Peter Rasmussen (prasmus3@jh.edu)
EN.705.603.82.SP23
HW6

Review Questions

1. Name two use cases for key-value databases.

Shopping cart

- A user shopping on Amazon selects products they would like to purchase. The size of the shopping cart is not expected to be large. The key-value database is well suited for such a use case because the user frequently selects (writes) items and unselects (writes) items. When the user is ready to check out, they access the cart, and the system reads the selected items for the user. Or the user does not make a purchase and the cart (collection of key-value pairs) is – depending on the application – preserved or discarded.

Mobile application user preference storing

- A user with a smartphone has some number of apps. Typically, apps allow the user to specify their preferences. For instance, CoinGecko is an app I use to track the performance of my crypto portfolio. I can select which tokens I would like to track. The keys in this case are the tokens and the values is the associated data (time series price data, token summary, website, etc.). I close the app and when I reopen it my preferences are saved because they were (presumably) written to a key-value database. The values update dynamically as price changes by the minute, but the keys remain the same so long as I don't update my preferences.

2. Describe two reasons for choosing a key-value database for your application.

- If I have a need to track transient attributes in a web application and I expect those attributes to be written and read in small batches, I would like a simple database (i.e., key-value database) that allows me to store those attributes as key-value pairs.
- If I expect simple queries (give me values for key X) then I don't need or want a database that supports execution of complex queries. Instead, I opt for a key-value database which meets the requirement without additional features I don't need.

3. Name two use cases for document databases.

- The New York Times website has a backend that must store a mix of written content (articles, blog posts, op-eds) and visual content (photos and videos). NYT website allows the user to search for articles, photos, and videos; therefore, the NYT needs a backend can quickly retrieve queried content. Document databases feature indexing and filtering capabilities well-suited for this purpose.

- Amazon sells millions of products on its website. Shoppers (like me) expect to get a list of products that match my query immediately after I execute it. Product attributes vary widely. For instance, the attributes of a new computer, which may include an extended warranty that I can buy, are different than the attributes of a plum, which may have an organic/non-organic attribute. Document-based databases are ideal for storing and indexing heterogeneous metadata that can be quickly served on demand to impatient users.

Peter Rasmussen (prasmus3@jh.edu)
EN.705.603.82.SP23
HW6

4. Describe two reasons for choosing a document database for your application.

- If I need a back-end for a website "with high volumes of reads and writes", a document database would be ideal, especially if that back-end must store articles, which a document database could easily handle.
- If I need to store items with varying metadata / attributes, it would be impractical to use a relational database, which would have a table for common attributes and a unique table for each set of non-common attributes. The relational database wouldn't scale if my database stored millions of such items. Instead, a document database flexibly accommodates such items.

5. Name two use cases for column family databases.

- US Census used Hadoop to store all the 2020 Decennial Census data. 2020 was the first time that citizens could complete the survey online, and US Census Bureau wanted to make sure all the online submissions were of sufficient quality. I worked on a team that analyzed the online survey submissions to ensure their quality. We required Hadoop because we frequently conducted analyses as they Census was conducted. There were massive pulls that then had to be reduced to summary statistics which we then provided to the government. I also worked for a project with USDA, which also uses Hadoop to store a significant portion of its data.
- Meta's Facebook uses column family databases to support its analytics operations and developed Cassandra, now maintained by Apache, to support its inbox search service.

6. Describe two reasons for choosing a column family database for your application.

- An application that must store "large amounts of data, read and write performance, and high availability" might require a column family database to support its operations.
- If the data is so large that multiple servers are needed, a column family, which partitions by column, might be needed.

7. Name two use cases for graph databases.

- Routing services such as Google maps use the A* algorithm to find the shortest path from point A to point B. A graph database natively stores the nodes and edges the A* algorithm uses to find the optimal path.
- LinkedIn may use a graph database for to find user's first-degree connections (people I've already connected with) and second- and third-degree people who are connected to some degree to my first-degree connections. A graph database is ideal for this use case, again because it natively stores the nodes and edges of the social network graph of LinkedIn users.

8. Describe two reasons for choosing a graph database for your application.

- If I needed to routinely find the shortest path through a network of connected entities, I'd want a graph database. This is because graph databases are structured in terms of nodes and edges (i.e., a graph) and that is the data structure that best meets my requirement.

Peter Rasmussen (prasmus3@jh.edu)

- If needed to analyze large social network graphs, I may want to store my data in a graph database. At the very least, I'd want a graph database on top of a column database to query such data (as some social network services do).

9. Name two types of applications well suited for relational databases.

- A relational database would be the best choice to manage a large company's employee data. Employee data fields should be already well known and therefore the relational database could be designed in advance. The company needs this data to be secure, to maintain its integrity, and be accessible by multiple users. The amount of data is limited (there are only so many employees and employee attributes) and so a column database is unnecessary.
- In traditional finance, the attributes of transaction data are known and long-established. Furthermore, banks must be sure their transactions ledgers are secure and have minimal error for legal and commercial reasons. Finally, banks need their transactions and accounts data to be accessible and allow read and write operations for multiple users. For these reasons, a traditional relational database is the best option.

10. Discuss the need for both NoSQL and relational databases in enterprise data management.

- Most sufficiently large enterprises have some sort of relational database to organize business data. These organizations require high data integrity, multiple users, and ACID transactions for this application. On the other hand, many companies need to organize their vast document stores. Think of a big corporation with innumerable job aids describing how to carry out particular tasks. This information can be easily managed using a NoSQL database such as MongoDB. So for the content management service component of enterprise data management, the NoSQL database is the best fit.