

Distributed Databases for Amazon Products

Erkang Chen, Haoming Guo, Mingji Xi

Viterbi School of Engineering, University of Southern California

Introduction

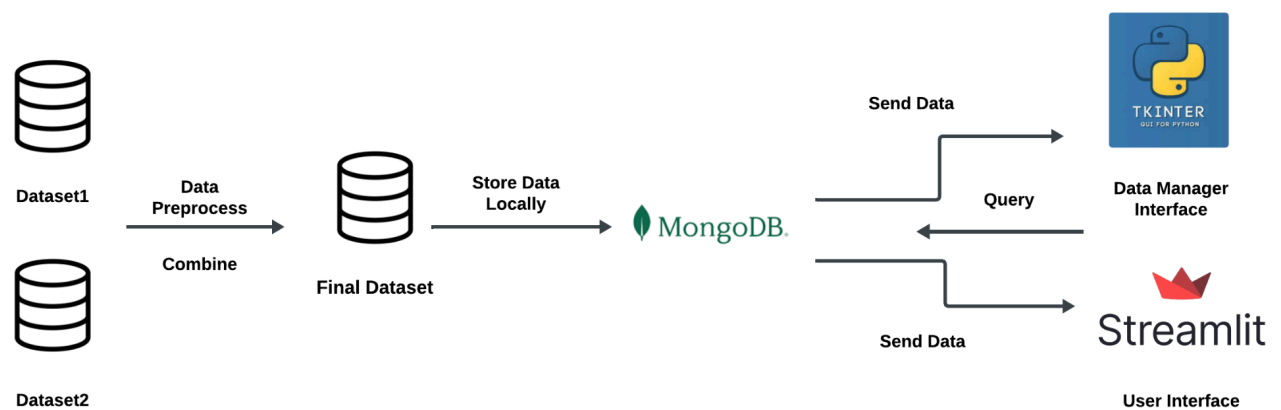
Transforming the landscape of e-commerce, our project harnesses MongoDB's power to architect a distributed database system tailored for Amazon products. Utilizing hash functions for data distribution across MongoDB nodes, we ensure optimal performance and scalability. With a focus on comprehensive CRUD operations through our data manager interface, and an intuitive user interface powered by Tkinter and Streamlit, we offer both managers and customers seamless control and navigation. This project represents a leap forward in e-commerce data management, promising enhanced efficiency, scalability, and user experience.

Planned Implementation

Through careful design, we have crafted a data schema that encompasses essential product attributes, including 'asin' (unique product ID), 'title', 'feature', 'price', 'imageURL', 'rank', and 'brand'. Central to our distribution strategy is the utilization of the 'asin' as the primary key, enabling us to implement sharding across our database effectively. By employing hashing techniques on the 'asin', we ensure the even distribution of product data across three primary database nodes, thereby achieving balanced load distribution and optimizing query performance. Additionally, we have allocated a dedicated database node specifically for backup and recovery processes. This strategic allocation enhances data durability and fortifies our system with a robust failover mechanism, essential for maintaining uninterrupted availability and safeguarding business continuity. Our database manager interface empowers administrators with comprehensive CRUD (Create, Read, Update, Delete) operations, affording them control over

the data management process. On the front-end, our user interface for the website is meticulously designed to deliver a seamless and intuitive customer experience. Equipped with features such as advanced search functionality, customizable filters including price sorting, and the ability to search by brand and product name, our interface enables customers to effortlessly navigate through product catalogs and locate their desired items with ease and efficiency.

Architecture Design



This flowchart illustrates the process of our distributed databases. The initial two datasets undergo preprocessing and are then combined into a final dataset. This final dataset is stored locally in MongoDB, a NoSQL database, for further operations. Data can be queried from MongoDB and sent to the Data Manager Interface built with Tkinter, a GUI toolkit for Python. Additionally, the data can be sent to a user interface created using Streamlit, which is used for building web applications quickly.

Implementation

Functionalities

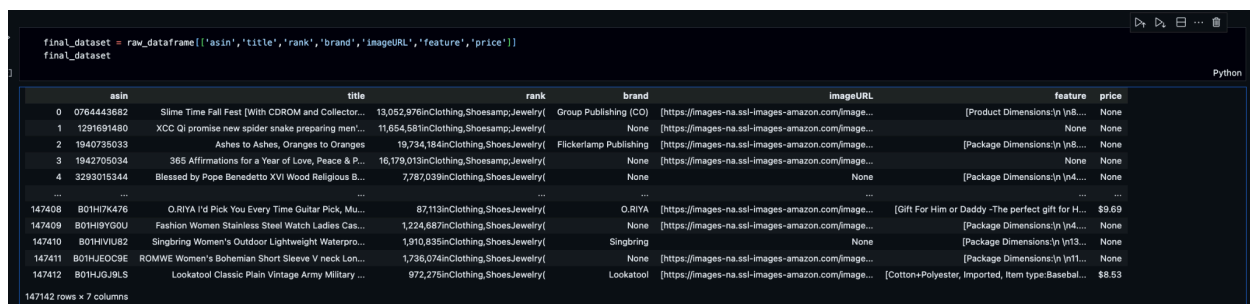
To construct this system, our group has developed a comprehensive three-step workflow. These steps are integral to our implementation:

1. **Data Cleaning and Loading:** This initial phase involves merging product data with metadata, cleaning, and enhancing the combined data to ensure it is primed for effective storage and retrieval.
2. **Database Manager Interface:** We have created a robust interface for database managers. This interface facilitates comprehensive CRUD (Create, Read, Update, Delete) operations, allowing for precise and efficient management of the database contents.
3. **User-Facing Web Interface:** The final step involves developing a visually appealing web interface that enables users to easily query product information. This interface is designed to be intuitive and user-friendly, ensuring a seamless experience for all users.

Below, we detail the specific functionalities created by our group, explaining what was developed and how each component contributes to the system.

In the first step of our project, our focus was on data cleaning and loading into a MongoDB database. We began by importing a dataset consisting of 80,000 entries from an Amazon Fashion JSON file into a pandas DataFrame. Subsequently, we loaded additional metadata from a CSV file. To integrate these two data sources, we matched entries based on the unique 'asin' feature,

ensuring that each product was correctly aligned across both datasets. This merging process resulted in a consolidated dataset containing 147,413 rows of unique product information. From this comprehensive dataset, we selected six features ('title', 'rank', 'brand', 'imageURL', 'feature', 'price') along with the 'asin' column to form the final DataFrame, which will be utilized for further analysis and operations. Here is a display for our final dataframe. (fig 1)



```
final_dataset = raw_dataframe[['asin','title','rank','brand','imageURL','feature','price']]
final_dataset
```

	asin	title	rank	brand	imageURL	feature	price
0	0764443682	Slime Time Fall Fest [With CDROM and Collector...	13,052,976inClothing,Shoes& Jewelry	Group Publishing (CO)	(https://images-na.ssl-images-amazon.com/image...	[Product Dimensions/\n \n8...	None
1	1291691480	XCC Qi promise new spider snake preparing men...	11,654,581inClothing,Shoes& Jewelry	None	(https://images-na.ssl-images-amazon.com/image...	None	None
2	1940735033	Ashes to Ashes, Oranges to Oranges	19,734,184inClothing,Shoes& Jewelry	Flickerlamp Publishing	(https://images-na.ssl-images-amazon.com/image...	[Package Dimensions/\n \n8...	None
3	1942705034	365 Affirmations for a Year of Love, Peace & R...	16,179,013inClothing,Shoes& Jewelry	None	(https://images-na.ssl-images-amazon.com/image...	None	None
4	3293015344	Blessed by Pope Benedetto XVI Wood Religious B...	7,787,039inClothing,Shoes& Jewelry	None	None	[Package Dimensions/\n \n4...	None
...
147408	B01H7K476	O.RIYA I'd Pick You Every Time Guitar Pick, Mu...	87,113inClothing,Shoes& Jewelry	O.RIYA	(https://images-na.ssl-images-amazon.com/image...	[Gift For Him or Daddy -The perfect gift for H...	\$9.69
147409	B01H9YG0U	Fashion Women Stainless Steel Watch Ladies Cas...	1,224,687inClothing,Shoes& Jewelry	None	(https://images-na.ssl-images-amazon.com/image...	[Package Dimensions/\n \n4...	None
147410	B01HJVIJ82	Singbring Women's Outdoor Lightweight Waterpro...	1,910,835inClothing,Shoes& Jewelry	Singbring	None	[Package Dimensions/\n \n13...	None
147411	B01HJEOC9E	ROMWE Women's Bohemian Short Sleeve V neck Lon...	1,736,074inClothing,Shoes& Jewelry	None	(https://images-na.ssl-images-amazon.com/image...	[Package Dimensions/\n \n11...	None
147412	B01HJGJ9LS	Lookatool Classic Plain Vintage Army Military ...	972,275inClothing,Shoes& Jewelry	Lookatool	(https://images-na.ssl-images-amazon.com/image...	[Cotton+Polyester, Imported, Item type:Basebal...	\$8.53

147142 rows x 7 columns

figure1

Upon examining the dataset, we identified several issues that necessitated further cleaning to make the data usable for our project. Specifically, the 'rank' field was formatted as a string containing numbers, commas, and characters, rather than as a pure number. Additionally, the 'price' column contained numerous missing values. To address these issues, we implemented several data cleaning steps: we converted the 'rank' to a numerical format using regular expressions to remove extraneous characters; we eliminated whitespace (represented as '\n \n') in the 'feature' columns; we removed rows with missing 'price' values and stripped the dollar sign ('\$') from the remaining entries; and we discarded titles that contained random, meaningless strings. After completing these cleaning steps, we successfully refined our dataset to 8,071

qualified samples, which are now ready for further use in our project. (fig2)

non_none_values

	_id	title	brand	imageURL	feature	price	rank
13	9654263246	X. L. Carbon Fiber Money Clip, made in the USA	Roar Carbon	[https://images-na.ssl-images-amazon.com/image...]	[Real Carbon Fiber, Made in USA, 5 year warran...	14.99	3725957.0
14	B00004T3SN	Shimmer Anne Shine Clip On Costume/Halloween C...	Shimmer Anne Shine	[https://images-na.ssl-images-amazon.com/image...]	[Shimmer Anne Shine Clip On Costume/Halloween ...	6.99	468314.0
36	B00007GDFV	Buxton Heiress Pick-Me-Up Framed Case	Buxton	[https://images-na.ssl-images-amazon.com/image...]	[Leather, Imported, synthetic lining, Flap clo...	16.95	43930.0
206	B00023JX9Y	Art Nouveau Sterling Silver Ornate Repousse He...	Silver Insanity	[https://images-na.ssl-images-amazon.com/image...]	[2&5/8" High and 3/4" Wide, Weight is Approx. ...	44.66	6343439.0
241	B0002PR25Y	Silver Forest Surgical Steel Turquoise Filigre...	Silver Forest	[https://images-na.ssl-images-amazon.com/image...]	[Shipping Weight: 1.28 ounces (View shipping r...	23.00	2852593.0
...
147397	B01H37KTG	Opal Gem Clip On Nose Ring Fake Non No Piercin...	Pierced Owl	[https://images-na.ssl-images-amazon.com/image...]	[Opal Gem Non Piercing Nose Clip, Silver Tone ...	7.99	134340.0
147406	B01HHVC958	Coxeer Venetian Masquerade Mask Halloween Mard...	Coxeer	[https://images-na.ssl-images-amazon.com/image...]	[The venetian mask is made of high-quality and...	12.98	67372.0
147407	B01H7FZLQ	O.RIYA Gifts I Pick You Always Forever, Father...	O.RIYA	[https://images-na.ssl-images-amazon.com/image...]	[Tell him he's special with this stainless ste...	11.96	540091.0
147408	B01H7K476	O.RIYA I'd Pick You Every Time Guitar Pick, Mu...	O.RIYA	[https://images-na.ssl-images-amazon.com/image...]	[Gift For Him or Daddy -The perfect gift for H...	9.69	87113.0
147412	B01HJGJ9LS	Lookatool Classic Plain Vintage Army Military ...	Lookatool	[https://images-na.ssl-images-amazon.com/image...]	[Cotton+Polyester, Imported, Item type:Basebal...	8.53	972275.0

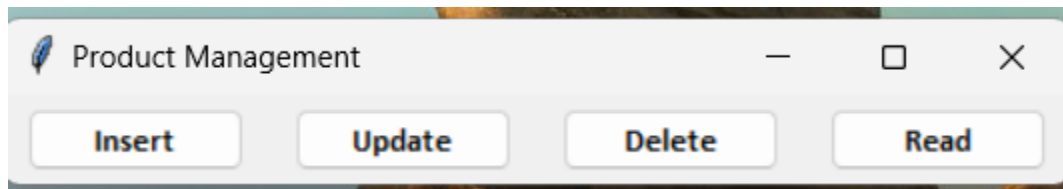
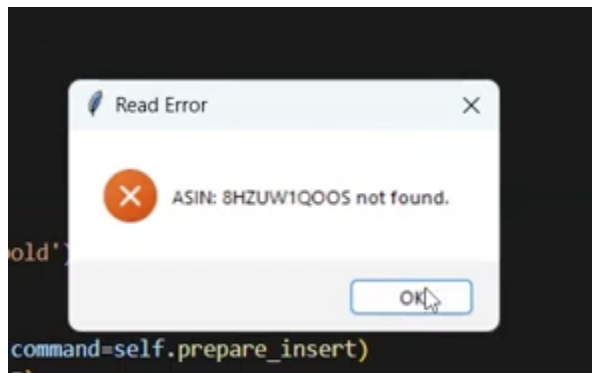
8071 rows x 7 columns

In alignment with our database architecture, we developed a sophisticated hash function to ensure balanced data distribution across three partitions, as outlined in Figure 3. Based on the results from this hashing process, we directed the data to the appropriate MongoDB distributed partitions. Additionally, we established a backup database to enhance data durability and ensure robust recovery options.

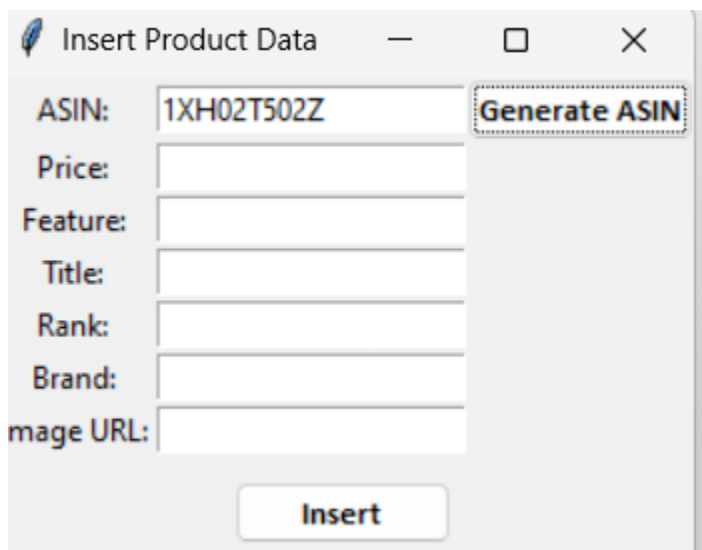
```
import hashlib

def complex_hash(asin, num_shards=3):
    # Use SHA-256 hash function to hash the ASIN
    hash_object = hashlib.sha256(asin.encode())
    # Get the hexadecimal representation of the hash
    hex_dig = hash_object.hexdigest()
    # Convert the hex digest to an integer
    int_hash = int(hex_dig, 16)
    # Use the modulo operation to get an index for the shard
    shard_index = int_hash % num_shards
    return shard_index
```

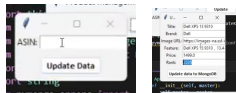
After data cleaning and loading, our group developed a user interface for the database manager to facilitate CRUD (Create, Read, Update, Delete) operations, as illustrated in Figure 4. This interface enhances user interaction and efficiency.



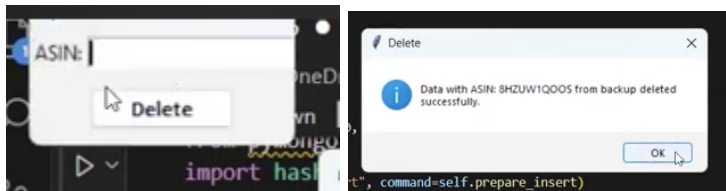
Create (Insert): Our interface enables managers to input product information into designated fields. Besides an 'Insert' button, we've implemented a 'Generate ASIN' button that generates a random, unique ASIN for new products, ensuring each entry is distinct.



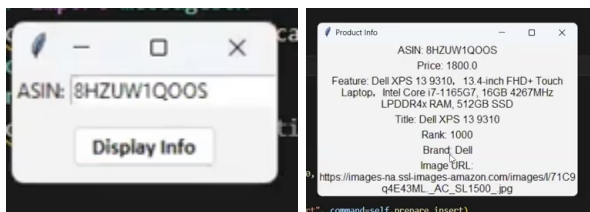
Update: Managers can search for a product using its ASIN, as shown in Figure 5. Upon validating the ASIN, a new window opens allowing the manager to update the product information. If the ASIN is invalid, the system captures and displays an error, as depicted in Figure 6.



Delete: Deletion operations are also ASIN-based. The system provides error notifications if the ASIN is invalid and confirms successful deletions with positive feedback.



Read (Query): The read operation utilizes ASIN as an index to fetch data. Errors in ASIN input are handled gracefully, displaying an error message if the ASIN is invalid. Valid queries display the results in a new window.

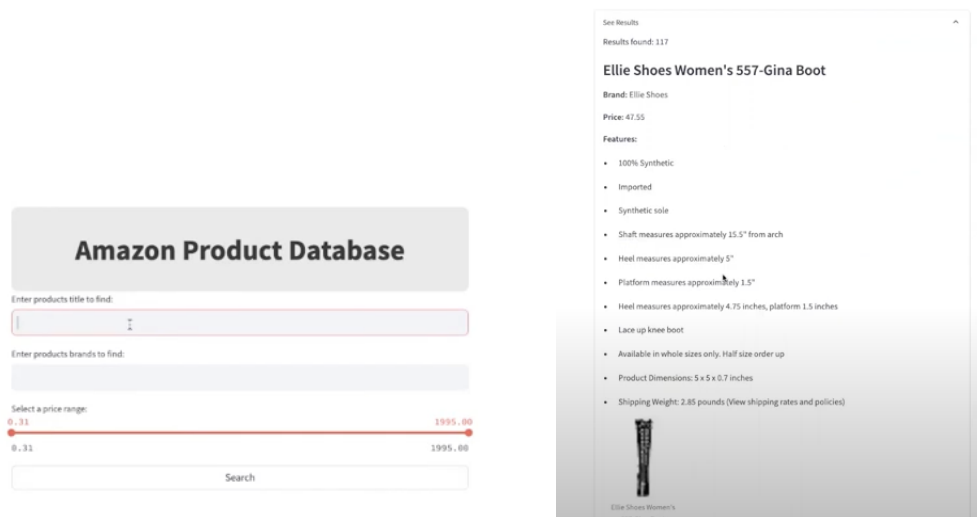


The third and final step in our project involved developing a user-facing web interface, designed to facilitate flexible and intuitive querying of the database. As depicted in Figure 1, this web interface allows users to define and apply various search conditions to tailor their queries. Users

can filter results based on product title, brand, and price range, using these filters either in combination or independently, according to their specific needs.

The interface is straightforward and user-friendly. Following a query, the interface not only displays the search results but also provides additional details like the total count of products found, housed under an expandable section. The results are presented in a well-designed format, as shown in Figure 2, ensuring that the product information is both accessible and easy to read.

This approach enhances user interaction, making it simple for users to obtain the information they need efficiently and effectively.



Tech Stack

Our project leverages a robust technology stack designed to optimize performance and user experience. At the core of our system, MongoDB serves as the distributed database, chosen for its flexibility and scalability in handling large datasets. For the backend processes, including data cleaning and CRUD operations, Python plays a crucial role, with its powerful libraries such as pandas for data manipulation and PyMongo for database interactions. The user interface for the database manager is developed using Tkinter, providing a reliable framework for building desktop applications. Additionally, for the user-facing web interface, we use Streamlit, which allows for the rapid development of interactive web applications. This combination of technologies ensures a seamless flow from backend data management to frontend user interaction, supporting complex queries and dynamic data presentation.

Learning Outcomes

The main challenge that our team faces is that none of us have any background knowledge of the front-end along with how to create an interface. Therefore, after searching and learning on the Internet, we decide to use tkinter library to build data manage interface and streamlit library to build our user interface. However, when we built our data manager interface, it was easy to build the initial four buttons for the CRUD process. When we tried to create a new interface when clicking the button and repeated it, we came across some difficulties that we clicked the button but the new interface did not show or an extra empty interface was shown. Meanwhile, we wanted to generate an random asin number that has similar form with a button. It

is difficult for us to generate a random asin number on the index when we tried to insert after click this button. For the user interface, our original thought is to create a website that everyone could view rather than a local website. However, since we store our data in local mongodb, it is hard to create a online website. Luckily, we asked for TA whether we could use a local website and TA answered that it was accepted to use a local website and then we use streamlit library to build our front end website.

Individual Contribution

Erkang Chen is responsible for searching for data, data cleaning and uploading data to MongoDB. Mingji Xi is responsible for creating the data manager interface for the CRUD process. Haoming Guo is responsible for designing the hash function and also the front-end website for user interface. All of them are participating in presenting the demo, recording the video and writing the final report.

Conclusion

In this project, we create a MongoDB distributed database for Amazon product information, showcasing the potential of MongoDB's distributed database architecture tailored for Amazon products. Through meticulous planning and implementation, we have crafted a system that promises enhanced efficiency, scalability, and user experience. Meanwhile, we encompass data cleaning and loading, robust CRUD functionalities for database managers, and an intuitive user interface for customers. We've ensured data accuracy, optimized query

performance, and fortified the system with backup and recovery mechanisms to maintain uninterrupted availability and safeguard business continuity.

Future Scope

In the future, we could improve our project in three parts, enhancing user interface, processing real-time data and using machine learning techniques. For enhancing user interface, we could incorporate more intuitive features, responsive design elements, and personalized recommendations based on user behavior. For real-time data processing, we could implement real-time data processing techniques to handle dynamic changes in product inventory, pricing, and customer preferences, ensuring timely updates and accurate information for both managers and customers. For machine learning techniques, we could use the knowledge from DSCI 552 to achieve predictive analytics, personalized product recommendations, and sentiment analysis of customer reviews in order to further enhance the platform's capabilities and user engagement.