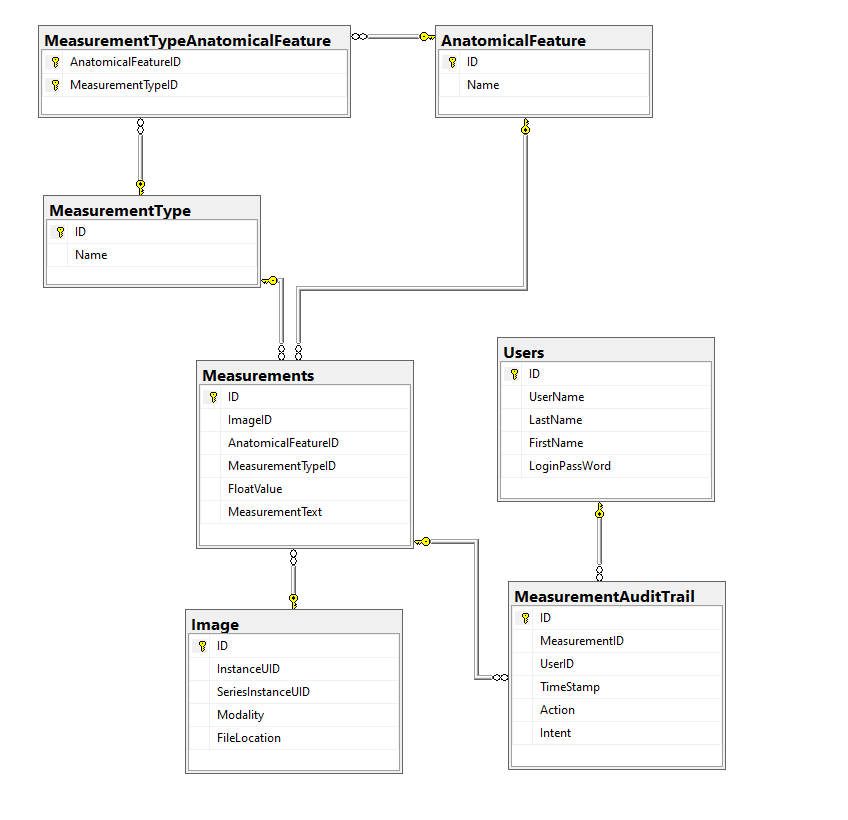
1. Server Side

Database schema as shown in following database diagram:



Stored procedures, triggers, views, user-defined table types:

1. We can create the user-defined data type for UID and Name

Create TYPE UID from nvarchar(64) NOT NULL

Create TYPE NameType from nvarchar(64) NOT NULL

1. We can create views about measurement: I created 2 views, one without user and audit information, one with user and audit information
2. We can create store procedure to generate the report of measurement audit records by anatomical feature, modality or between

sp\_MeasurementAuditByAnatomicFeature

sp\_MeasurementAuditByModality

sp\_MeasurementAuditBetweenDates

We can also create procedure to add or update the measurement in Measurements table and insert an audit record in MeasurementAuditTrail table

1. Based on what can be the most frequent queries and characteristic of the frequent query, we can create index on the view created in b). If there is not too much update operation for the database table, it will be good idea to create index to improve the database query performance. Should refer to SQL Index Architecture and design guide at <https://learn.microsoft.com/en-us/sql/relational-databases/sql-server-index-design-guide?view=sql-server-ver16>
2. I don’t see the need for user-defined table type for given use case.

Security Concern:

1. User Password should be encrypted.
2. Application should start with a login page for user to log in with password.
3. Create user groups and assign the proper permission to certain data and database operations by managing the user accounts and database roles
4. Create and schedule the database backup
5. Limit the access to the physical sever and network

Scalability:

1. Database server hardware upgrade, like more CPU, disk and RAM
2. Database Schema design: use efficient data types, use database indexing, divide the large table into smaller, more manageable one,
3. Optimize the queries by using tool to monitor the performance, identify the bottleneck.
4. Caching the frequently accessed data in memory.
5. Partition the database into smaller parts and distributed them in the cluster across multiple notes.
6. Gaining more understanding of the configuration and fine tuning by optimize the configuration.
7. Upgrade to cloud base database to take advantage of the cloud infrastructure and scaling tools
8. Client Side

The application is a .Net Core WPF project, this is my first time creating and working on the .Net Core project.

I use Entity Frame Core for creating the data model from SQL database. Current implementation did not handle the concurrency operations. It also don’t sync the database changes by the multiple clients.