Beyzanur YILDIZ

1. **What are the communication methods between microservices in Lagom Framework? What are the uses cases of these methods? Why?**

   In the Lagom Framework, there are two basic ways to communicate between microservices. The first category is service calls. Service calls, which can be either synchronous or asynchronous (streaming), enable services to connect with one another using public APIs and established protocols (HTTP and WebSockets). Second, publishing messages to a broker like Apache Kafka further decouples communication. This is referred to as event streaming. It's a tried-and-true, scalable, and effective method of communicating across services.

2. **What are the methods to filter a non-primary key columns in CQL. Are they useful for production?**

   CQL non primary keys
   If you “know what you are doing”, you can force the execution of this query by using ALLOW FILTERING and that makes most of the functions usable for non primary keys. But for executing a forced query leader and other nodes will spend more resource. And thus the amount of work required to complete the query increases as the cluster size grows and you will not see the linear horizontal scaling that Cassandra is famous for.

3. **Is denormalization useful for Cassandra architecture? Why?**

   Denormalization is an important practice in Cassandra table design because Cassandra does not support joins or derived tables.

4. **What are the advantages of event sourcing?**

   There are various advantages to sourcing events:

   It overcomes one of the most difficult aspects of establishing an event-driven architecture by allowing events to be consistently published whenever state changes. It primarily avoids the object-relational impedance mismatch problem since it persists events rather than domain objects.
   It provides an audit history of all modifications made to a company entity that is 100 percent dependable.
   It allows for temporal queries to determine the status of an entity at any given point in time.
   Business logic based on event sourcing is made up of loosely connected business entities that exchange events. This makes migrating from a monolithic application to a microservice architecture much easier.

5. **In a CQRS(Command Query Responsibility Segregation) paradigm, can we read a record from microservice state? Is this approach the best practice? Why?**

   CQRS is a crucial pattern for querying between microservices. The CQRS design pattern can be used to reduce complex queries and inefficient joins. This pattern divides the read and update processes in a database. CQRS suggests creating two databases, one for reading and one for writing, using "separation of concerns" notions. This enables us to read and write from many database formats, such as no-sql for reading and relational databases for crud operations.

6. **Suppose an application that has many Lagom Microservices uses Cassandra Database(no-sql) for readside. How can you gather(make relation) of two different microservice's data tables?**

   Using a distributed NoSQL database enables the same application design - widely distributed, scalable, and adaptable to a variety of scenarios. In the future, you may make the most of your microservices by combining application and database design.