Questions and Answers

# How would news be passed to the frontend?

There are two solutions to answer this questions either the frontend will ask the server give me my news, or the server will push news to front end. The solution proposed push solution. For real devices like IPhone, Android devices, desktops and Laptops the system can use push notifications. For web pages, there are multiple solutions like HTML5 Server-Sent Events.

# As the business grows, there are many more users. How could you scale such a system? Where would you store the user session data?

The system is built using stateless micro services and it can be easily scale. It also can uses clouds solutions. For session, it is recommended to have no sessions, if not then we can use cookies specially if the session size is small like less than 2k. if not possible then we may use database, may be Radis or other key value databases, or use a readymade session management solution.

# Since news will not change after a while, how would you handle with news that are older than 1 year to control the capacity of the database? How does this affect your persistence logic?

Knowing the queries for the database is very important to answer such question, but here are few solutions:

1. Always keep your database clean. In our case for example after notifying a device the record which is found on the DeviceNews table can be deleted.
2. Find a write key to divide your data, may be newsCreationDate is good in this case. Divide your data to multiple files based on the newsCreationDate. Create an index based on newsCreationDate. Now every query should have newsCreationDate included with it, even update query.
3. We can move the older news to an ArchiveDB and here we do the following:
   1. Identify what need to be archives from the database.
   2. Have a view with union to point to archived database and let the view query use newsCreationDate in its selection criteria..
   3. Identify a partition key to partition the database.
   4. When moving the data disable database logging. May be using prober statements with table lock and row lock.
   5. Have an index on newsCreationDate so the view will get data faster.

# For database transactions there are different isolation levels? What is the reasoning behind it. Why would you choose one over the other. What would be a typical use case given the scenario above?

Serializable level provides read, write and range locks which will prevents phantom, non-repeatable reads and dirty reads problems. Repeatable reads level provides read and write locks. This will suffer from phantom problem. Read committed level this provides reads locks and suffer from phantom and non-repeatable reads problem. Read uncommitted level provides no locks and it is the fastest ones, but it suffer from all the problems of previous levels plus dirty reads problem.

Even though read uncommitted isolation level will cause dirty reads, unrepeatable reads, and phantom reads but this is acceptable due the system nature. An example of this is: a device logged out when a system receives a news and though it is not logged in because of dirty reads and unrepeatable reads problem. So even if the received news current transaction find it needs to notify a logged in device, and another transaction occurs at the same time changing the device state to logged out, then the notification system will not be able to deliver the message and the backend services will find that the device is logged out and will not try to resend it back. This situation could occur when the notification system is sending a news to a device while the device is logging out. Another example of issues that may occur is: a user search for news but could not get the latest ones because of phantom problems. Even though this situation could occur but the notification system will notify the user about the missing news. The same problem could occur when a user search about some news and after few micro seconds of replying to user the system receives a new news from news server.

# What is an optimistic lock exception in JPA?

It means that a transaction had been committed on entities that are trying to do delete or update. This exception occurs on commit, where a version checking of data occurs. The exception occurs when committing entity and the system finds the version for entity is different.

# What is a stale object, what is a detached object?

Detached objects becomes detached when their container closed. They still valid and can be attached to another container. Stale objects have an invalid state and cannot be used. On a detached or persistence object if the database data changed, then that object can become a stale object.

# How would you deal with them given the asynchronous scenario above?

To avoid stale data problems, do not have long lived session or transactions. Using @Version and optimistic lock will catch stale problems. Using read uncommitted will prevents this from occurs, but may effect data consistency.