eBay Search Service

Isabel Giang and Maxwell Wenger

CSS490 Group Y4

January 26, 2021

Contents

Overview

This document is intended to explain why the eBay Search Service is being created and a suggested approach to implement the eBay Search Service for the purposes of being audited for technical feasibility by an engineer.

This document will explain the following:

- the API designs for each eBay Search Service API
- details needed for implementation and to assess if this solution is technically viable
- changes that must be made to the eBay Master Service to use the eBay Search Service

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.

Temporary Solution

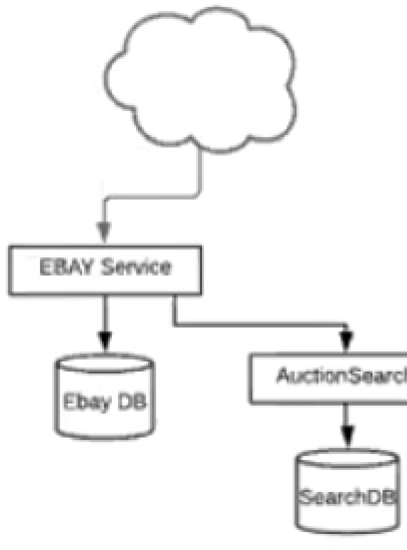
As a bandaid solution to temporarily address this problem, we have added an AuctionStatus index to the Auction table of the eBay Master Service database. This allows us to immediately access all active auctions instead of being forced to scan the entire Auction table to find active auctions before querying 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 more permanently, we want to create a separate search service that will handle searching for auctions by keyword and/or category. We will call this new service the eBay Search Service.

The search service will have its own database that can only be changed by external users via its API. We will *delegate* the eBay Master Service's searching to this new Search Service, which will allow us to only keep information relevant to searching in the Search Service's database.



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.

At a minimum, the eBay Search Service should support searching in the following ways:

- Searching for all auctions
- Searching for auctions based on auction status (closed, open, pending, cancelled)
- Searching for auctions based on keywords
- Searching for auctions based on category
- Searching for auctions based on keywords and category

Each search operation will only allow "AND" operations between search terms. For example, searching for auctions that have"red" AND "yellow" in their titles.

"OR" operations must be done by forming a union of the results from multiple calls. For example, to get the results for auctions that have "red" OR "yellow" in their titles, you must combine the results of searching for auctions that have "red" in their titles, and the results of searching for auctions that have "yellow" in their titles.

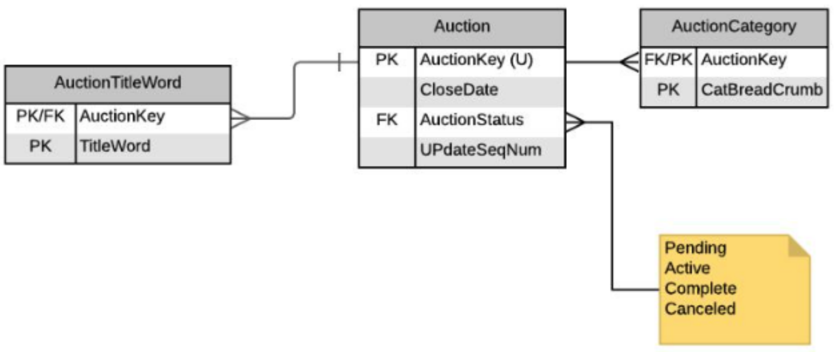
eBay Search Service API Specification

API List

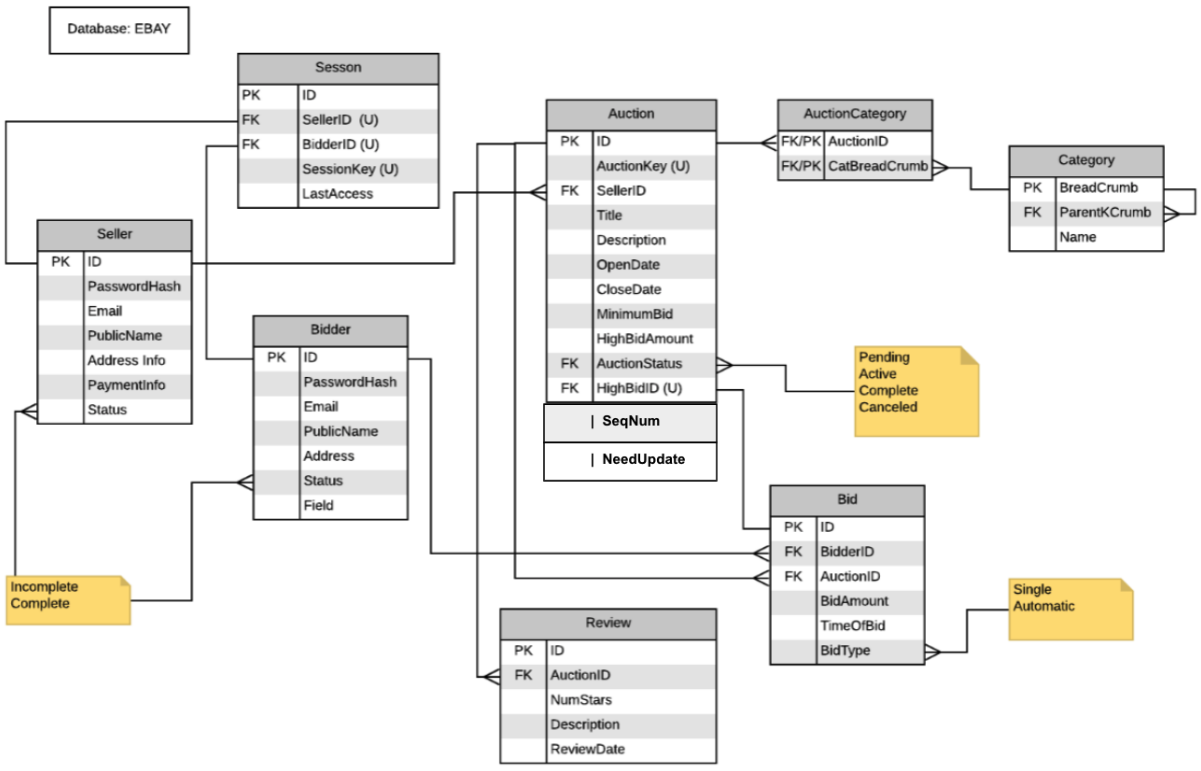
API Name	Description
saveSearchableAuction	Updates or creates a searchable auction
findAuctions	Finds auctions based on the given keywords and/or category.

Schemas

SearchDB



MasterDB


Database: EBAY
Sesson
PK ID
FK SellerID (U)
FK BidderID (U)
SessionKey (U)
LastAccess
Seller
PK ID
PasswordHash
Email
PublicName
Address Info
PaymentInfo
Status
Bidder
PK ID
PasswordHash
Email
PublicName
Address
Status
Field
Incomplete
Complete
Auction
PK ID
AuctionKey (U)
FK SellerID
Title
Description
OpenDate
CloseDate
MinimumBid
HighBidAmount
FK AuctionStatus
FK HighBidID (U)
SeqNum
NeedUpdate
AuctionCategory
FK/PK AuctionID
FK/PK CatBreadCrumb
Category
PK BreadCrumb
FK ParentKCrumb
Name
Pending
Active
Complete
Canceled
Bid
PK ID
FK BidderID
FK AuctionID
BidAmount
TimeOfBid
BidType
Single
Automatic
Review
PK ID
FK AuctionID
NumStars
Description
ReviewDate

API Descriptions

saveSearchableAuction

Creates an auction if no auction that corresponds with the given auction key exists. Otherwise, updates the auction that corresponds with the given auction key.

Input

- **auctionKey** - Public key that uniquely identifies an auction from the eBay Master Service.
- **title** - Title of the auction
- **closeDate** - Closing date of the auction
- **categories** - List of category breadcrumbs
- **status** - Status of the auction (closed, open, pending, cancelled)
- **seqNum** - Sequence number of the auction record. Each time an update is performed for this record, it is incremented by one. Requests made with `seqNum` < `UpdateSeqNum` for this record are ignored.

```
{
    "auctionKey": <string>,
    "title": <string>,
    "closeDate": <string>,
    "categories": [ <string>, ...],
    "status": <string>,
    "seqNum": <int>
}
```

Output

Scenario	Response
Successfully stored an auction	`{ "success": true, "seqNum": <int> }`
Auction key does not match with an existing auction	`{ "success": false, "error": "InvalidAuction" }`
`SeqNum` < `UpdateSeqNum` for this auction.	`{ "success": false, "error": "OutOfOrder" }`

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** - Keywords that auction results will be matched to
- **category** - Category breadcrumb
- **status** - Status of the auction (closed, open, pending, cancelled)
- **lastAuctionKey** - The last auction key will be last auction not included in the results that are numResults long. If given, results will be returned starting from this page.
- **numResults** - Number of auctions returned per page

```
{
    "keywords": [ <string>, ...], // optional
    "status": <string>, // optional
    "categories": [<string>, ...], // optional
    "lastAuctionKey": <number>, // optional
    "numResults": <number>
}
```

Output

scenario	response

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

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

eBay Search Service Internals

This section describes the suggested internal processes to successfully implement each API with a focus on changes in the database layer of the eBay Search Service.

Initial Creation

When the eBay Search Service is initially created, the auctions from the eBay Master Service will need to have its auction records from the MasterDB be loaded into the SearchDB. This can be performed using calls of `saveSearchableAuction`.

saveSearchableAuction

Each new record and record update performed through this API is done as a transaction. If no auction key is given, a record cannot be created, and the transaction is immediately cancelled.

If an auction key is given and the auction key *does* match to a record in the SearchDB's auction table, the API will only update an auction if the given `seqNum` is greater than `UpdateSeqNum`. If it does, `UpdateSeqNum` will be set to `SeqNum` Otherwise, we ignore the request and return an `OutOfOrder` error response.
If the auction key *does not* match to a record in the SearchDB's auction table, we create a new auction record with `UpdateSeqNum` equal to 0.

To create a new auction, first, keywords are generated from the given `title`. Keywords are generated by splitting the title by whitespace.
Ex: "Red Beanie Baby" → [“Red”, “Beanie”, “Baby”]

- New keywords will only be generated and updated if a new title is provided.
- If new keywords are generated, the previously generated keywords that match with the given auction key that are not duplicates of keywords generated from the new title will be deleted
- If an empty string is given as the title (“”), all keywords will be deleted, but no new keywords will be generated.

Each keyword is added as a record in the `AuctionTitleWord` table, matched with the given `AuctionKey`.

Next, categories are processed from the given list of category breadcrumbs, `categories`. All previous categories that match with the given auction key will be removed, in case the categories have changed.

Each category is added as a record in the `AuctionCategory` table, matched with the given `AuctionKey`.

Finally, `AuctionStatus`, `CloseDate`, and `UpdateSeqNum`, and `AuctionKey` are created as a record in the `Auction` table and committed as a transaction. Each value overwrites the previous value. If any non-required parameters are not given, their values in the SearchDB will not change.

findAuctions

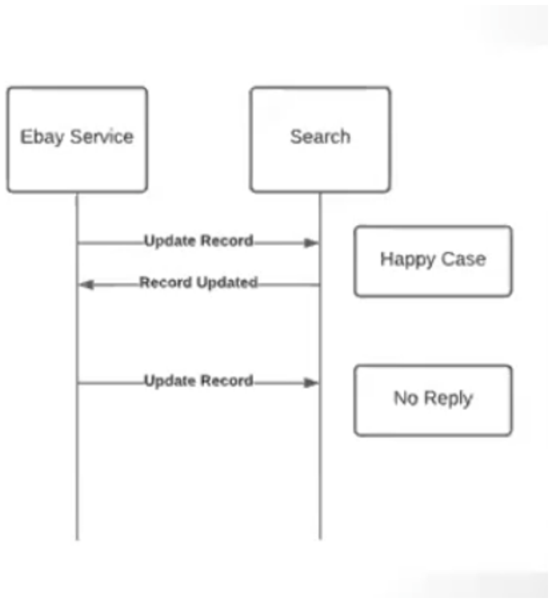
Find auctions builds a query based on the provided information. It generates this query as an “and” operation, so results passed back are only those that fit all the criteria you passed to findAuction. It handles pagination by having the user keep track of the last auction it saw, and request a certain number after that. So from our ordered query, we will pull numResults of results back, starting from one after the last auction key. If the last auction key was deleted/does not exist at the time of the call, it will return numResults number of results starting from the first value again. This helps get around the issue of missed auctions that were added during the search session.

Changes to eBay Master Service

MasterDB Schema Changes

NeedUpdate Field

The `NeedUpdate` field has been added to the `Auction` record to handle dropped update messages from the eBay Search Service. In the case where there's no reply from the eBay Search Service, there must be a way to guarantee that we will try again until the eBay Search Service has also been updated.



If this field is `true`, we will say this means the eBay Master Service is in the process of updating this record, and has yet to receive a message from the eBay Search Service indicating it has been updated on the SearchDB.

We suggest created some kind of mechanism, such a CRON job, to run periodically to check if there are any auctions with this flag set to `true`. If they are `true`, send a message to the eBay Search Service asking again to update this record on the SearchDB table.

SeqNum Field

The `SeqNum` field has been added to the `Auction` record to help track the order of update messages, so we know which update message each record is waiting for.

If a record has `SeqNum` equal to 2, then we know it is waiting for the update message from the eBay Search Service with sequence number 2.

API Definition Changes

createAuction

The `createAuction` API will need to adopt the responsibility of sending messages to the eBay Search Service's `saveSearchableAuction` API to populate the SearchDB as new auctions are created.

For each new auction, `createAuction` will:

1. begin a new transaction
2. create a record in the MasterDB's Auction table
3. set `SeqNum` to 0
4. set `NeedUpdate` to `true`
5. call eBay Search Service's `saveSearchableAuction` API to trigger a transaction to create a new auction in the SearchDB.
6. commit the transaction

The eBay Master Service will then wait to receive a response from the eBay Search Service. If a successful response with `"success":"true"` is returned, `createAuction` should:

1. begin a new transaction
2. set `NeedUpdate` to `false`
3. increment `SeqNum` by 1
4. commit the transaction

If no response is returned after an appropriate timeout period, `createAuction` should return a successful response to the `createAuction user` and do nothing.

updateAuction

The `updateAuction` API will need to adopt the responsibility of sending messages to the eBay Search Service's `saveSearchableAuction` API whenever an auction is updated with new information.
For each new auction update, `updateAuction` will:

1. begin a new transaction
2. update a record in the MasterDB's Auction table
3. set `NeedUpdate` to `true`
4. call eBay Search Service's `saveSearchableAuction` API to trigger a transaction to update an auction in the SearchDB
5. commit the transaction

The eBay Master Service will then wait to receive a response from the eBay Search Service. If a successful response with `"success":"true"` is returned, `updateAuction` should:

1. begin a new transaction
2. set `NeedUpdate` to `false`
3. increment `SeqNum` by 1
4. commit the transaction

If no response is returned after an appropriate timeout period, `updateAuction` should return a successful response to the `updateAuction user` and do nothing.

findOpenAuctions

This API from the eBay Master Service calls the eBay Search Service's `findAuctions`. The parameters passed to the eBay Search Service's `findAuctions` API will be the same as the parameters passed to the eBay Master Service.

After waiting for a response, the eBay Master Service should receive a list of auction keys in the form of JSON. To return the same information that is expected from the API specification for `findOpenAuctions`, for each auction key, the eBay Master Service will need to perform a `JOIN` with the MasterDB Seller table, filtering on `AuctionKey` to get `Title`, `Description`, `OpenDate`, `ClosingDate`, and `HighBid`.

NOTE: For `updateAuction` and `createAuction`, assume that sending update messages after timeouts will be handled by some mechanism that searches for records where `NeedUpdate` is `true`. *See NeedUpdate Field for more information.*

Logging

This section describes a suggested format for this service's application logs.

```
{
    "start": <string>,
    "duration": <string>,
    "api": <string>,
    "params":
        {
            "original": {
                "auctionkey": <string>,
                "title": <string>,
                "closeDate": <string>,
                "status": <string>,
                "category": [<string>, ...],
                "seqNum": <int>
            },
            "updated": {
                "auctionkey": <string>,
                "title": <string>,
                "closeDate": <string>,
                "status": <string>,
                "category": [<string>, ...],
                "seqNum": <int>
            }
        }
}
```

- **start** The time the service first receives the call.
- **duration** The amount of time it takes from when the service receives the call to when the service responds.
- **api** The name of the API that is called.
- **params** The original and updated data that the API changed. Optional data that was not changed is not included in either the updated or original logs.