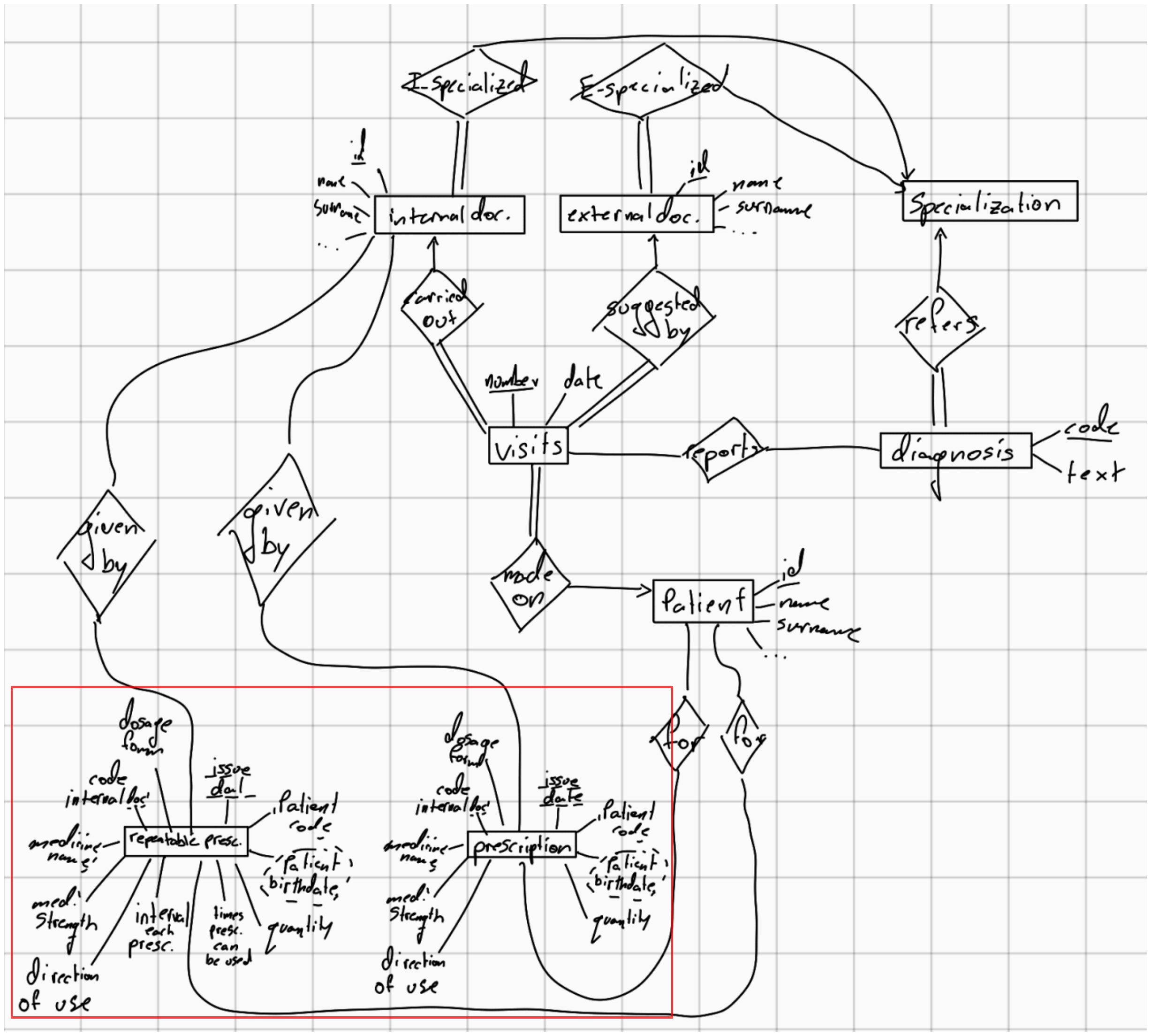
Relational Schema

The relational schema looks like this:

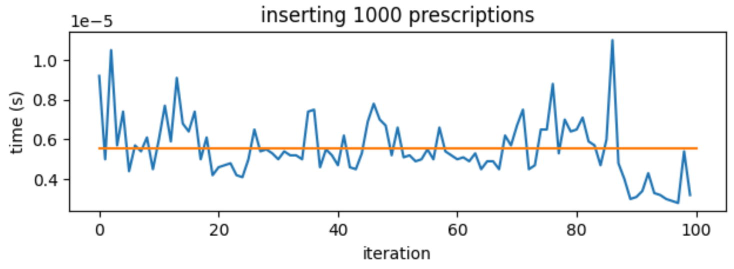


The repeatable prescriptions as well as the normal prescriptions are in full relation (many-many) to the internal doctors (who give the prescription) and to the patients (who receive the prescription). The relationships are many to many due to the fact that many internal doctors might give different (or the same) prescription to each patient, and each patient can receive many prescriptions at a time or during their existence in the system.
The primary keys on the other hand are determined by the *issued date* (when the prescriptions were given to the patient) and the name of the actual prescription. But this means even if two different doctors give out the same prescription on the same day, the prescription will be the same.
An attribute is redundant, the *patient's date of birth* can be found by looking up the patient directly, thus I didn't include it in the implementations of the tables in python and it's labeled as a weak attribute.

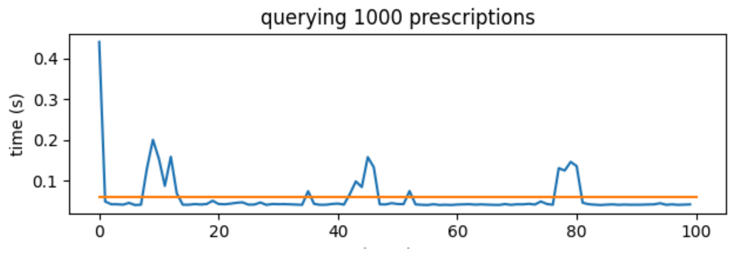
Time efficiency analysis of SQL vs NoSQL methods

To analyze the efficiency of the two methods I decided to insert and query 1000 prescriptions 100 times, these are the results:

USING SQL

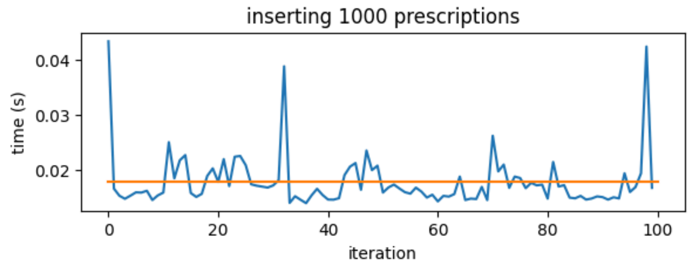


On average it takes SQLAlchemy 0.000006 seconds to insert 1000 prescriptions.

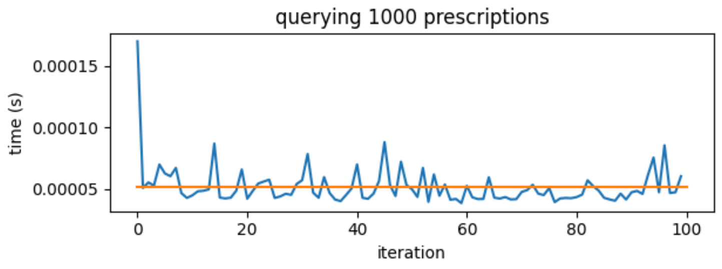


On average it takes SQLAlchemy 0.06 seconds to query all the 1000 prescriptions.

USING NOSQL



On average it takes pymongo 0.018 seconds to insert 1000 prescriptions.



On average it takes pymongo 0.00005 seconds to query all the 1000 prescriptions.

To summarize, when inserting data the SQL solution takes on average 0.000006 seconds to insert 1000 prescriptions, while if we compare the NoSQL solution (mongodb a document database), it takes 0.018 seconds to complete the data insertions for this instance of testing.

But if we consider querying the data the SQL solution takes 0.06 seconds while the NoSQL solution takes 0.00005 seconds.
The results show that for lots of data it's more efficient to use SQL to insert data but NoSQL to query the same data.

Functionality & Complexity of SQL vs NoSQL methods

As a relational database model, SQL requires more complexity when querying the data, but it results in lower space usage compared to a NoSQL solution like mongodb. For example when relating a prescription to a certain internal doctor multiple times, the relational database solution relates the prescription instance to one particular internal doctor instance for each prescription inserted. On the contrary the document solution would need to recreate the same internal doctor instance for each prescription added. This would be the same for the Patient attribute or any other type of metadata which can have a relation.
This results in a NoSQL solution having higher disk space usage but more efficient querying capabilities when compared to the SQL solution. This concept becomes more relevant with larger amounts of data.

Robustness of SQL vs NoSQL methods

The ability to have relational data gives an SQL solution more data integrity but can lack fault tolerance. A NoSQL solution can deal with more impacting faults. I have noticed this while creating the scripts for prescription insertion time analysis. Using SQLAlchemy when accidentally adding duplicated data it would throw an error so I had to create a for loop checking for duplicates, whilst with mongodb I didn't need to worry about this since it doesn't have unique primary key constraints.

Advantages and Drawbacks of SQL & NoSQL solutions

SQL solutions provide data integrity thanks to their constraints which improves data robustness but reduces flexibility and error tolerance. Thus NoSQL solutions allow for duplicated data which might not be ideal depending on the case.

SQL solutions allow for more complex querying capabilities thanks to their relational data model, whilst the NoSQL solution is less advanced in this sector. Although more complex querying results in higher computation time.

SQL solutions allow for time and space efficient data insertion keeping their complexity. Whilst NoSQL takes more time and space to insert data keeping it's simplicity.