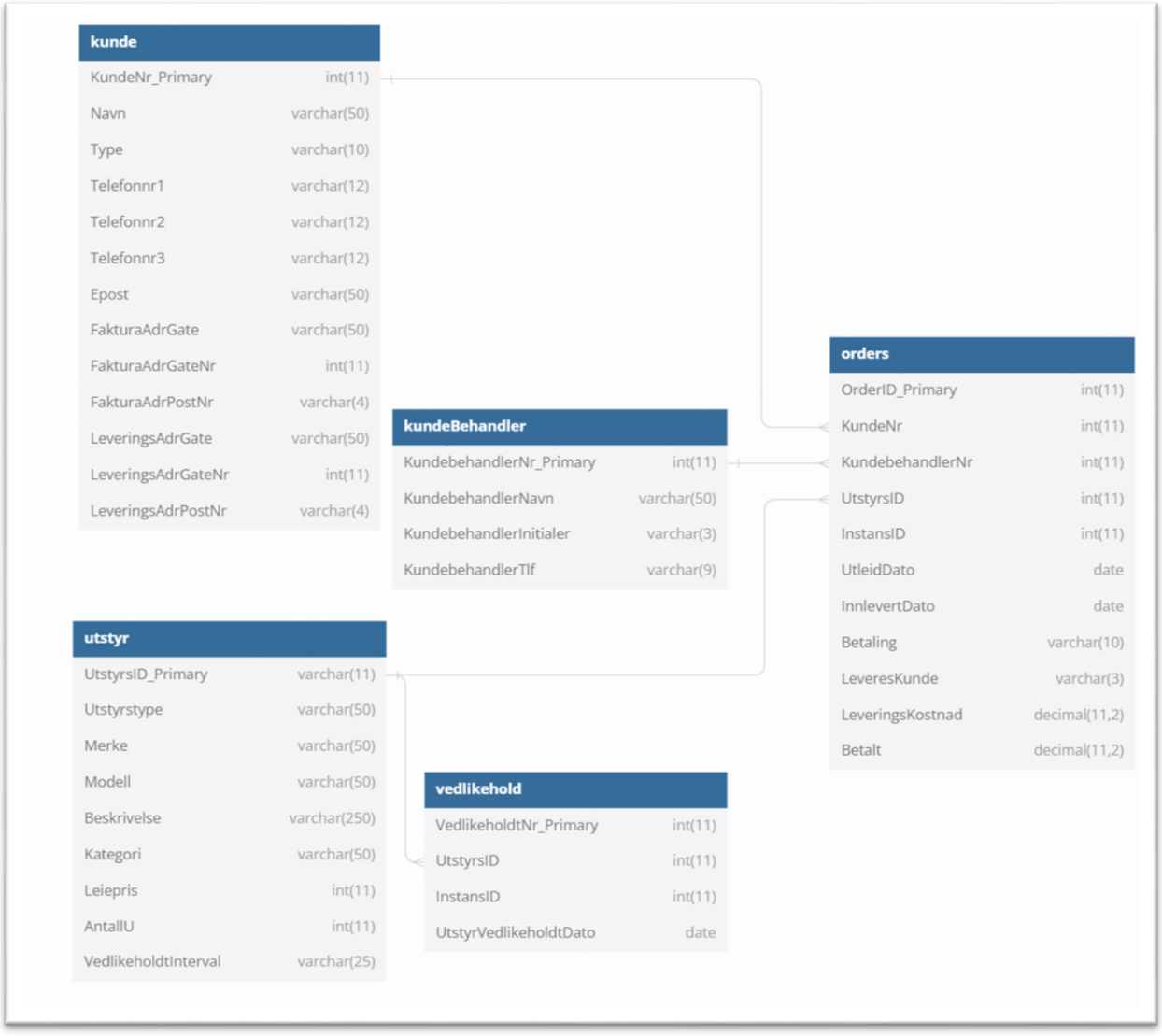
INTRO

The way the assignment was interpreted was that a better and more functional system was to be created in place of the existing one using SQL. This meant breaking down the existing table present in the XLSX file attached into multiple smaller and more reusable tables. The solution was to create separate tables for everything that could be separated. Then making it possible to link them back together with the use of primary keys. This allows for the lowest possible repetition between the tables while keeping all necessary functionality. Everything after was simply a matter of following the assignment point by point to create the ER-diagram, produce the SQL requests, author this report, and lastly record the video.

ER-diagram



Why Use SQL

The reason SQL is a step up from the previous way of doing things (the dreaded XL sheet) is that it both standardizes everything to a greater extent and eliminates repetition.

If we start by looking at the first point, standardization. The use of SQL allows for both easier compatibility and integration with other systems. Conventionally you would also have a front-end application to both enter and retrieve information from the database. This is something that would be a lot more difficult and a lot less practical with software like XL, which is a self-containing package that is not meant to be expanded to wider use. This would mean that you could have more systems in place to protect the valuable data like debit/credit card information and such. The scope of what one can do with an SQL database is simply wider and more adaptive to one's needs.

The second point, repetition. It is also critical as it allows for making changes without having to make that same change multiple times in various locations. As well as making the database structure much easier to change if the need arises. It also means that you are not storing the same information multiple times without justification. And mitigates the chance of potential errors in the information stored.

The downsides, and yes there are some. Are that it takes a lot more time to set up from the get-go and may simply be overkill in a lot of instances. If you have a large database, then the answer is obvious, SQL, no questions asked. But with something smaller it might take longer, and it might be more costly to set up than the time and money it saves can justify.

In this case an SQL database is clearly the better choice for the above-mentioned reasons.

The Prosses of Making the Tables

For starters, the attached xlsx file had to be deciphered. The way the file is set up is as 1 big table that contains everything. This leads to it being hard to read even with only 6 entries. This meant breaking it down into smaller more manageable tables.

Next was the "kunde" table. It was chosen as the first to be created as it was the simplest point from which to start. It was the most straightforward one of them all as the columns were already nicely lined up in the xlsx file and "kundeNr" provided a natural primary key. The prosses was repeated for "kundeBehandler". From here it was downhill and would only be more difficult.

Next up was the "utstyr" table. Although mostly straight forward, ended up posing a challenge with the maintenance dates which included both the previous maintenance date and the next. The reason this was a problem is because some of the entries within the table could have multiple instances (like 233-1 and 233-2). Not wanting to have entries within the table that were 95% alike or God forbid lose any data, it was decided to split it up and move things around. First the instance was not specified within the "utstyr" table and instead only the total number of instances. Then the maintenance dates were moved to a separate table ("vedlikehold") where the last maintenance date and the instance were specified then linked with the "utstyr" table. The date for the next maintenance date was converted to an interval in time and entered in the "utstyr" table. Thus, making the next maintenance date possible to calculate based on the existing entries within the table and solving the problem elegantly.

For the last table, "orders" it was all about linking the previous tables into something usable. This meant taking all the primary keys from the other tables and linking them to the "orders" table. Afterwards it was a simple case of adding the other columns.

Throughout the process there were a few times were going back to fix or change the tables was necessary. For instance, adding a column that was neglected or modifying an existing one. But thanks to the way the tables were designed, that being in a way that minimized repetition. This ended up being an uneventful and easy part of the process.

The result was structure seen in the ER-diagram.

The SQL Requests

As for the SQL requests. It was a straightforward part of the assignment. Because of the excellent way the tables were structured it was easy to link them together and get results specified in the assignment. The only drawback being that the SQL requests ended up quite lengthy.