MAIN ARCHITECTURAL PROBLEM:

PROBLEM:

The company's DBA's strongly advocate SQL Server for data storage and insist that the data be store in third normal form. But third normal form has negative performance implications and can cause application code to be harder to maintain.

SOLUTIONS AND DISCUSSIONS:

While Sql Server may be great for transactional consistency, it may be a poor choice if the system needs to handle billions of transactions. Other developers on the team may advocate MongoDb for storing the transactions. They've read that MongoDb scales really well, and part of that reason is that it's not a relational database and there are no third normal form requirements. But Mongo doesn't provide support for transactional consistency except between a parent and its direct children (think order and orderliness). So that could be a big gotcha with a financial system. And making up for that could requires lots of extra code that wouldn't be needed if the database supported transactions.

Back to Sql Server, there really is no reason that you have to store the data in third normal form, it's just what most corporate DBAs and relational database architecture books say you should do. "It's a best practice." And apparently it's what the company's DBA's are demanding. But if the architect is even considering MongoDb, they should surely consider a denormalized solution in Sql Server as part of the possible solution set of architectures.

And these decisions are intertwined, so for example, if support for billions of transactions is needed then an event driven architecture using a farm of databases might make sense, but if you go that way, then you won't have transactions showing up in account in real time. You see this with PayPal for example. It's possible to make a purchase with PayPal and then log into your PayPal account with another browser and the transaction isn't in your account. Come back 10 minutes later and check again with that browser and there it is. It's not an uncommon situation in big systems using event driven architecture and a database farm. I've seen it in At&t's billing system too. Is that a desirable architectural feature? Of course not. It's an unfortunate by product of the architecture decisions that were made to accommodate billions of transactions. So in this case real time processing was at odds with the need for high transaction volume.

Interestingly, these designs decisions will be almost as much about the people representing various positions as they will be about the technology their position advocates. So maybe the architect is leaning towards Sql Server using a denormalized database and leveraging Sql Server's transactional consistency. But will the DBAs allow it? The DBA's may like denormalized data much better if it's not in their Sql Server databases and is living in MongoDb instead. But maybe the VP of IS doesn't want to introduce a new database into the mix, and says Mongo isn't happening. And the Director of IS Ops isn't too excited about supporting it either. But maybe the use cases are starting to make MongoDb look like it might have an edge over SqlServer.

Author By: Rehman Ghani