Schattenspeicher

Daniel Kutasi
Garching, 24. Oktober 2017

Seminar:
Implementierungstechniken für Hauptspeicherdatenbanksysteme

1 Motivation
2 Explizit verwalteter Schattenspeicher
3 Betriebssystem eigener Schattenspeicher

1 Motivation

Bestellung,
Banküberweisung, …

- wenige Daten
- sehr schnell

1 Motivation

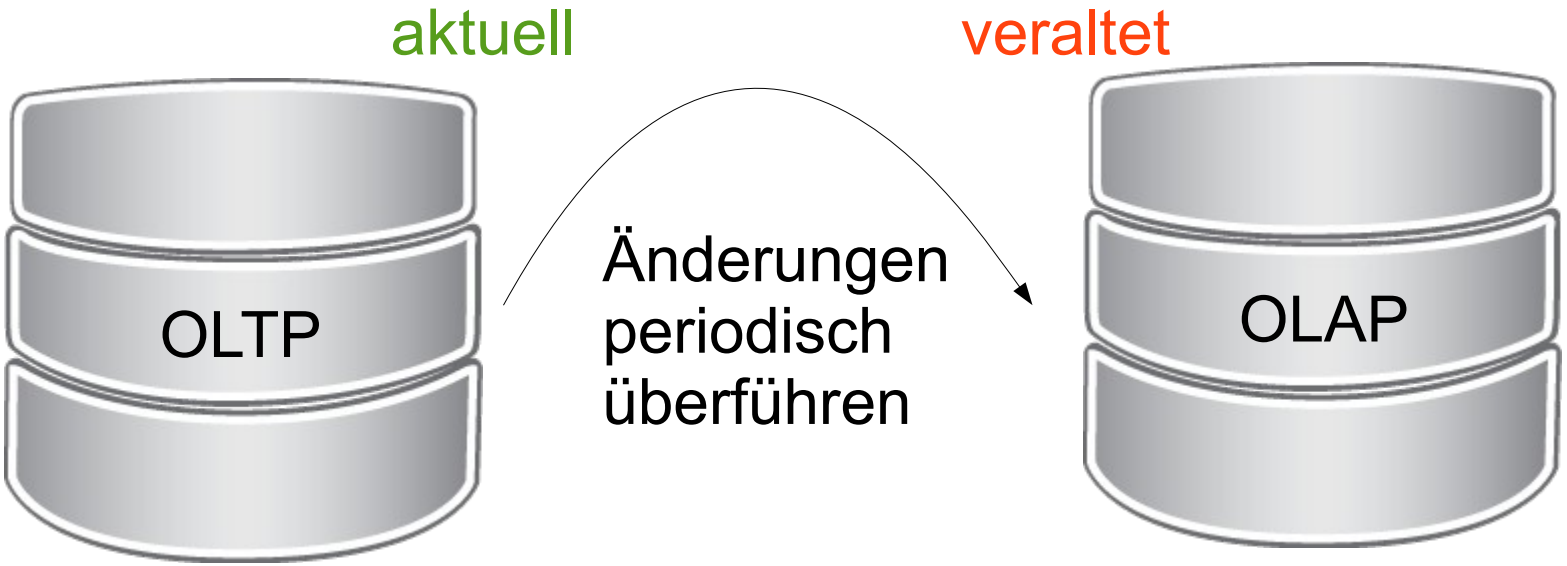
Bestellung,
Banküberweisung, …

- wenige Daten
- sehr schnell

Statistiken zu
Verkäufen ...

- viele Daten
- sehr langsam

1 Motivation



Bestellung,
Banküberweisung, …

- wenige Daten
- sehr schnell

Statistiken zu
Verkäufen ...

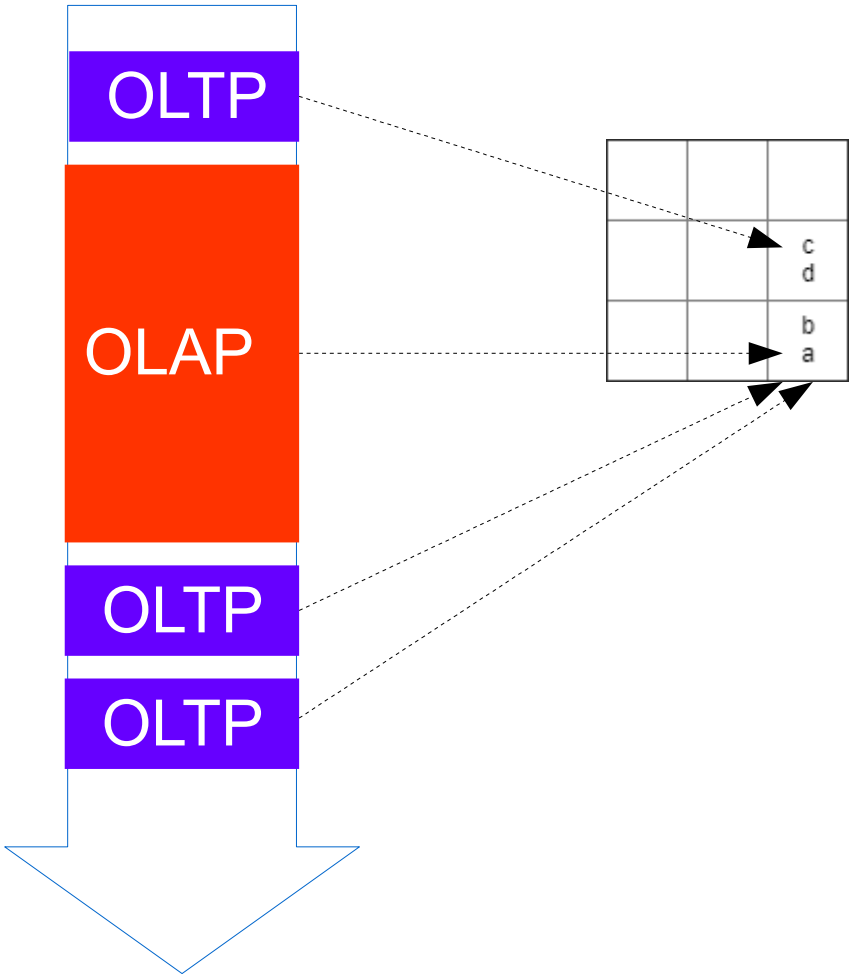
- viele Daten
- sehr langsam

1 Motivation

OLTP + OLAP

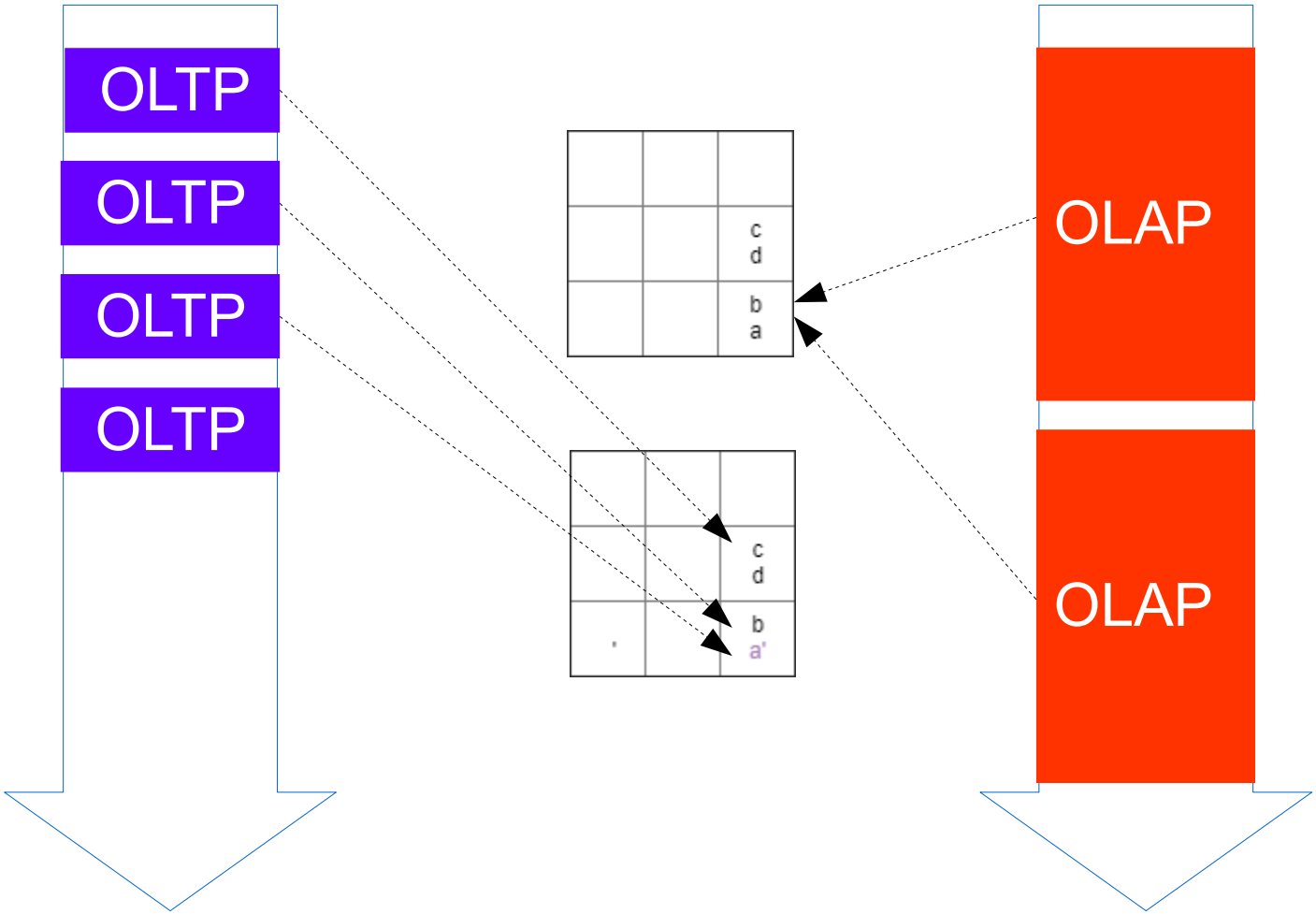
HTAP

1 Motivation



1 Motivation

HTAP mit Schattenspeicher



1 Motivation
2 Explizit verwalteter Schattenspeicher
3 Betriebssystem eigener Schattenspeicher

2 Explizit verwalteter Schattenspeicher

2.1 Tupel-Schattenspeicher

2.2 Originäres Schattenspeicherkonzept

2.3 Twin-Objekt-Verfahren

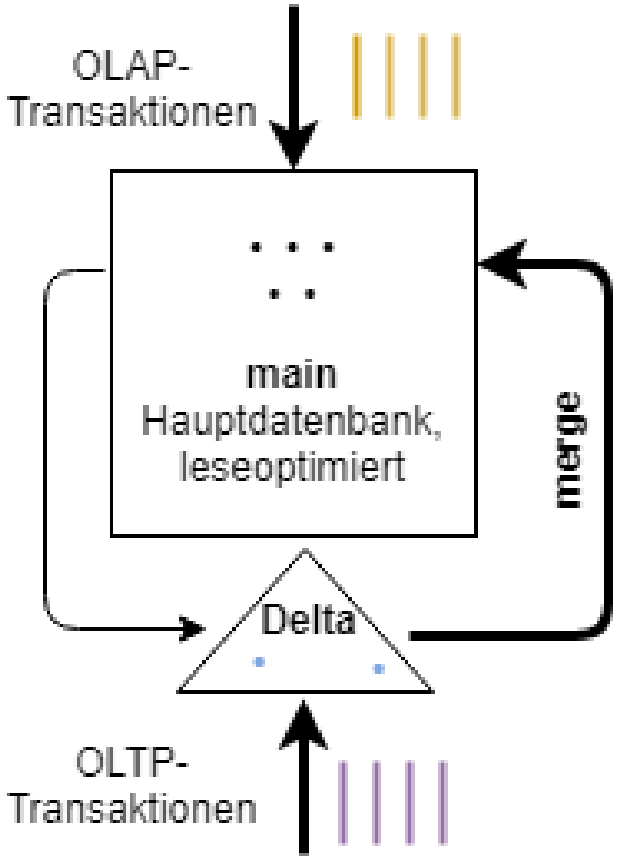
2 Explizit verwalteter Schattenspeicher

2.1 Tupel-Schattenspeicher

2.2 Originäres Schattenspeicherkonzept

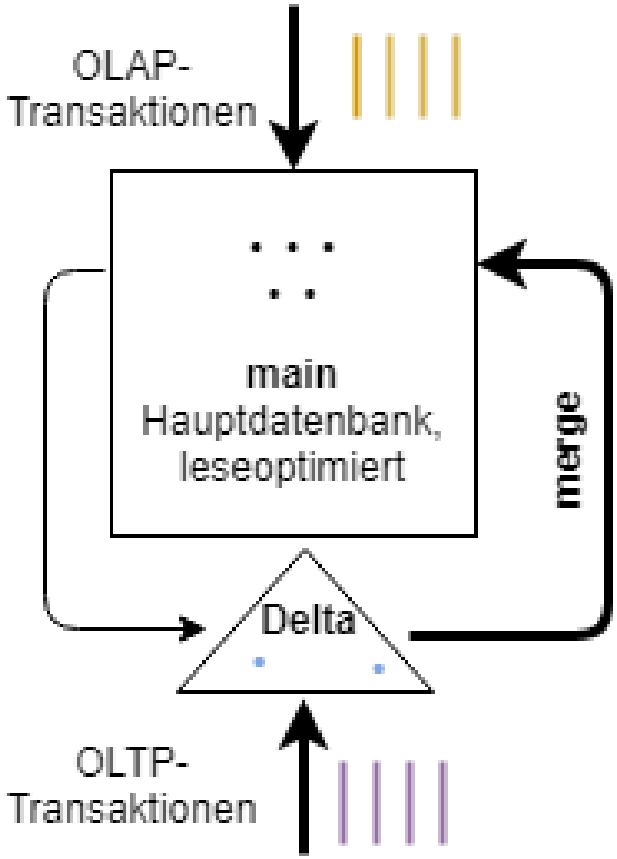
2.3 Twin-Objekt-Verfahren

2.1 Tupel-Schattenspeicher Konzept

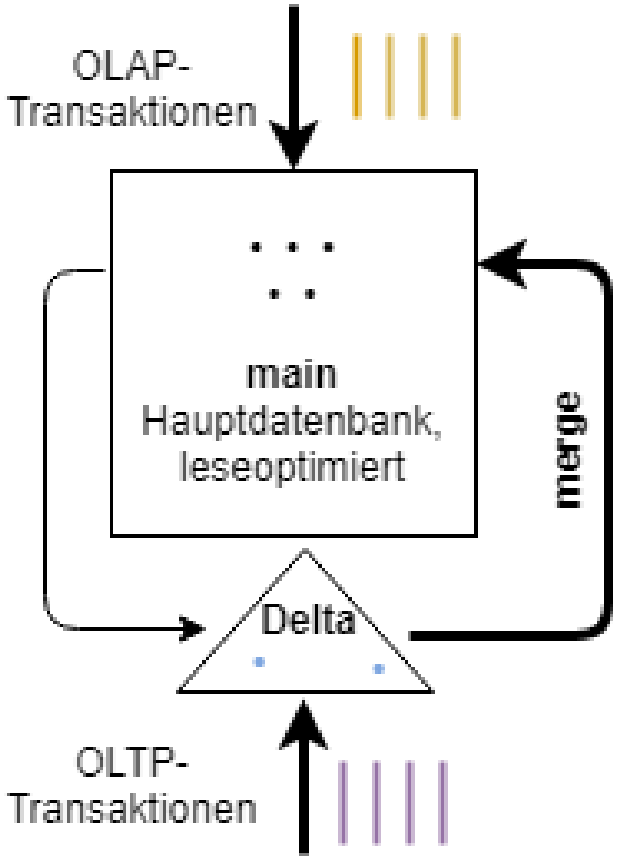


2.1 Tupel-Schattenspeicher Konzept

- typisch: *Column Stores*

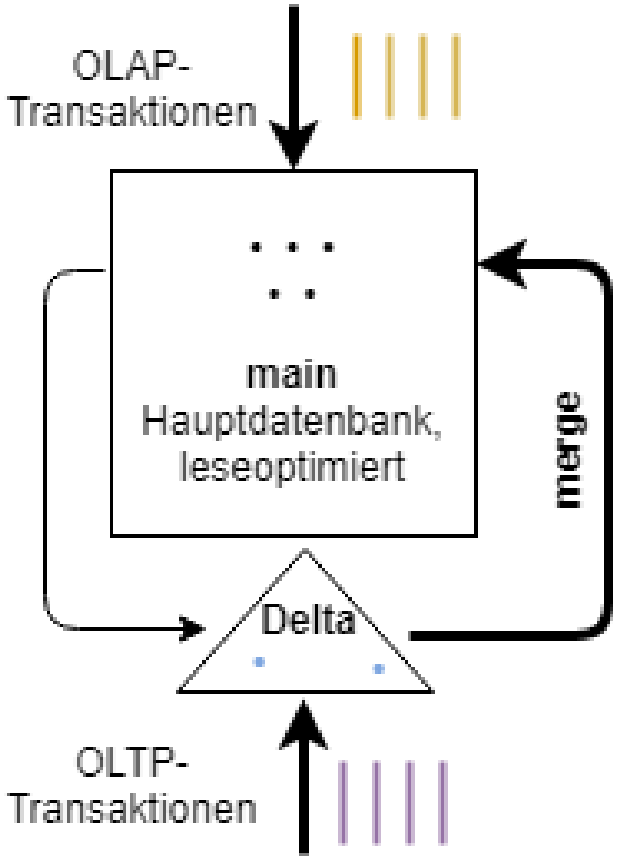


2.1 Tupel-Schattenspeicher Konzept



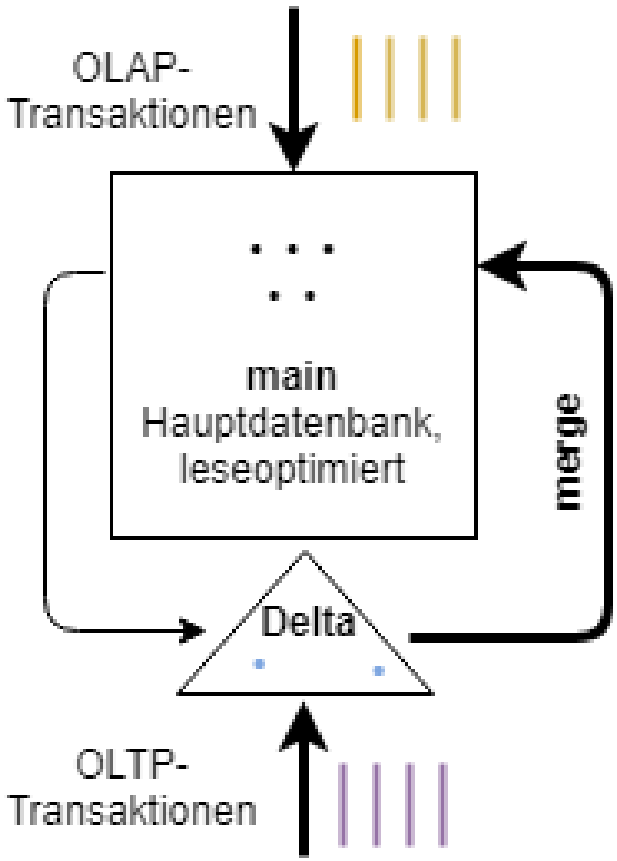
- typisch: *Column Stores*
- *merge* ist Flaschenhals (viel Zeit & Platz)

2.1 Tupel-Schattenspeicher Konzept



- typisch: *Column Stores*
- *merge* ist Flaschenhals (viel Zeit & Platz)
- Scan langsamer, da erst im Delta nachgesehen werden muss

2.1 Tupel-Schattenspeicher Konzept



- typisch: *Column Stores*
- *merge* ist Flaschenhals (viel Zeit & Platz)
- Scan langsamer, da erst im Delta nachgesehen werden muss
- → *Positional Tree* enthält Pos. der Tupel im Delta
  → zwischen zwei nicht invalidierten Tupel volle Geschwindigkeit des Scans

2.1 Tupel-Schattenspeicher Implementierung

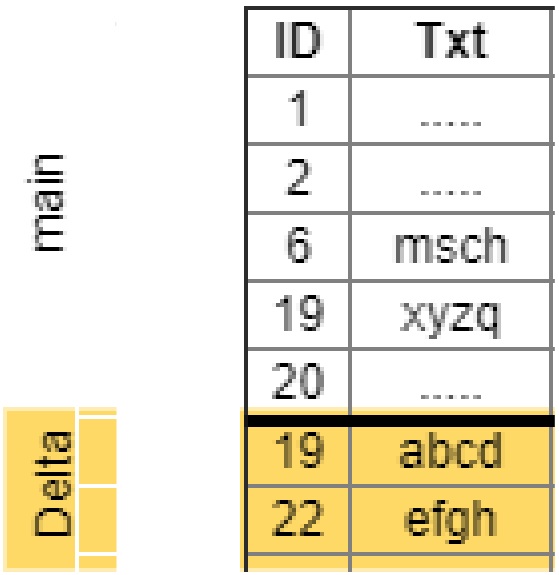
main

ID	Txt
1	.....
2	.....
6	msch
19	xyzq
20	.....

Delta

19	abcd
22	efgh

2.1 Tupel-Schattenspeicher Implementierung



2.1 Tupel-Schattenspeicher Implementierung

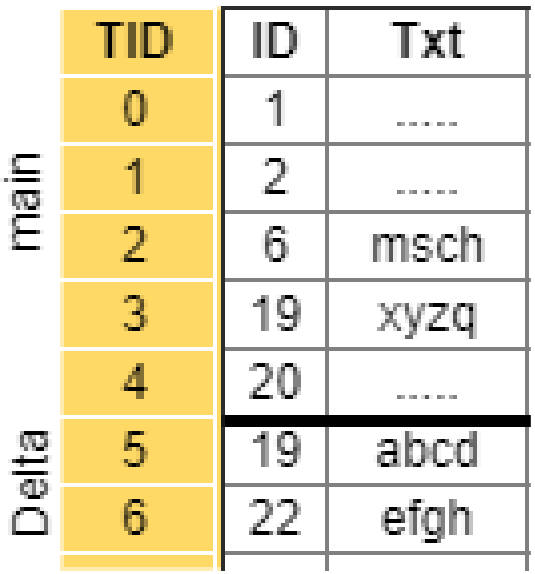
main

ID	Txt
1	.....
2	.....
6	msch
19	xyzq
20	.....

Delta

ID	Txt
19	abcd
22	efgh

2.1 Tupel-Schattenspeicher Implementierung



	TID	ID	Txt
main	0	1	.....
	1	2	.....
	2	6	msch
	3	19	xyzq
	4	20	.....
Delta	5	19	abcd
	6	22	efgh

2.1 Tupel-Schattenspeicher Implementierung

	TID	ID	Txt	ShadowPtr
main	0	1	.....	-
	1	2	.....	-
	2	6	msch	-1
	3	19	xyzq	5
	4	20	.....	-
Delta	5	19	abcd	3
	6	22	efgh	-

2.1 Tupel-Schattenspeicher Implementierung

main

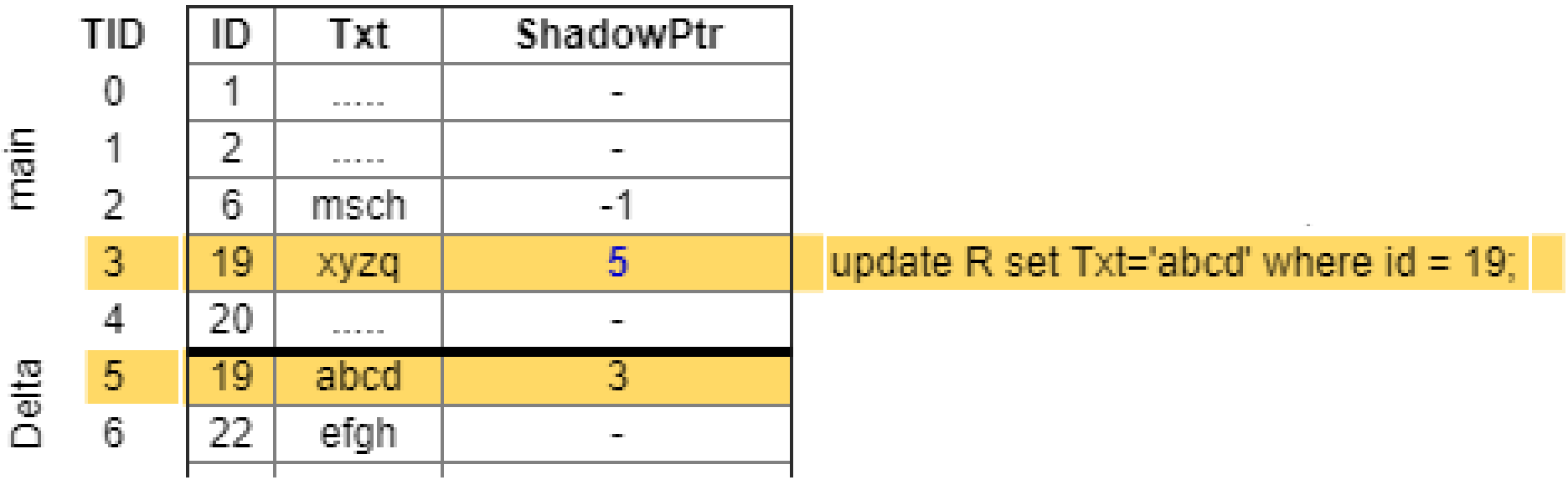
TID	ID	Txt	ShadowPtr
0	1	.....	-
1	2	.....	-
2	6	msch	-1
3	19	xyzq	5
4	20	.....	-

Delta

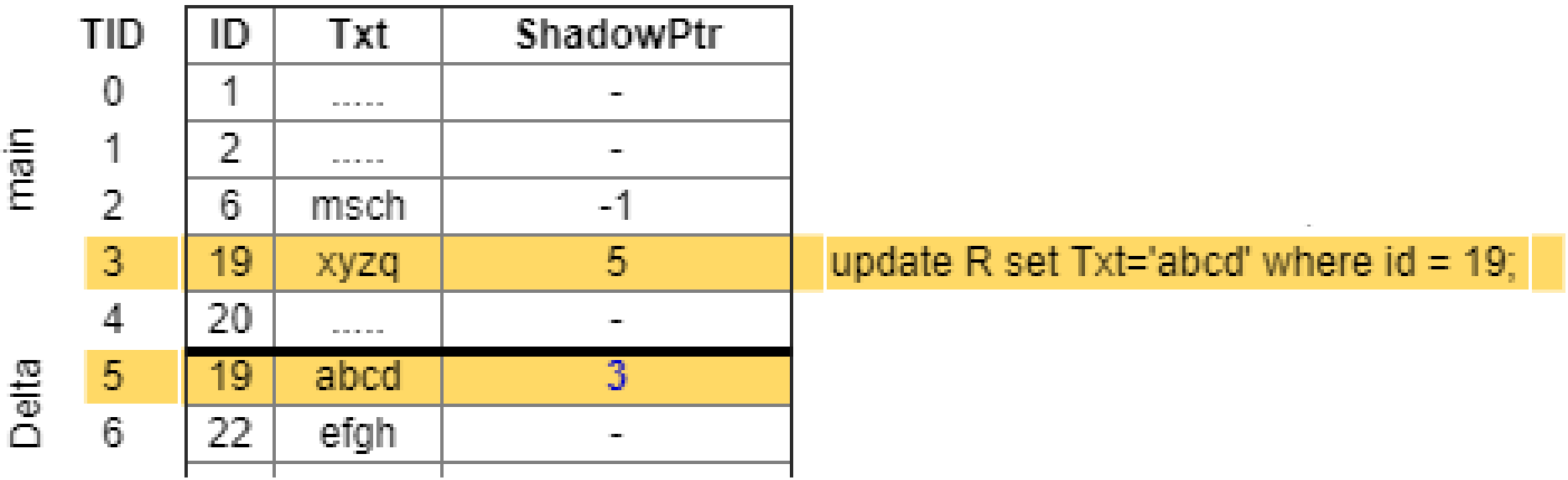
TID	ID	Txt	ShadowPtr
5	19	abcd	3
6	22	efgh	-

update R set Txt='abcd' where id = 19;

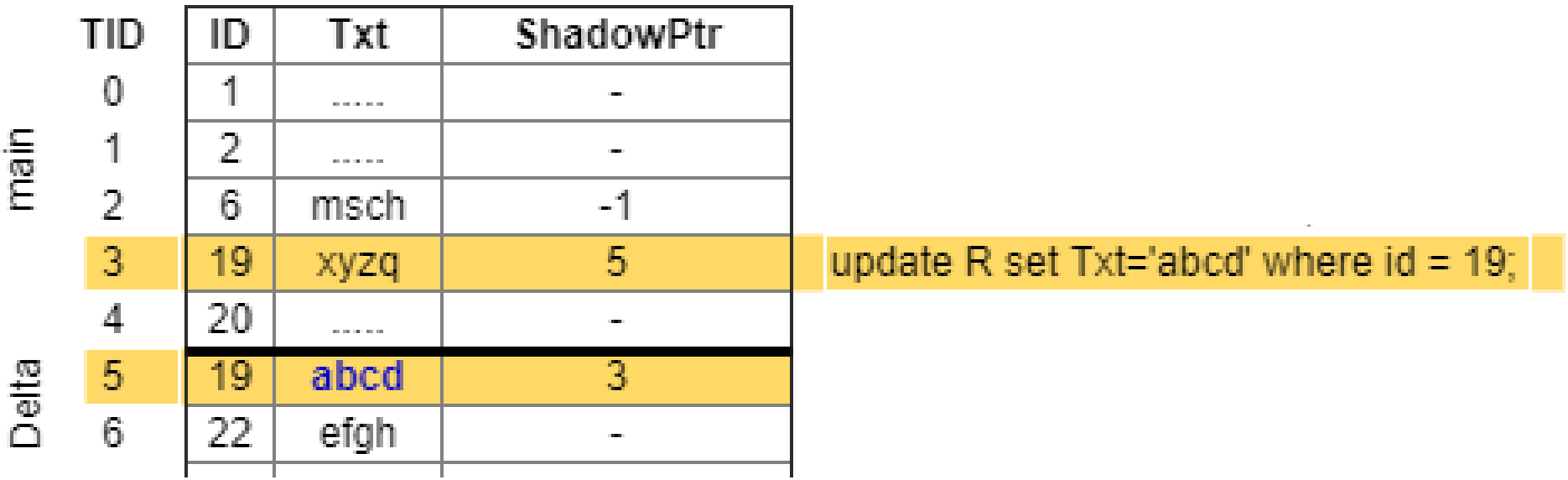
2.1 Tupel-Schattenspeicher Implementierung



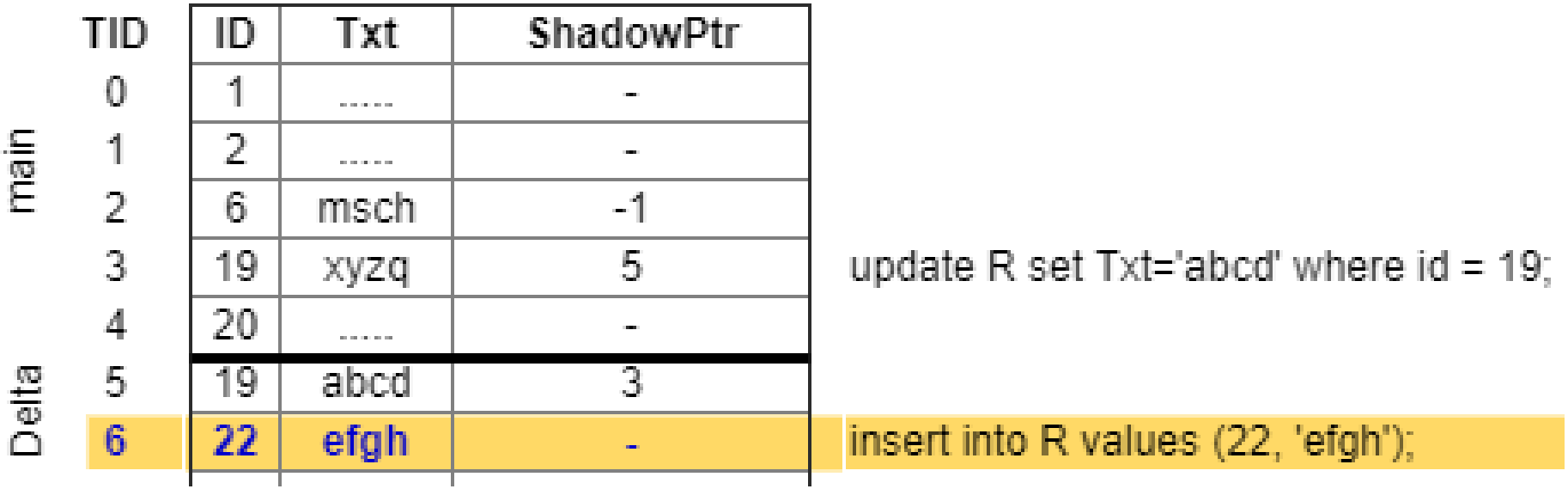
2.1 Tupel-Schattenspeicher Implementierung



2.1 Tupel-Schattenspeicher Implementierung



2.1 Tupel-Schattenspeicher Implementierung



2.1 Tupel-Schattenspeicher Implementierung

	TID	ID	Txt	ShadowPtr	
main	0	1	.....	-	
	1	2	.....	-	
	2	6	msch	-1	delete from R where id = 6;
	3	19	xyzq	5	update R set Txt='abcd' where id = 19;
	4	20	.....	-	
Delta	5	19	abcd	3	
	6	22	efgh	-	insert into R values (22, 'efgh');

2 Explizit verwalteter Schattenspeicher

2.1 Tupel-Schattenspeicher

2.2 Originäres Schattenspeicherkonzept

2.3 Twin-Objekt-Verfahren

2 Explizit verwalteter Schattenspeicher

2.1 Tupel-Schattenspeicher

2.2 Originäres Schattenspeicherkonzept

2.3 Twin-Objekt-Verfahren

2.2 Originäres Schattenspeicherkonzept

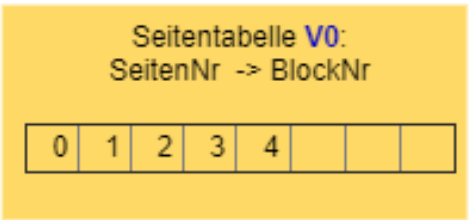
Seitentabelle V0:
SeitenNr -> BlockNr

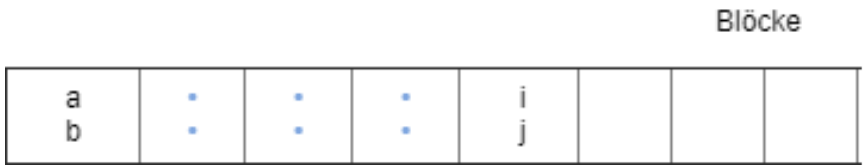
0	1	2	3	4			

Blöcke

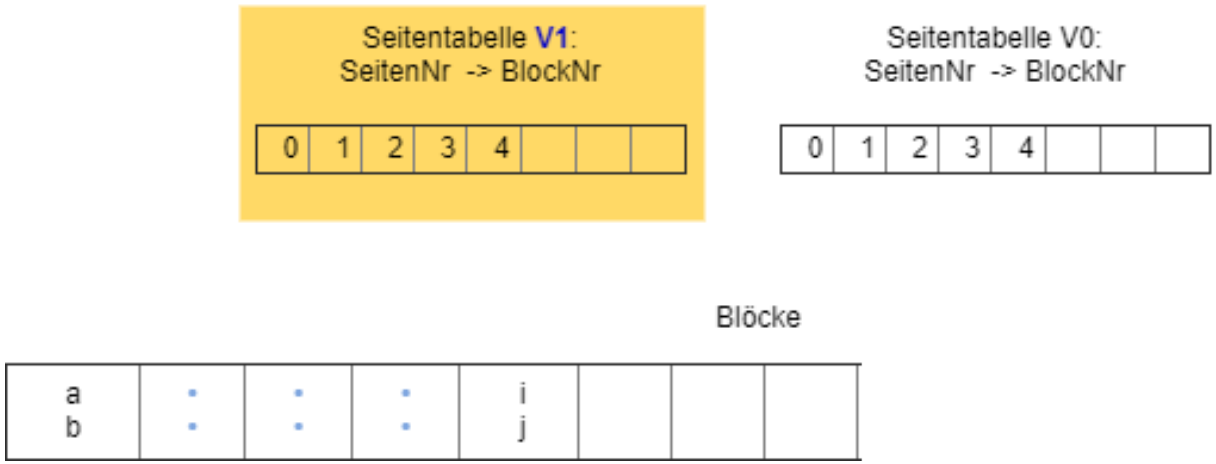
a b	· ·	· ·	· ·	i j			

2.2 Originäres Schattenspeicherkonzept

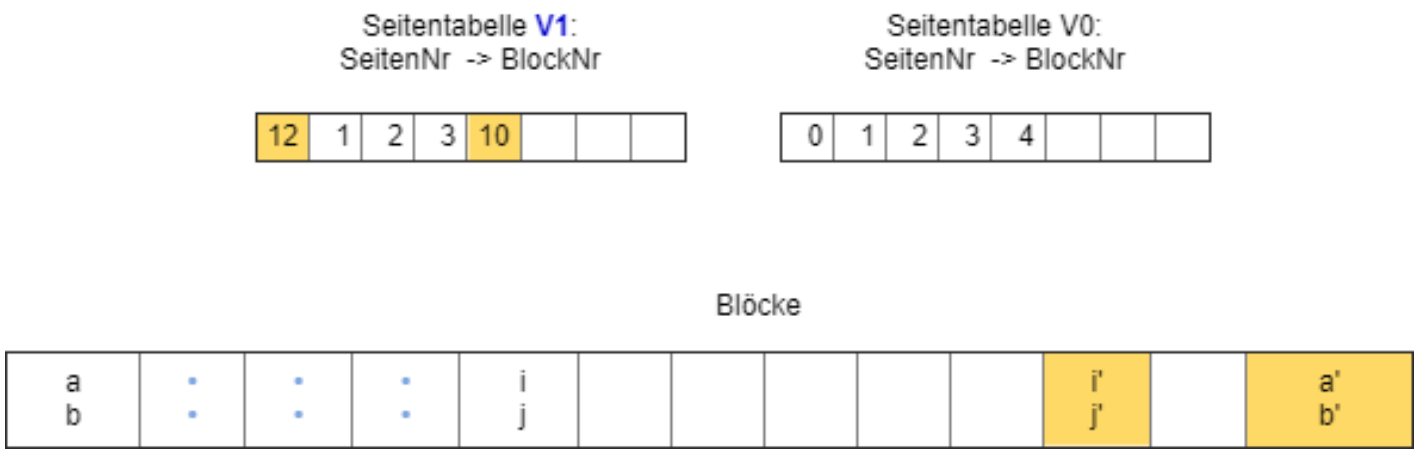




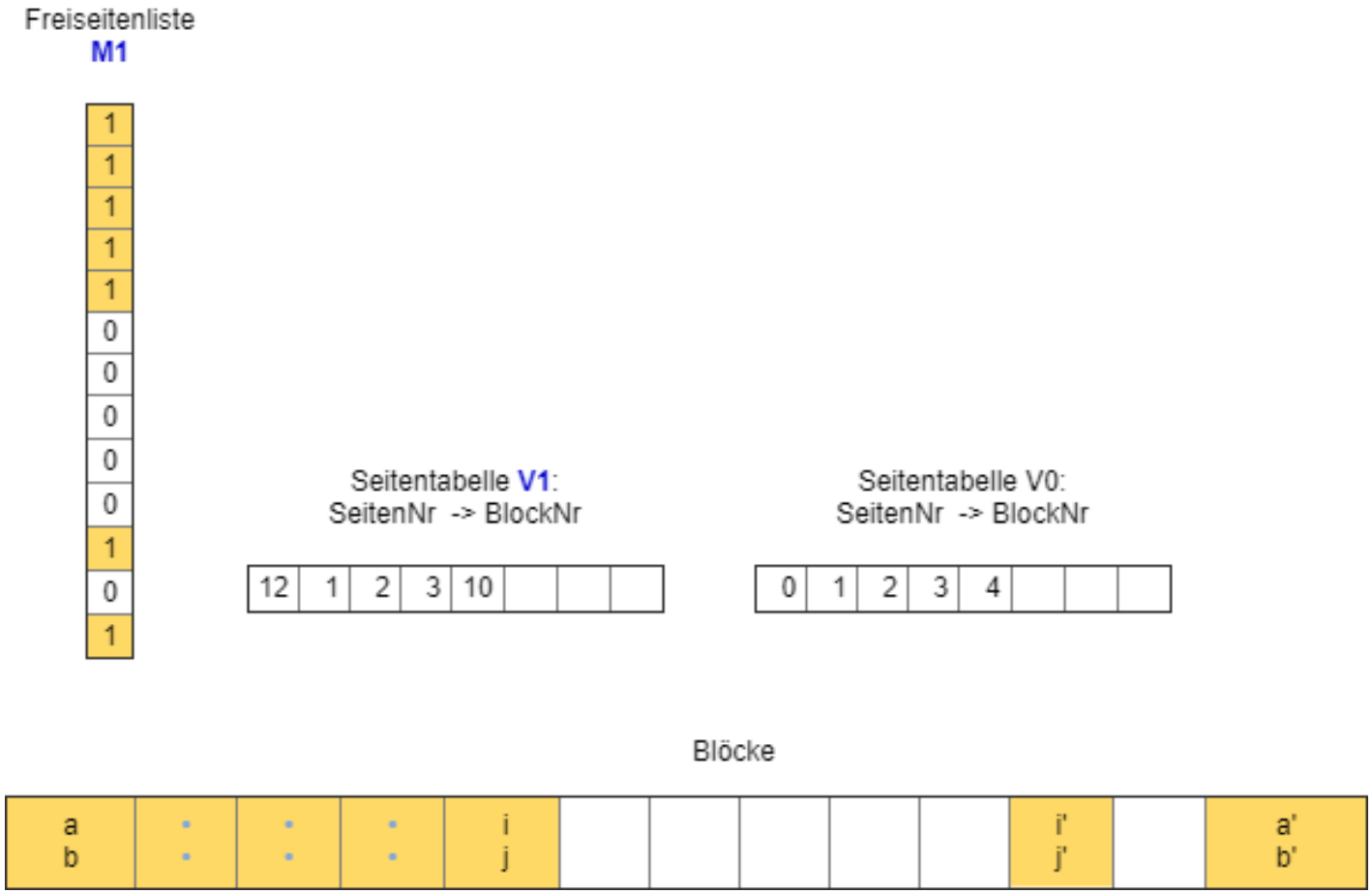
2.2 Originäres Schattenspeicherkonzept



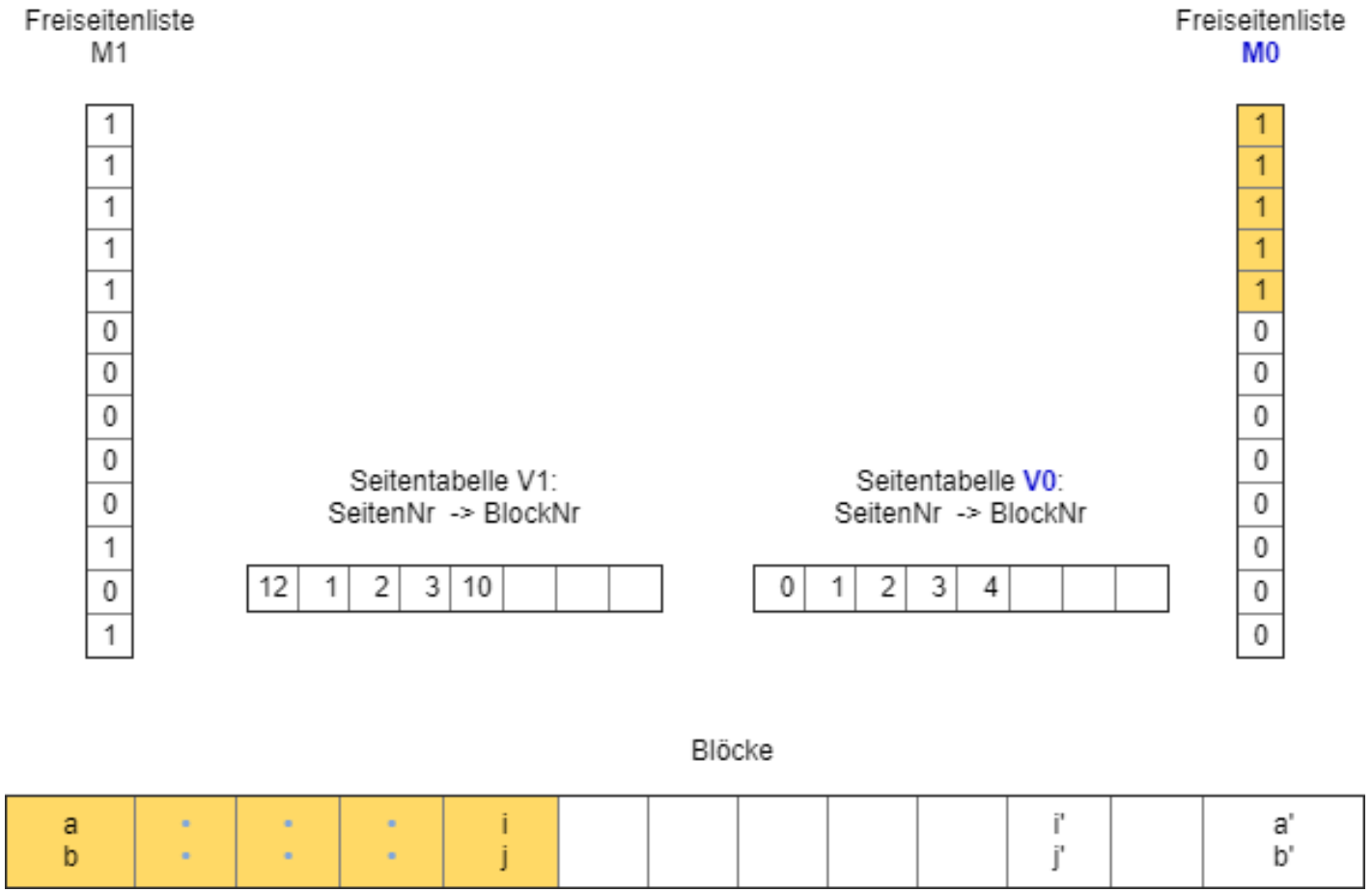
2.2 Originäres Schattenspeicherkonzept



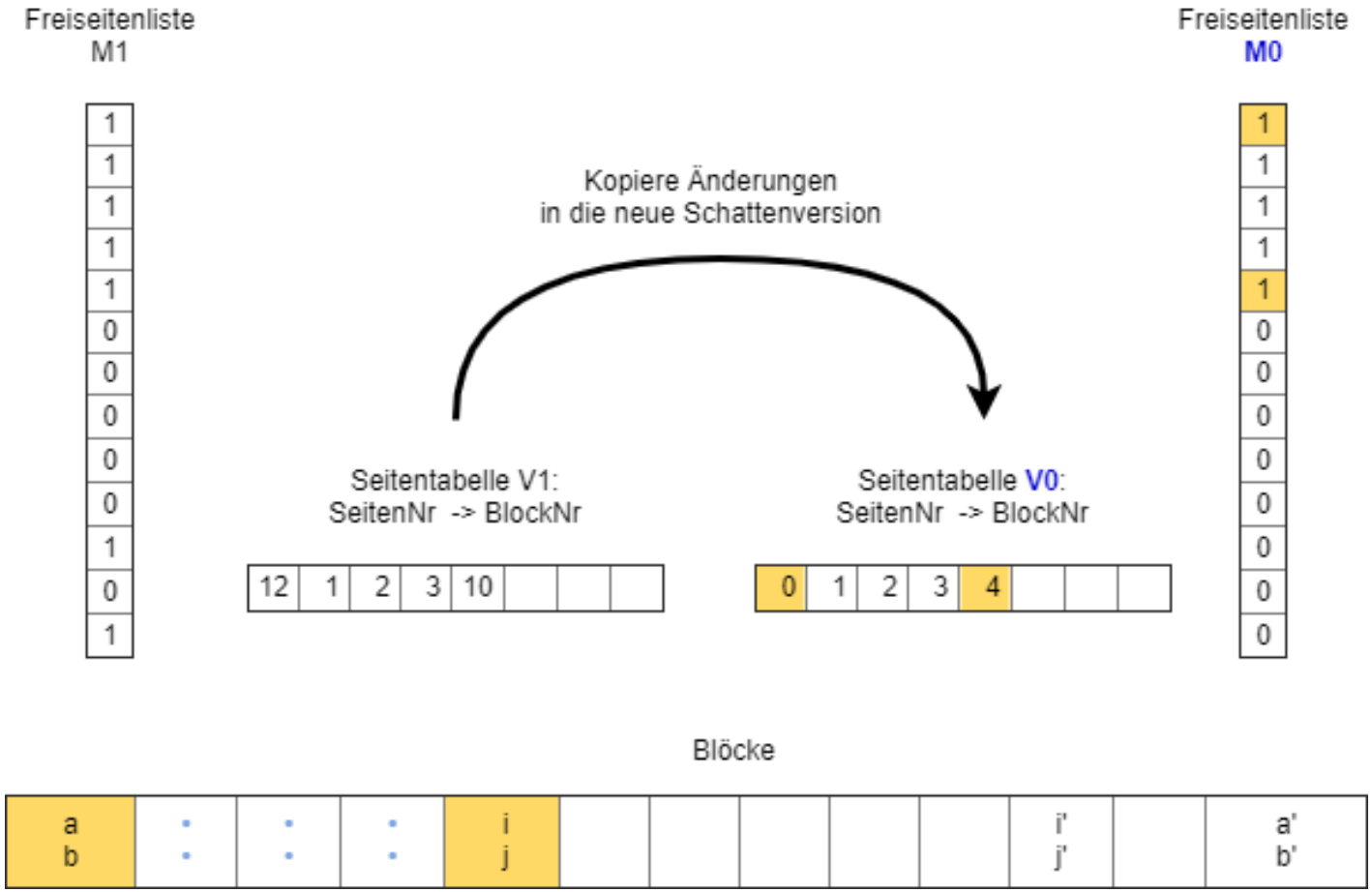
2.2 Originäres Schattenspeicherkonzept



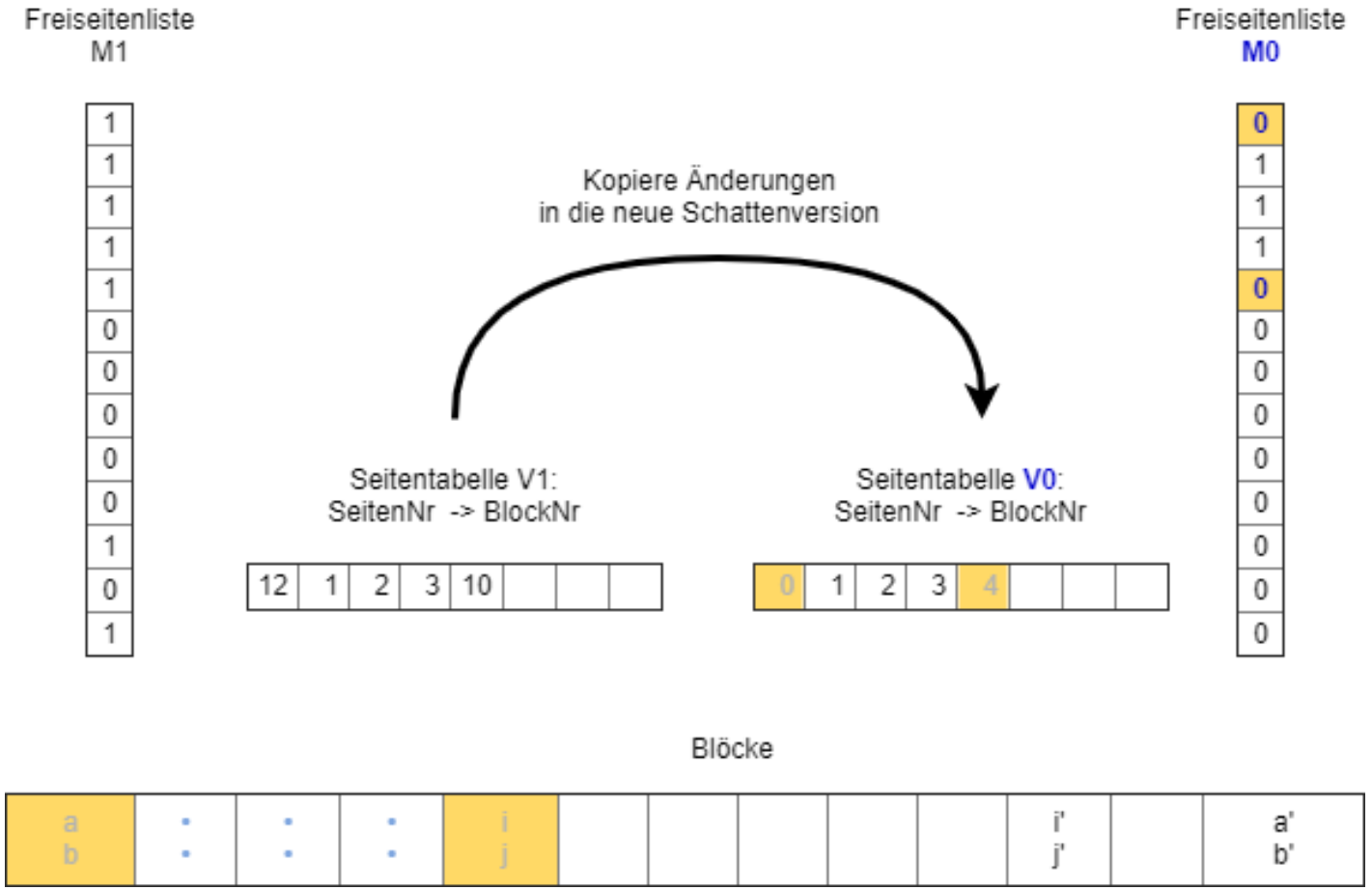
2.2 Originäres Schattenspeicherkonzept



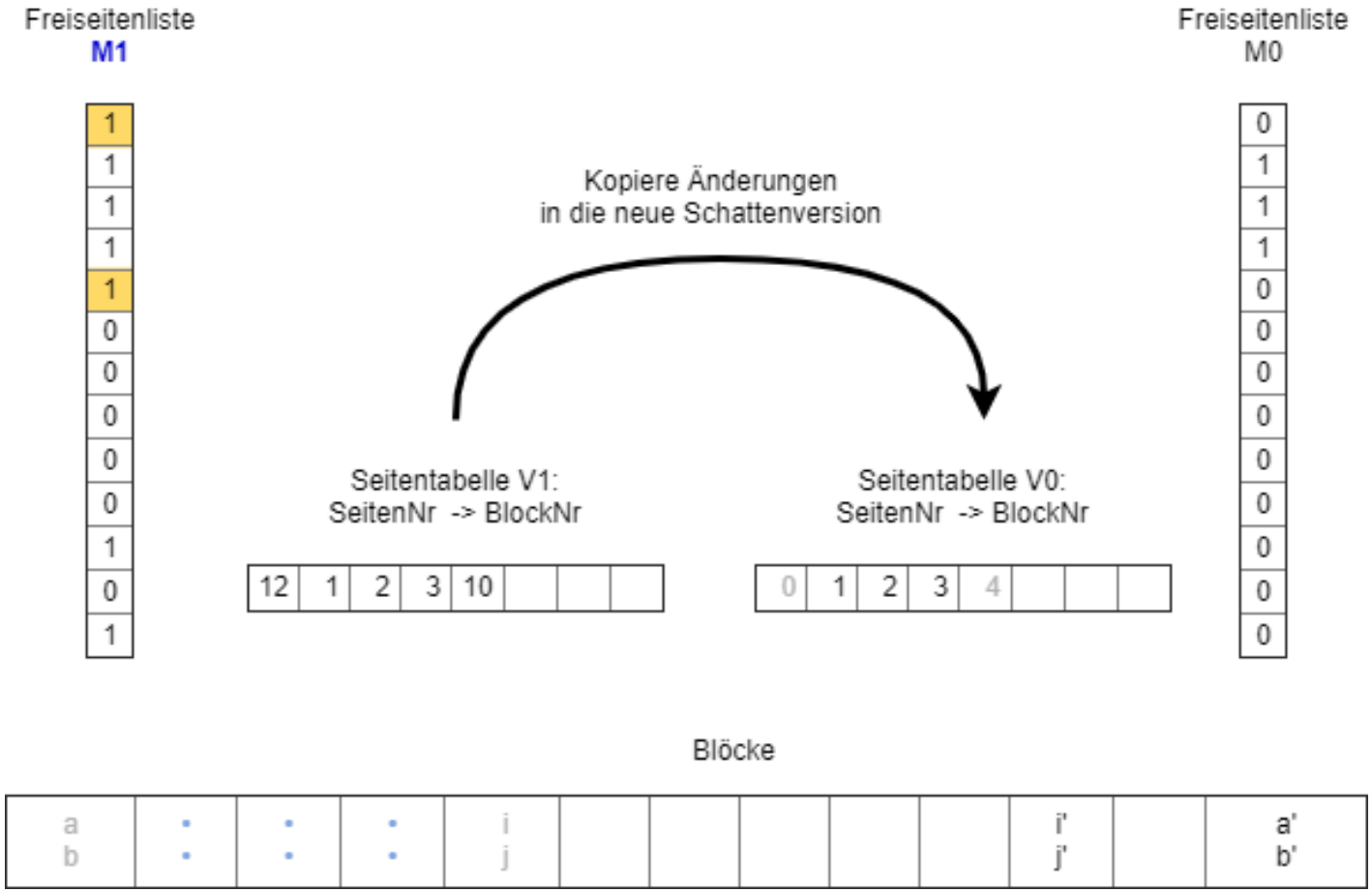
2.2 Originäres Schattenspeicherkonzept



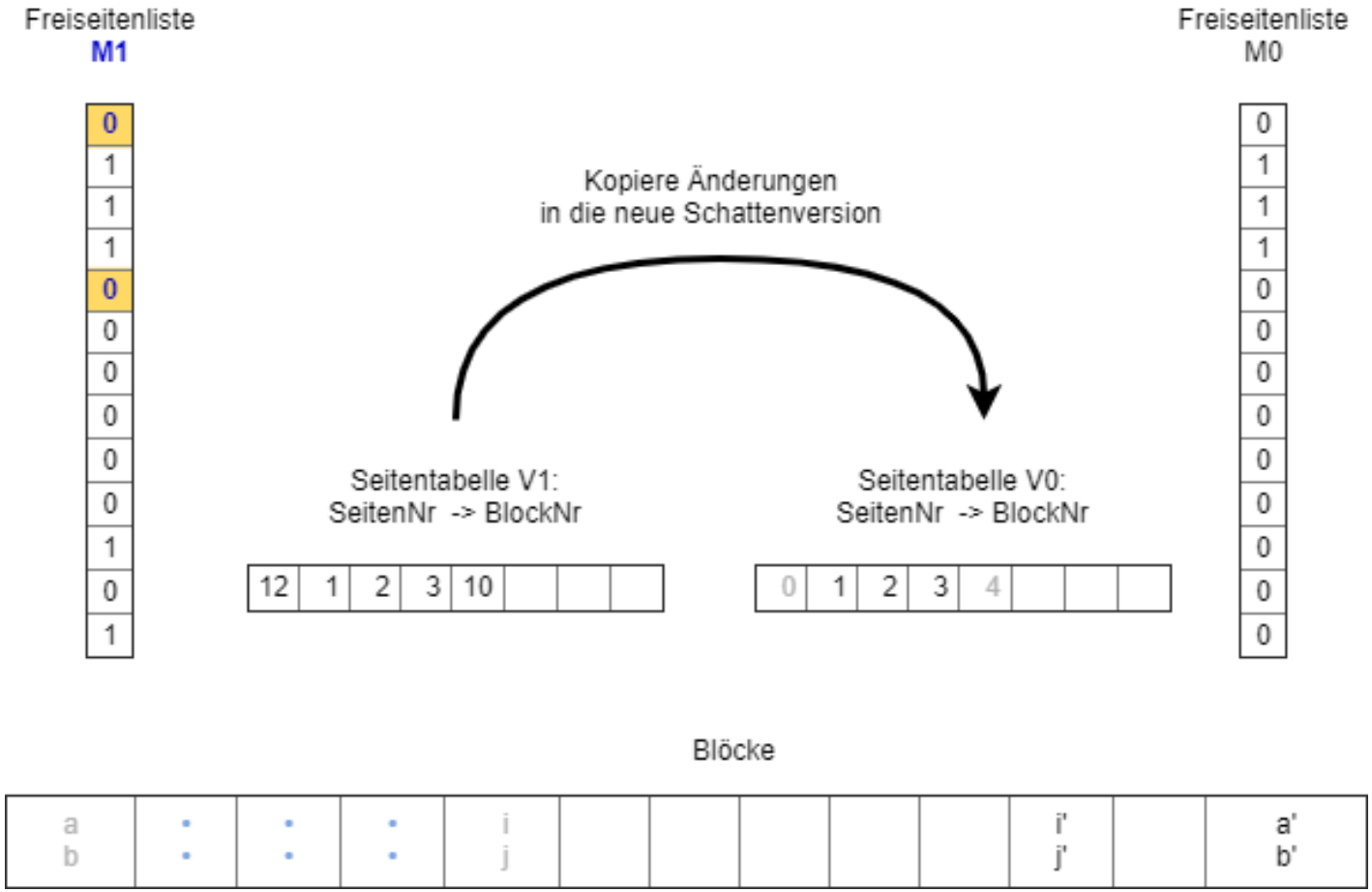
2.2 Originäres Schattenspeicherkonzept



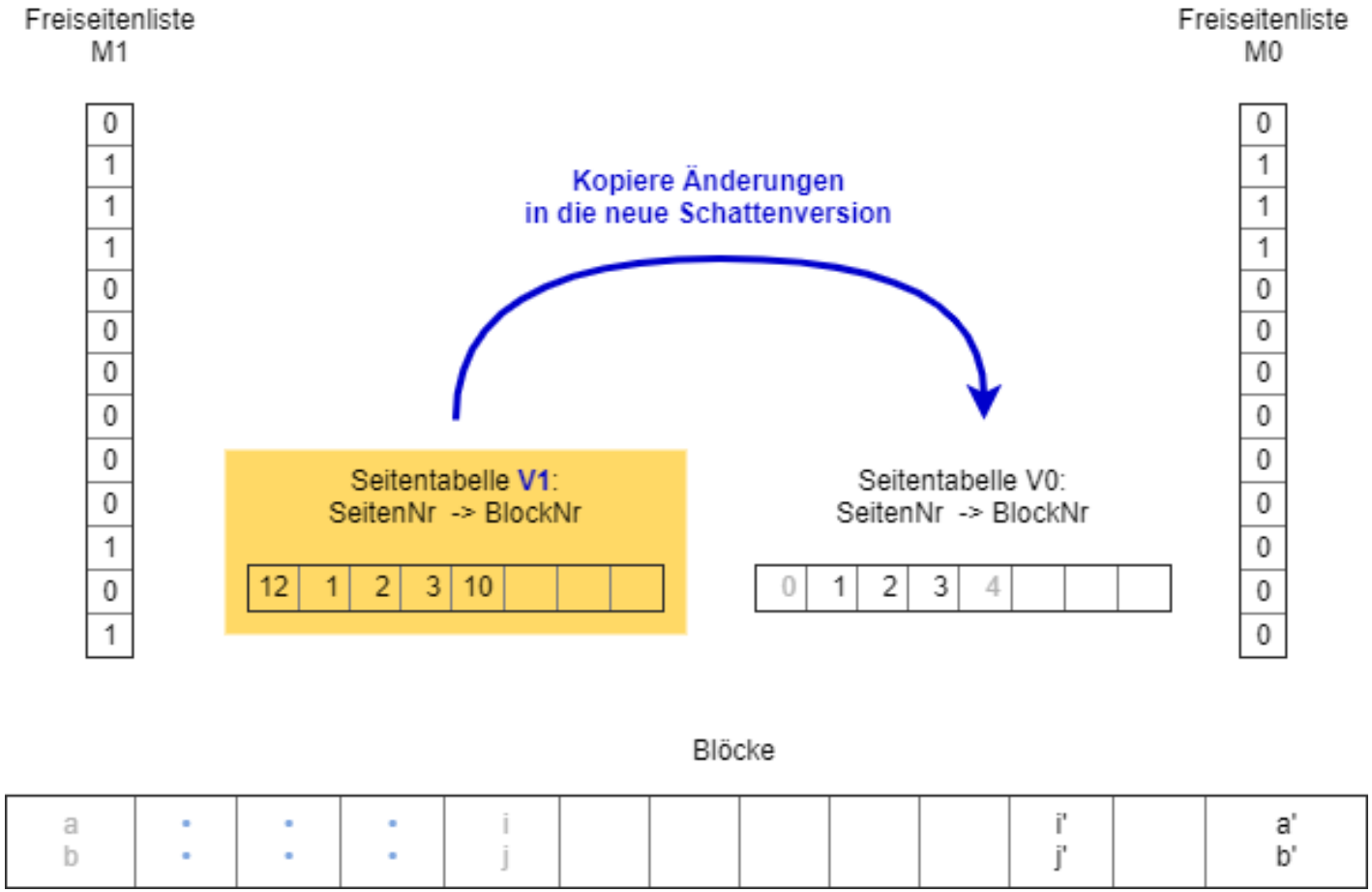
2.2 Originäres Schattenspeicherkonzept



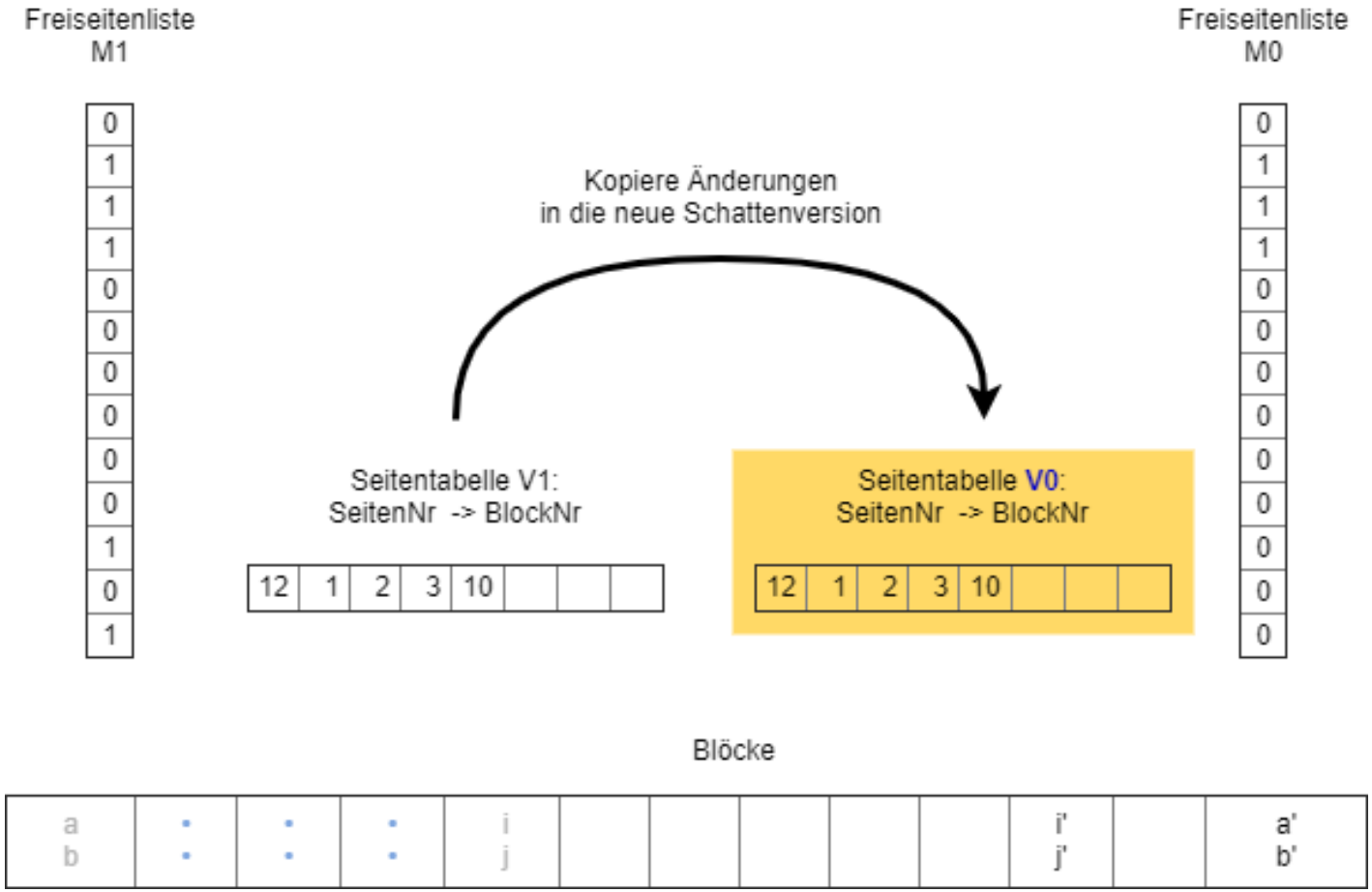
2.2 Originäres Schattenspeicherkonzept



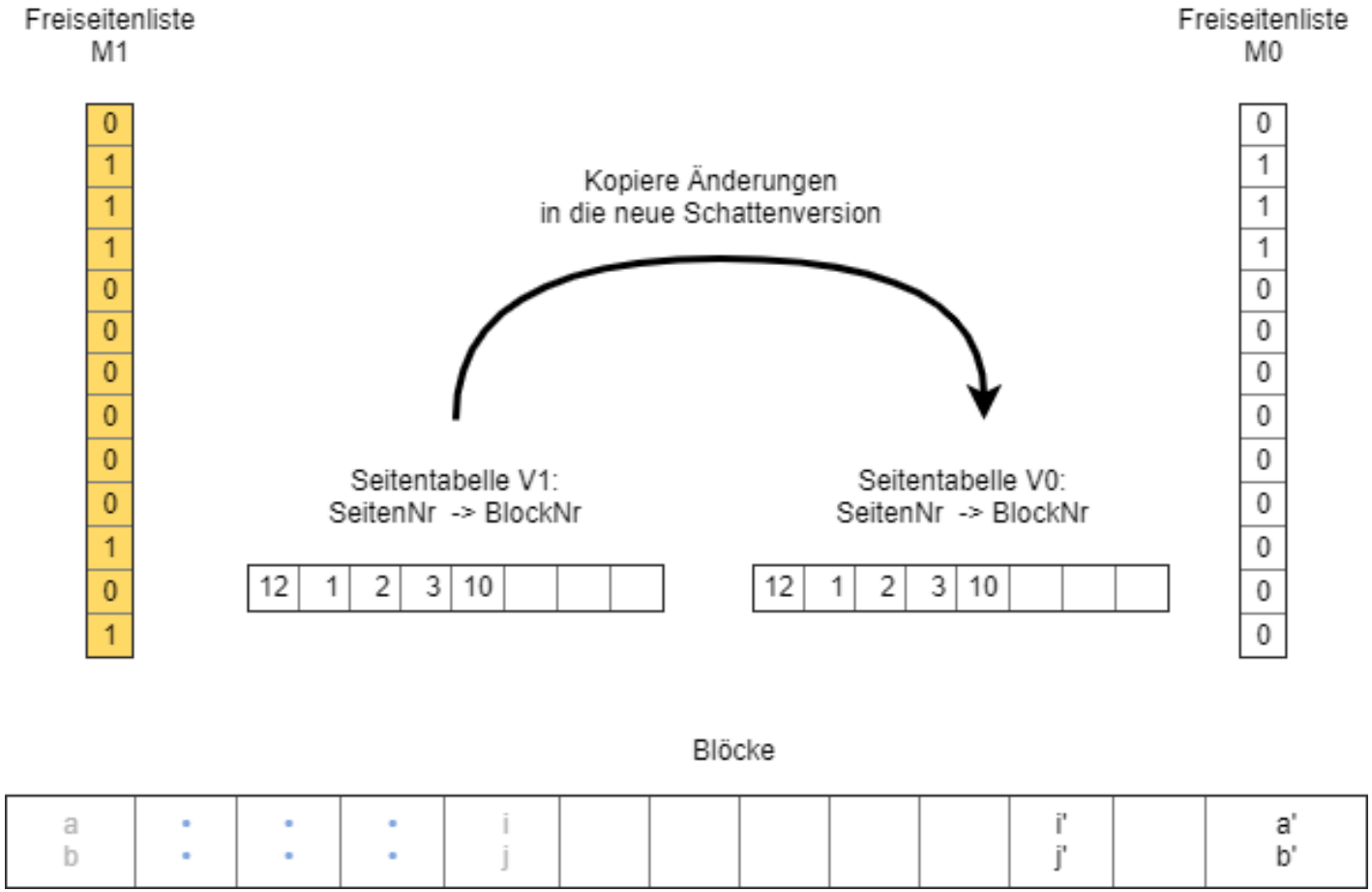
2.2 Originäres Schattenspeicherkonzept



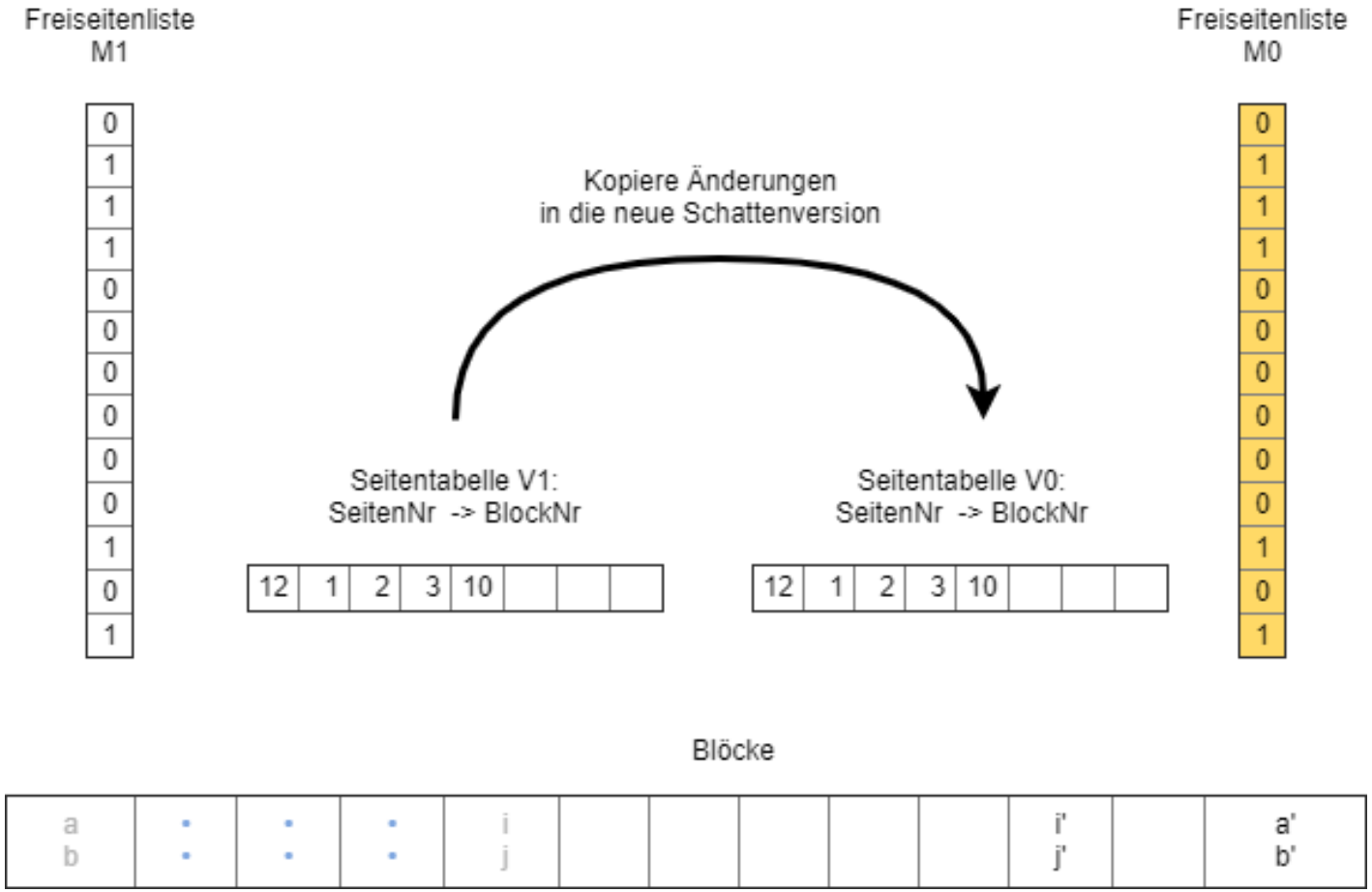
2.2 Originäres Schattenspeicherkonzept



2.2 Originäres Schattenspeicherkonzept



2.2 Originäres Schattenspeicherkonzept



2 Explizit verwalteter Schattenspeicher

2.1 Tupel-Schattenspeicher

2.2 Originäres Schattenspeicherkonzept

2.3 Twin-Objekt-Verfahren

2 Explizit verwalteter Schattenspeicher

2.1 Tupel-Schattenspeicher

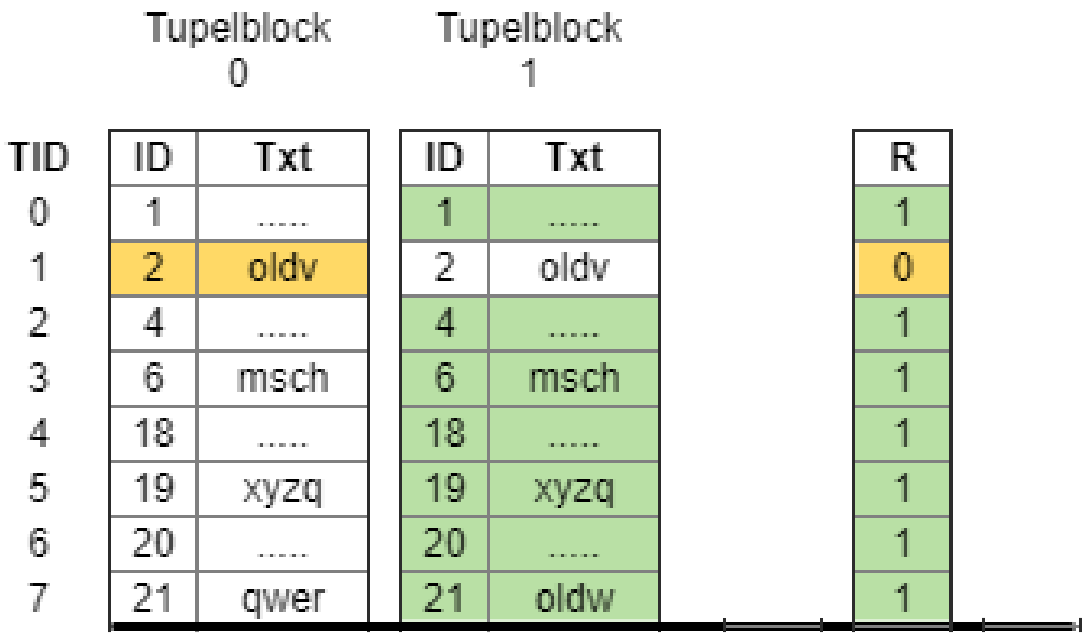
2.2 Originäres Schattenspeicherkonzept

2.3 Twin-Objekt-Verfahren

2.3 Twin-Objekt-Verfahren

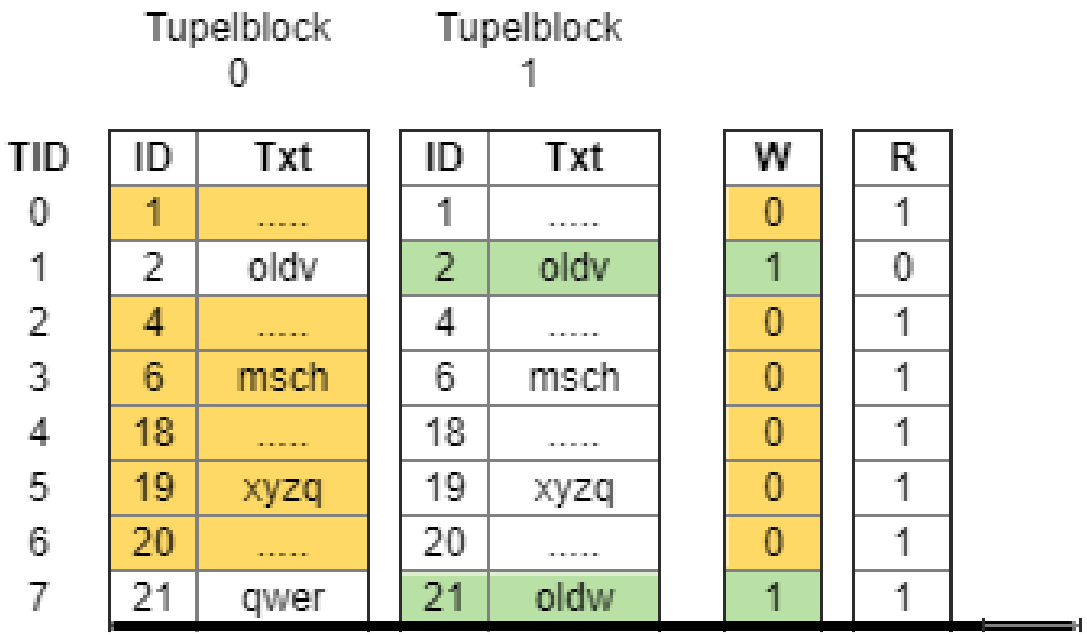
	Tupelblock 0		Tupelblock 1	
TID	ID	Txt	ID	Txt
0	1	.....	1	.....
1	2	oldv	2	oldv
2	4	.....	4	.....
3	6	msch	6	msch
4	18	.....	18	.....
5	19	xyzq	19	xyzq
6	20	.....	20	.....
7	21	qwer	21	oldw

2.3 Twin-Objekt-Verfahren



	Tupelblock 0		Tupelblock 1		
TID	ID	Txt	ID	Txt	R
0	1	.....	1	.....	1
1	2	oldv	2	oldv	0
2	4	.....	4	.....	1
3	6	msch	6	msch	1
4	18	.....	18	.....	1
5	19	xyzq	19	xyzq	1
6	20	.....	20	.....	1
7	21	qwer	21	oldw	1

2.3 Twin-Objekt-Verfahren



	Tupelblock 0		Tupelblock 1			
TID	ID	Txt	ID	Txt	W	R
0	1	.....	1	.....	0	1
1	2	oldv	2	oldv	1	0
2	4	.....	4	.....	0	1
3	6	msch	6	msch	0	1
4	18	.....	18	.....	0	1
5	19	xyzq	19	xyzq	0	1
6	20	.....	20	.....	0	1
7	21	qwer	21	oldw	1	1

2.3 Twin-Objekt-Verfahren

	Tupelblock 0		Tupelblock 1				
TID	ID	Txt	ID	Txt	W	R	D
0	1	.....	1	.....	0	1	0
1	2	oldv	2	oldv	1	0	0
2	4	.....	4	.....	0	1	0
3	6	msch	6	msch	0	1	0
4	18	.....	18	.....	0	1	0
5	19	xyzq	19	xyzq	0	1	0
6	20	.....	20	.....	0	1	0
7	21	qwer	21	oldw	1	1	0

2.3 Twin-Objekt-Verfahren

	Tupelblock 0		Tupelblock 1					
TID	ID	Txt	ID	Txt	W	R	D	
0	1	.....	1	.....	0	1	0	
1	2	oldv	2	oldv	1	0	0	
2	4	.....	4	.....	0	1	0	
3	6	msch	6	msch	0	1	1	delete from R where id = 6;
4	18	.....	18	.....	0	1	0	
5	19	xyzq	19	xyzq	0	1	0	
6	20	.....	20	.....	0	1	0	
7	21	qwer	21	oldw	1	1	0	

2.3 Twin-Objekt-Verfahren

	Tupelblock 0		Tupelblock 1					
TID	ID	Txt	ID	Txt	W	R	D	
0	1	.....	1	.....	0	1	0	
1	2	oldv	2	oldv	1	0	0	
2	4	.....	4	.....	0	1	0	
3	6	msch	6	msch	0	1	1	delete from R where id = 6;
4	18	.....	18	.....	0	1	0	
5	19	xyzq	19	xyzq	0	1	0	
6	20	.....	20	.....	0	1	0	
7	21	qwer	21	oldw	1	1	0	
8	22	efgh	22	efgh	0	1	0	insert into R values (22, 'efgh');

2.3 Twin-Objekt-Verfahren

	Tupelblock 0		Tupelblock 1					
TID	ID	Txt	ID	Txt	W	R	D	
0	1	.....	1	.....	0	1	0	
1	2	oldv	2	oldv	1	0	0	
2	4	.....	4	.....	0	1	0	
3	6	msch	6	msch	0	1	1	delete from R where id = 6;
4	18	.....	18	.....	0	1	0	
5	19	xyzq	19	xyzq	0	1	0	update R set Txt='abcd' where id = 19;
6	20	.....	20	.....	0	1	0	
7	21	qwer	21	oldw	1	1	0	
8	22	efgh	22	efgh	0	1	0	insert into R values (22, 'efgh');

2.3 Twin-Objekt-Verfahren

	Tupelblock 0		Tupelblock 1					
TID	ID	Txt	ID	Txt	W	R	D	
0	1	.....	1	.....	0	1	0	
1	2	oldv	2	oldv	1	0	0	
2	4	.....	4	.....	0	1	0	
3	6	msch	6	msch	0	1	1	delete from R where id = 6;
4	18	.....	18	.....	0	1	0	
5	19	abcd	19	xyzq	0	1	0	update R set Txt='abcd' where id = 19;
6	20	.....	20	.....	0	1	0	
7	21	qwer	21	oldw	1	1	0	
8	22	efgh	22	efgh	0	1	0	insert into R values (22, 'efgh');

2.3 Twin-Objekt-Verfahren

	Tupelblock 0		Tupelblock 1					
TID	ID	Txt	ID	Txt	W	R	D	
0	1	.....	1	.....	0	1	0	
1	2	oldv	2	oldv	1	0	0	
2	4	.....	4	.....	0	1	0	
3	6	msch	6	msch	0	1	1	delete from R where id = 6;
4	18	.....	18	.....	0	1	0	
5	19	abcd	19	xyzq	0	0	0	update R set Txt='abcd' where id = 19;
6	20	.....	20	.....	0	1	0	
7	21	qwer	21	oldw	1	1	0	
8	22	efgh	22	efgh	0	1	0	insert into R values (22, 'efgh');

2 Explizit verwalteter Schattenspeicher

2.1 Tupel-Schattenspeicher

2.2 Originäres Schattenspeicherkonzept

2.3 Twin-Objekt-Verfahren

1 Motivation
2 Explizit verwalteter Schattenspeicher
3 Betriebssystem eigener Schattenspeicher

1 Motivation
2 Explizit verwalteter Schattenspeicher
3 Betriebssystem eigener Schattenspeicher

3 Betriebssystem eigener Schattenspeicher

- 3.1 ein OLTP-Thread, ein OLAP-Thread
- 3.2 mehrere OLAP-Threads
- 3.3 mehrere OLTP-Threads
- 3.4 Implementierung

3 Betriebssystem eigener Schattenspeicher

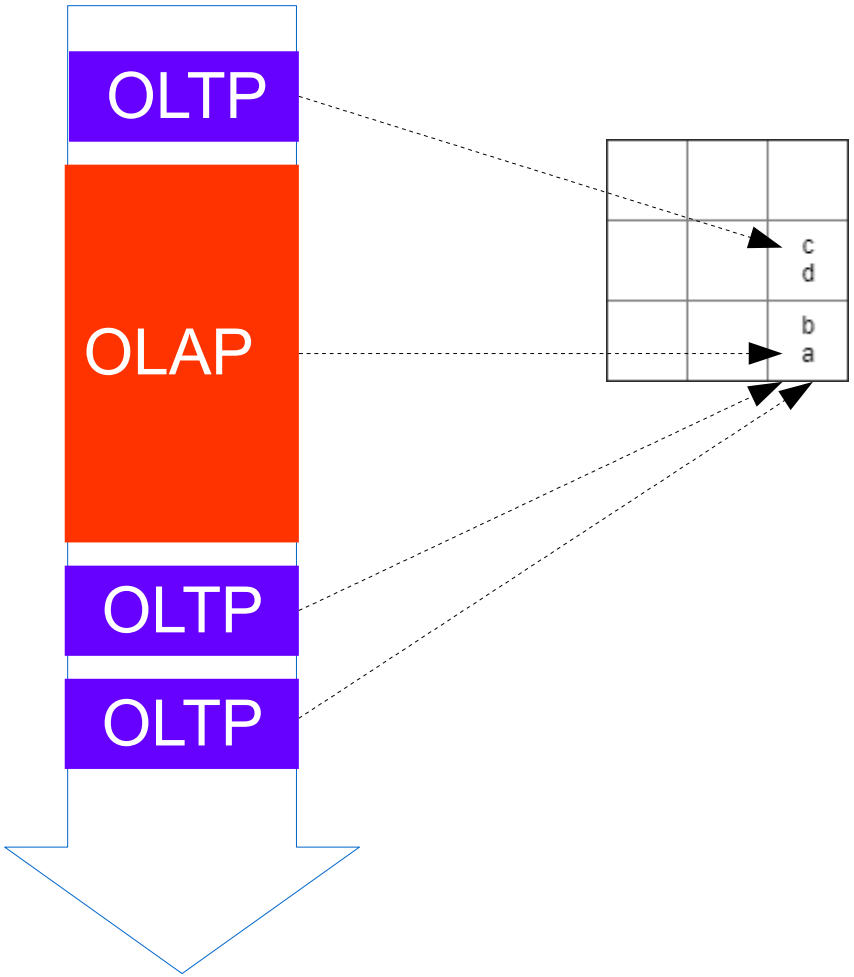
3.1 ein OLTP-Thread, ein OLAP-Thread

3.2 mehrere OLAP-Threads

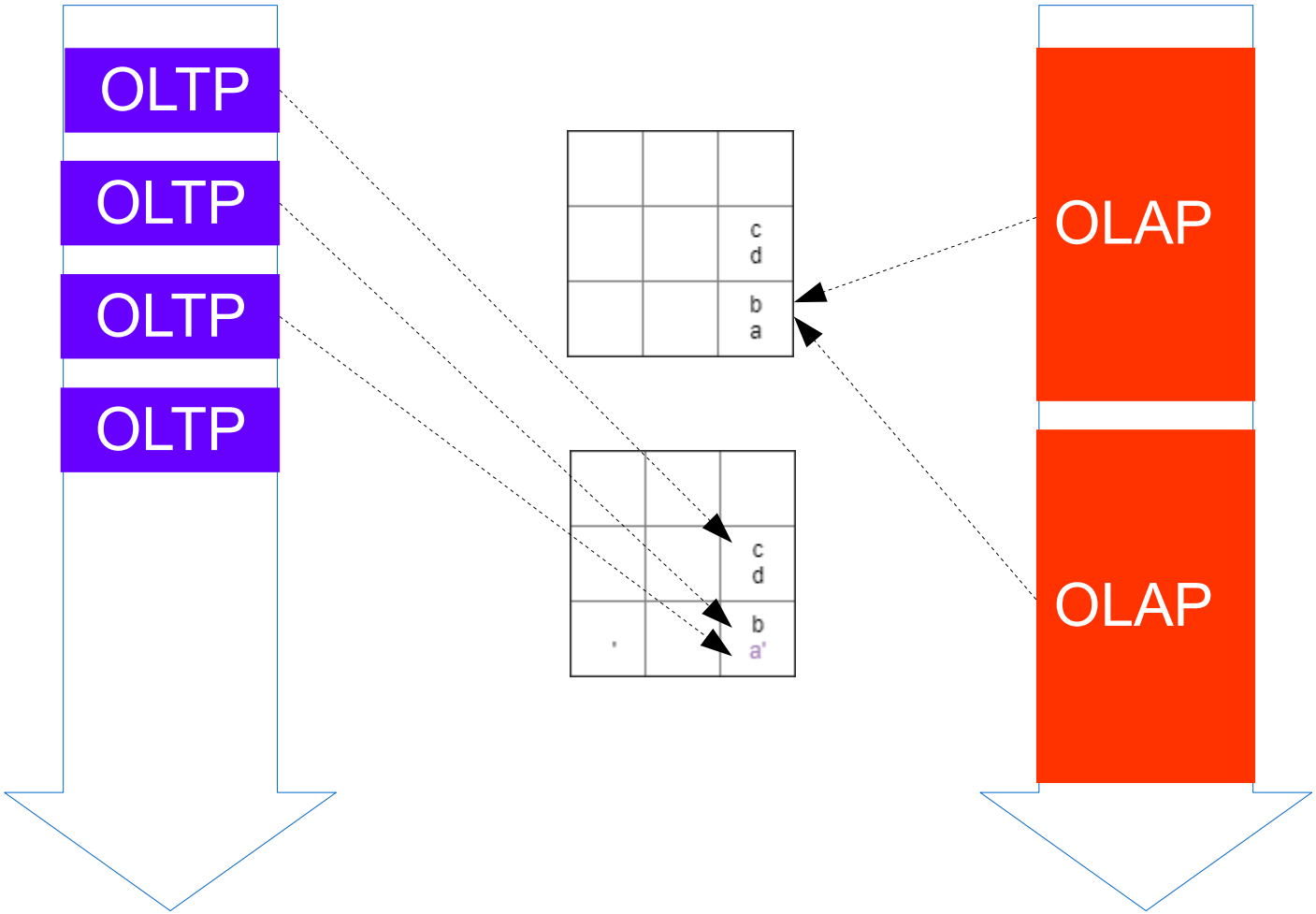
3.3 mehrere OLTP-Threads

3.4 Implementierung

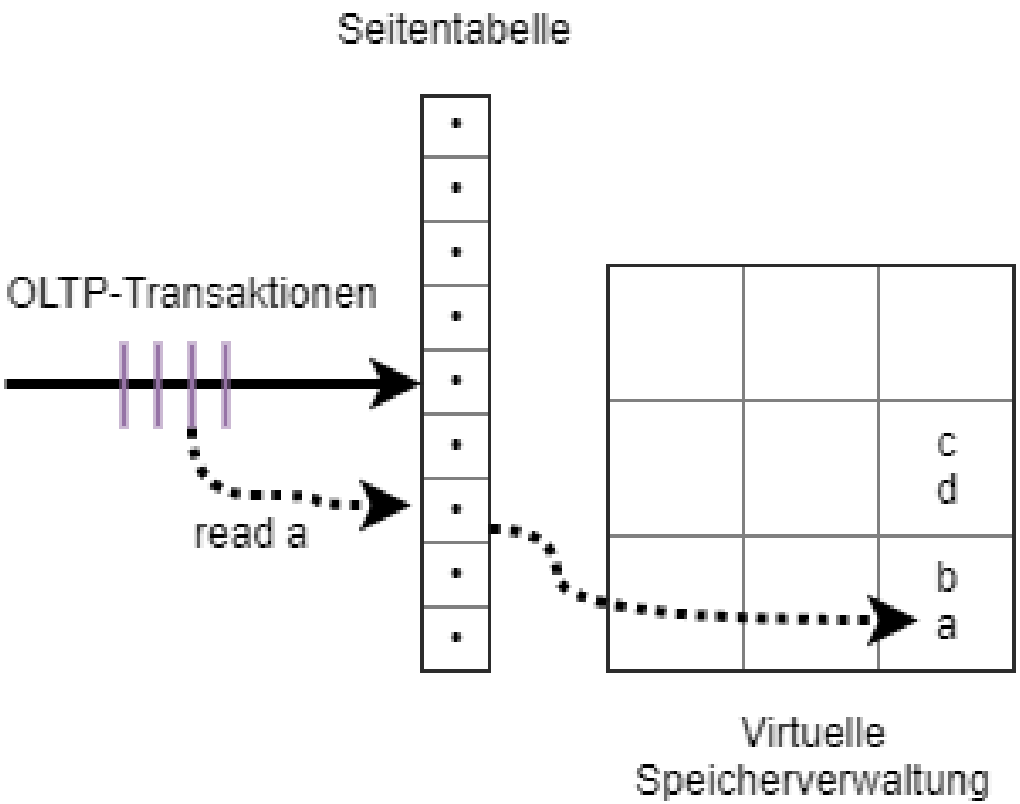
3.1 ein OLTP-Thread, ein OLAP-Thread



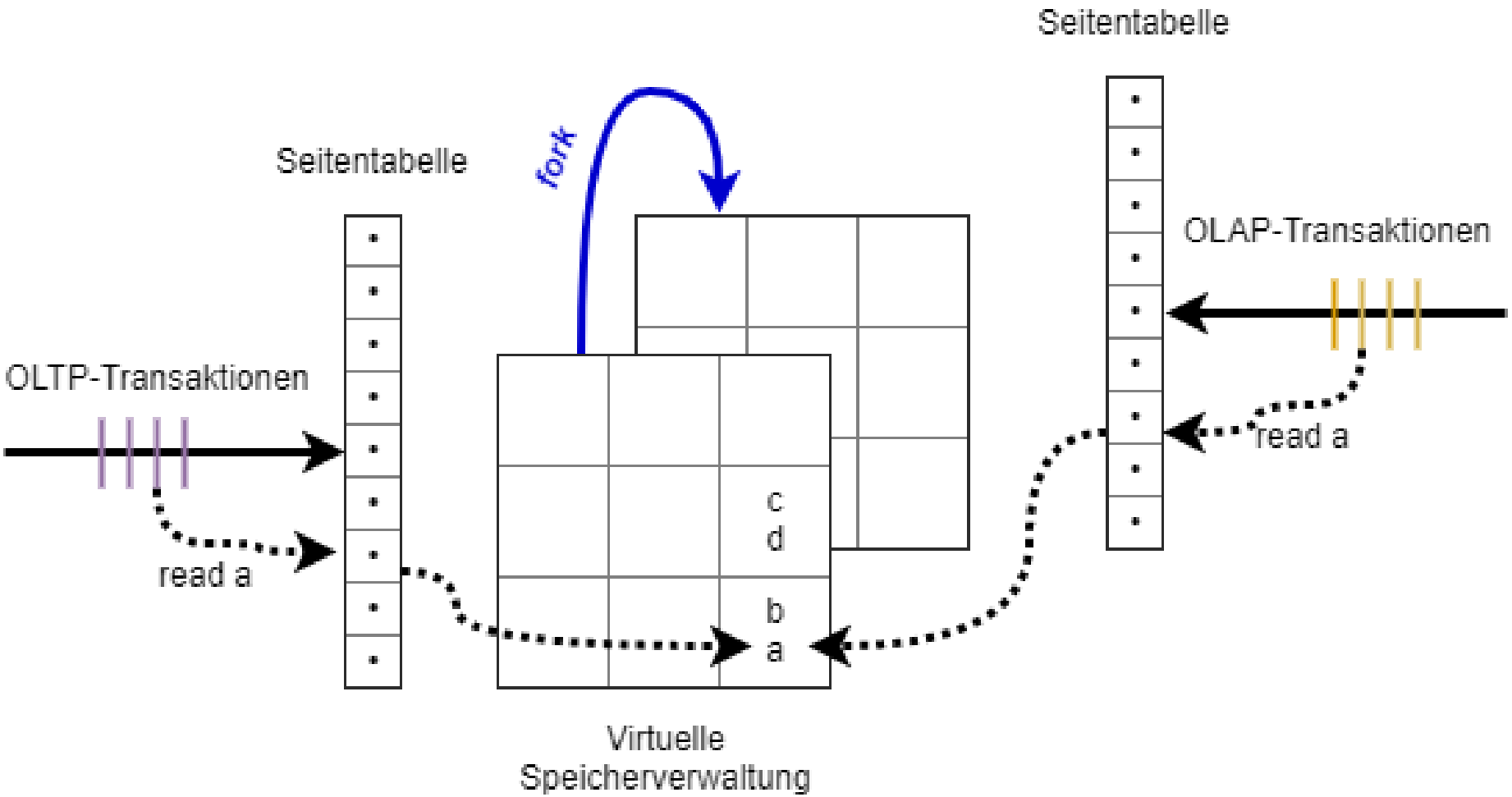
3.1 ein OLTP-Thread, ein OLAP-Thread



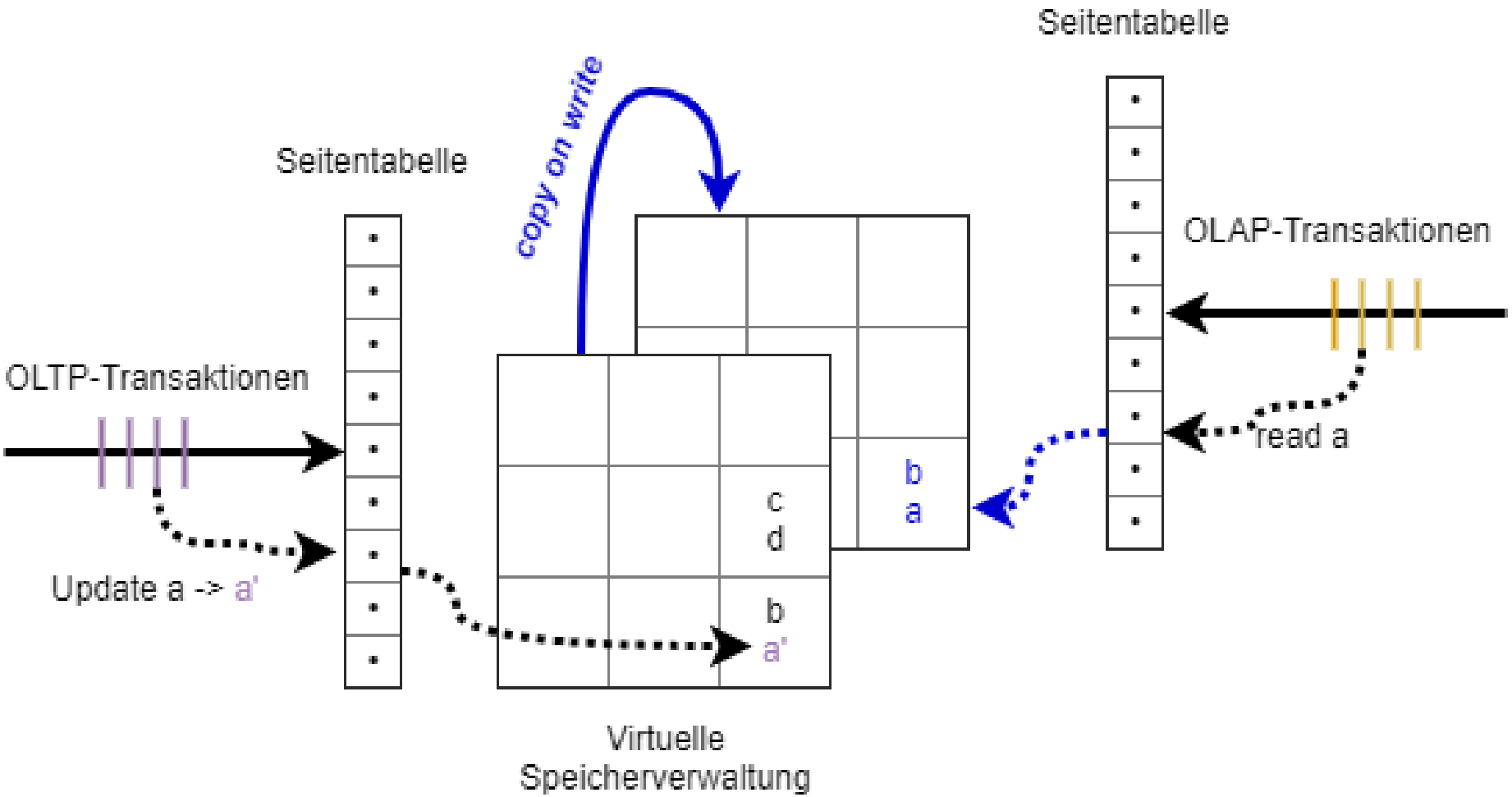
3.1 ein OLTP-Thread, ein OLAP-Thread



3.1 ein OLTP-Thread, ein OLAP-Thread



3.1 ein OLTP-Thread, ein OLAP-Thread



3 Betriebssystem eigener Schattenspeicher

3.1 ein OLTP-Thread, ein OLAP-Thread

3.2 mehrere OLAP-Threads

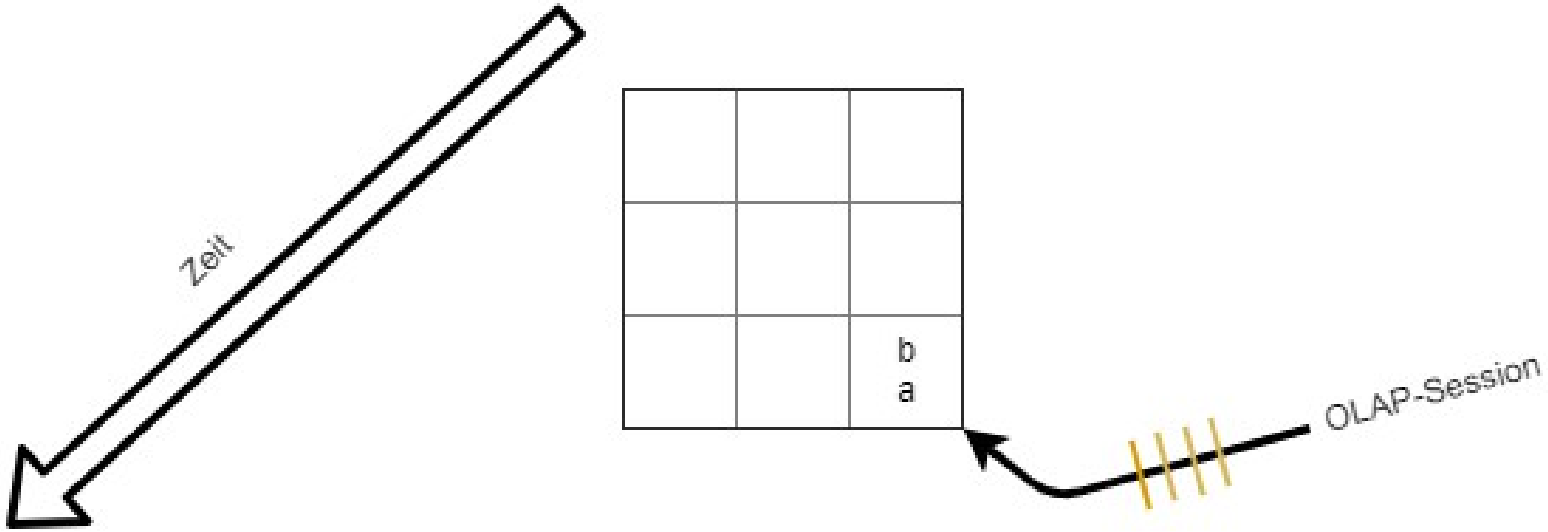
3.3 mehrere OLTP-Threads

3.4 Implementierung

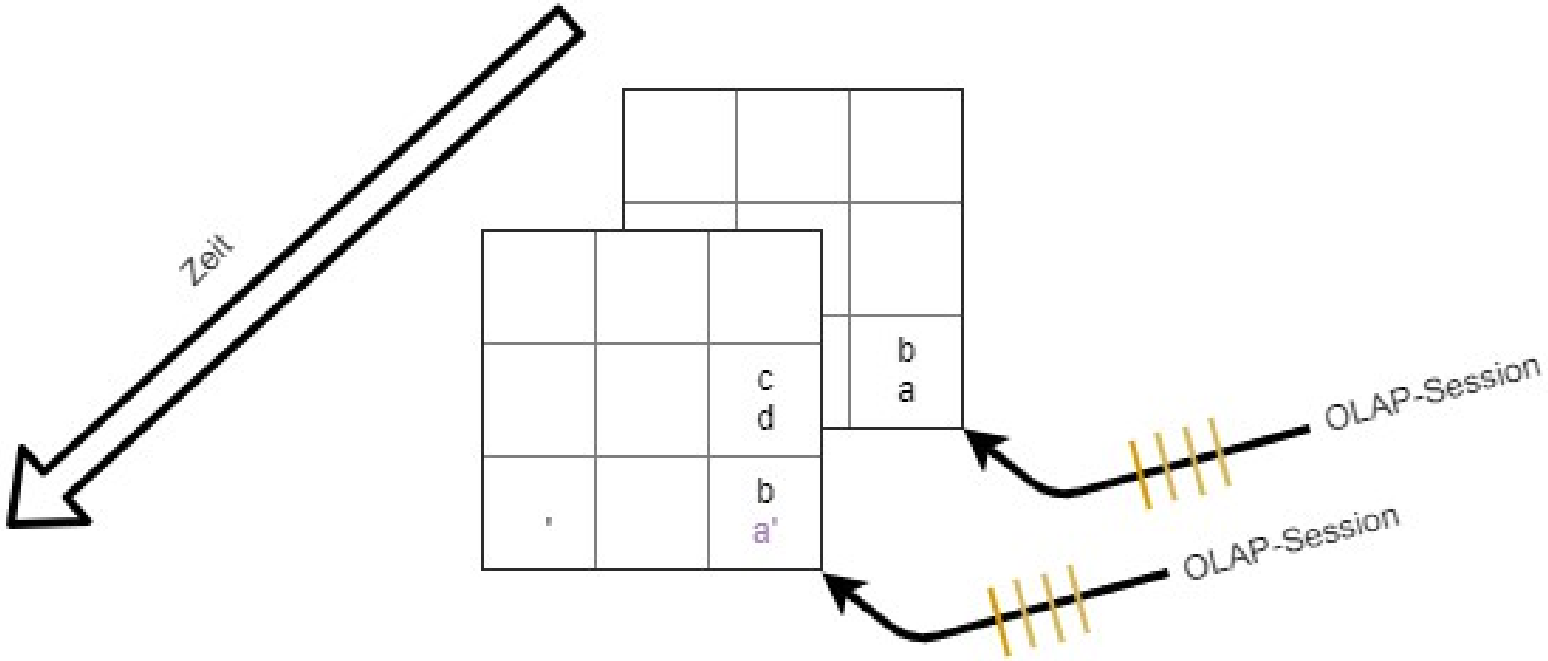
3 Betriebssystem eigener Schattenspeicher

- 3.1 ein OLTP-Thread, ein OLAP-Thread
- 3.2 mehrere OLAP-Threads
- 3.3 mehrere OLTP-Threads
- 3.4 Implementierung

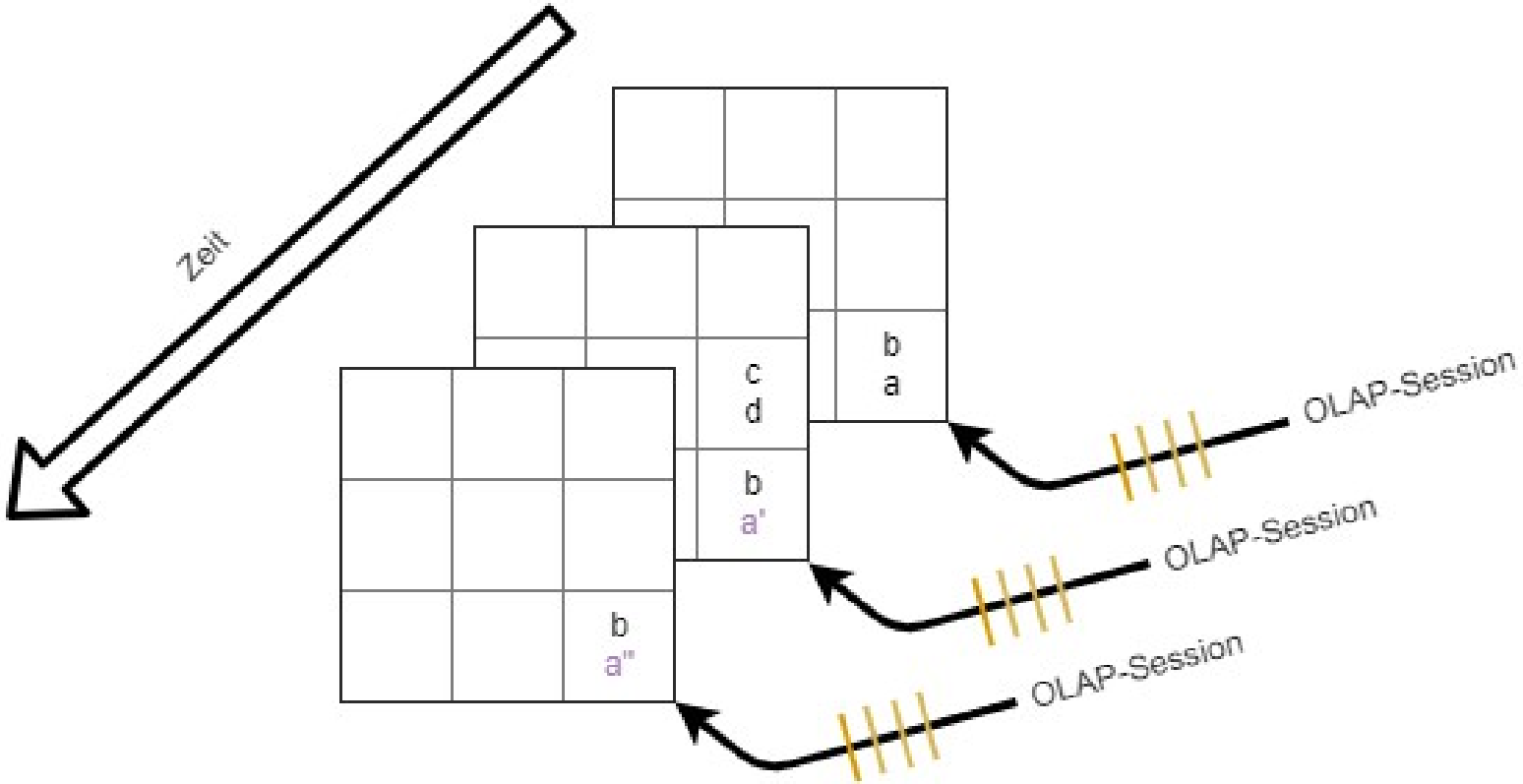
3.2 mehrere OLAP-Threads



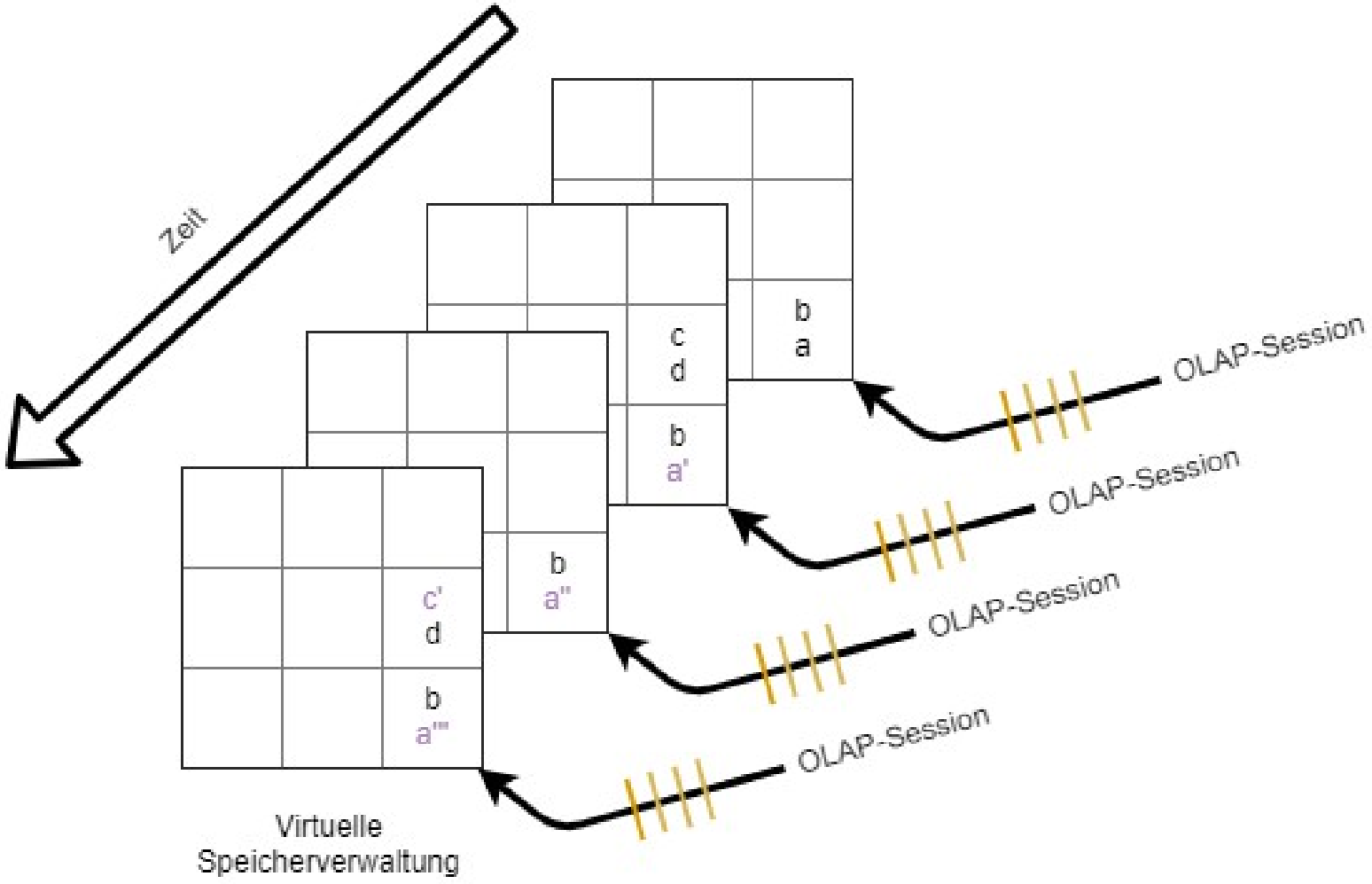
3.2 mehrere OLAP-Threads



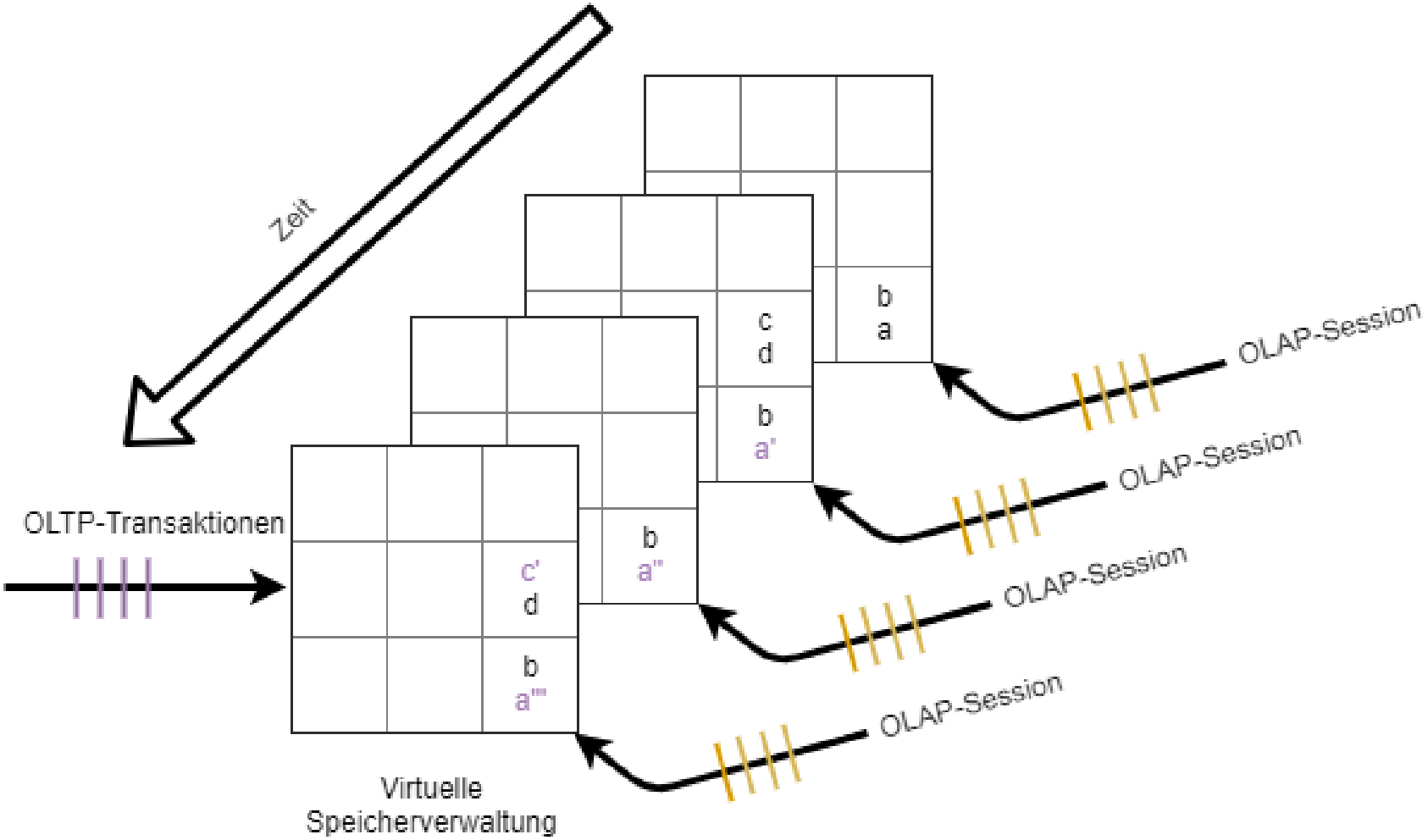
3.2 mehrere OLAP-Threads



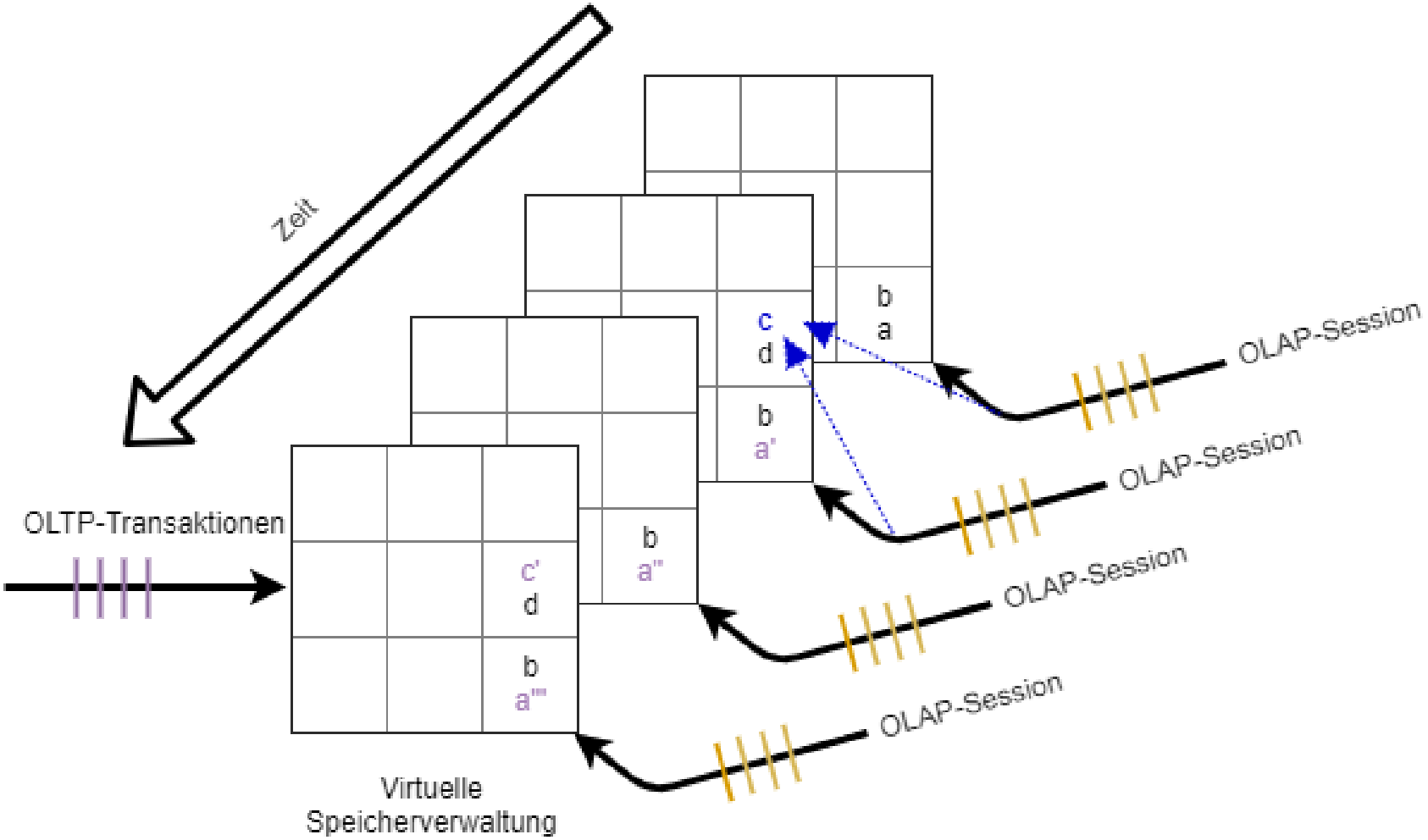
3.2 mehrere OLAP-Threads



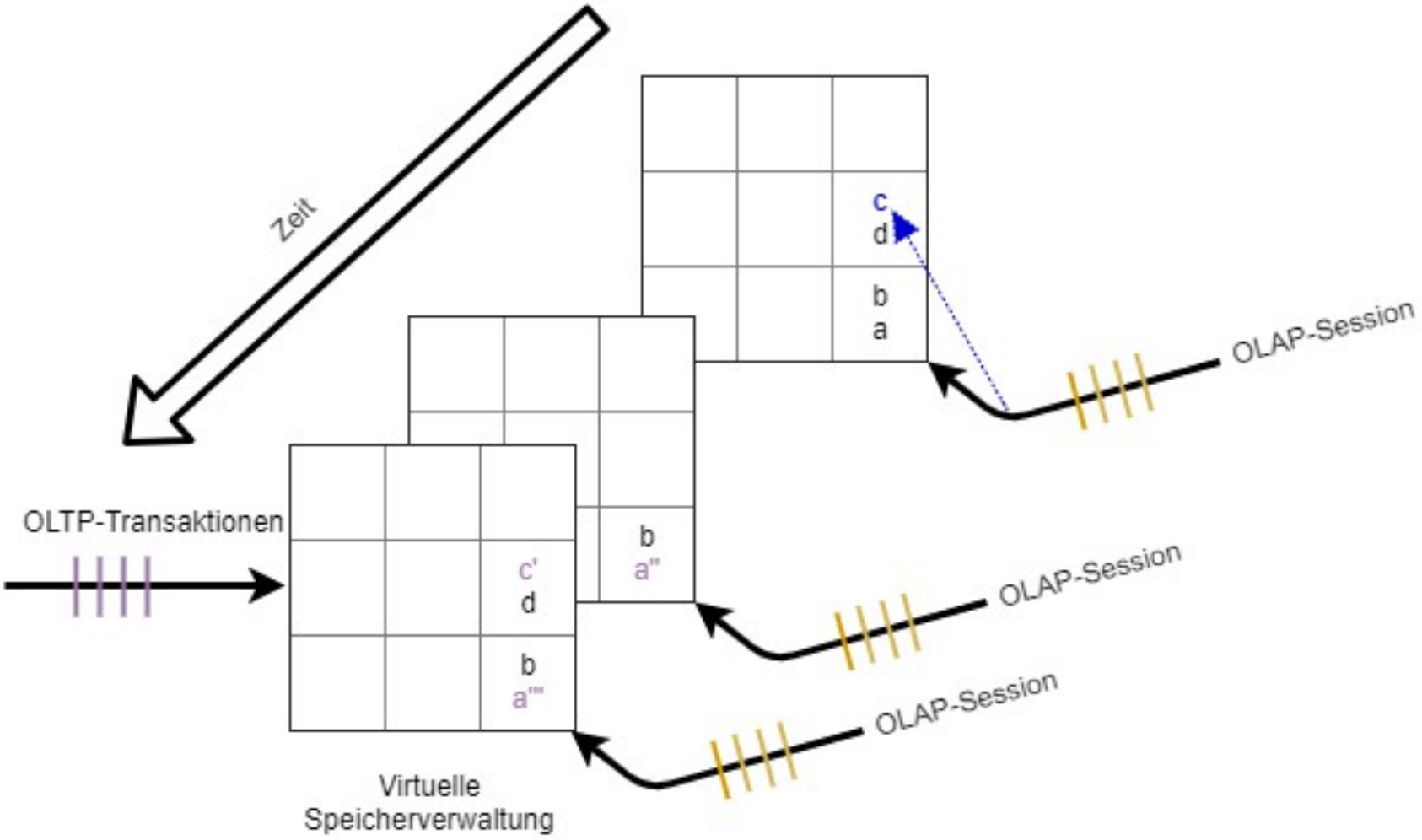
3.2 mehrere OLAP-Threads



3.2 mehrere OLAP-Threads



3.2 mehrere OLAP-Threads



3 Betriebssystem eigener Schattenspeicher

- 3.1 ein OLTP-Thread, ein OLAP-Thread
- 3.2 mehrere OLAP-Threads
- 3.3 mehrere OLTP-Threads
- 3.4 Implementierung

3 Betriebssystem eigener Schattenspeicher

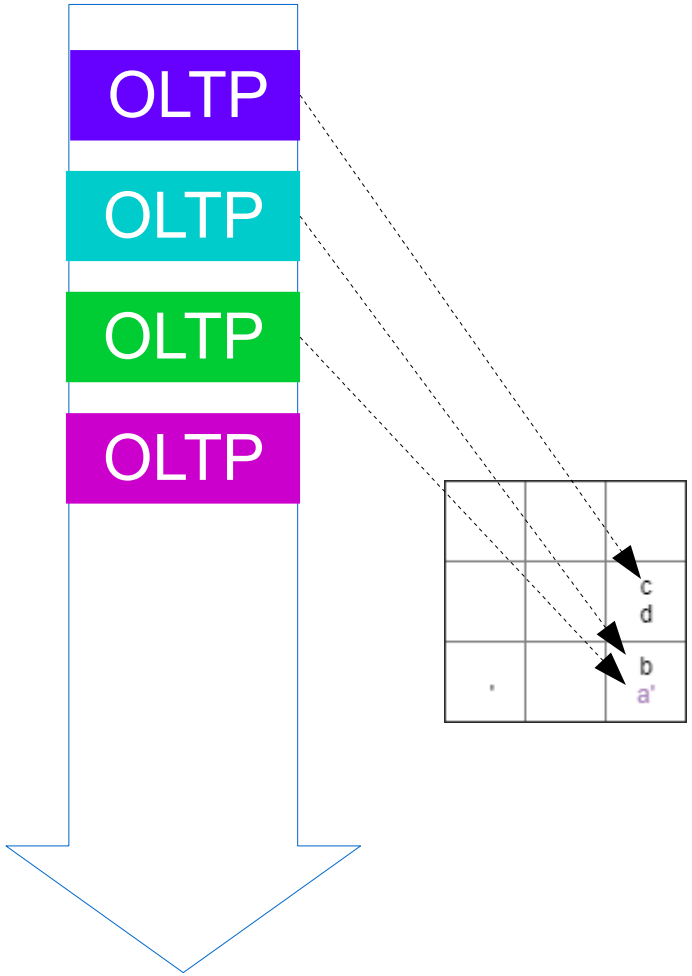
3.1 ein OLTP-Thread, ein OLAP-Thread

3.2 mehrere OLAP-Threads

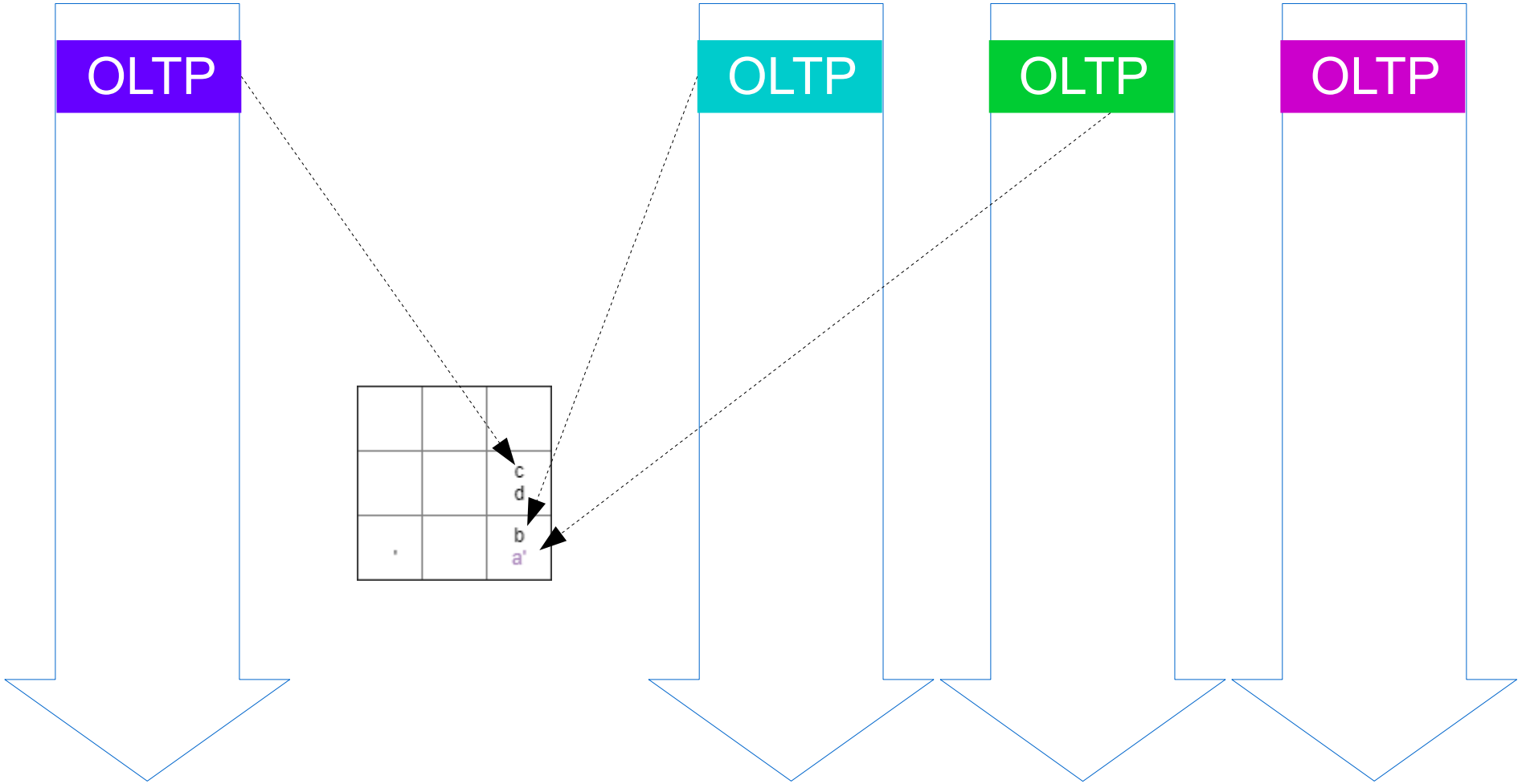
3.3 mehrere OLTP-Threads

3.4 Implementierung

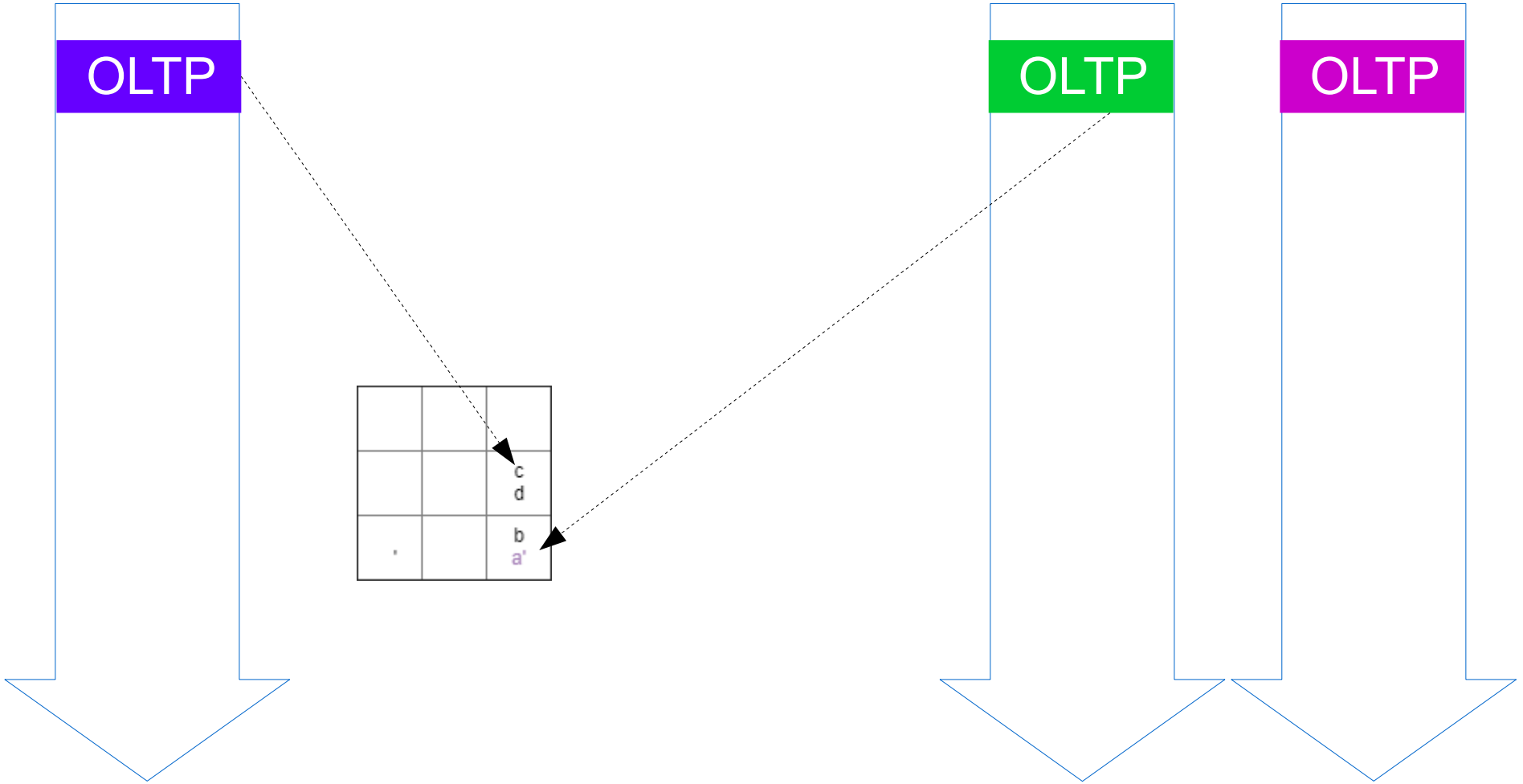
3.3 mehrere OLTP-Threads



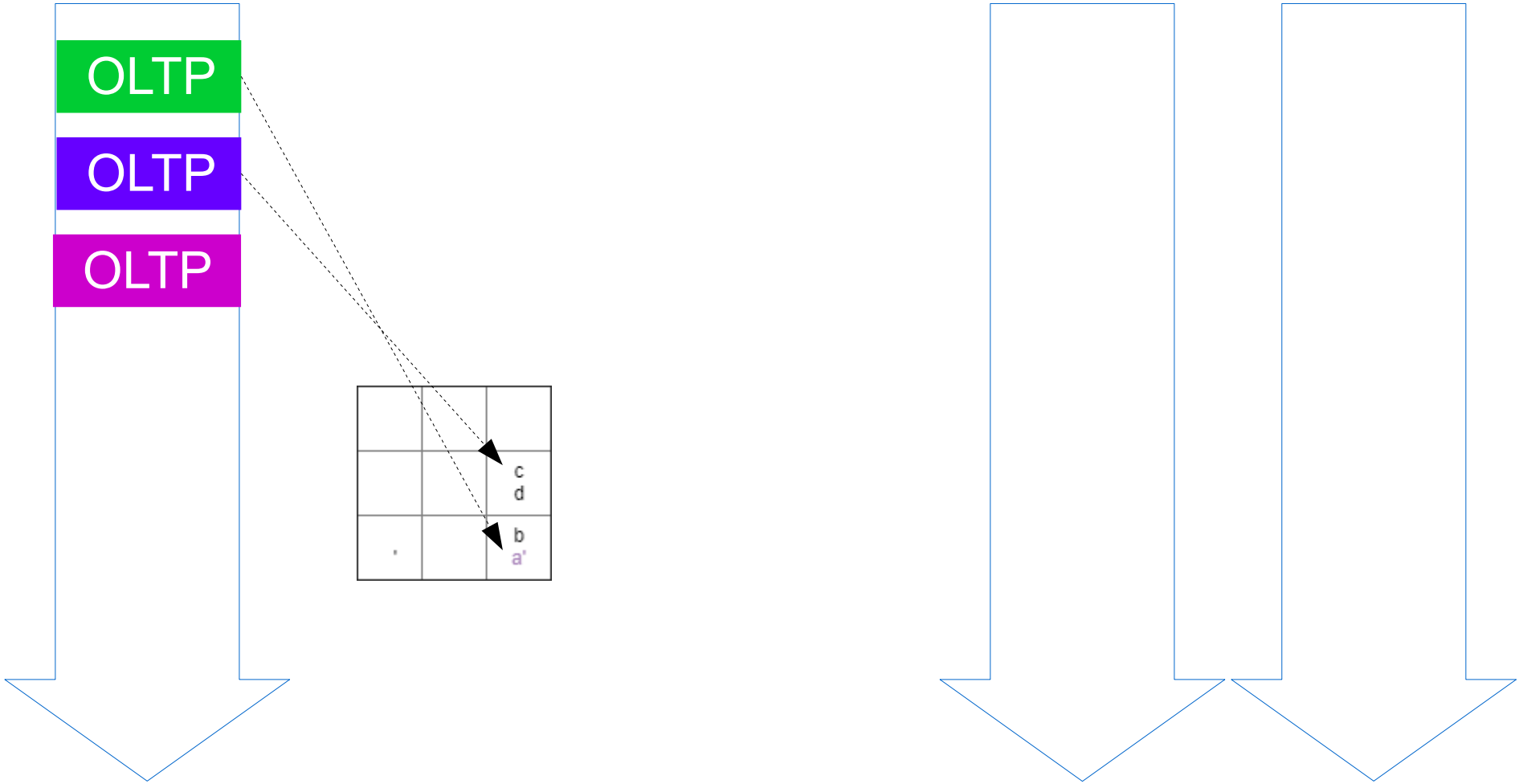
3.3 mehrere OLTP-Threads



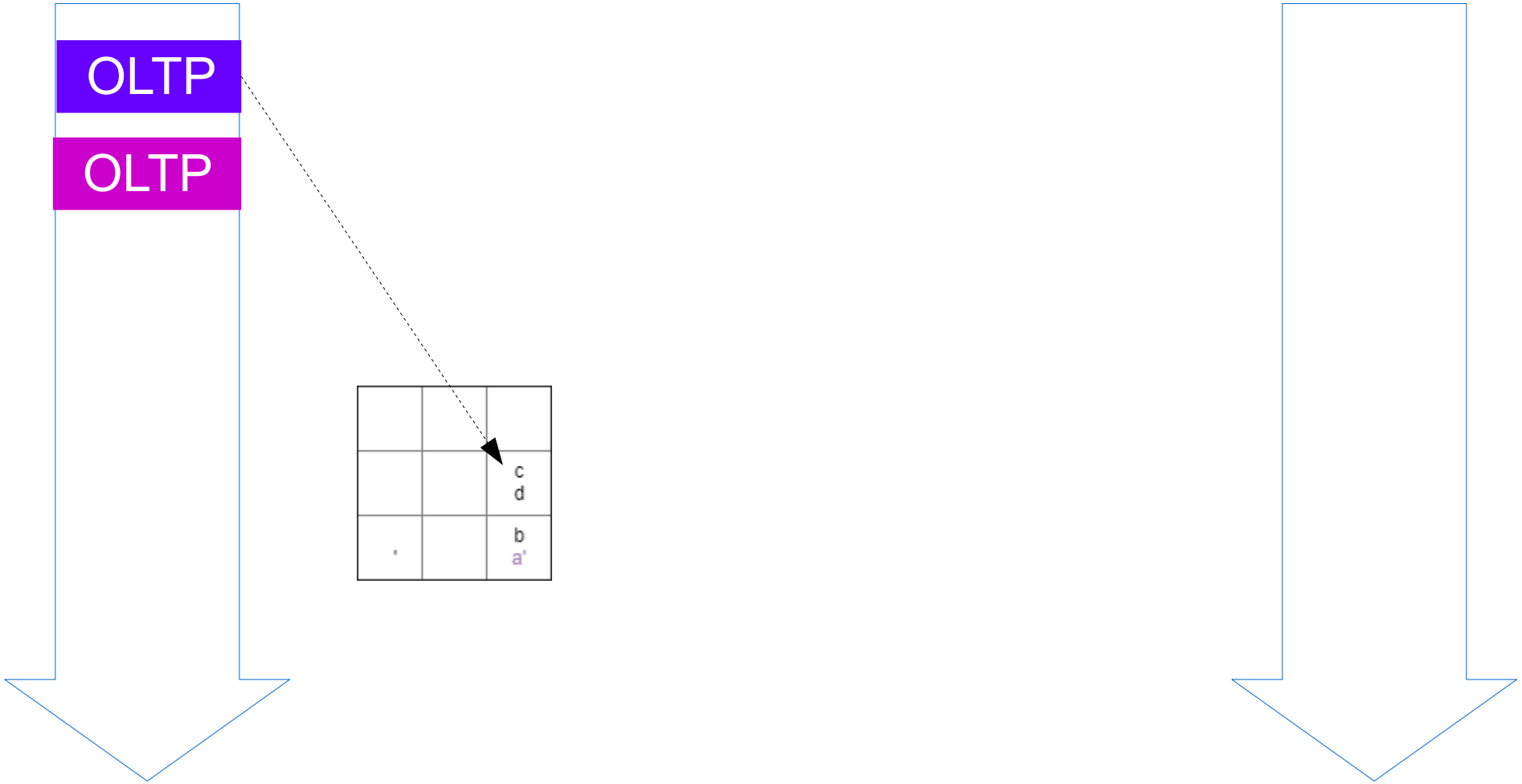
3.3 mehrere OLTP-Threads



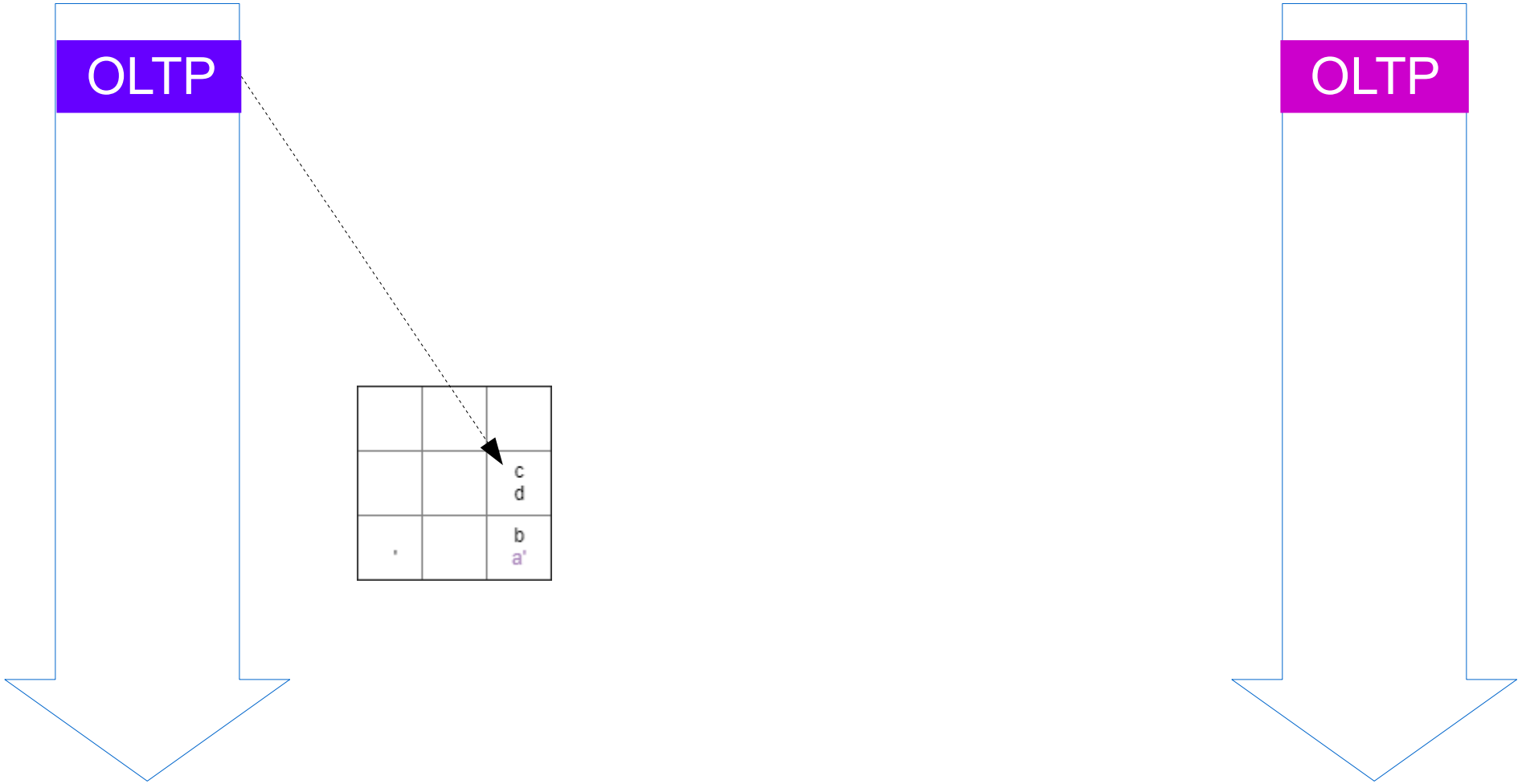
3.3 mehrere OLTP-Threads



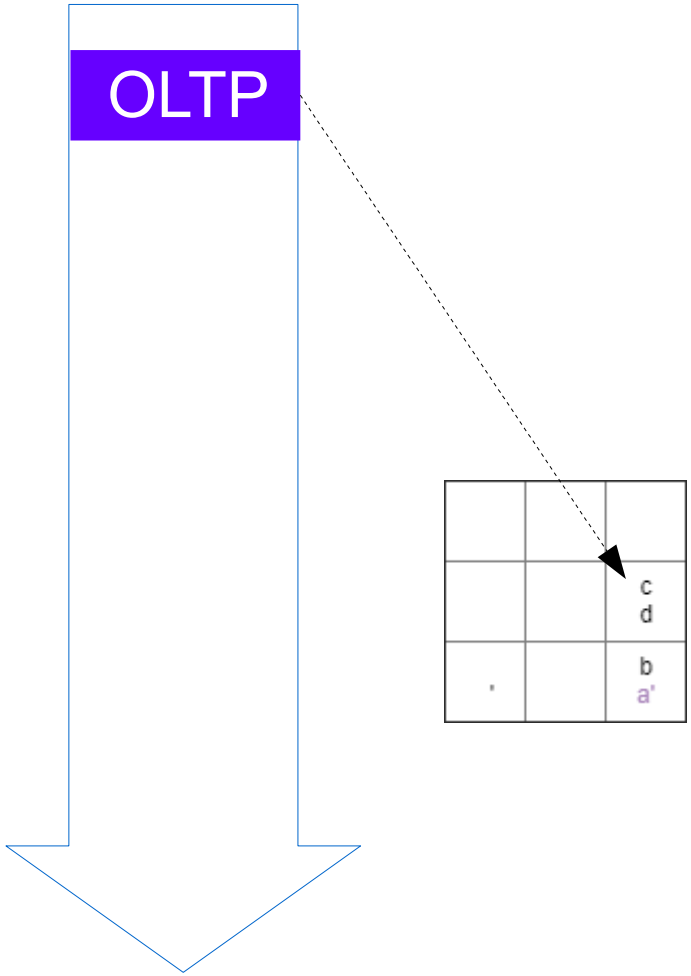
3.3 mehrere OLTP-Threads



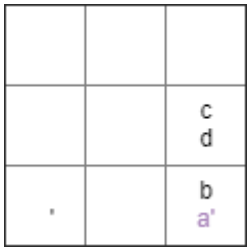
3.3 mehrere OLTP-Threads



3.3 mehrere OLTP-Threads



3.3 mehrere OLTP-Threads



3 Betriebssystem eigener Schattenspeicher

3.1 ein OLTP-Thread, ein OLAP-Thread

3.2 mehrere OLAP-Threads

3.3 mehrere OLTP-Threads

3.4 Implementierung

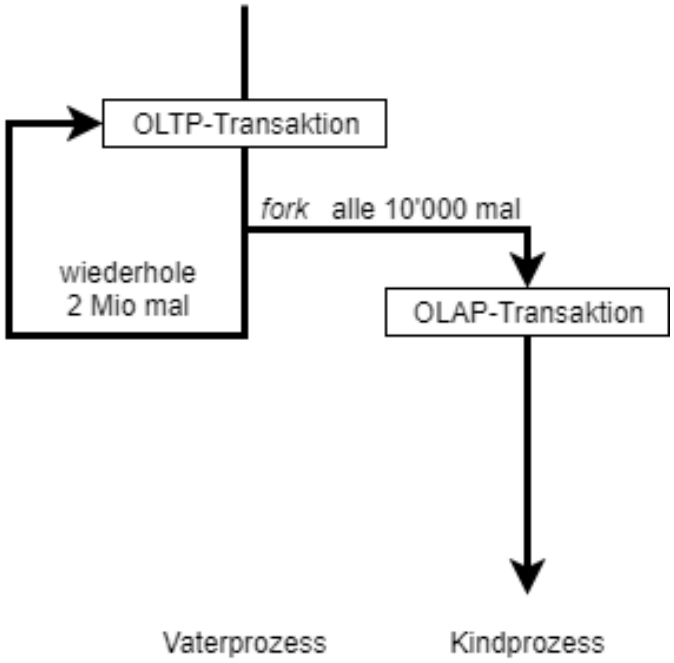
3 Betriebssystem eigener Schattenspeicher

- 3.1 ein OLTP-Thread, ein OLAP-Thread
- 3.2 mehrere OLAP-Threads
- 3.3 mehrere OLTP-Threads
- 3.4 Implementierung

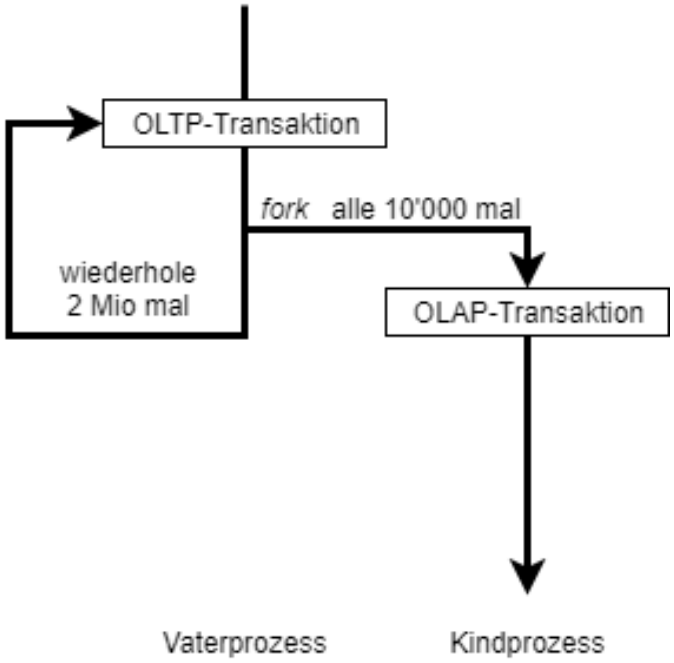
3.4 Implementierung

```
#include <vector>       // vector

// ...

// main-memory database
struct order {
    unsigned int orderId;
    unsigned int customerId;
    double totalAmount;
};
vector<order> orders;
```

3.4 Implementierung

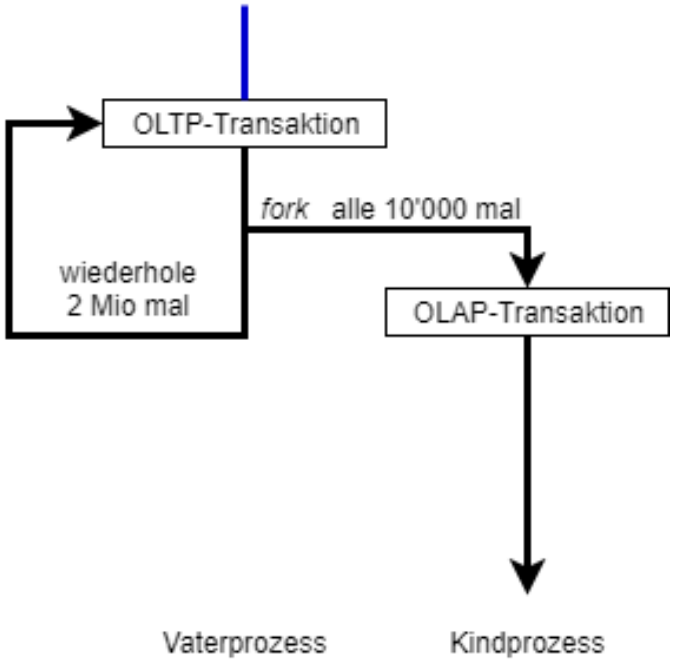


3.4 Implementierung



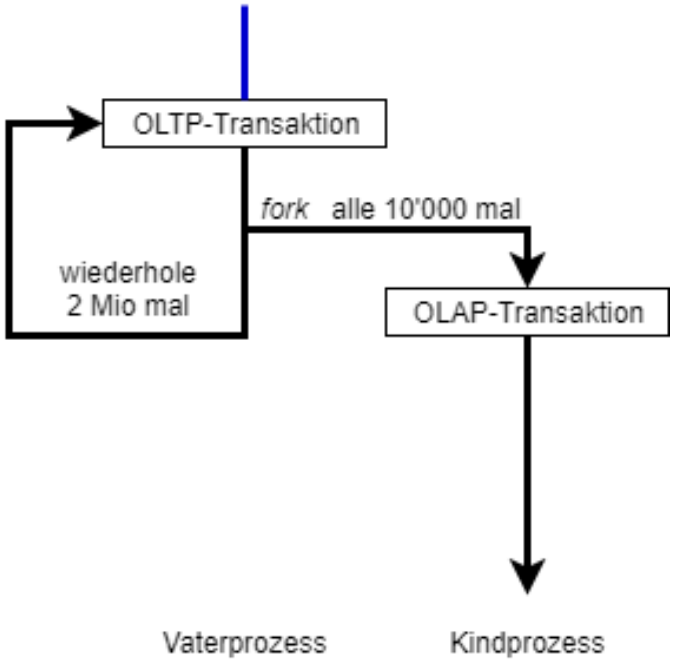

```
int main() {
    initialize();
    auto numberOfTransactions = 2000000;
    auto olapEvery = 10000;

    for (auto i=1;
    i <= numberOfTransactions; ++i) {
        oltp();
        if (i % olapEvery == 0) {
            olap();
        }
    }

    return 0;
}
```

3.4 Implementierung



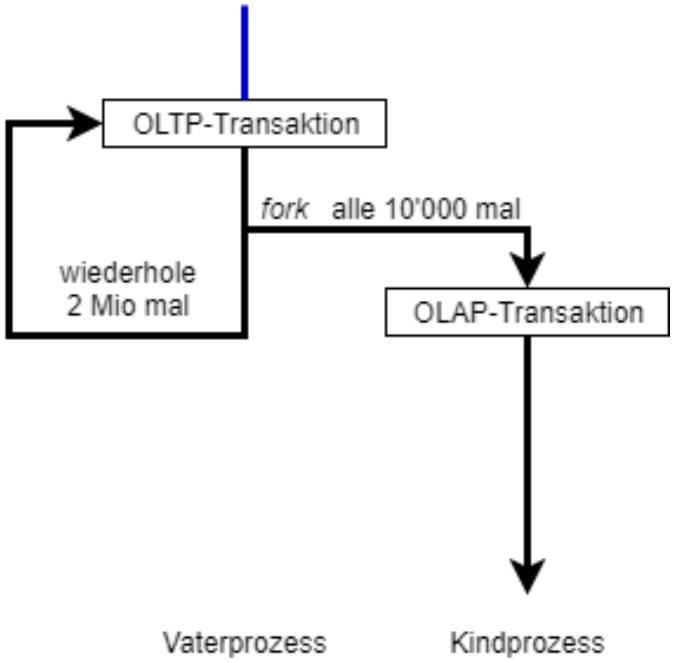

```
int main() {
    initialize();
    auto numberOfTransactions = 2000000;
    auto olapEvery = 10000;

    for (auto i=1;
    i <= numberOfTransactions; ++i) {
        oltp();
        if (i % olapEvery == 0) {
            olap();
        }
    }

    return 0;
}
```

3.4 Implementierung



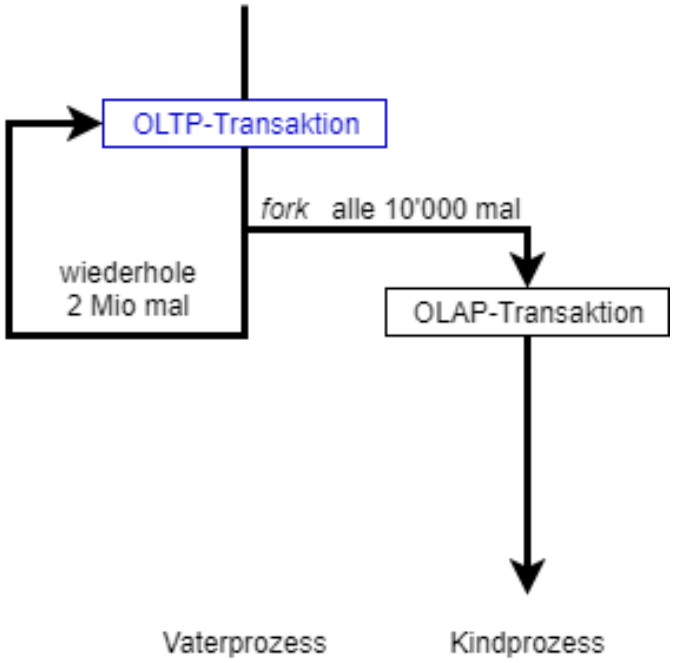

```
#include <stdlib.h>      // srand, rand
#include <time.h>        // time

// ...

using namespace std;

// seed random number generator
void initialize() {
    srand (time(NULL));
}
```

3.4 Implementierung



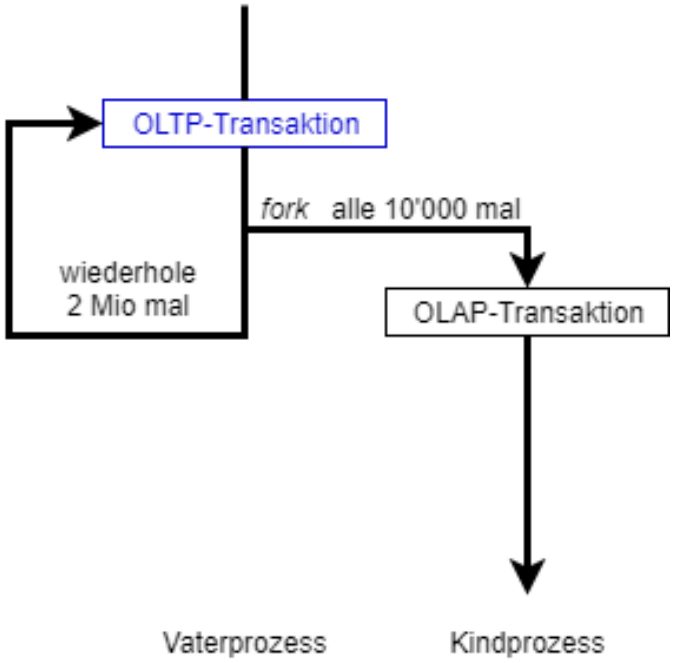

```
int main() {
    initialize();
    auto numberOfTransactions = 2000000;
    auto olapEvery = 10000;

    for (auto i=1;
    i <= numberOfTransactions; ++i) {
        oltp();
        if (i % olapEvery == 0) {
            olap();
        }
    }

    return 0;
}
```

3.4 Implementierung



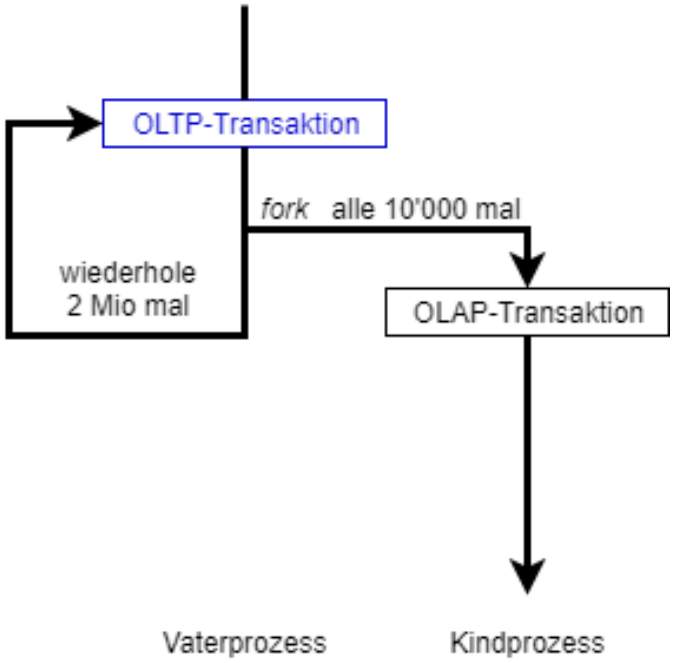

```
int main() {
    initialize();
    auto numberOfTransactions = 2000000;
    auto olapEvery = 10000;

    for (auto i=1;
    i <= numberOfTransactions; ++i) {
        oltp();
        if (i % olapEvery == 0) {
            olap();
        }
    }

    return 0;
}
```

3.4 Implementierung



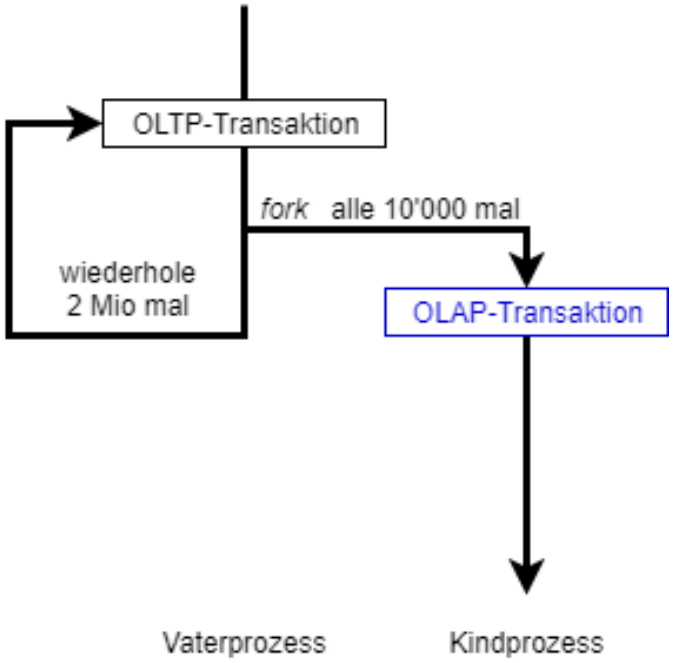

```cpp
#include <vector>        // vector
#include <stdlib.h>      // srand, rand
// ...

unsigned int currentId = 1;

// execute OLTP transaction: place an
order with a random amount
void oltp() {
    orders.push_back({ currentId,
    currentId++, randomAmount() });
}

// generate random amount (1.00 to 999.99)
double randomAmount() {
    auto euros = rand() % 999 + 1;
    auto cents = rand() % 99 + 0;
    return euros + cents / 100.0;
}
```

3.4 Implementierung



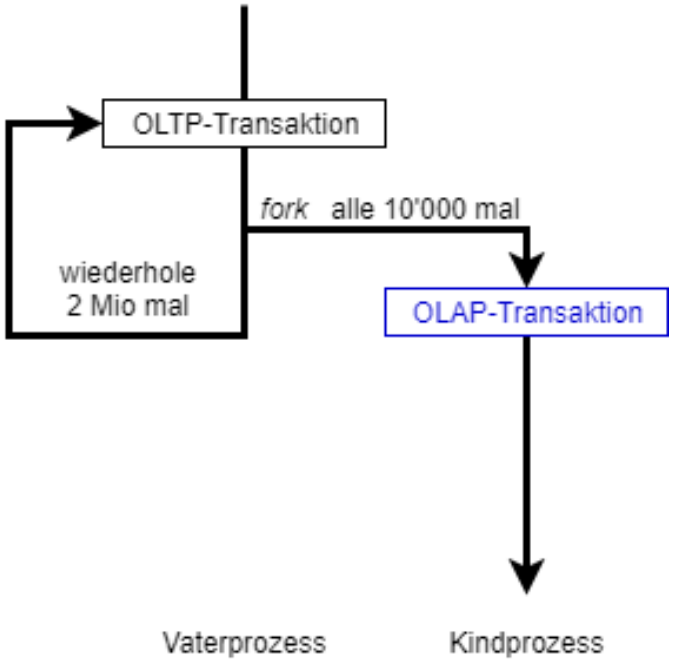

```
int main() {
    initialize();
    auto numberOfTransactions = 2000000;
    auto olapEvery = 10000;

    for (auto i=1;
    i <= numberOfTransactions; ++i) {
        oltp();
        if (i % olapEvery == 0) {
            olap();
        }
    }

    return 0;
}
```

3.4 Implementierung



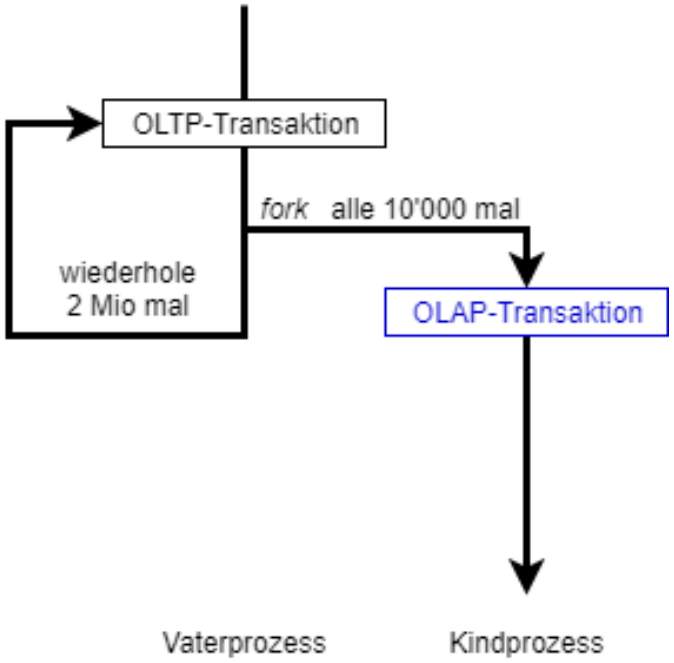

```
int main() {
    initialize();
    auto numberOfTransactions = 2000000;
    auto olapEvery = 10000;

    for (auto i=1;
    i <= numberOfTransactions; ++i) {
        oltp();
        if (i % olapEvery == 0) {
            olap();
        }
    }

    return 0;
}
```

3.4 Implementierung



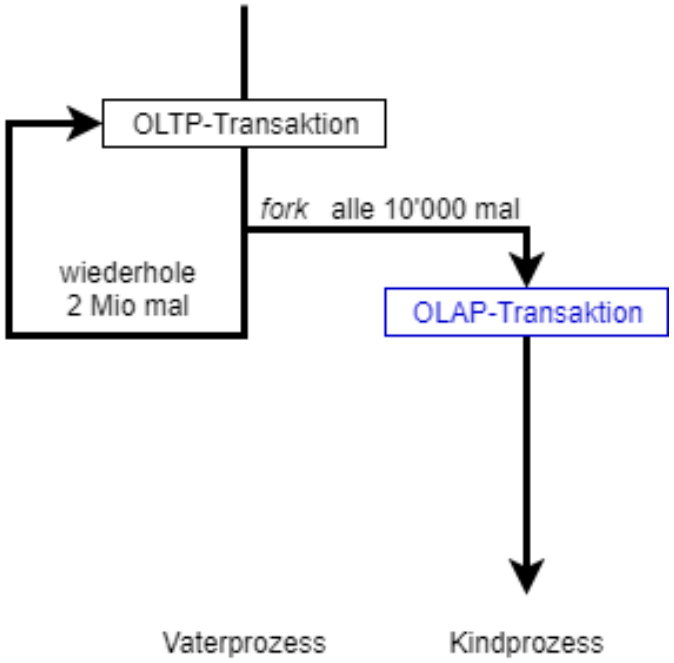

```
#include <unistd.h>      // fork
#include <stdio.h>       // printf
// ...
```

3.4 Implementierung



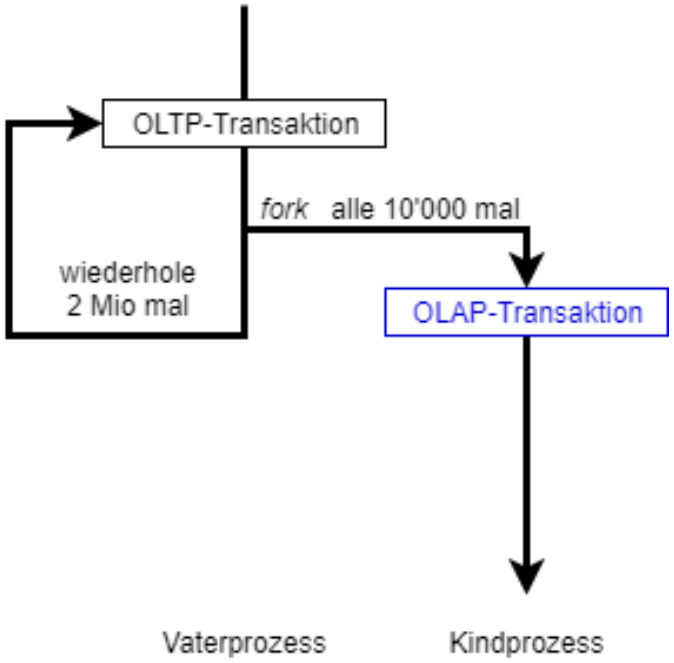

```
// avg of total amount and # of orders
void olap() {
    // fork the parent process, store PID
    pid_t pid = fork();

    // only do OLAP if this process is a
    // child process (pid == 0);
    if (pid == 0) {
        auto sum = 0.0;
        auto count = orders.size();
        for (order ord : orders) {
            sum += ord.totalAmount;
        }
        auto average = sum / count;
        printf("average total amount =
        %.2f | number of orders = %lu\n",
        average, count);
        exit(0);    // delete snapshot
    }
}
```

3.4 Implementierung



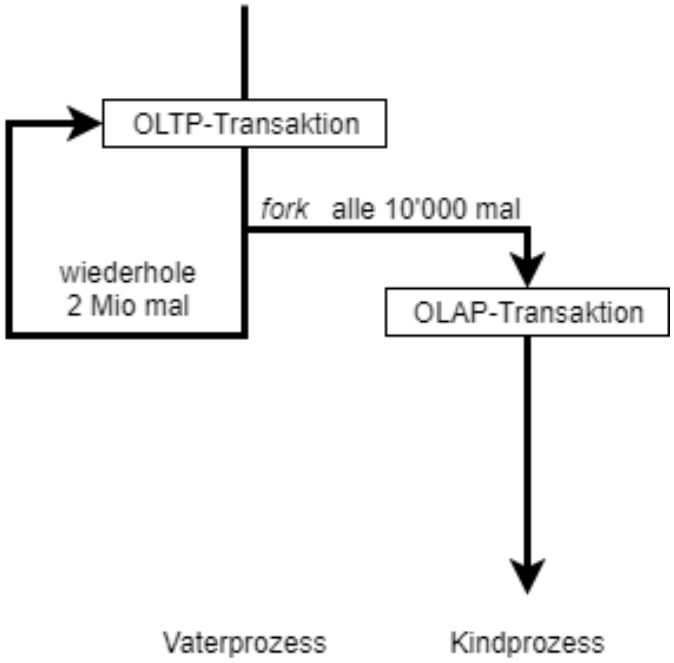

```
// avg of total amount and # of orders
void olap() {
    // fork the parent process, store PID
    pid_t pid = fork();

    // only do OLAP if this process is a
    // child process (pid == 0);
    if (pid == 0) {
        auto sum = 0.0;
        auto count = orders.size();
        for (order ord : orders) {
            sum += ord.totalAmount;
        }
        auto average = sum / count;
        printf("average total amount =
        %.2f | number of orders = %lu\n",
        average, count);
        exit(0);    // delete snapshot
    }
}
```

3.4 Implementierung




```
int main() {
    initialize();
    auto numberOfTransactions = 2000000;
    auto olapEvery = 10000;

    for (auto i=1;
    i <= numberOfTransactions; ++i) {
        oltp();
        if (i % olapEvery == 0) {
            olap();
        }
    }

    return 0;
}
```

3.4 Implementierung




```
int main() {
    initialize();
    auto numberOfTransactions = 2000000;
    auto olapEvery = 10000;

    for (auto i=1;
    i <= numberOfTransactions; ++i) {
        oltp();
        if (i % olapEvery == 0) {
            olap();
        }
    }

    return 0;
}
```