eBay Search Service

Isabel Giang and Maxwell Wenger

CSS490 Group Y4

January 26, 2021

Contents

Overview

Problem Statement

In the last few months, eBay users have reported increasingly slow response times from eBay's website when searching for auctions. Application log analysis for the eBay Master Service has confirmed that our FindAuctions API for the eBay Master service takes significantly longer to return a response when we search for auctions using keywords. It also takes longer to search for a large number of auctions. This is negatively impacting

eBay's user experience for existing users and is hindering the website's chances of being adopted by new users.

Performing further analysis on the eBay Master service schema shows that the FindAuctions API must scan the entire table to first find active auctions. Then it must scan those active auctions to find auctions that have titles with the keywords we are looking for. To

solve this problem, we have created an additional search database that contains only the necessary auction data required to for search functionality. We have then indexed the database by both keyword and category. This speeds up the FindAuctions API enough

to fix the poor user experience temporarily, but this will not be enough as the company grows. The number of records in the Auction table, and subsequently, the number of active auctions will increase at an exponential rate.

Solution

Current Solution

To partially address this problem, we have added an AuctionStatus index to the Auction table so we can immediately access all active auctions instead of search for all auctions every time we query with keywords.

This speeds up the FindAuctions API enough to fix the poor user experience temporarily, but this will not be enough as the company grows. The number of records in the Auction table, and subsequently, the number of active auctions will increase at an exponential rate.

Next Steps

To solve this, we want to create a separate search service as another step towards eBay's move to a microservice oriented architecture that will handle searching for auctions by keyword and/or category.

We will call this new service 'AuctionSearch'. This search service will have its own database that will only be externally accessible via its API.

Whenever a new FindAuction API request is made, this service will perform the required business logic and databases accesses instead of the eBay Master service.

eBay Search Service API Specification

createSearchableAuction

This creates an entry in the search database for the auction, all of the auction's keywords, and the auction category.

Input

```
{
    "auctionKey": <string>,
    "title": <string>,
    "closeDate": <string>,
    "status": <string>,
    "breadcrumb": <string>
}
```

Output

Scenario	Response
Successfully created search entry.	`{ "success": true }`
TODO: Make error states	Yeah! What he said!

updateSearchableAuction

Update searchable auction will update fields, keywords, and categories of an existing auction in the search table. New keywords will be generated if a new title is provided. The new keywords will be added to the database, and keywords associated with they auction key but were not again generated from the updated title will be removed from the database. The fields will be based on the auction key, therefore the auction key may not be updated. Any optional fields not included in the message to the service will not be changed.

input

```
{
    "auctionkey": <string>,
    "title": <string>, // optional
    "closedate": <string>, // optional
    "status": <string>, // optional
    "breadcrumb": <string> // optional
}
```

output

scenario	response
successfully updated search entry.	`{ "success": true }`
provided auction key does not exist in the search database.	`{ "success": false, "exception": "auctionkeynotfound" }`
todo: make error states	yeah! what he said!

logging

```
{
    "start": <string>,
    "duration": <string>,
    "api": <string>,
    "params":
        {
            "original": {
                "auctionkey": <string>,
                "title": <string>, // optional
                "closedate": <string>, // optional
                "status": <string>, // optional
                "breadcrumb": <string> // optional
            },
            "updated": {
                "auctionkey": <string>,
                "title": <string>, // optional
                "closedate": <string>, // optional
                "status": <string>, // optional
                "breadcrumb": <string> // optional
            }
        }
}
```

findAuctions

findAuctions will return a number of results based on the user's query. The parameters search by "anding" the results together (e.g. if you ask for a keyword "red", "LG" with the category "ELE:PHO" signifying phones in electronics, red LG phones will be returned, but not red Nokia phones, or red LG refrigerators). To perform searches that "or" search terms, multiple searches must be made. lastAuctionKey and numResults are to be used for pagination. findAuctions will return numResults number of entries ordered from oldest to newest, starting from the lastAuctionKey. If no lastAuctionKey is provided, findAuctions will return numResults number of results starting from the first result.

input

```
{
    "keywords": [ <string>, ...], // optional
    "status": <string>, // optional
    "breadcrumb": <string>, // optional
    "lastAuctionKey": <string>, // optional
    "numresults": <string>
}
```

findAuctions will return a number of results based on the user's query. The parameters search by "anding" the results together (e.g. if you ask for a keyword "red", "LG" with the category "ELE:PHO" signifying phones in electronics, red LG phones will be returned, but not red Nokia phones, or red LG refrigerators). To perform searches that "or" search terms, multiple searches must be made. lastAuctionKey and numResults are to be used for pagination. findAuctions will return numResults number of entries ordered from oldest to newest, starting from the lastAuctionKey. If no lastAuctionKey is provided, findAuctions will return numResults number of results starting from the first result.

input

```
{
    "keywords": [ <string>, ...], // optional
    "status": <string>, // optional
    "breadcrumb": <string>, // optional
    "lastAuctionKey": <string>, // optional
    "numresults": <string>
}
```

output

scenario	response
Successfully search with found results.	`{ "success": true, "results": [ auctionKey <string>, ...] }`
Successfully search with no results.	`{ "success": true, "results": [ ] }`
todo: make error states	yeah! what he said!

output

scenario	response

Successfully search with found results.	`{`
	`    "success": true,`
	`    "results": [ auctionKey <string>, ...]`
	`}`

Successfully search with no results.	`{`
	`    "success": true,`
	`    "results": [ ]`
	`}`

todo: make error states	yeah! what he said!

eBay Search Service Intervals

Changes to eBay Master Service

createAuction

Creates an auction as it did before, although now the createAuction service is responsible for calling createSearchableAuction to create a search entry for the auction.
The impact on the database from createAuction is documented in the master api documentation. The impact on the database from createSearchableAuction is documented in section createSearchableAuction.
If three calls to the search service fails, the call will be queued up to be attempted again with other failed calls at a regular interval.

updateAuction

Update auction will work as it did before. Except, now, update auction will call updateSearchableAuction with the updated parameters it will receive.
The impact on the database from updateAuction is documented in the master api documentation. The impact on the database from updateSearchableAuction is documented in section updateSearchableAuction.
If three calls to the search service fails, the call will be queued up to be attempted again with other failed calls at a regular interval.