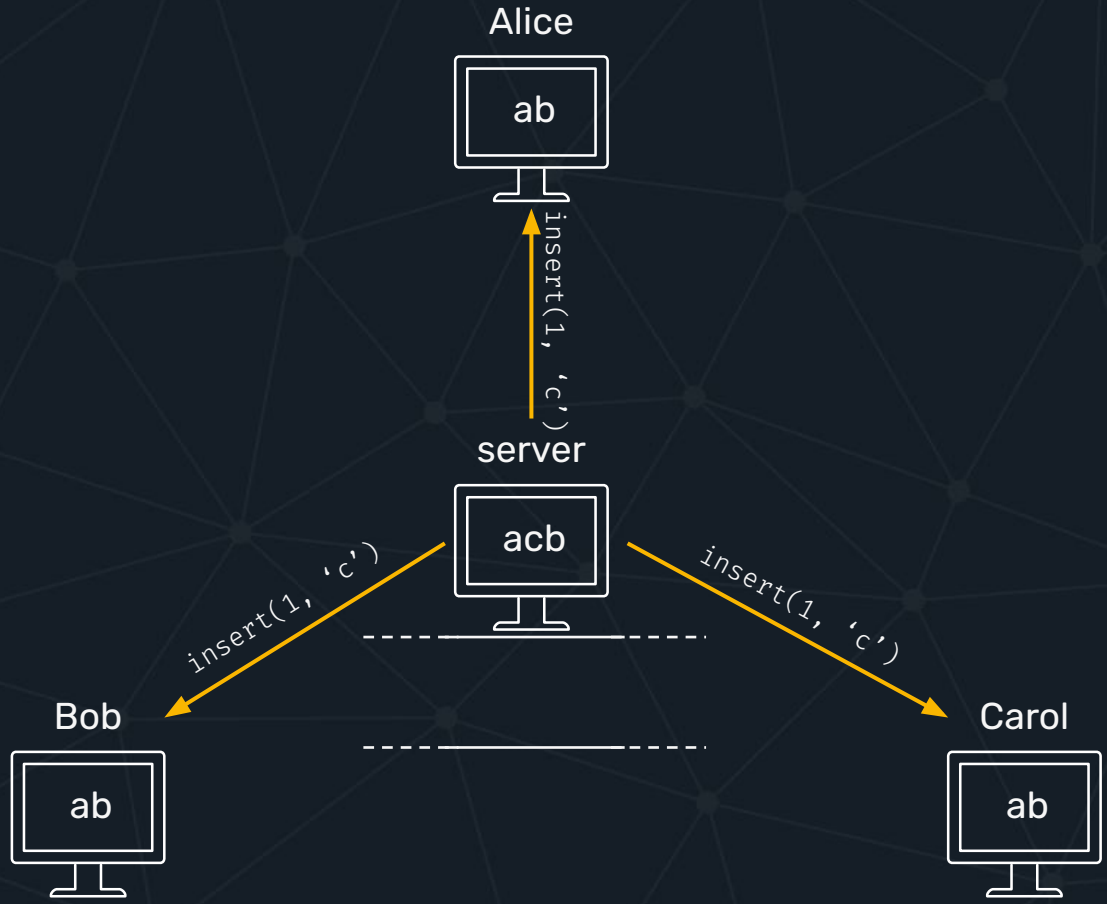
SERVER-DRIVEN TEXT EDITOR



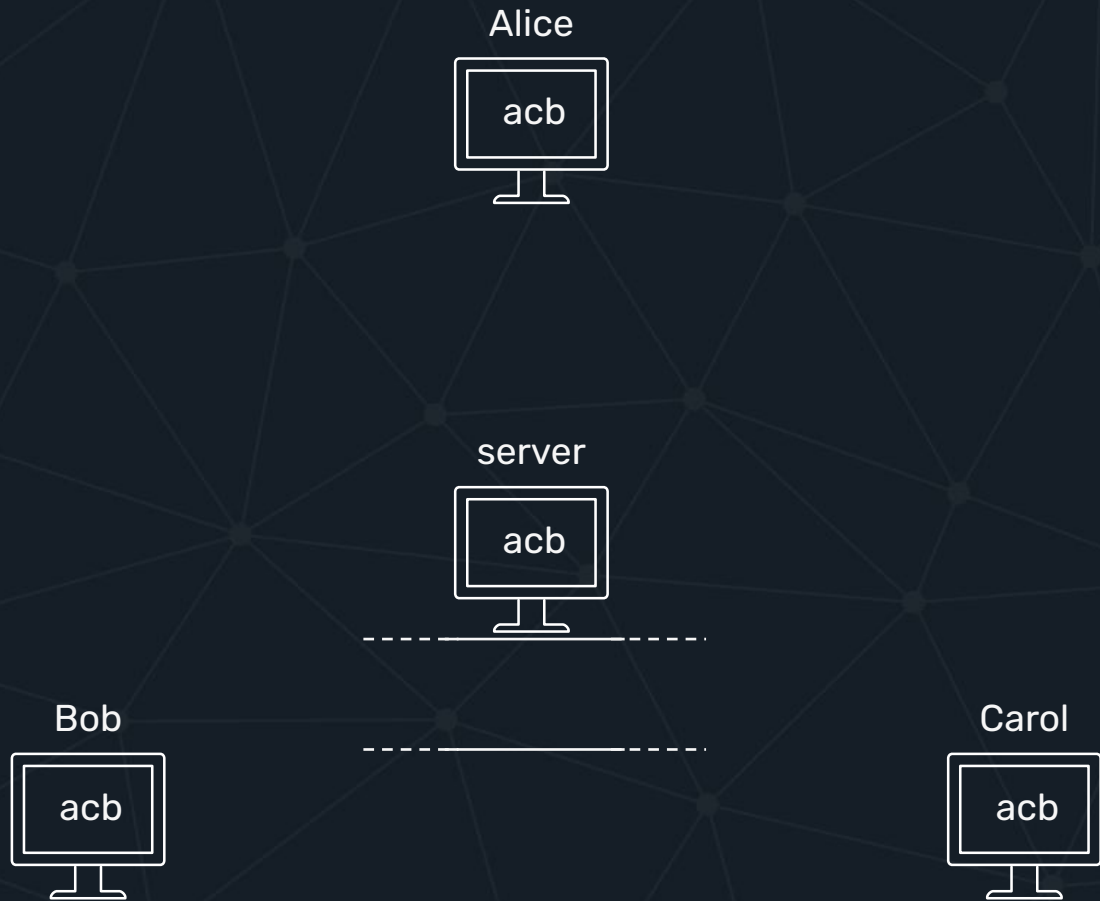
SERVER-DRIVEN TEXT EDITOR



SERVER-DRIVEN TEXT EDITOR



SERVER-DRIVEN TEXT EDITOR



ISSUES

1. Latency
2. Online-only / Network issues
3. Characters interleaving
4. Server: bottleneck & point of failure

FIELDS SHARING SIMILAR PROBLEMS

- 1. Collaborative text editors*
- 2. Cross-continental data replication*
- 3. Vehicle apps*
- 4. Remote areas*
- 5. One-account/multi-device sync*

CONFLICT AVOIDANCE

**Decisions are made by
quorum**

“Let the majority decide on
the correct order.”

CONFLICT AVOIDANCE

**Decisions are made by
quorum**

“Let the majority decide on
the correct order.”

CONFLICT RESOLUTION

**Decisions are made
individually**

“Given enough context
everyone should come to the
same conclusion.”

CONFLICT RESOLUTION

IT'S ALL ABOUT PRESERVING THE ORIGINAL INTENT

SEMANTICS MATTER

1. Insert characters one after another
 2. Fixing a typo
 3. Move an element
 4. Move a range of elements
1. Insert characters at positions X , $X+1$, $X+2$, etc.
 2. Insert/remove character at position X
 3. Remove an element then re-insert it
 4. Delete then re-insert range of elements

DON'T USE INDEXES FOR POSITIONING

THEIR MEANING CHANGES OVER TIME

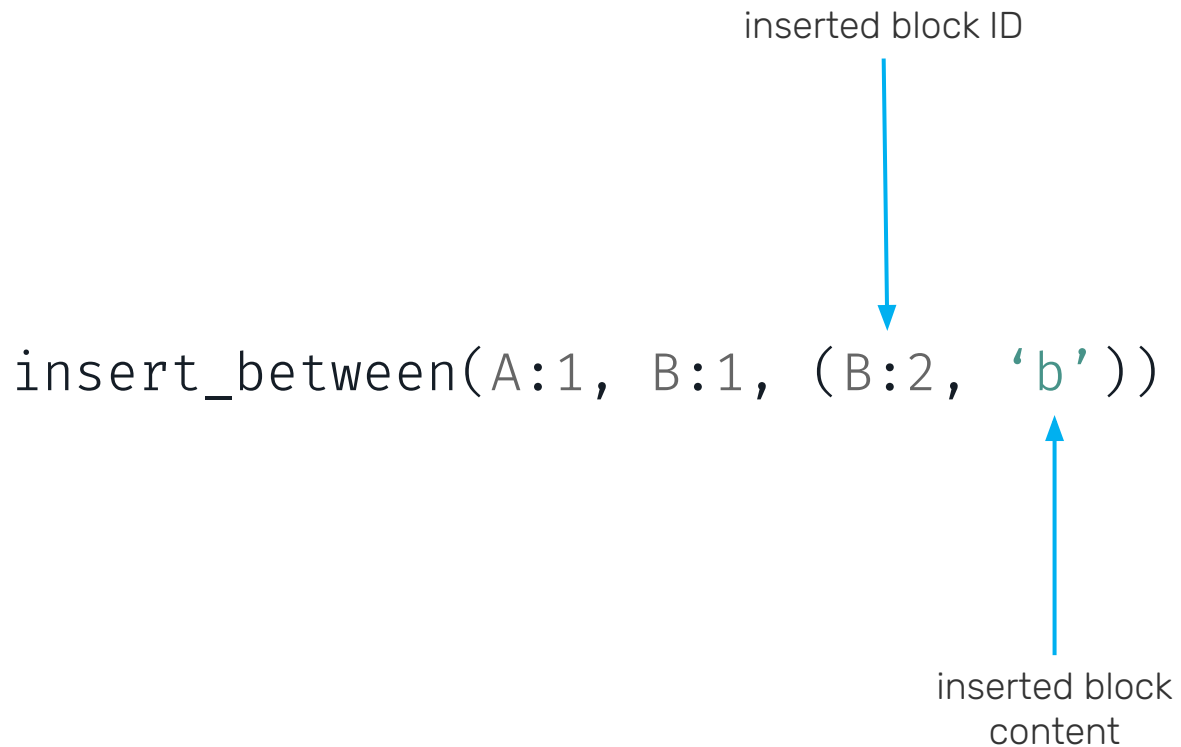
LAMPORT CLOCK



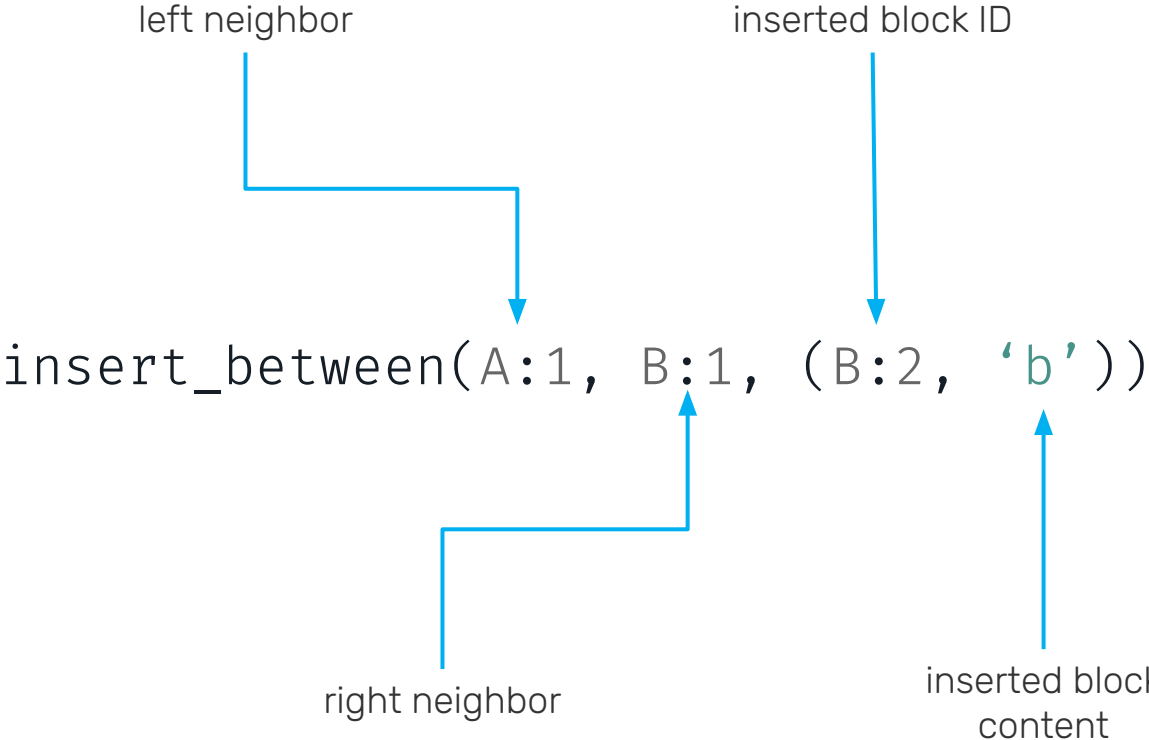
CONTEXT-AWARE INSERTION

```
insert_between(A:1, B:1, (B:2, 'b'))
```

CONTEXT-AWARE INSERTION



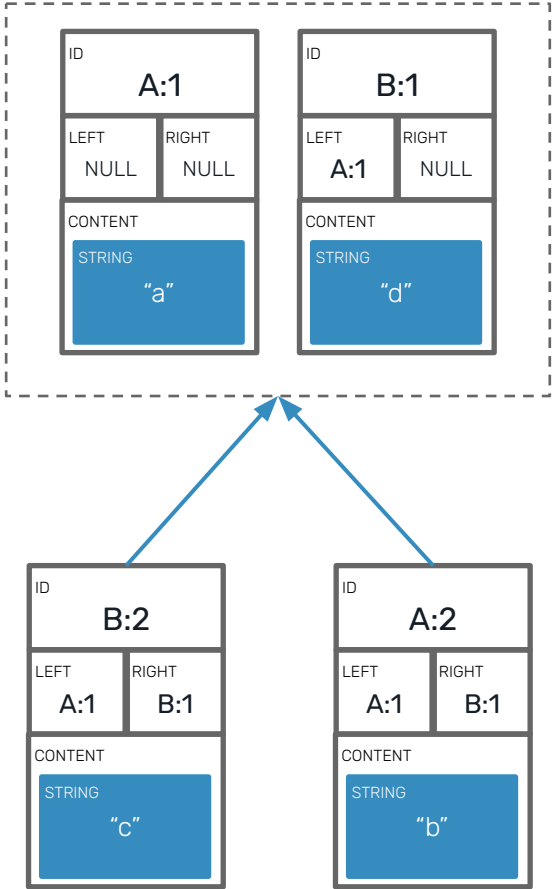
CONTEXT-AWARE INSERTION



CONFLICT RESOLUTION

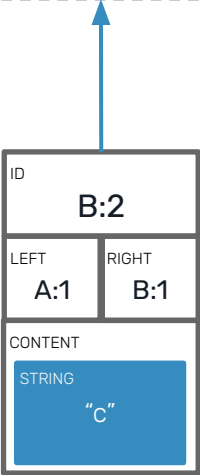
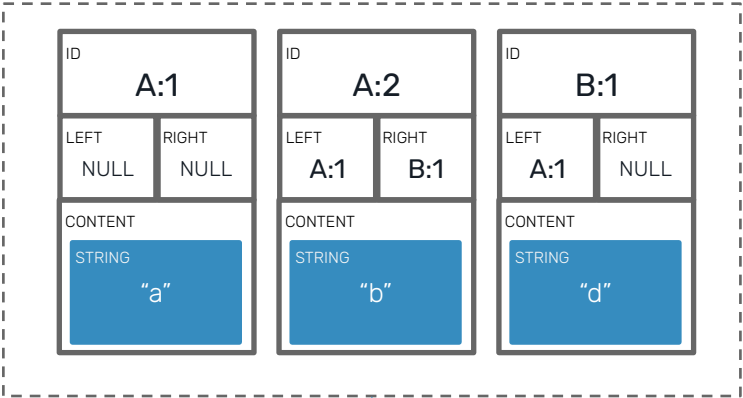
CONFLICT RESOLUTION

Document state



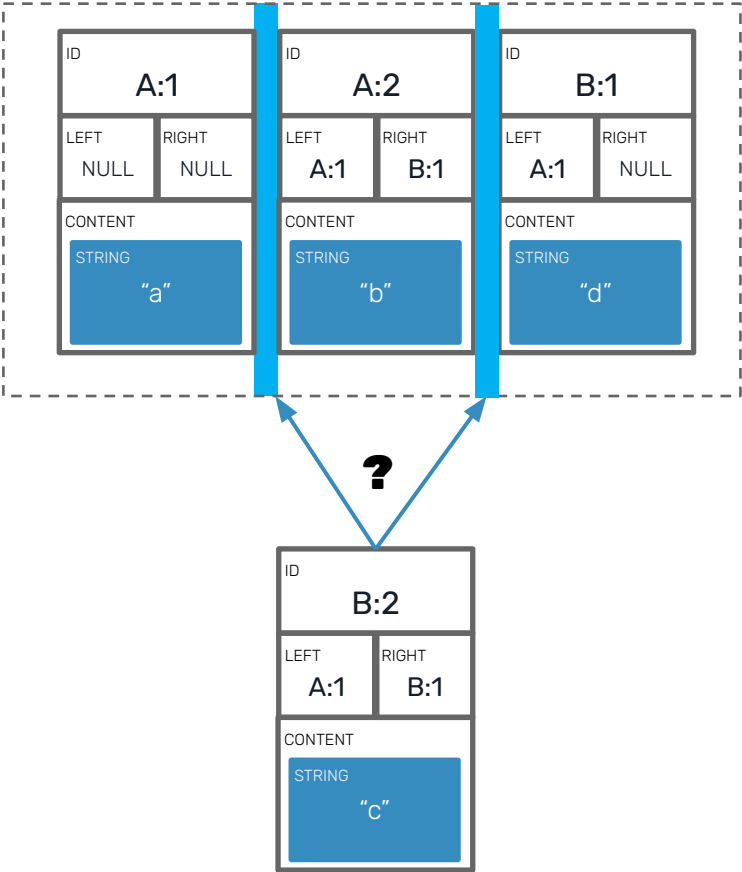
CONFLICT RESOLUTION

Document state



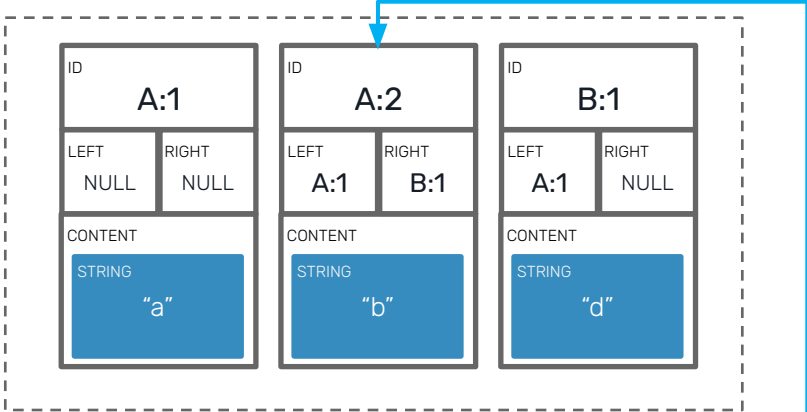
CONFLICT RESOLUTION

Document state

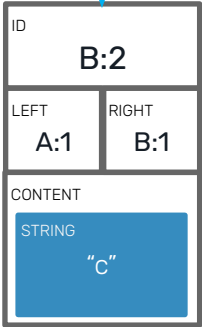


CONFLICT RESOLUTION

Document state

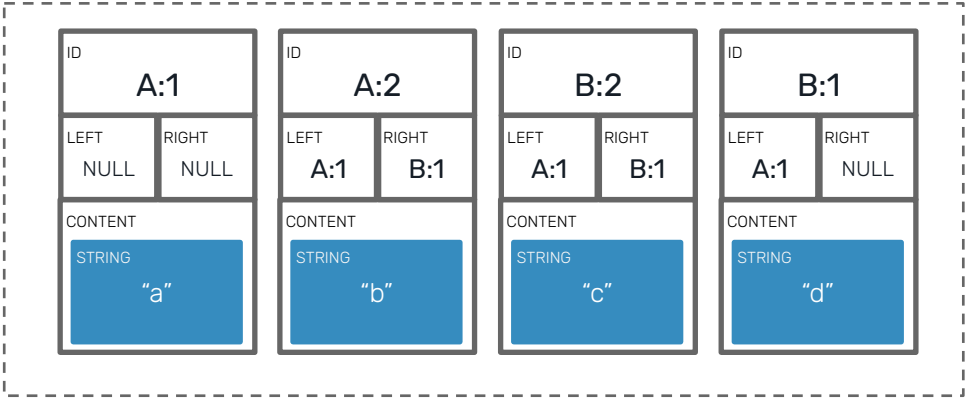


Use block IDs to skip over blocks with lower precedence



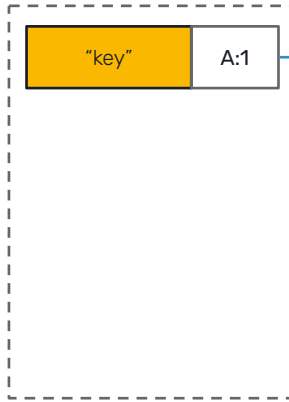
CONFLICT RESOLUTION

Document state

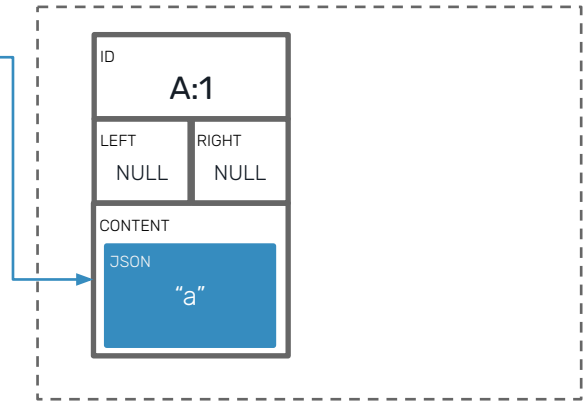


YATA FOR MAPS

YMap



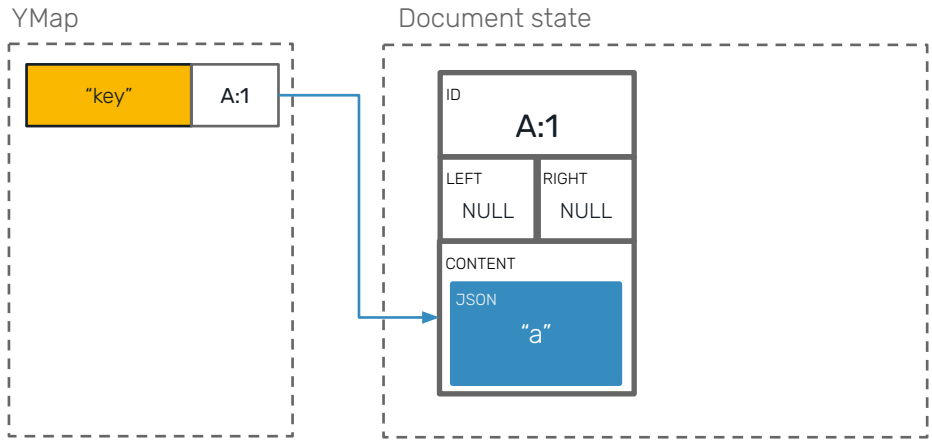
Document state



YMAP

ENTRY INSERTION

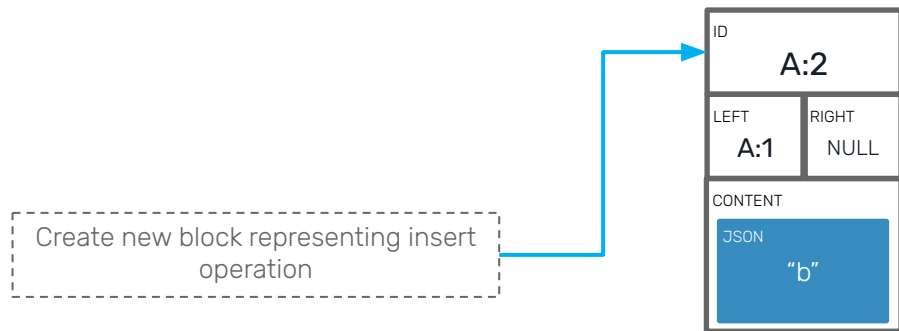
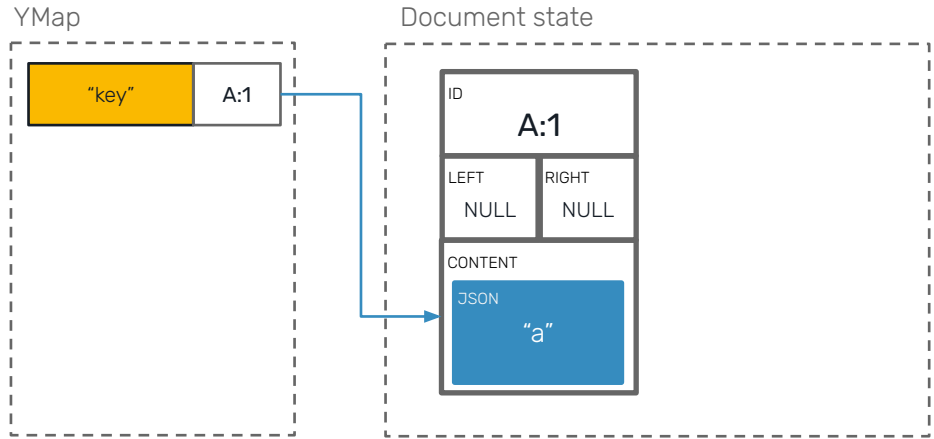
```
ymap.set('key', 'b')
```



YMAP

ENTRY INSERTION

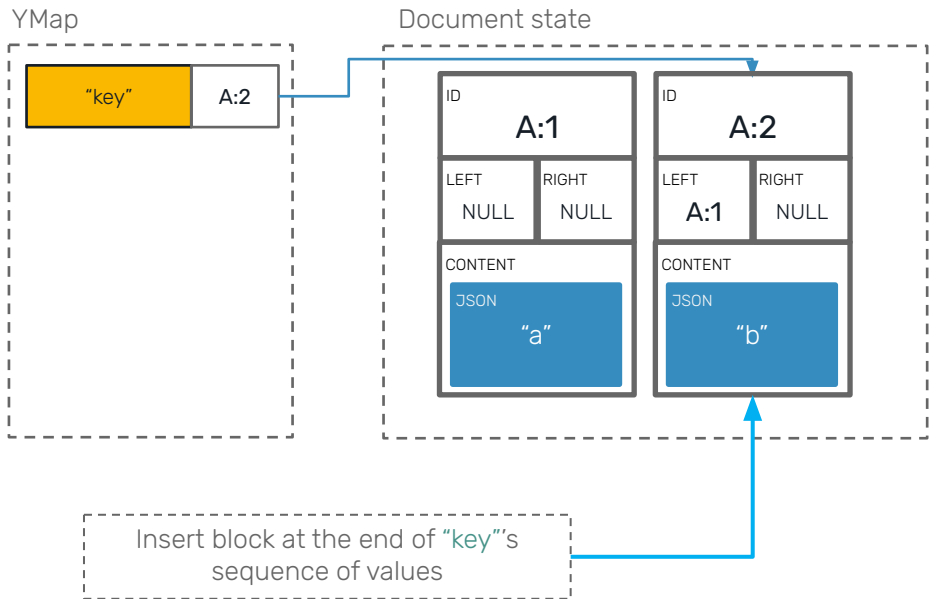
```
ymap.set('key', 'b')
```



YMAP

ENTRY INSERTION

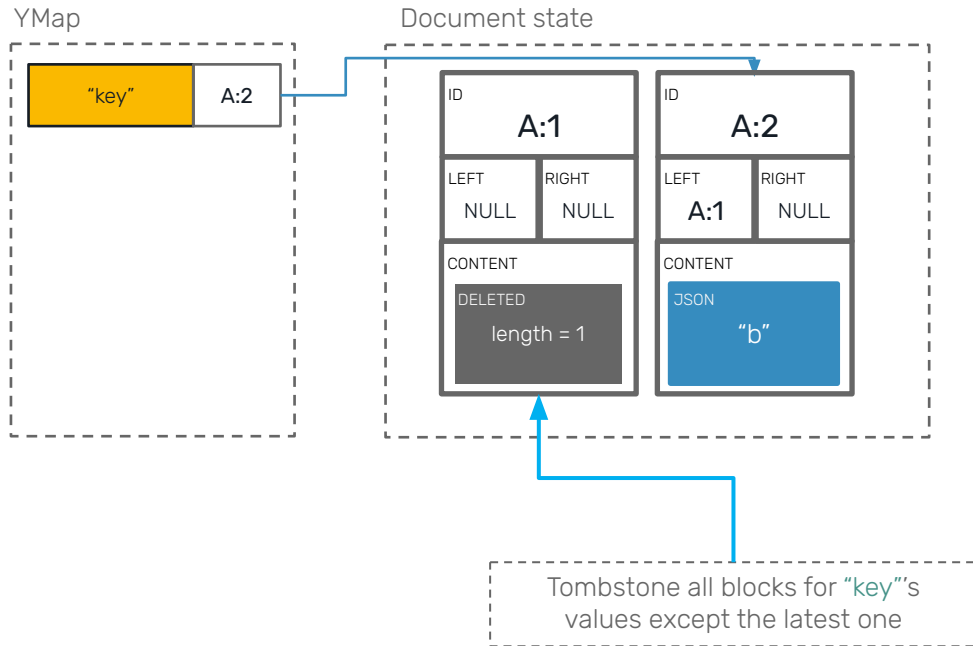
```
yomap.set('key', 'b')
```



YMAP

ENTRY INSERTION

```
ymap.set('key', 'b')
```



YMAP

ENTRY INSERTION

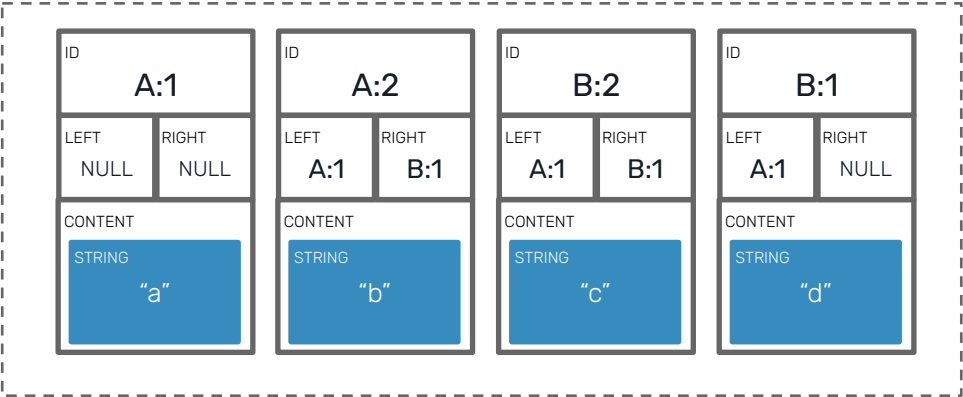
MOVING ELEMENTS



WHY DON'T JUST USE REMOVE + INSERT?

MOVING ELEMENTS

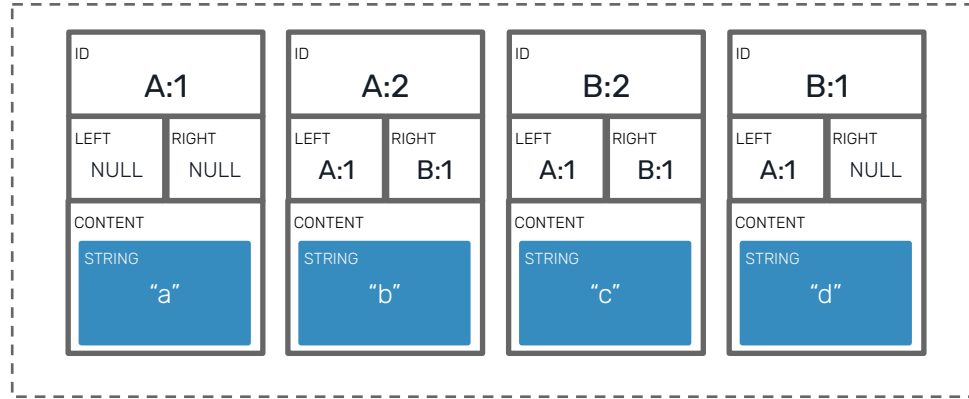
Document state



MOVING ELEMENTS

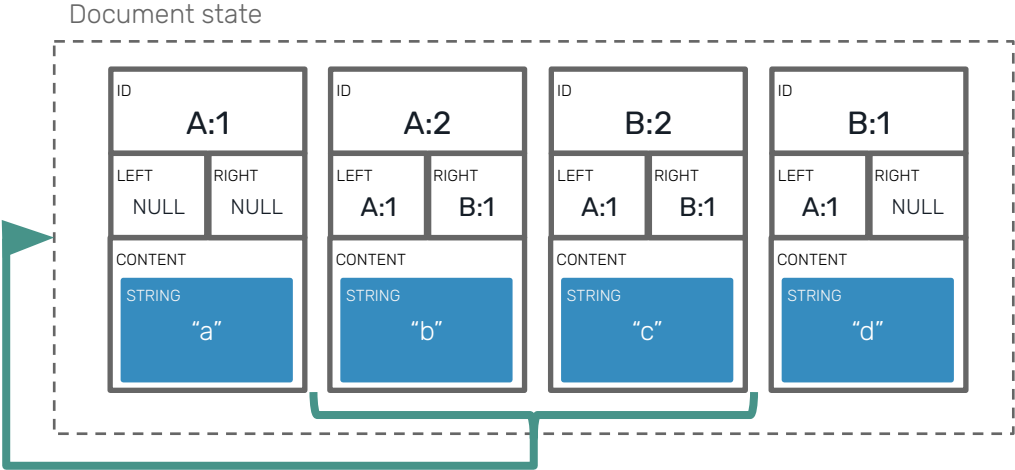
```
doc.move(1..2, 0)
```

Document state



MOVING ELEMENTS

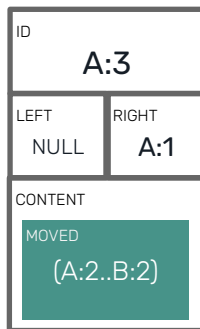
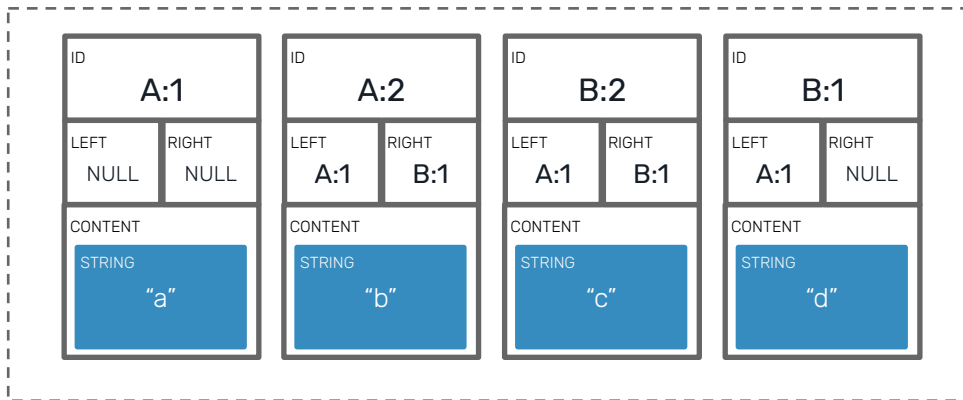
```
doc.move(1..2, 0)
```



MOVING ELEMENTS

```
doc.move(1..2, 0)
```

Document state

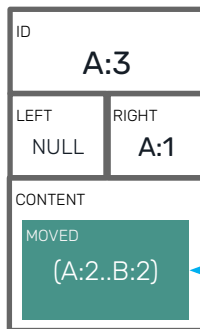
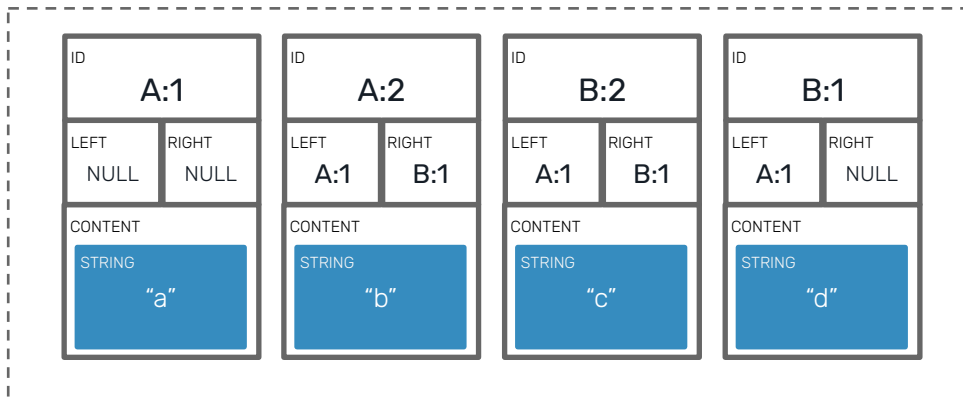


Create new block representing move operation

MOVING ELEMENTS

```
doc.move(1..2, 0)
```

Document state

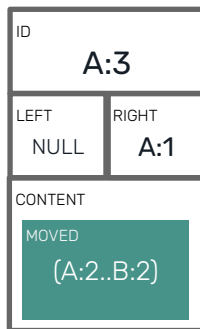
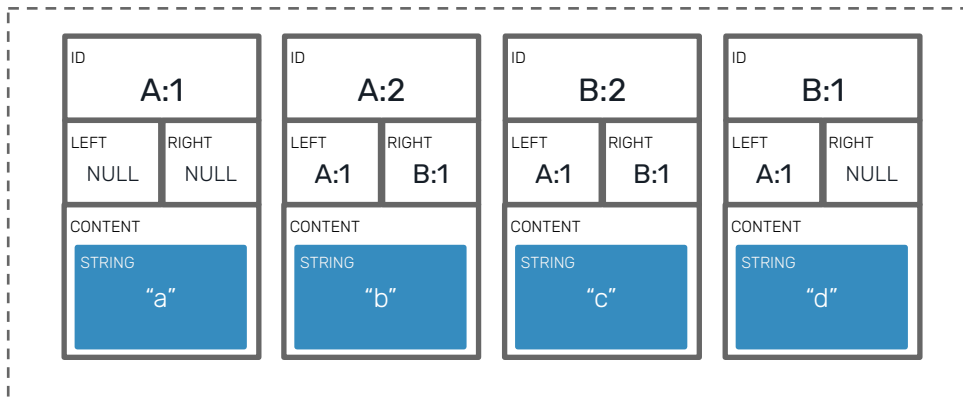


Range 1..2 maps onto continuous sequence of blocks from A:2 to B:2

MOVING ELEMENTS

```
doc.move(1..2, 0)
```

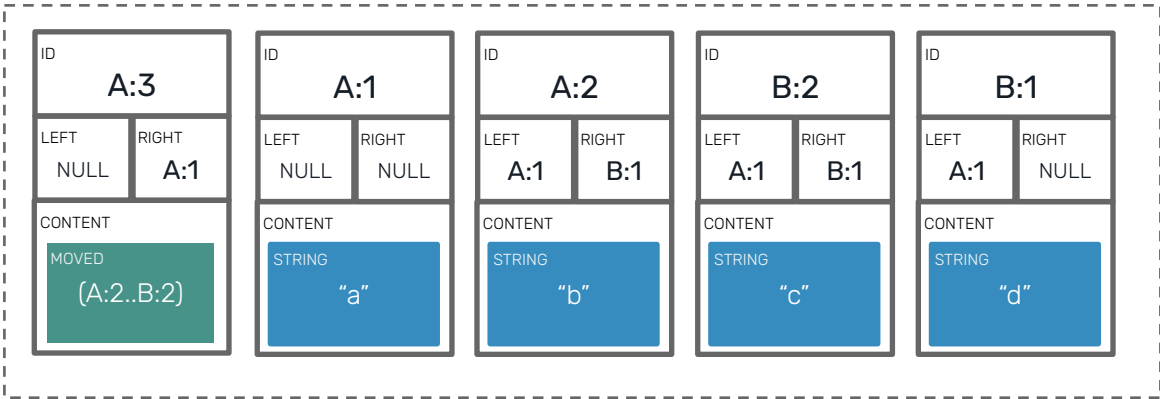
Document state



Destination index 0 suggests to insert this block before A:1

MOVING ELEMENTS

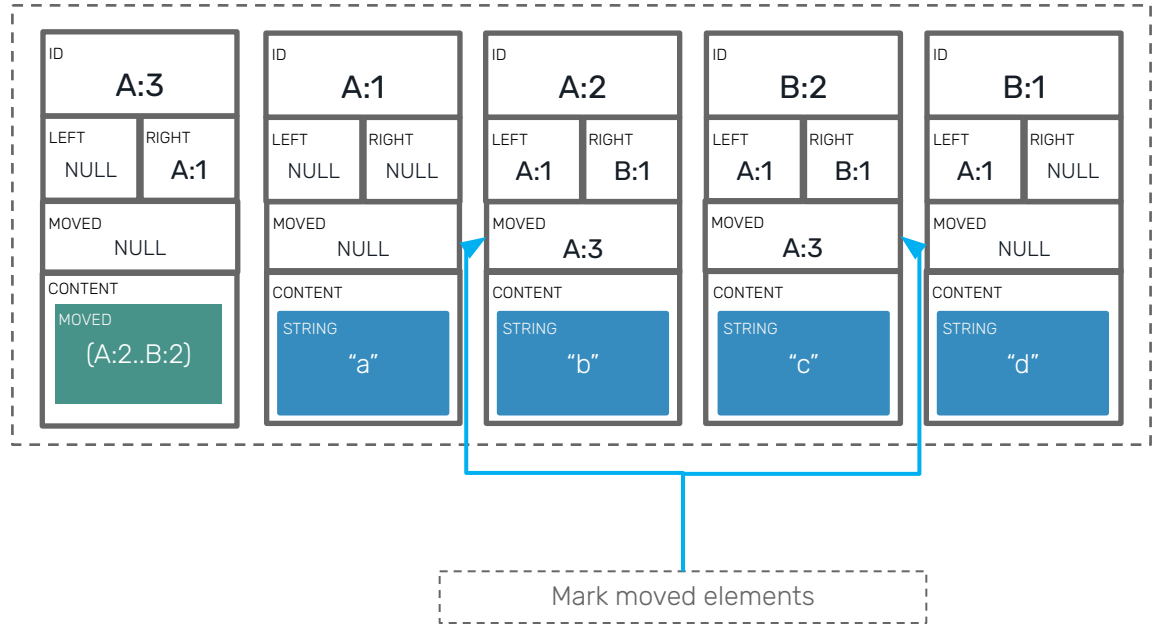
Document state



MOVING ELEMENTS

```
doc.move(1..2, 0)
```

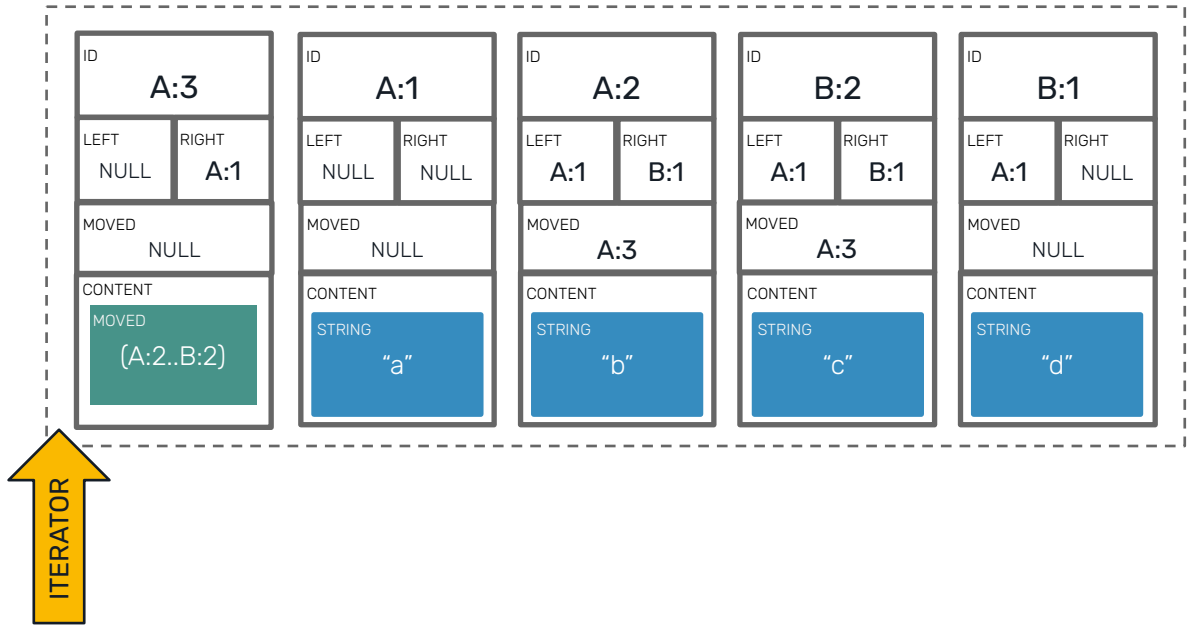
Document state





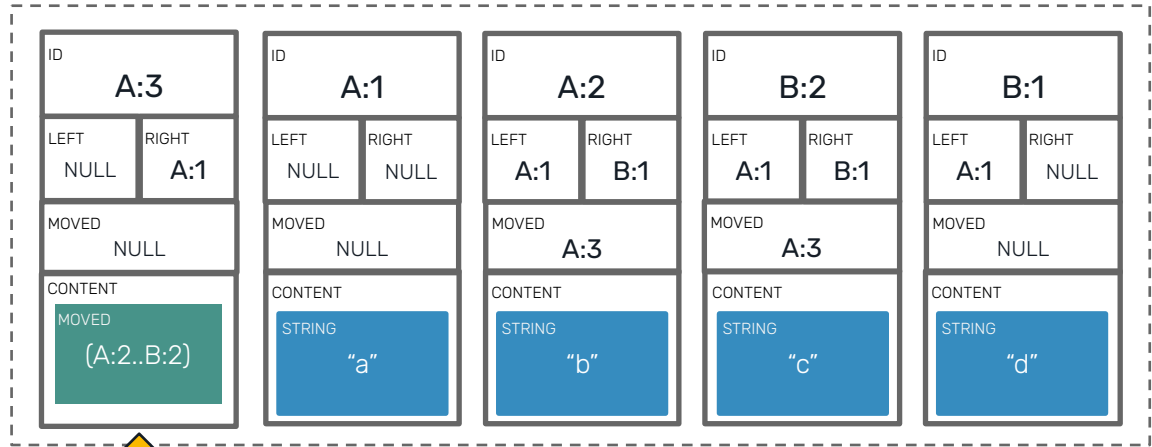
HOW ARE WE GOING TO READ THAT?

Document state

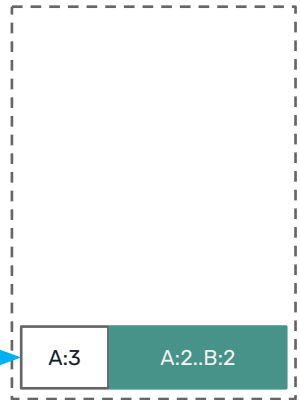


**READING MOVED
ELEMENTS**

Document state



Move stack

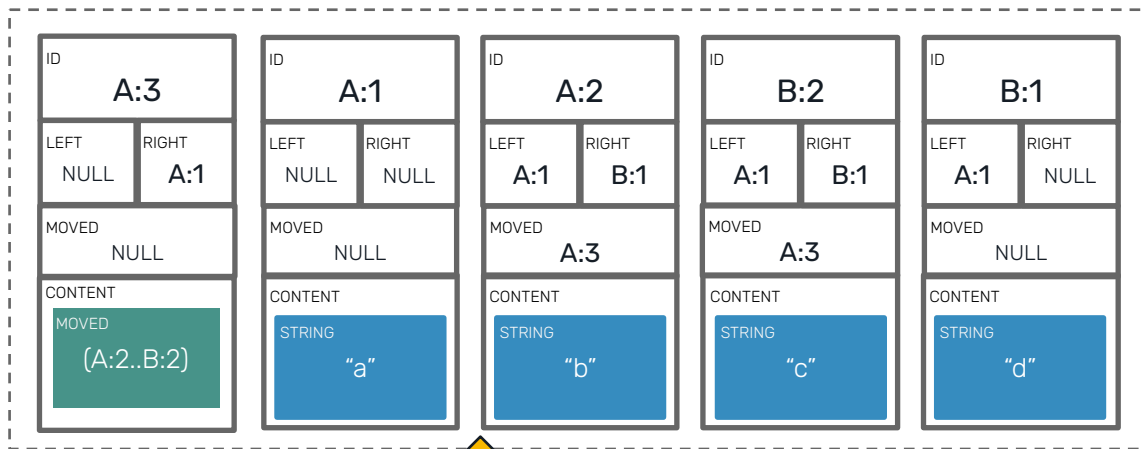


Move frame informs if we're currently within moved range context

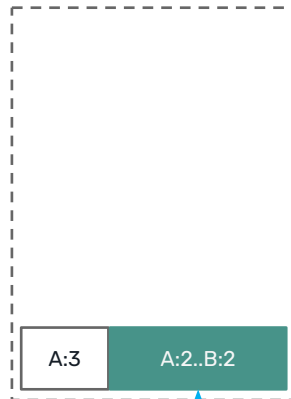


READING MOVED ELEMENTS

Document state



Move stack

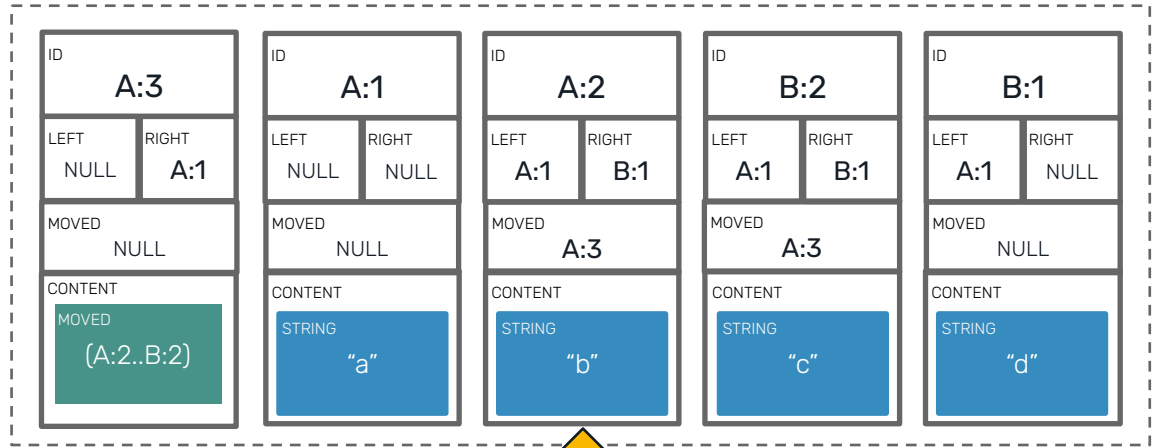


Jump to the beginning of moved range

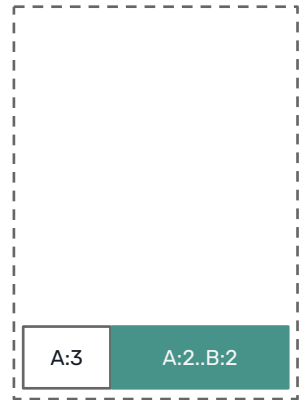


**READING MOVED
ELEMENTS**

Document state

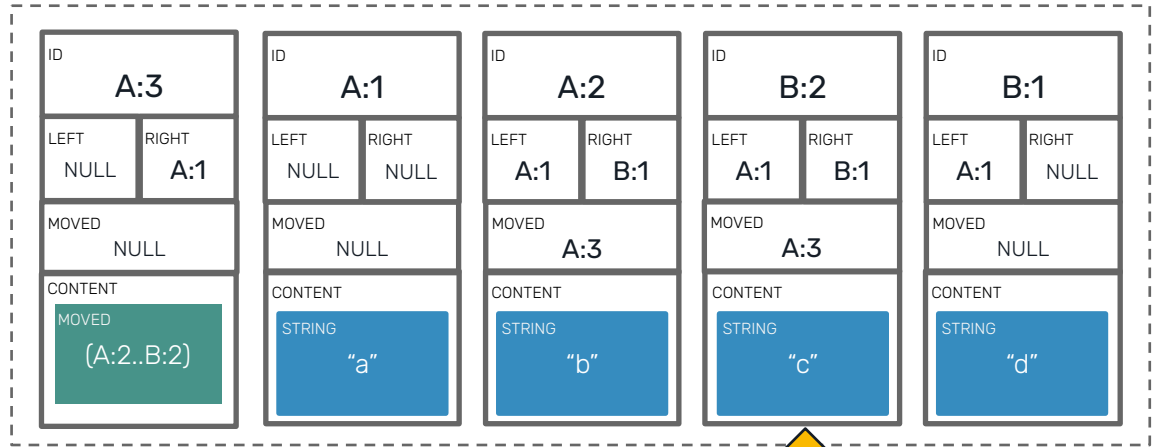


Move stack

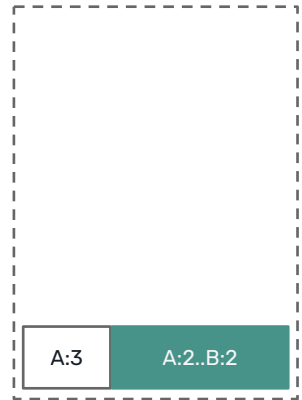


**READING MOVED
ELEMENTS**

Document state

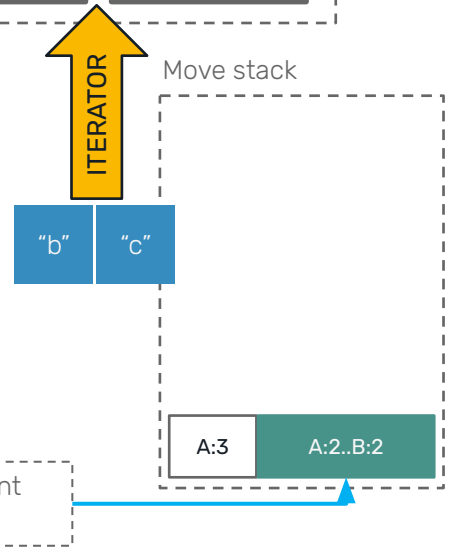
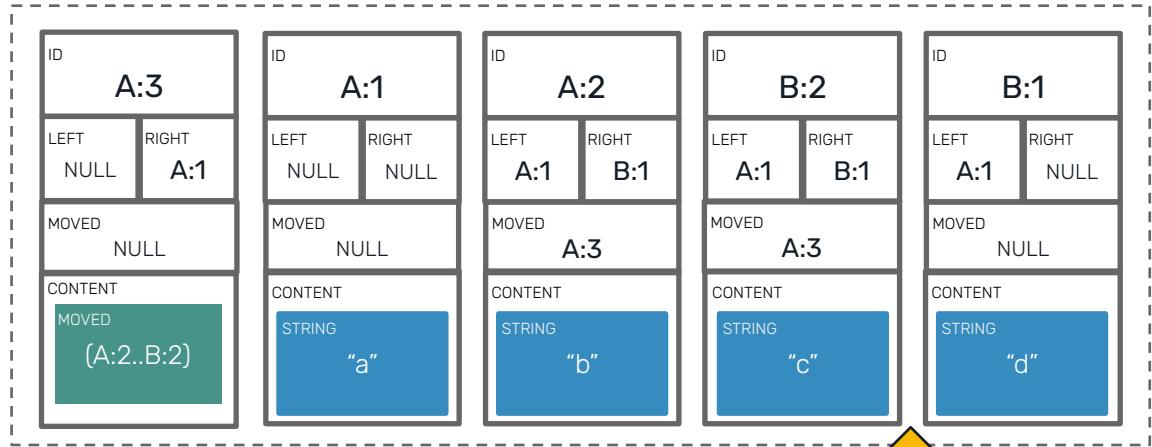


Move stack



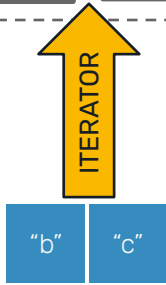
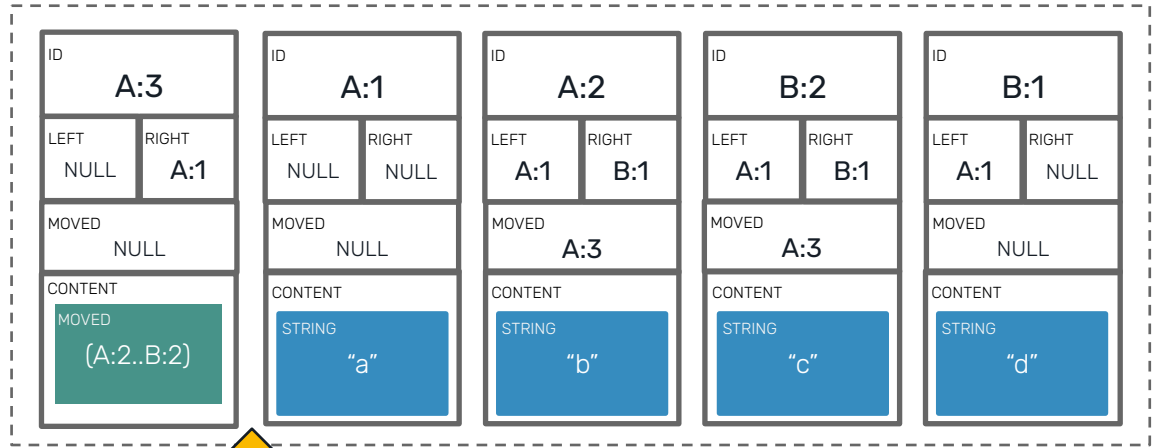
**READING MOVED
ELEMENTS**

Document state



READING MOVED ELEMENTS

Document state

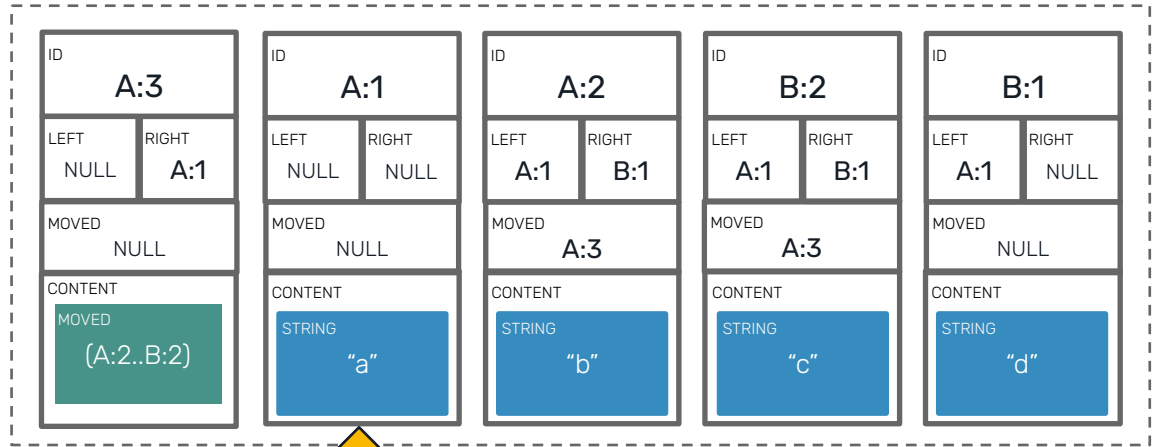


Move stack



**READING MOVED
ELEMENTS**

Document state

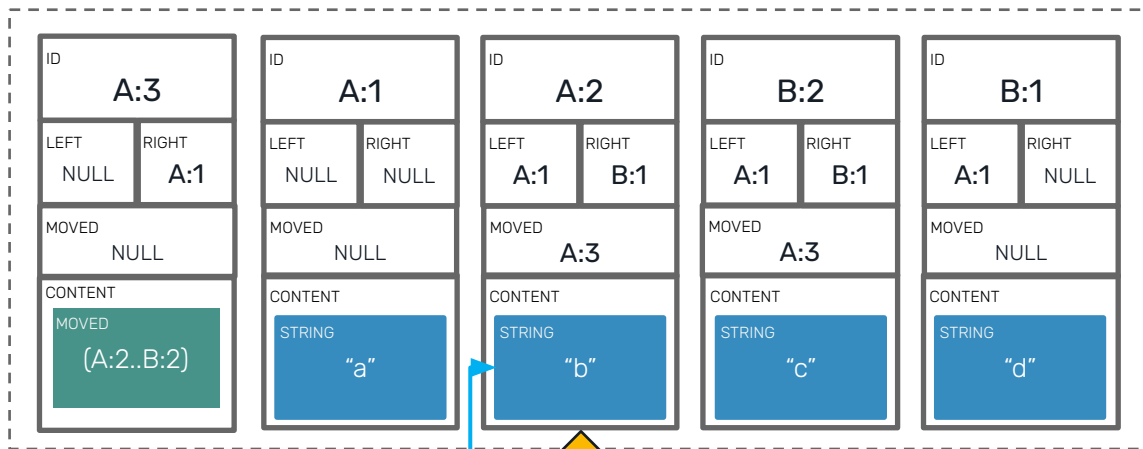


Move stack

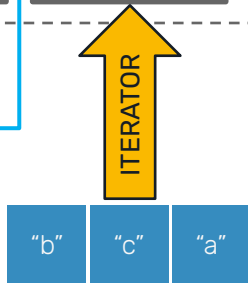


**READING MOVED
ELEMENTS**

Document state



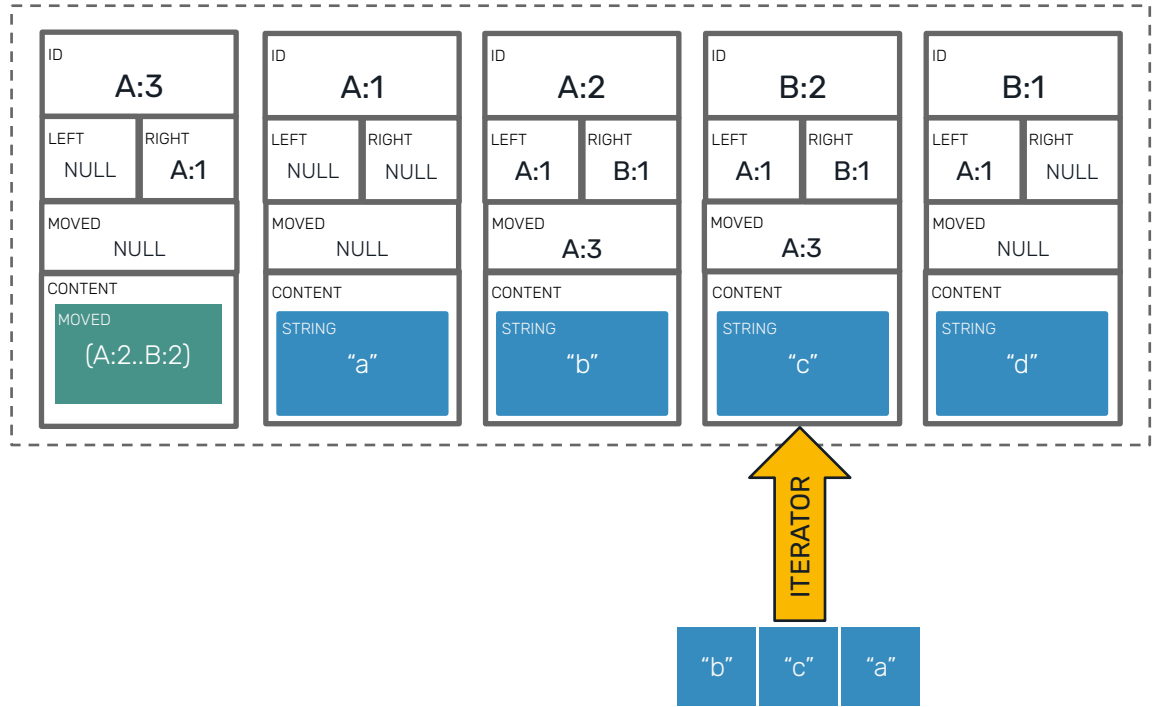
Skip over moved blocks that aren't part of a current move frame



Move stack

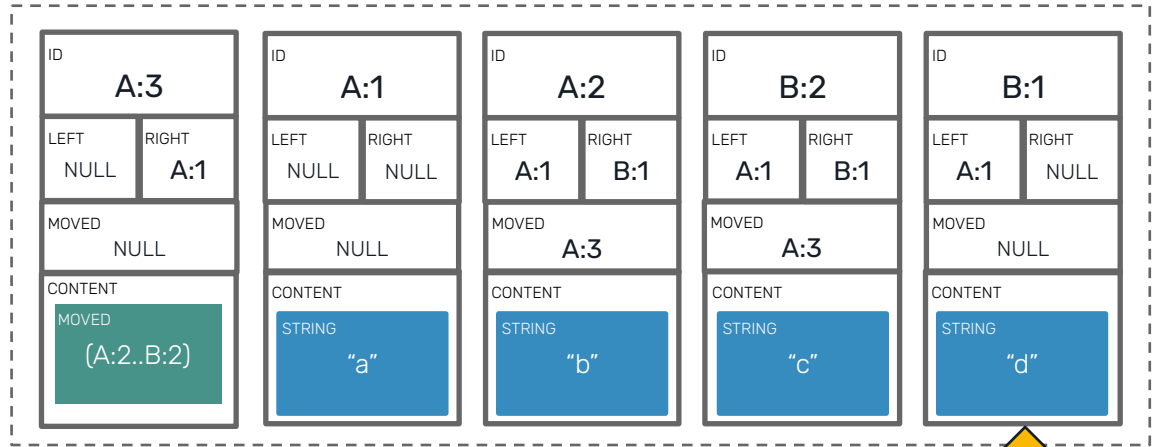
**READING MOVED
ELEMENTS**

Document state



**READING MOVED
ELEMENTS**

Document state



**READING MOVED
ELEMENTS**

OPTIMIZATIONS

1. KNOW YOUR ENVIRONMENT

KNOW THE DIFFERENCE

	Peer to Peer	Client / Server
Examples	Yjs/Yrs, Automerge	RiakDB, AntidoteDB, DynamoDB
Ops / sec.	few* (related to single user activity)	> 1000 ops / sec
Collaborators	unknown, limited control	known, under full control
Network / connections	heterogenous, unreliable	Homogenous, fairly stable
Data volume	fits in memory (hopefully)	greater than disk

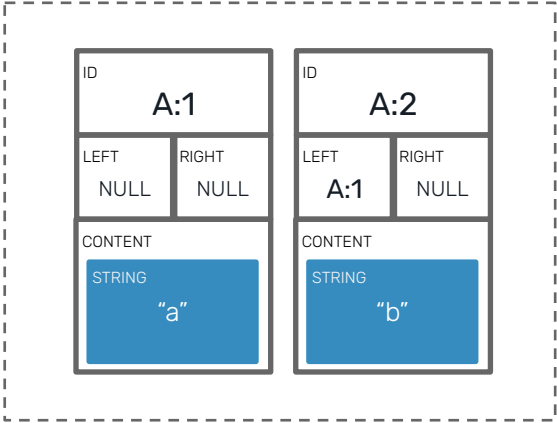
OPTIMIZATIONS

2. REDUCE NUMBER OF OPERATIONS

OPTIMIZATIONS

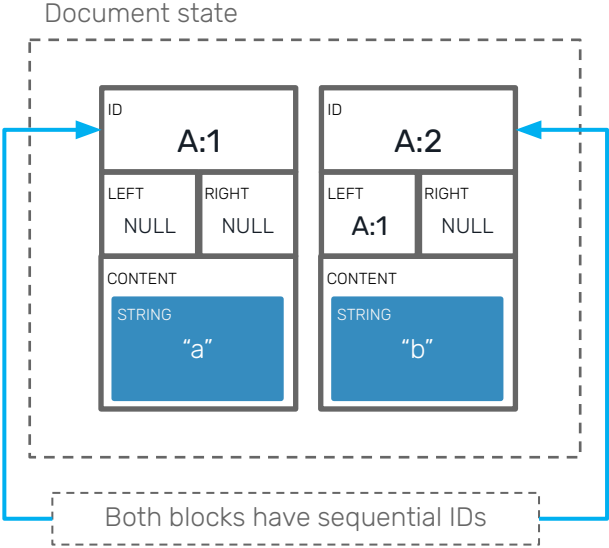
BLOCK MERGING

Document state



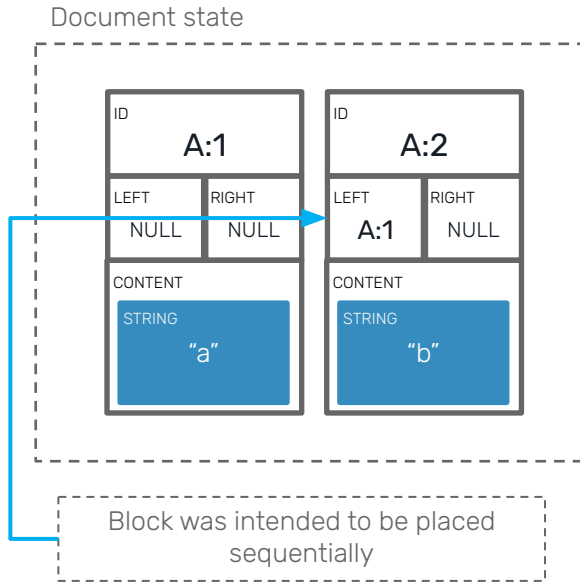
OPTIMIZATIONS

BLOCK MERGING



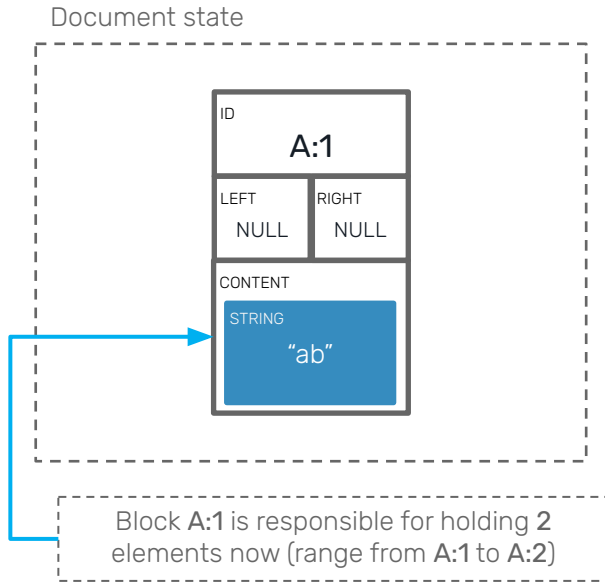
OPTIMIZATIONS

BLOCK MERGING



OPTIMIZATIONS

BLOCK MERGING

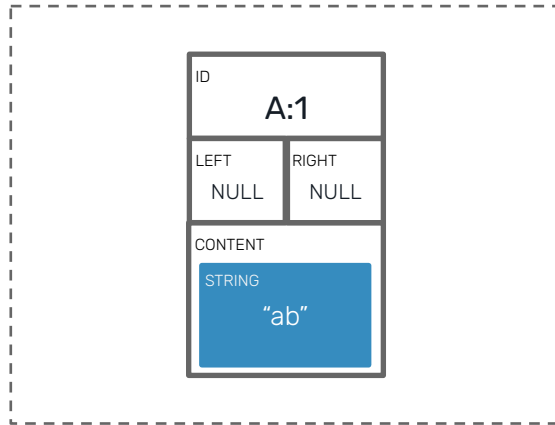


OPTIMIZATIONS

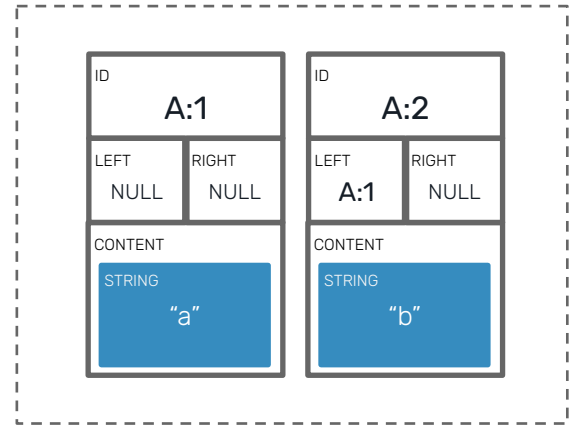
BLOCK MERGING

These two representations are logically equivalent

Document state



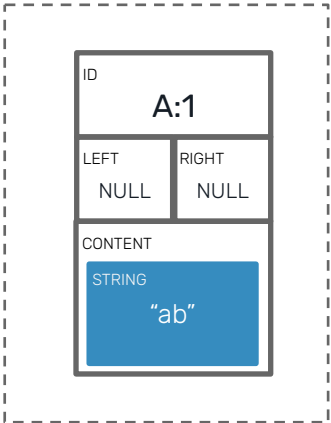
Document state



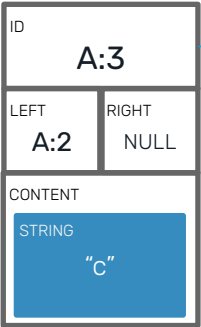
OPTIMIZATIONS

BLOCK MERGING

Document state



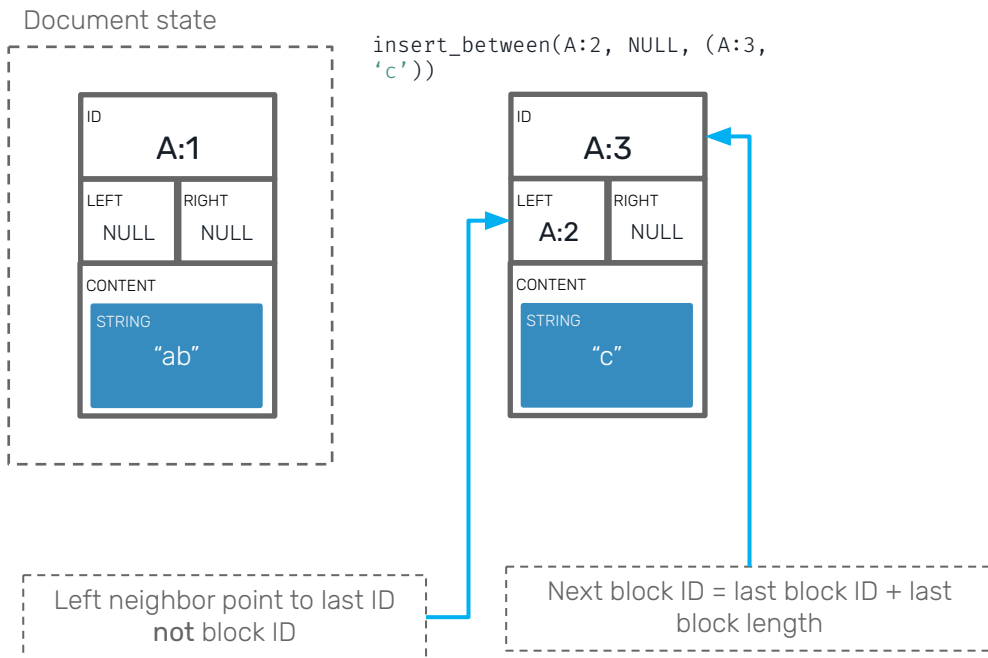
```
insert_between(A:2, NULL, (A:3, 'c'))
```



Next block ID = last block ID + last block length

OPTIMIZATIONS

BLOCK MERGING

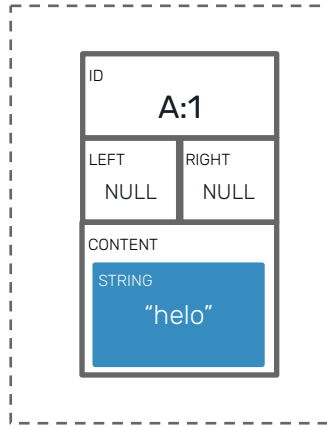


WHAT IF WE WANT TO INSERT ELEMENT WITHIN BLOCK?

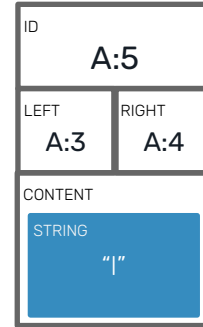
OPTIMIZATIONS

BLOCK SPLITTING

Document state

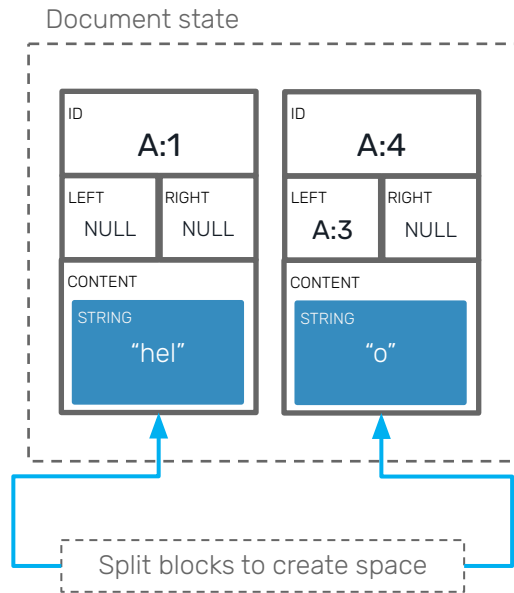


`insert_between(A:3, A:4, (A:5, 'l'))`

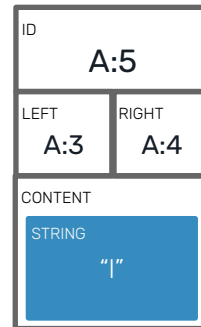


OPTIMIZATIONS

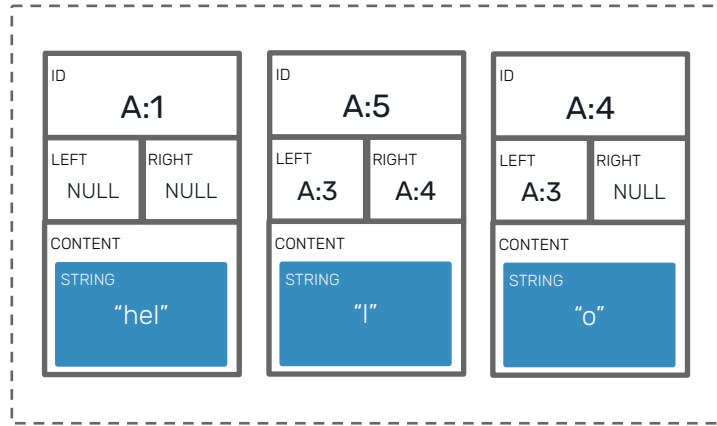
BLOCK SPLITTING



`insert_between(A:3, A:4, (A:5, 'l'))`



Document state



OPTIMIZATIONS

BLOCK SPLITTING

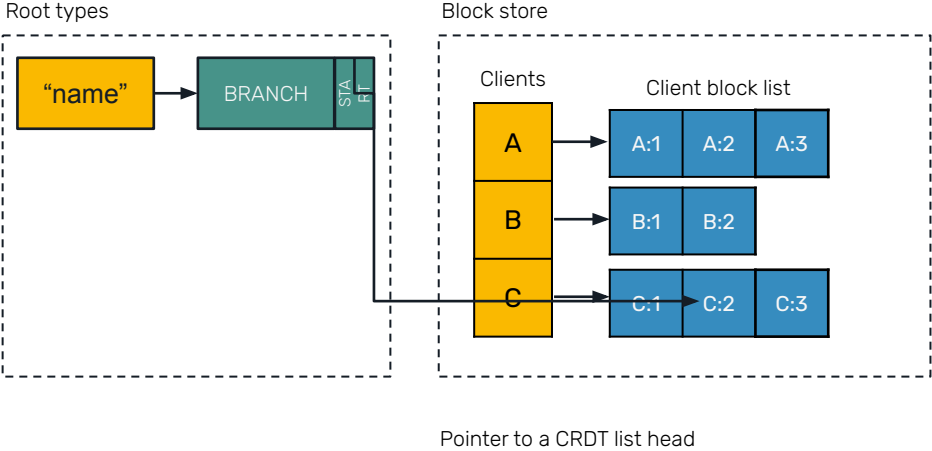
OPTIMIZATIONS

3. IMPROVE BLOCK INSERTION TIME

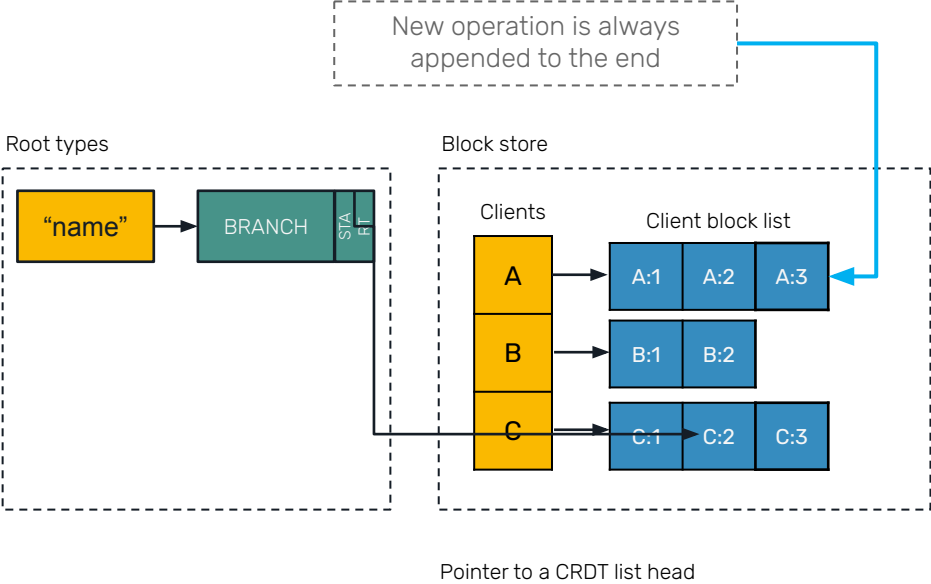


PREFER SEQUENTIAL ACCESS WHENEVER POSSIBLE

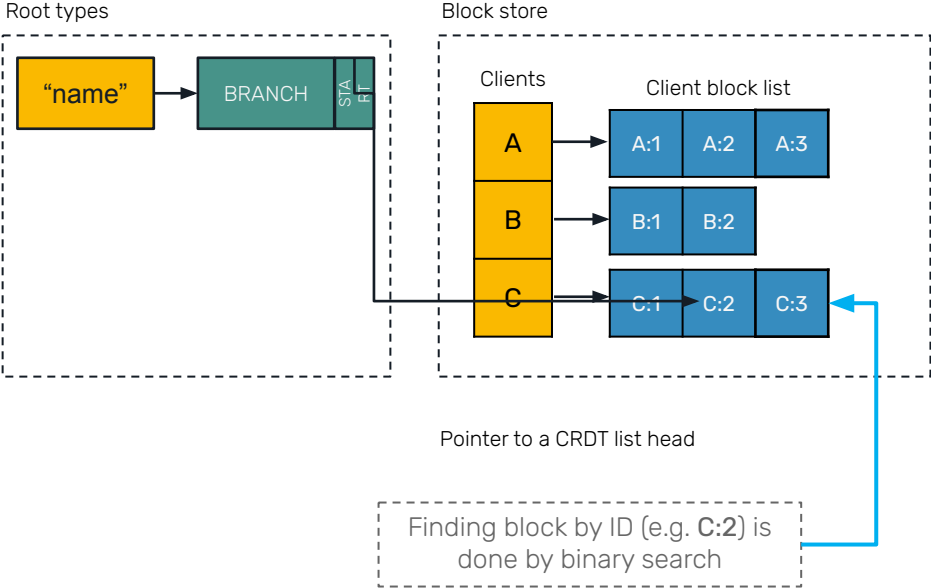
DOCUMENT BLOCK STRUCTURE UNDER THE HOOD



DOCUMENT BLOCK STRUCTURE UNDER THE HOOD



DOCUMENT BLOCK STRUCTURE UNDER THE HOOD



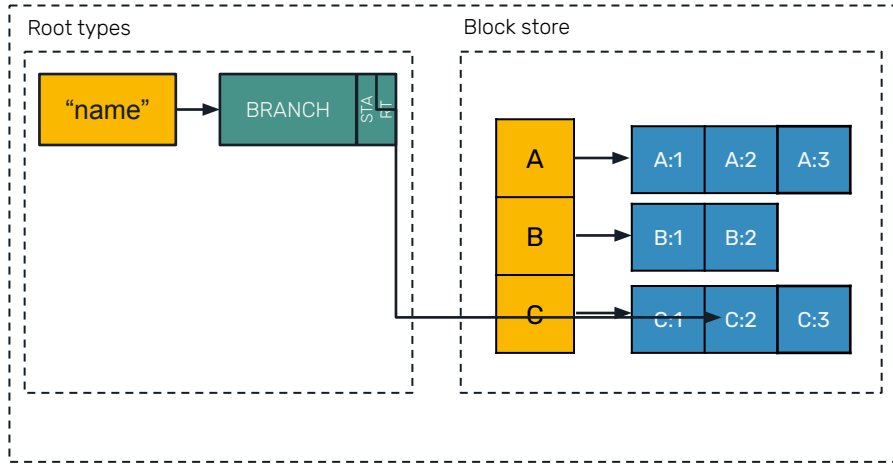


IT'S ALSO BETTER FOR SYNCHRONIZING DATA

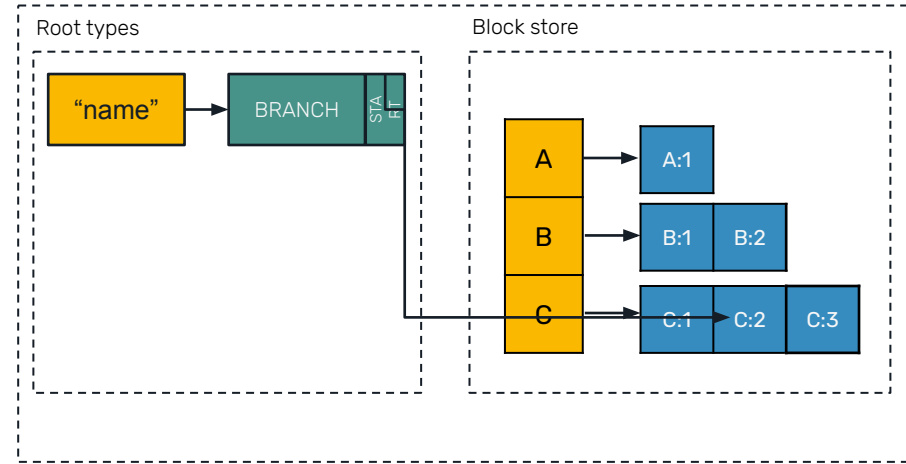
DELTA REPLICATION



Alice



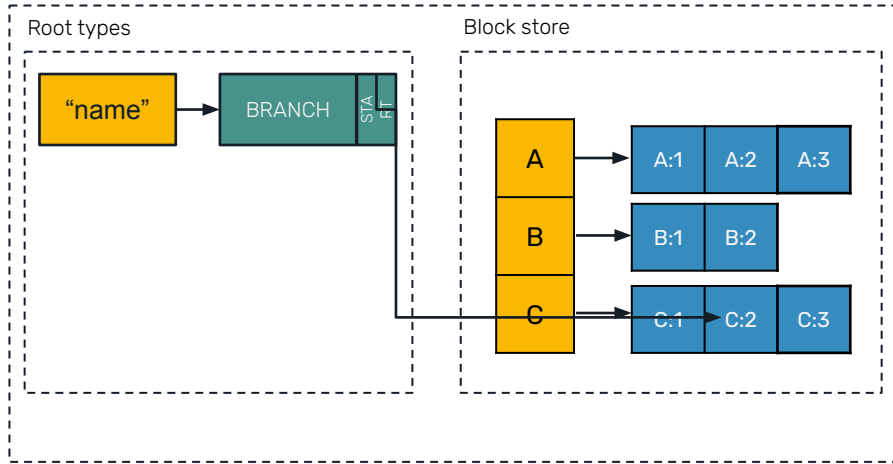
Bob



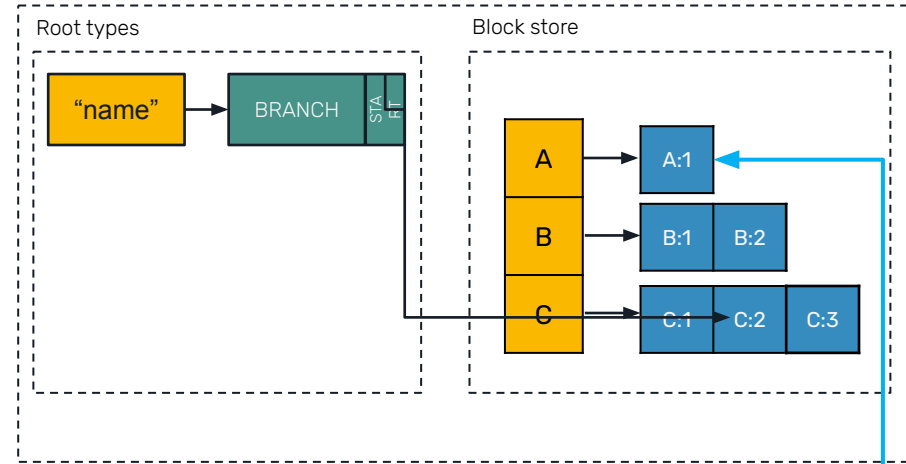
DELTA REPLICATION



Alice



Bob

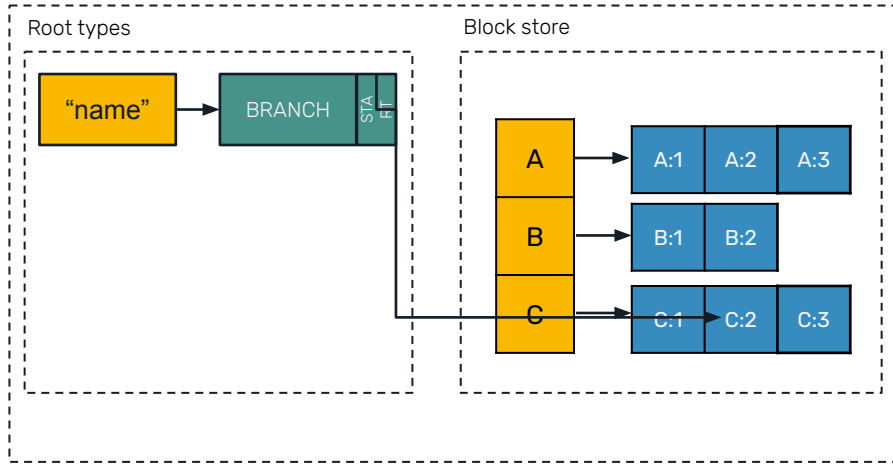


Bob is missing some of the updates from Alice

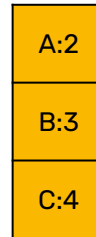
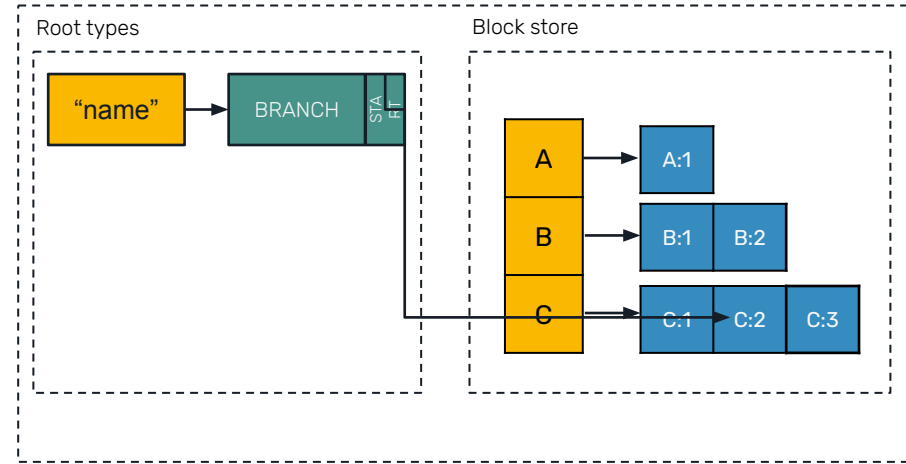
DELTA REPLICATION



Alice



Bob

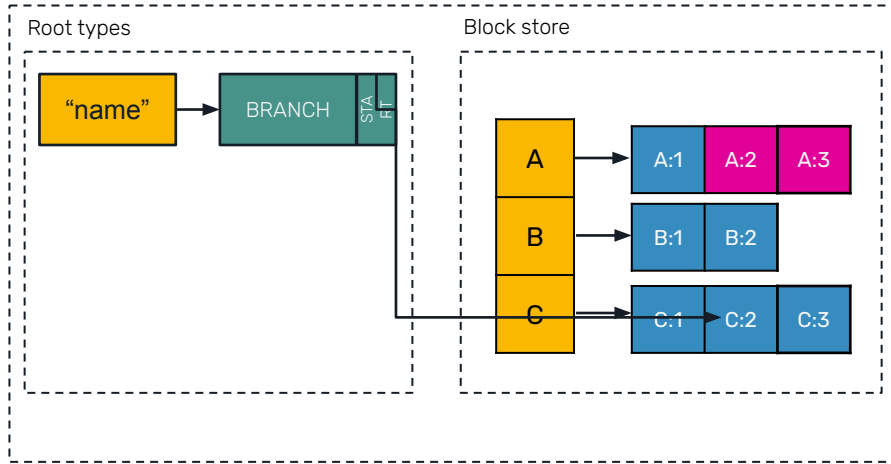


Bob creates a vector clock of his most recent updates

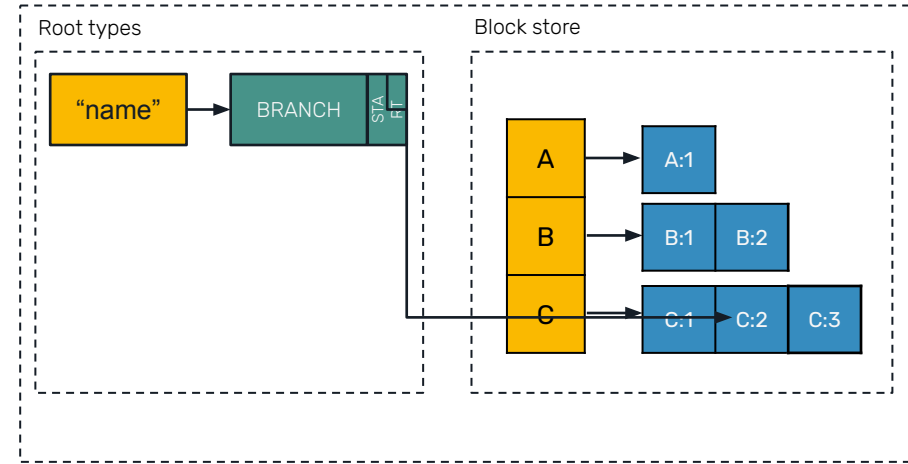
DELTA REPLICATION



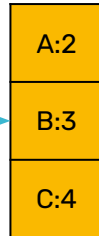
Alice



Bob



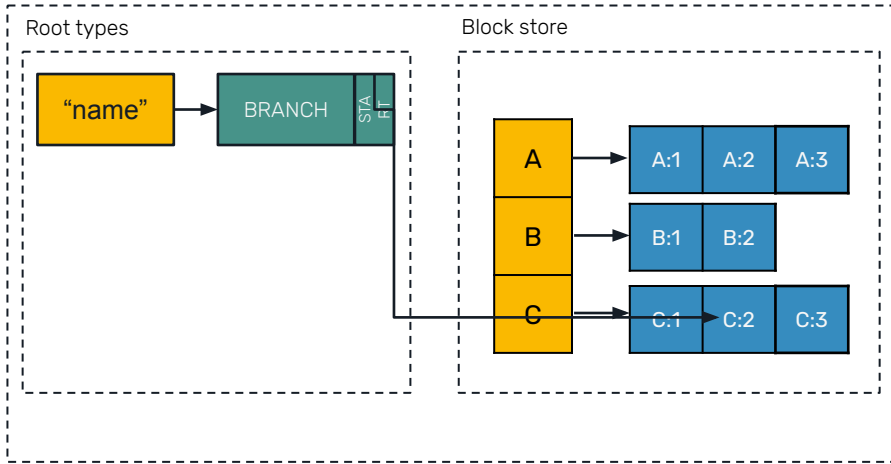
Alice compares Bob's vector clock against her own known state



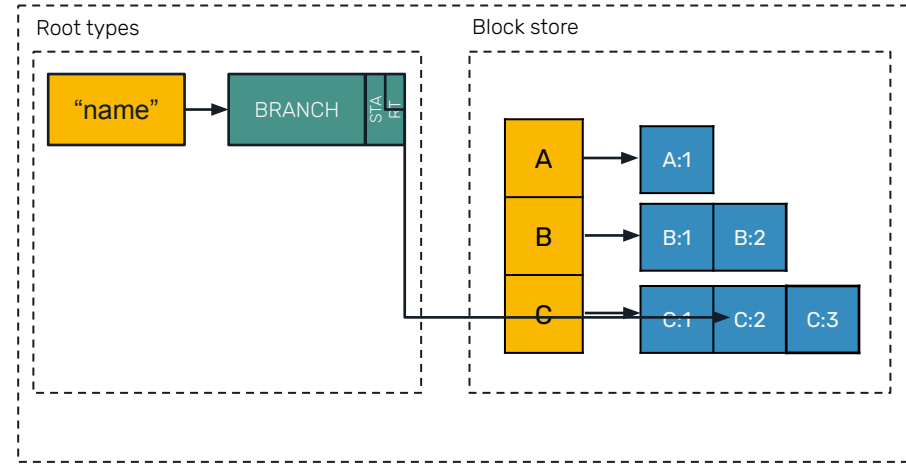
DELTA REPLICATION



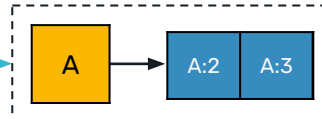
Alice



Bob



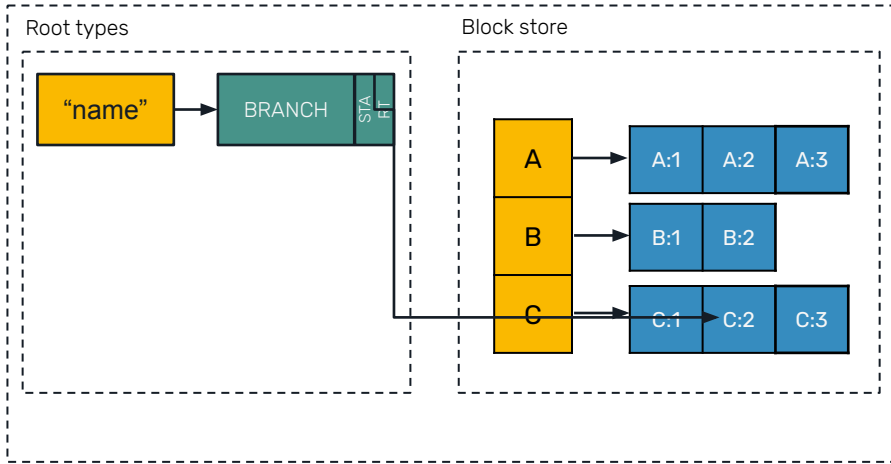
Alice produces a delta with blocks that Bob is missing



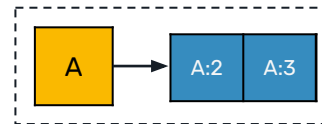
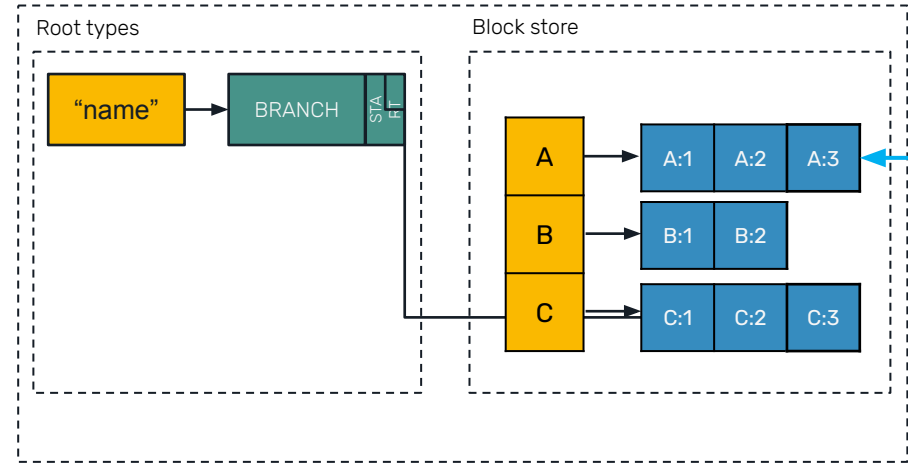
DELTA REPLICATION



Alice



Bob



Bob applies incoming updates on his side

OPTIMIZATIONS

4. INSERTING ELEMENTS CONSECUTIVELY

```
let doc = Doc::new();  
let txt = doc.transact().get_text("text");  
txt.insert(&mut doc.transact(), 0, "hello");  
txt.insert(&mut doc.transact(), 5, "world");
```

OPTIMIZATIONS

CACHING LATEST POSITION



**MAPPING BETWEEN USER INDEX AND BLOCK ID
INTRODUCES COST**

```
let doc = Doc::new();
let txt = doc.transact().get_text("text");
let mut cursor = txt.seek(0);
cursor.insert(&mut doc.transact(), "hello");
cursor.insert(&mut doc.transact(), "world");
```

OPTIMIZATIONS

CURSORS

SUMMARY

CRDT PROJECTS

- Yjs/Yrs: <https://crates.io/crates/yrs>
- Automerge: <https://automerge.org/>
- Ditto.live: <https://www.ditto.live/>
- RiakDB: <https://riak.com/>
- Amazon DynamoDB: <https://aws.amazon.com/dynamodb/>
- AntidoteDB: <https://www.antidotedb.eu/>

REFERENCES

- CRDTs deep dive: <https://bartoszsypytkowski.com/tag/crdt/>
- List of aggregated CRDT articles: <https://crdt.tech>
- Making CRDTs faster: <https://josephg.com/blog/crdts-go-brrr/>

THANK YOU