Bazar.com (Lab2)

Introduction

In this time, we will upgrade and improve the performance of our application due to increasing of loading on the basic server from lab1. We will use caching (proxy) in the implicitly in the frontend server. The catalog servers (two servers replicated) can send invalidate messages to the frontend (proxy) when an object in the database is updated.

By this process, the frontend removes the dirty object from cache so that if the user requests the object again, the proxy will have the clean object in its cache!

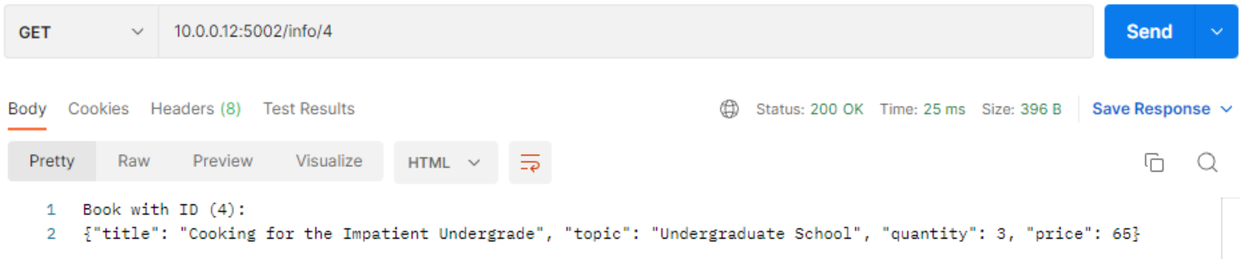
We created two types of caches, lookup cache & search cache both of them hold the tag – value. In the case of lookup cache, the tag is the ID of the book, and the value is the book object coming from DB. In case of search cache, the tag is the topic, and the value is an array of books that its topic is the same as the tag.

When the user performs purchase operation, the catalog edits the quantity in the stock, send to the other replica to update its quantity too, so that saving the consistency manner. Order1 talks with Catalog1 server and Order2 talks with Catalog2 server.

Demo

We will show you a GET method which gets a book by ID (info operation), as we request, the frontend server (same as proxy) will cache the object, so in the second time will fetch the object from cache, we will also show you the difference between response times:

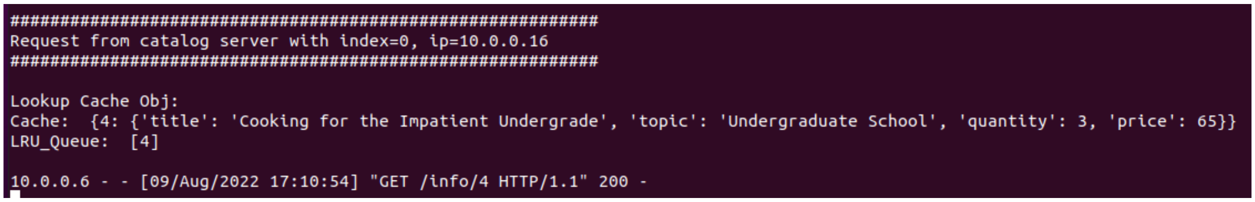
First time without caching:



```
Book with ID (4):
{"title": "Cooking for the Impatient Undergrade", "topic": "Undergraduate School", "quantity": 3, "price": 65}
```

We can see that the time = 25ms.

From console of frontend (the request is done from catalog has IP address: 10.0.0.16:5000)

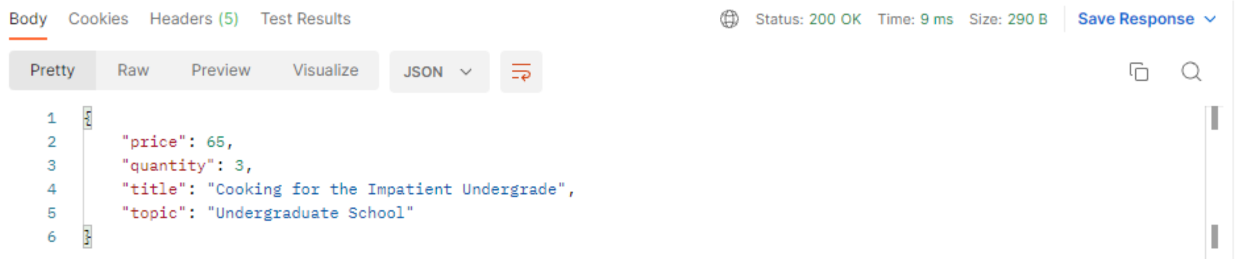


```
############################################################
Request from catalog server with index=0, ip=10.0.0.16
############################################################

Lookup Cache Obj:
Cache:  {4: {'title': 'Cooking for the Impatient Undergrade', 'topic': 'Undergraduate School', 'quantity': 3, 'price': 65}}
LRU_Queue:  [4]

10.0.0.6 - - [09/Aug/2022 17:10:54] "GET /info/4 HTTP/1.1" 200 -
```

Request the same object, will fetch from cache with less response time:

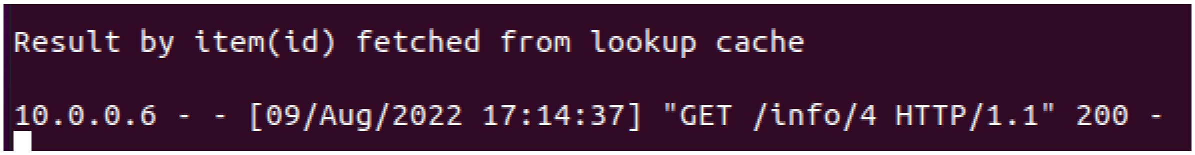


Status: 200 OK Time: 9 ms Size: 290 B

```
{
    "price": 65,
    "quantity": 3,
    "title": "Cooking for the Impatient Undergrade",
    "topic": "Undergraduate School"
}
```

We can see the time = 9ms, which is a large difference!!

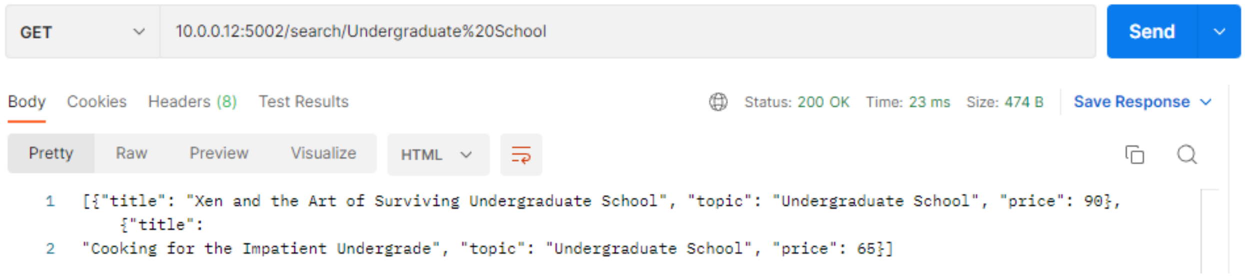
From frontend server's console:



```
Result by item(id) fetched from lookup cache

10.0.0.6 - - [09/Aug/2022 17:14:37] "GET /info/4 HTTP/1.1" 200 -
```

Now we will fetch all books with topic "Undergraduate%20School", the request should be done from the 2nd replica with IP address: 10.0.0.9:5003:

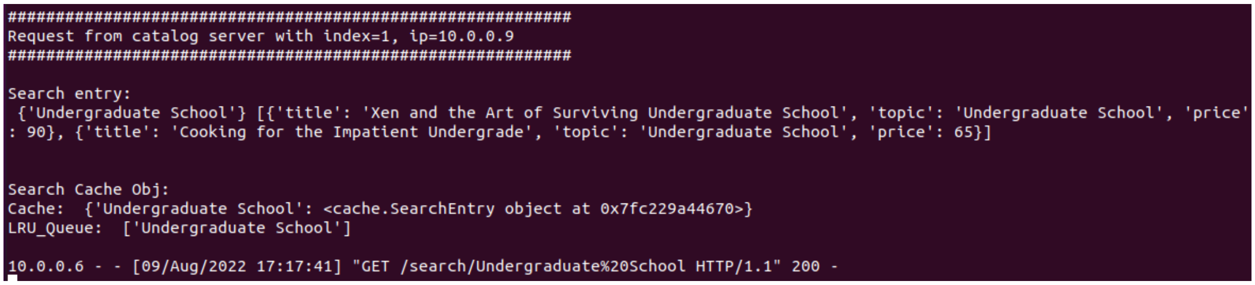


GET 10.0.0.12:5002/search/Undergraduate%20School

Status: 200 OK Time: 23 ms Size: 474 B

```
[{"title": "Xen and the Art of Surviving Undergraduate School", "topic": "Undergraduate School", "price": 90},
    {"title":
"Cooking for the Impatient Undergrade", "topic": "Undergraduate School", "price": 65}]
```

Response time is 23ms. From console of frontend server:



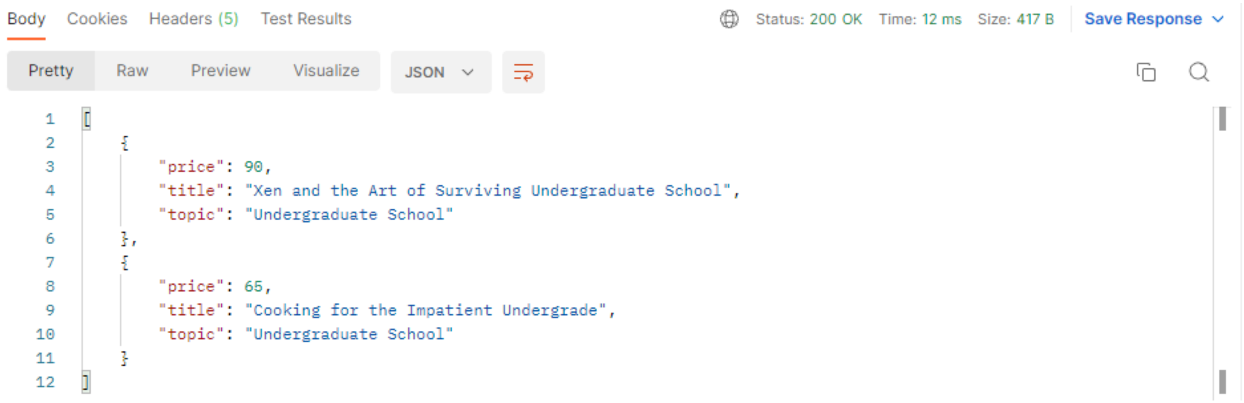
```
############################################################
Request from catalog server with index=1, ip=10.0.0.9
############################################################

Search entry:
 {'Undergraduate School'} [{'title': 'Xen and the Art of Surviving Undergraduate School', 'topic': 'Undergraduate School', 'price'
: 90}, {'title': 'Cooking for the Impatient Undergrade', 'topic': 'Undergraduate School', 'price': 65}]


Search Cache Obj:
Cache:  {'Undergraduate School': <cache.SearchEntry object at 0x7fc229a44670>}
LRU_Queue:  ['Undergraduate School']

10.0.0.6 - - [09/Aug/2022 17:17:41] "GET /search/Undergraduate%20School HTTP/1.1" 200 -
```

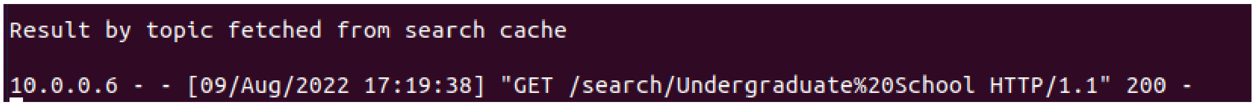
Retry the same request:



Status: 200 OK Time: 12 ms Size: 417 B

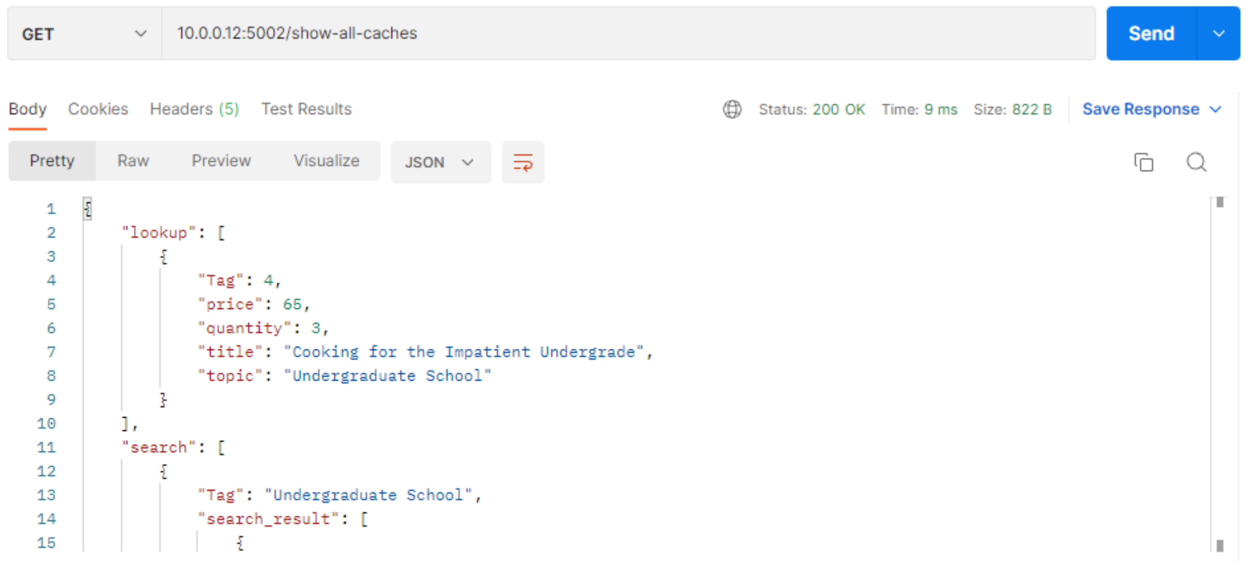
```
[
    {
        "price": 90,
        "title": "Xen and the Art of Surviving Undergraduate School",
        "topic": "Undergraduate School"
    },
    {
        "price": 65,
        "title": "Cooking for the Impatient Undergrade",
        "topic": "Undergraduate School"
    }
]
```

Response time = 12ms!!



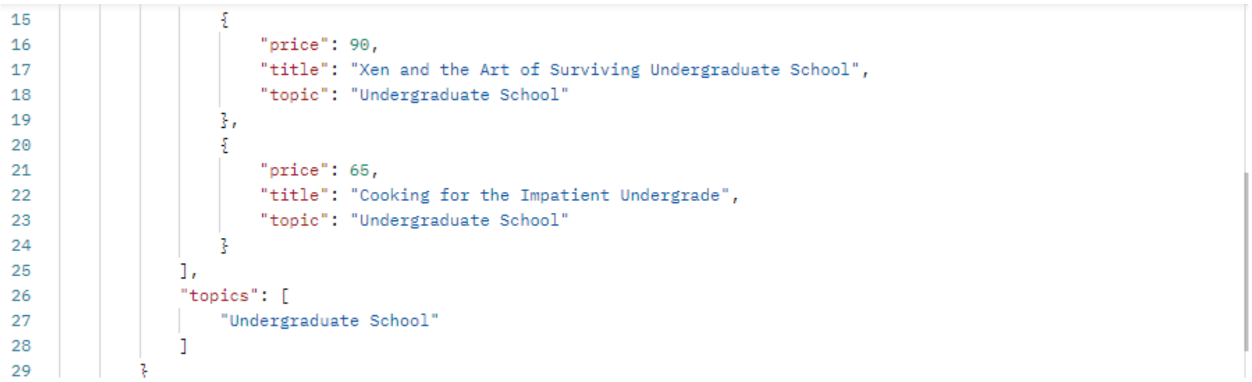
```
Result by topic fetched from search cache
10.0.0.6 - - [09/Aug/2022 17:19:38] "GET /search/Undergraduate%20School HTTP/1.1" 200 -
```

Now we will show you the cache, we build an API to see the content of the cache:



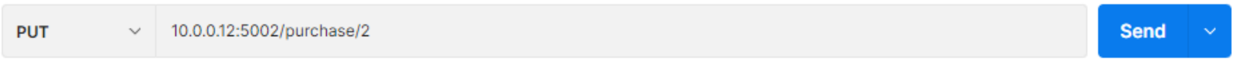
GET 10.0.0.12:5002/show-all-caches

Status: 200 OK Time: 9 ms Size: 822 B



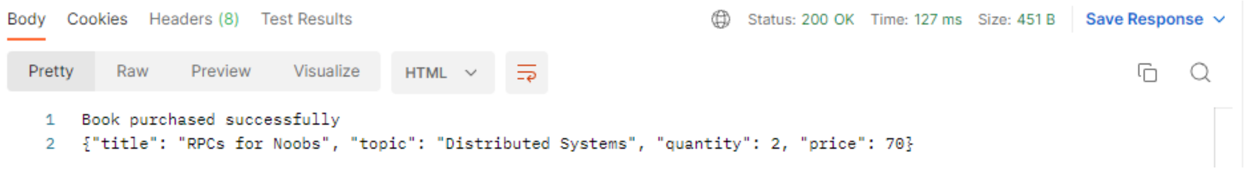
```
{
    "lookup": [
        {
            "Tag": 4,
            "price": 65,
            "quantity": 3,
            "title": "Cooking for the Impatient Undergrade",
            "topic": "Undergraduate School"
        }
    ],
    "search": [
        {
            "Tag": "Undergraduate School",
            "search_result": [
                {
                    "price": 90,
                    "title": "Xen and the Art of Surviving Undergraduate School",
                    "topic": "Undergraduate School"
                },
                {
                    "price": 65,
                    "title": "Cooking for the Impatient Undergrade",
                    "topic": "Undergraduate School"
                }
            ],
            "topics": [
                "Undergraduate School"
            ]
        }
```

Now we will do a purchase operation, we have in two DBs all books with quantity = 3. We are going to buy a book with id = 2. So, let's see:



```
10.0.0.12:5002/purchase/2
```

PUT method, the result and response time:

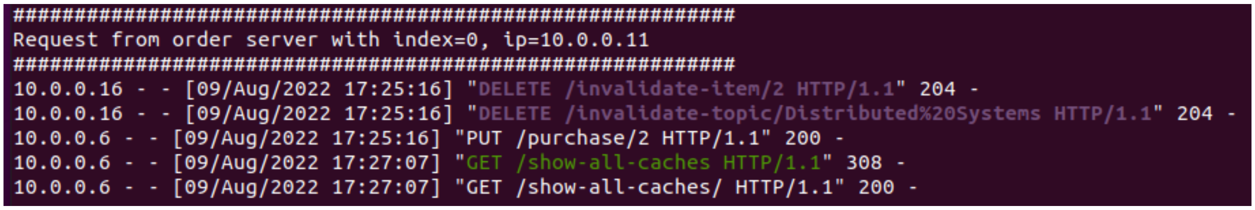


```
Book purchased successfully
{"title": "RPCs for Noobs", "topic": "Distributed Systems", "quantity": 2, "price": 70}
```

Response time = 127!! Relatively High!!

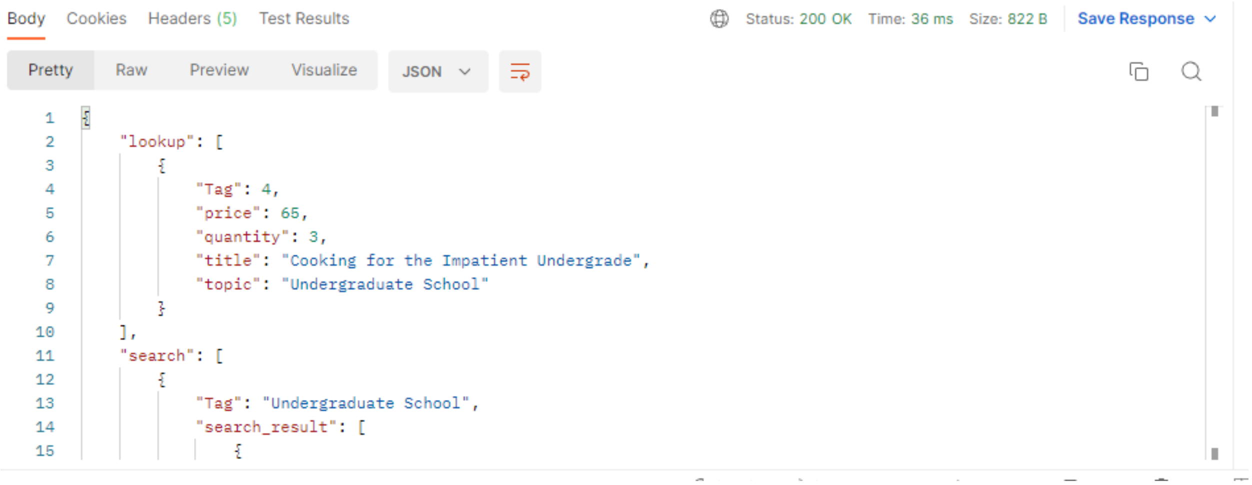
The cache will not affect, because the book with id = 2 is not in cache before!

Console (front-end):



```
############################################################
Request from order server with index=0, ip=10.0.0.11
############################################################
10.0.0.16 - - [09/Aug/2022 17:25:16] "DELETE /invalidate-item/2 HTTP/1.1" 204 -
10.0.0.16 - - [09/Aug/2022 17:25:16] "DELETE /invalidate-topic/Distributed%20Systems HTTP/1.1" 204 -
10.0.0.6 - - [09/Aug/2022 17:25:16] "PUT /purchase/2 HTTP/1.1" 200 -
10.0.0.6 - - [09/Aug/2022 17:27:07] "GET /show-all-caches HTTP/1.1" 308 -
10.0.0.6 - - [09/Aug/2022 17:27:07] "GET /show-all-caches/ HTTP/1.1" 200 -
```

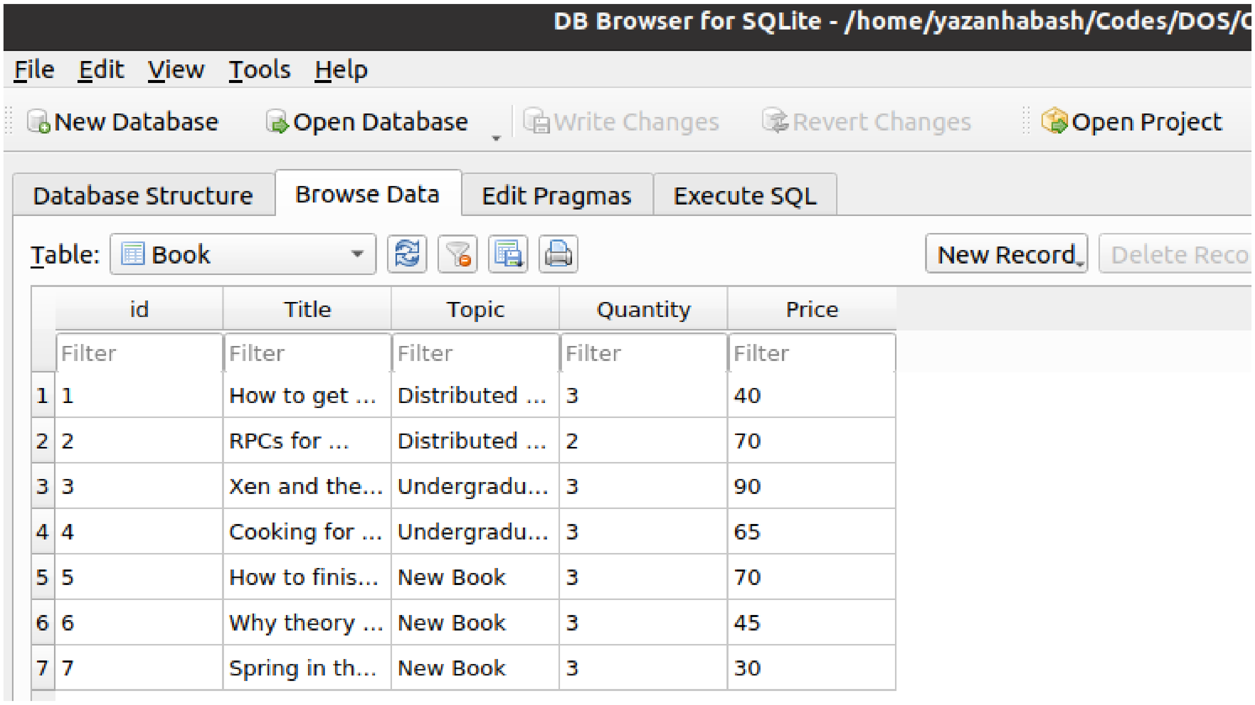
We can see the steps of requests: invalidation first, purchase to order server (first replica: 10.0.0.11) then we did see-all-caches (cache not affected) as above:



```
{
    "lookup": [
        {
            "Tag": 4,
            "price": 65,
            "quantity": 3,
            "title": "Cooking for the Impatient Undergrade",
            "topic": "Undergraduate School"
        }
    ],
    "search": [
        {
            "Tag": "Undergraduate School",
            "search_result": [
                {
```

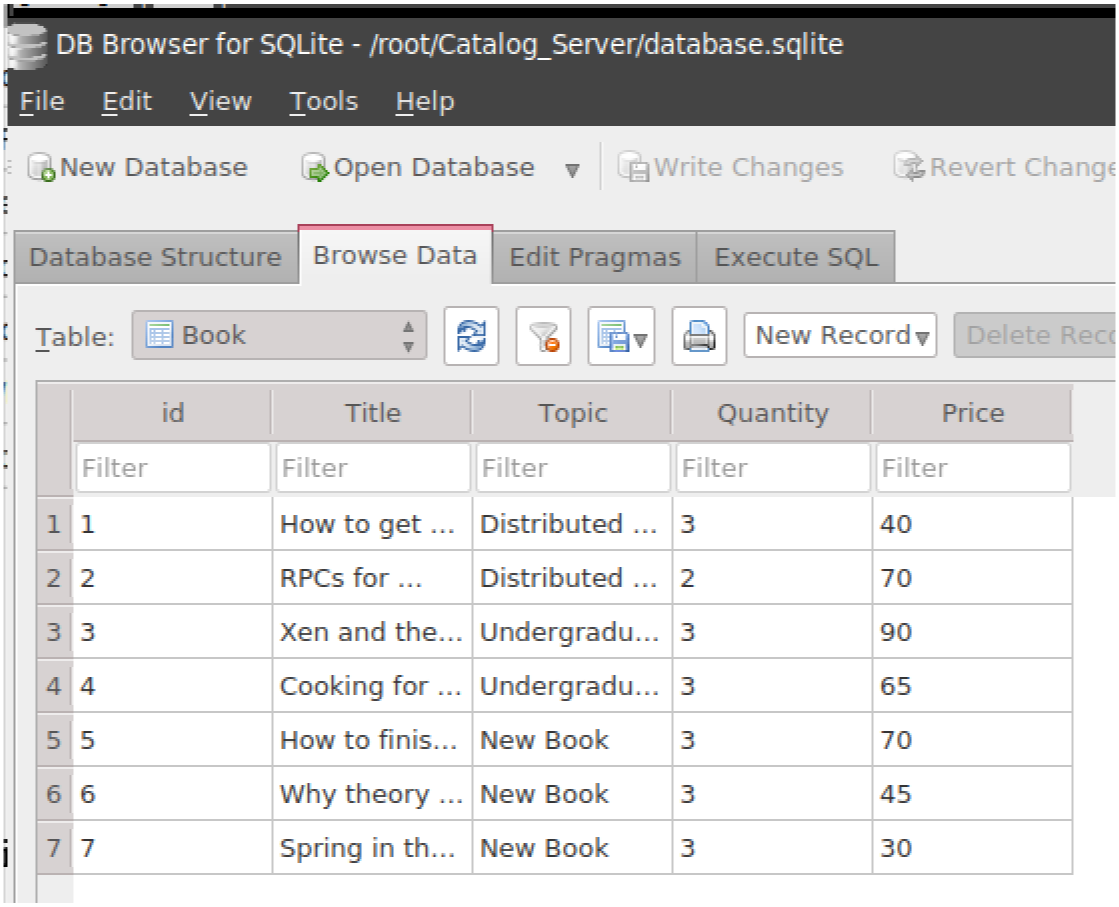
See the databases, the values become 2 instead of 3:

First replica (Ubuntu 20.04):



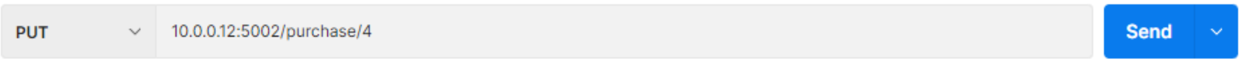
	id	Title	Topic	Quantity	Price
	Filter	Filter	Filter	Filter	Filter
1	1	How to get ...	Distributed ...	3	40
2	2	RPCs for ...	Distributed ...	2	70
3	3	Xen and the...	Undergradu...	3	90
4	4	Cooking for ...	Undergradu...	3	65
5	5	How to finis...	New Book	3	70
6	6	Why theory ...	New Book	3	45
7	7	Spring in th...	New Book	3	30

Second replica (Puppy Linux):



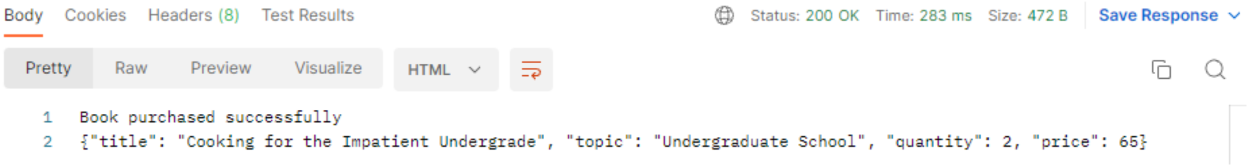
	id	Title	Topic	Quantity	Price
	Filter	Filter	Filter	Filter	Filter
1	1	How to get ...	Distributed ...	3	40
2	2	RPCs for ...	Distributed ...	2	70
3	3	Xen and the...	Undergradu...	3	90
4	4	Cooking for ...	Undergradu...	3	65
5	5	How to finis...	New Book	3	70
6	6	Why theory ...	New Book	3	45
7	7	Spring in th...	New Book	3	30

Purchase book with id = 4 (exists in cache):



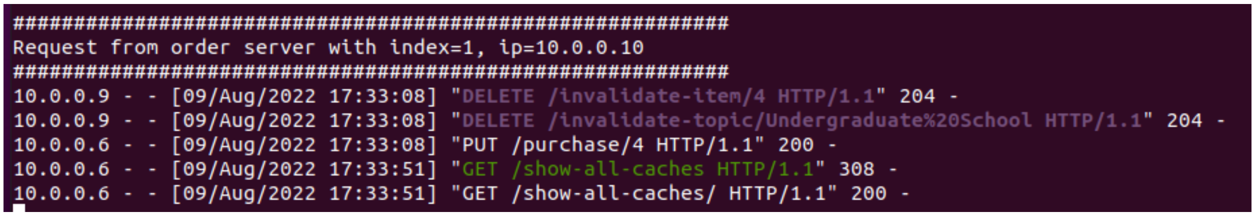
```
10.0.0.12:5002/purchase/4
```

Result:



```
Book purchased successfully
{"title": "Cooking for the Impatient Undergrade", "topic": "Undergraduate School", "quantity": 2, "price": 65}
```

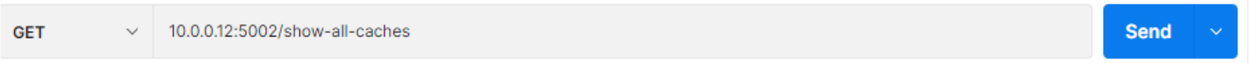
Console of front-end:



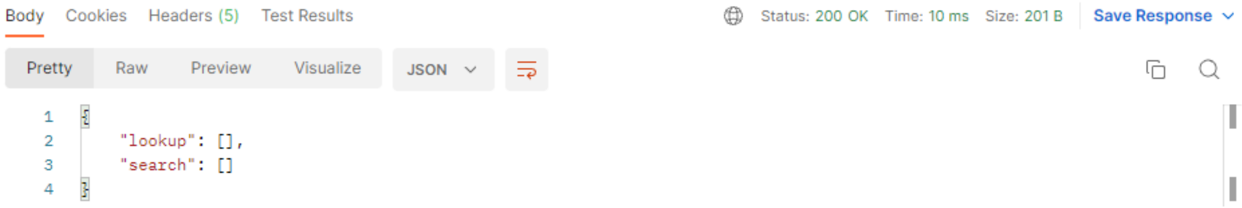
```
############################################################
Request from order server with index=1, ip=10.0.0.10
############################################################
10.0.0.9 - - [09/Aug/2022 17:33:08] "DELETE /invalidate-item/4 HTTP/1.1" 204 -
10.0.0.9 - - [09/Aug/2022 17:33:08] "DELETE /invalidate-topic/Undergraduate%20School HTTP/1.1" 204 -
10.0.0.6 - - [09/Aug/2022 17:33:08] "PUT /purchase/4 HTTP/1.1" 200 -
10.0.0.6 - - [09/Aug/2022 17:33:51] "GET /show-all-caches HTTP/1.1" 308 -
10.0.0.6 - - [09/Aug/2022 17:33:51] "GET /show-all-caches/ HTTP/1.1" 200 -
```

We can see the steps of requests: invalidation first, purchase to order server (second replica: 10.0.0.9) then we did see-all-caches (cache affected, becomes empty):

The caches will be empty due to invalidate messages from catalog replicas:



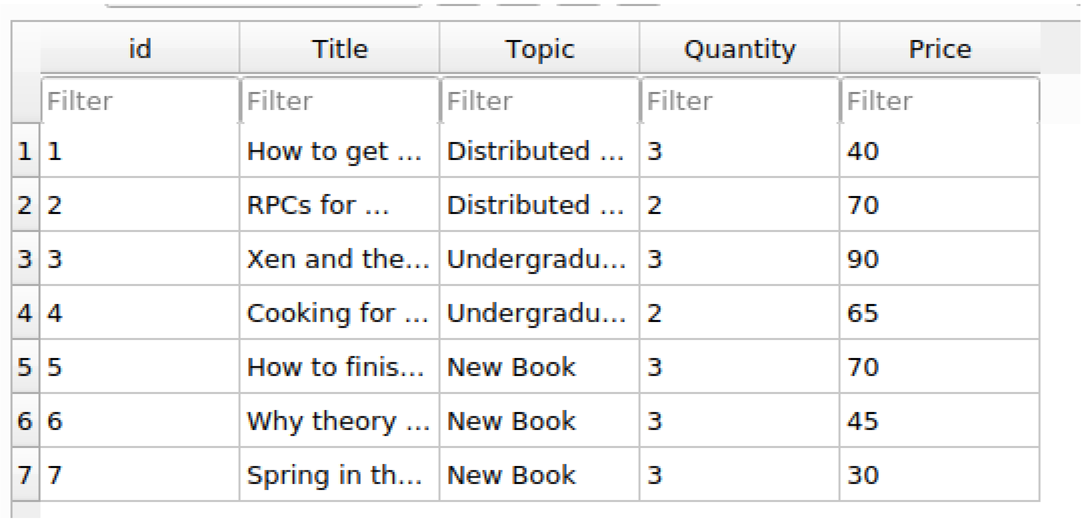
```
10.0.0.12:5002/show-all-caches
```



```
{
    "lookup": [],
    "search": []
}
```

See DBs:

Ubuntu (first replica), book id = 4, quantity becomes 2:



	id	Title	Topic	Quantity	Price
	Filter	Filter	Filter	Filter	Filter
1	1	How to get …	Distributed …	3	40
2	2	RPCs for …	Distributed …	2	70
3	3	Xen and the…	Undergradu…	3	90
4	4	Cooking for …	Undergradu…	2	65
5	5	How to finis…	New Book	3	70
6	6	Why theory …	New Book	3	45
7	7	Spring in th…	New Book	3	30

Second replica (Puppy Linux), as the same as above:

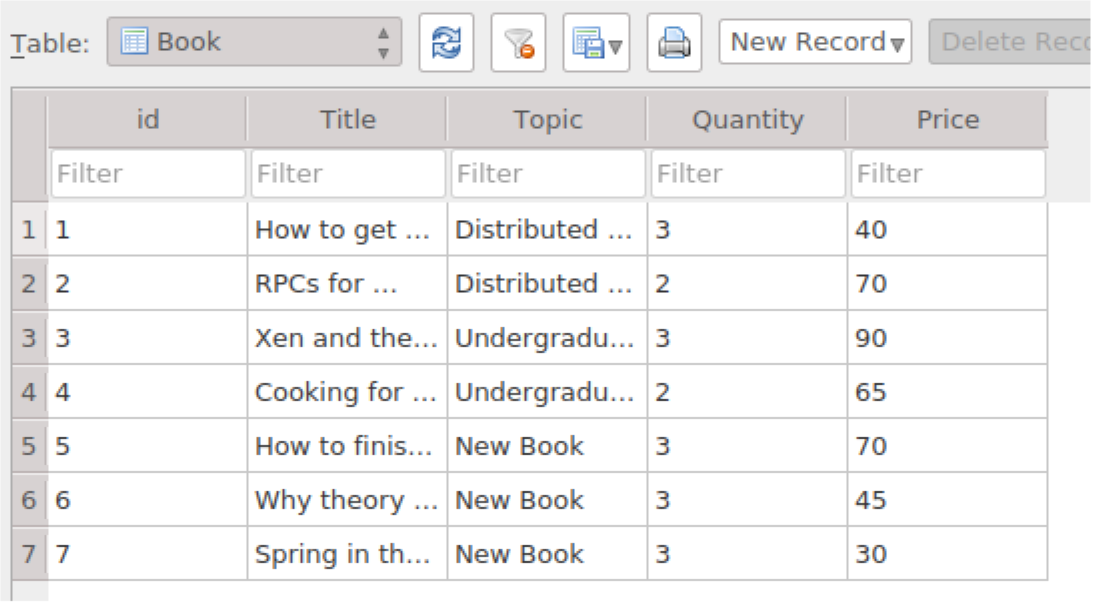


Table: Book

	id	Title	Topic	Quantity	Price
	Filter	Filter	Filter	Filter	Filter
1	1	How to get …	Distributed …	3	40
2	2	RPCs for …	Distributed …	2	70
3	3	Xen and the…	Undergradu…	3	90
4	4	Cooking for …	Undergradu…	2	65
5	5	How to finis…	New Book	3	70
6	6	Why theory …	New Book	3	45
7	7	Spring in th…	New Book	3	30

Yazan Habash & Ashraf Hab-Rumman