University College of North Denmark

Top-up program - Software development

1st Semester / Class PSU0217 / Group 12

Project report

Group members

Miroslav Pakanec, Edgaras Aušvicas, Dominykas Žigas

Supervisors

Rasmus Christiansen Knap

Jens Henrik Vollesen

Submission date

May 30th 2017

University College of North Denmark

Software development / PSU0217 - Group 12

Abstract:

This project consists of researching, planning, implementation and testing of the Georgia Tech Library system. It mainly focuses on aspects from database development and testing. The system is a Client-Server web application developed in C# and ASP.NET MVC using the Agile and SCRUM development methodologies.

Preface:

We are a first semester students of a Software development program at University College of Northern Denmark. Our team consists of three members representing two European countries - Slovakia and Lithuania.

This report describes our development process of the Georgia Tech Library system. During the process, we have learned a number of crucial concepts, especially from the area of testing and databases.

Problem statement:

Large libraries, such as Georgia Tech Library, need to keep track of large amounts of data - they need to manage all their books, members, orders and reservation.To do this, they need to store query this data efficiently.

Can we design a system, that would help Georgia Tech Library to manage information about their members, books, reservations and borrowing activities? Can we design a database that is capable of storing and retrieving data efficiently? And finally, can we ensure the quality of the delivered product by applying various testing techniques and testing types?

Content

Figures and tables

1. Introduction

The report describes the implementation and testing of an application for Georgia Tech Library (GTL). The software is a tool for both librarians and employees of the library as well as other members. On one hand, it should serve as a tool for librarians to manage all the books, orders and members of the library. On the other hand, it should help users and members of the library to order books more easily as well as provide knowledge and summary of their orders and activities.

The report primarily focuses on two parts - testing and database. The testing part of the report describes the creation, execution and summarization of the test plan and test strategy. It attempts to answer the question - How can we assure the quality of the GTL system? It also describes how the system should be developed and implemented, so that it is highly testable.

The second part of the report focuses on the implementation of the database. It describes the steps how the database is designed and created from the business requirements. Furthermore, the performance of the database is analyzed and evaluated.

2. Test

The Test section focuses on the test aspects of the GTL project. It can be divided into three sub-sections - test planning (creating a test plan, understanding what needs to be tested an how), creating testable architecture and finally on implementation (implementation of various testing techniques as well as how testing is performed on different levels).

2.1 Test Plan

The following section describes the master test plan for the GTL project. It mainly specifies scope, approach, resources and schedule of intended testing activities. Generally, a test plan serves as a tool for communication between project team members, peers, managers and other stakeholders.

2.1.1 Scope

Defining a testing scope is a critical assignment, because the available resources (such as time or budget) for testing and quality assurance are limited. It is common that significant amounts of time go to the development of the new features and there is little time left for the quality assurance, therefore it has to be used wisely.

The scope determines, which features will be tested as well as how they are going to be tested (which types of test are and are not going to be performed). Several criteria were taken into consideration - the amount of time, the importance, priority and frequency of usage of customer's requirement as well as its risk according to the product risk analysis (see Risk analysis - 2.1.11).

User stories to be tested were chosen based on their priority assigned to them during sprint 0 using the Moscow scale. User stories with priority Won't and Could were completely excluded from the testing scope. According to the System requirement document (see 5.1.1), only requirements with from SR-1 to SR-14 were included in the scope.

Additionally, only functional tests were included within the scope. Functional testing will be performed on different test levels (see Test levels - 2.1.3). Component testing will be relevant only in the Core module (see System Architecture - 5.2) and it is necessary that every feature within the scope is unit tested with adequate test coverage (see Exit criteria - 2.1.9). Integration tests will be performed on system components that integrate with external components, such as the database. System test will be performed only on requirements with the highest criticality - those are, according to the risk analysis, risks with the priority higher than 15.

Non-functional tests, installation tests, live tests, alpha and beta tests were excluded from the scope.

2.1.2 Test objectives

Test objectives have been set on different test level, because each level tries to achieve a different goal.

Component test objectives:

- Find defects as early as possible and minimize the cost to fix them
- Find programmers defects created during development process
- Ensure that individual software component functions properly in isolation
- Ensure quick detection of defect's cause.
- Improve code quality

Integration test objectives:

- Verify that software components properly interact with other components
- Verify that software integrates with database correctly.

System test objectives:

- Confirm that the system works as intended for the end users
- Assure that the final result meets customer requirements

2.1.3 Test levels

The purpose of having different test levels is to split testing phase into logical phases and to set boundaries between them.[1] Test levels are - component tests, integration tests and system tests.

Each test level fulfills different objectives (see Test objectives - 2.1.2) and have their advantages/ disadvantages. The test pyramid model was followed in order to capture those advantages.

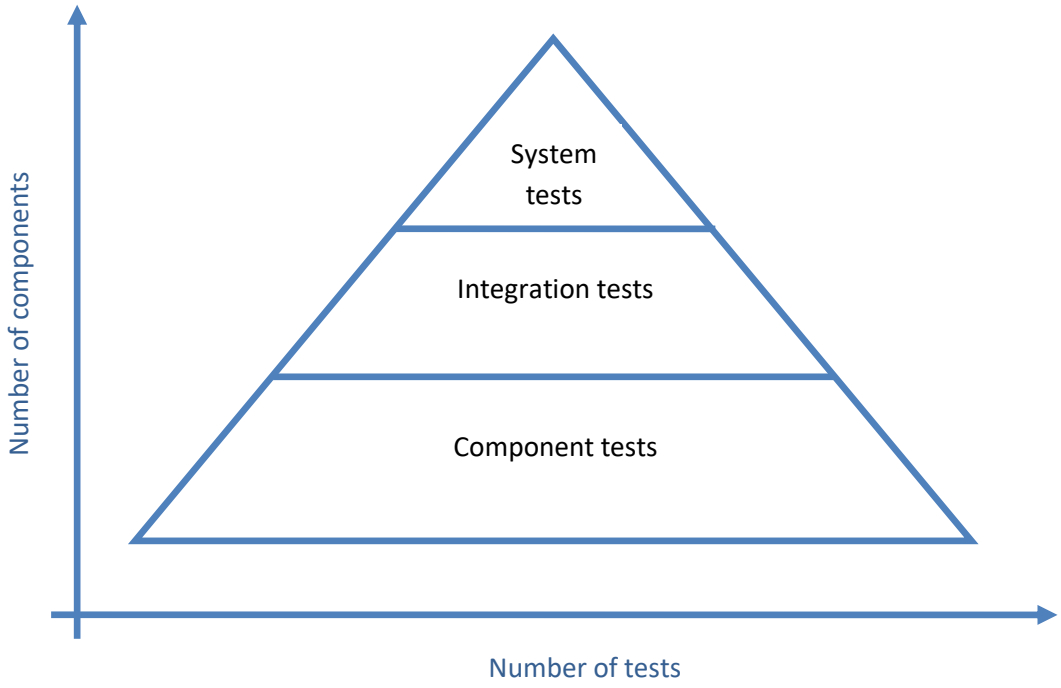


Figure 1 - Testing Pyramid

Disadvantages of the high-level tests:

- increased execution time
- increased cost to develop and maintain
- the possibility of false negatives

Advantages of higher level tests:

- high coverage
- more business relevant

[1] Adapted from http://www.requirementdriventesting.com/how-to-test-strategy (Retrieved in 05/2017)

2.1.4 Defect resolution

When a defect is found, it is written into the defect-log document (see Defect log - 5.1.2), where defect attributes must be specified. Once the defect is acknowledged, the defect resolution process begins. The process consists of 4 steps - prioritize risk, schedule fix, fix the defect and report resolution.

Risk prioritization - The purpose of risk prioritization is to identify, whether given defect is new or was previously reported, as well as to determine what is the priority to fix it and what steps should be undertaken to minimize the impact. In order to properly prioritize the risk, three step prioritization scale is used - critical, major or minor.

Fix schedule - The fix for the defect is scheduled depending on its priority.

Defect fix - it is required to correct and verify both program and documentation deliverables to remove the defect from the system.

Report resolution - Once the defect is fixed, other team members have to be notified that the defect has been fixed, while information such as, nature of the fix and when is the fix going to be released are provided. [2]

2.1.5 Test tools

For testing on every test level, MS Test framework is going to be used. Additionally System tests will be achieved with selenium web driver configured for Google Chrome web browser.

2.1.6 Deliverables

Test Deliverables are the artifacts, which are given to the stakeholders of software project during the software development lifecycle. There are different test deliverables at every phase of the software development. Some test deliverables are provided before testing phase, some are provided during the testing phase and some after the testing cycles is over.[3] Deliverables are:

- test case specification document
- risk analysis
- system requirement specification
- test plan
- test strategy

[2] Adapted from - http://www.defectmanagement.com/defectmanagement/defresolution.htm (Retrieved in 05/2017)
[3] Adapted from - http://www.softwaretestingmentor.com/what-are-test-deliverables (Retrieved in 05/2017)

- test trace-ability matrix
- test summary report
- defect log

2.1.6.1 Test case specification document

A test case specification is a document that specifies the inputs, predicted results, and a set of execution conditions for a test. Test cases were specified differently depending upon how they would be performed. If a test case is a component test or an integration test, only the following attributes are specified - test case id, title, requirement id, input and output parameters, environmental needs and expected result (pass/ fail). However, if a test case is an end-to-end test it is also necessary to define risk id, description, precondition test steps, procedure test steps (in a form of action - verification) and finally post condition test steps. The sample of an end-to-end test case specification can be seen on figure number 2 and the whole test case specification document is listed in the appendix (see section 5.1.3).

Test case ID: TC-S-1
Description: Test that user can be registered with valid ssn and password and non-verified member role is assigned to him. User is automatically logged in and is redirected to the home page.

Test items: SR-8, SR-9, SR-10
Risk IDs (R-ID): 7,
Input specification: TCIS-1
Expected result: User is registered successfully.
Environmental needs:
- Google chrome browser is installed on the machine
- Connection to the database is established.
- AppConfig file of SystemTest project has valid database configuration.
- Member with ssn "111-00-0000" is not present in the database.
- Firewall does not block chromedriver.exe

Precondition steps:

Test procedure steps

Action test step	Verification test step
Navigate Google Chrome browser to URL: "localhost:60764/Account/Register"	Browser is on URL: "localhost:60764/Account/Register"
Enter value "111-00-0000" to the input field with name "Ssn" and defocus the input field.	No validation error message is displayed.
Enter value "MyPassword1." to the input field with name "Password" and defocus the input field.	No validation error message is displayed.
Enter value "MyPassword1." to the input field with name "Confirm Password" and defocus the input field.	No validation error message is displayed.
Left click on the submit button.	No validation error message is displayed. Browser is redirected to the URL: "http://localhost:60764/".
	On the right side of the top navbar is a message: "Hello 111-00-0000!" indicating that user with Ssn "111-00-0000" is successfully logged in.
	Verify that in the database, user with Ssn "111-00-0000" has assigned a role with ID: 2 and his password is hashed.

Postcondition steps:
- Remove user with Ssn: "111-00-0000" from the database.

Figure 2 - End to end test case sample

2.1.6.2 Traceability matrix

The purpose of the traceability matrix (requirement to test case traceability matrix) is to verify that the test cases capture testing of all the system interface requirements at least once. The traceability matrix captures following attributes - Id, Requirement id, Requirement description, status, system components, a software module, test case id and implementation and testing dates. A sample from the traceability matrix can be seen on table number 1 and the traceability matrix document is listed in the appendix (see section 5.1.4).

ID	SR-ID	Requirement description	Status	System components	Software Module	TC-ID	Implemented	Tested
1	1	Librarian is able to add a book with valid ISBN.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-6	25.04.2017	26.04.2017
2	4	Every user is able to see a catalogue of books.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-1	25.04.2017	26.04.2017
...	...	...	...	...	...	...	...	...
7	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-I-1	25.04.2017	08.05.2017

Table 1 - Requirement to test case traceability matrix sample

2.1.6.3 Test Summary document

Test summary report is a document that contains a summary of test activities and final test results. It serves stakeholders to evaluate the quality of the tested product. This type of deliverable is prepared when the testing is completed. Table number 2 shows the results of the final test run.

Level	# test cases	# passed	# failed	Pass rate (%)	Fail rate (%)
Component	325	320	5	98,46153846	1,538461538
Integration	92	82	10	89,13043478	10,86956522
End to end	10	9	1	90	10
Total	427	411	16	96,2529274	3,892944039

Table 2 - Test Summary

The graphical representation of the test run can be seen on figure number 3.

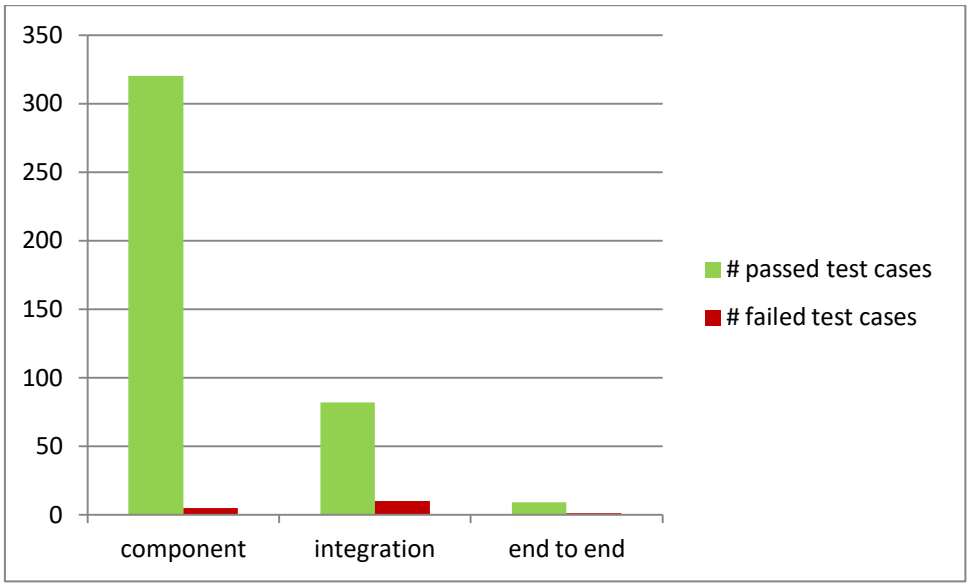


Figure 3 - Final test run pass/ fail comparison

2.1.7 Test schedule

The test schedule is the essential attribute that defines the timelines for the testing activities. It must be planned according to the development schedule.[4] Table number 3 shows the schedule of all testing activities.

ID	Task	Start	End
1	Business understanding	10.02.2017	20.02.2017
2	Create test plan	20.02.2017	05.03.2017
3	Risk analysis	05.03.2017	10.03.2017
4	Test case preparation	10.03.2017	15.04.2017
5	Test framework development	15.04.2017	30.04.2017
6	Test development	01.05.2017	15.05.2017
7	Test execution	16.05.2017	16.05.2017
8	Test evaluation	17.05.2017	20.05.2017

Table 3- Test schedule

Separation of the test schedule and the development schedule was the biggest challenge when the test plan was created. It was because of the agile development methodology chosen for the GTL project. A lot of testing activities (such as component or integration testing) were the responsibility of developers and were performed on the sprint basis. Therefore it was impossible to e.g prepare test cases for integration with the database before the development started - it had to be done before every sprint started, when sprint backlog was created.

[4] Adapted from http://reqtest.com/testing-blog/sample-test-plan (Retrieved in 05/2017)

On the other hand, the test plan very well complemented the end to end testing. The end to end testing did not start before the development was finished. During the development, risk analysis was made and afterward, test cases were created accordingly. Before the testing could begin, an automated test framework (see Test framework - 2.3.5.2) needed to be developed. After the tests were implemented, the execution and evaluation was performed. The test plan as well as the development plan can be seen on figure number 4.

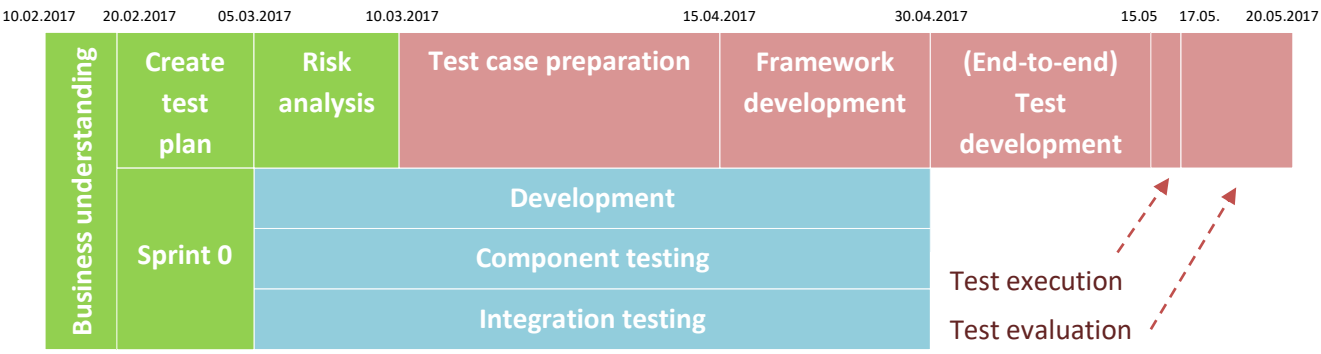


Figure 4 - Test and development schedule

2.1.8 Entry criteria

Entry criteria are necessary conditions that must be in place before testing can begin. All the criteria are to prevent testing time being wasted at any stage in development. They break down into two sets of responsibilities - responsibilities of the test team and the development team.

Before testing can begin, test team is responsible for developing the test plan, which must be later approved by the stakeholders. Additionally, acceptance criteria, test cases and test scripts must be developed and the testing environment has to be properly set up.

Development team is responsible for delivering a working version of the software. Furthermore, the software has to be properly unit tested, before its released to the testing environment. [5]

[5] Adapted from - http://www.coleyconsulting.co.uk/testing-entry-criteria.htm (Retrieved in 05/2017)

2.1.9 Exit Criteria

Exit criteria are used to determine whether a given test activities have been completed or not. The exit criteria for the GTL project are:

- Test coverage of 75% of the Core module and 50% of the Data Access module (see System Architecture - 5.2) has been met.
- All high risk areas (with priority greater than 15 according to the risk analysis) are completely tested.
- All planned tests have been run.
- There are no high severity defects left outstanding.

2.1.10 Test strategy

Test strategy is critical in the making of a test plan. When a test strategy is being selected, a number of factors (e.g. risks, skill, objectives, regulations, ...) must be considered.

One of the most crucial factors that have influenced the decision making, were risks. After a project risk analysis, it was concluded that not meeting the deadline is the biggest threat of the project. The number of user requirements was high and there was a limited time. Additionally, skills of the team had to be considered - every team member lacked testing experience and was in the process of learning. On top of that, project requirements were very well set. Test strategy that would satisfy the needs of the stakeholders had to be selected.

After consideration of these factors, analytical risk based strategy in combination with regression-averse strategy were selected.

Risk based testing is prioritizing the features, modules and functions of the Application Under Test based on impact and likelihood of failures. It involves assessing the risk based on the complexity, business criticality, usage frequency, visible areas, defect prone areas, etc. [6]

According to test objectives (see Test objectives - 2.1.2 , one of the goals is to find defects as early as possible. Moreover, the application is being developed using Agile development methodology. Due to these reasons, a regression-averse strategy was chosen as well. Regression-averse strategy provides the possibility to respond to frequent changes, by having a set of automated procedures that allow to detecting regression defects.

[6] Quotes from http://www.guru99.com/risk-based-testing.html (Retrived in 05/2017)

2.1.11 Risk analysis

Constraints such as limited time or budget can negatively impact the amount of functionality and quality of the project. Product risk analysis is a method that helps testers divide limited resources and focus on the parts that have the highest priority.

Firstly, the product risks were determined based on the system requirements document (see System requirements specification document - 5.1.1). Secondly the likelihood of each risk as well as its impact was specified, using the 5 scale system (very low, low, medium, high and very high). Finally, each product risk was prioritized (priority = likelihood * impact).

R-ID	SR	Product risk	Likelihood	Impact	Priority
1	14	Number of book copies of a book is updated (decreased) while they are loaned and they get lost.	5	5	25
2	15	Librarian removes a book that has loaned book copies.	5	5	25
3	11	Standard user loans a non existing book copy.	4	5	20
4	11	Standard user loans a book that cannot be loaned (is not available).	4	5	20

Table 4- Risk analysis sample

Legend:

Very low	1
Low	2
Medium	3
High	4
Very high	5

2.2 Testability

Testability is a degree to which a system or component facilitates the establishment of test criteria and performance of tests to determine whether those criteria have been met. In other words, it is the ease with which a computer program can be tested.

Testability follows directly from a good design and architecture of the software, therefore It is a responsibility of developers to make software testable. The following are the common design challenges that had to be overcome in order to improve testability.

2.2.1 Object initialization and business logic

Objects should not have many responsibilities. A very common way of how to make system non-testable, is to combine the object instantiation (the new operator) with business logic. This automatically creates dependencies that are difficult to overcome and there is no control of testing in isolation.

By separating these two responsibilities, it is possible to instantiate a small sub-portion of the application in the test. Additionally this removes conditionals and if statements to one place (see Conditionals - 2.2.4), providing a performance and maintenance benefit, because the same logical decision does not have to be done over and over again. This separation can be achieved, for example, using a factory or dependency injection, using an Inversion of Control container.

In the application, constructor dependency injection was used, using the Unity inversion of control container. This gives the possibility to mock all the dependencies in the tests, ensuring that components work as intended in isolation. The mocking was performed using Moq library (see Mocking framework - 2.3.3.2). Figure number 5 show the usage of dependency injection in the BookCopyRepository. Book copy data access interface as well as book copy mapper are injected via constructor. Figure number 6 shows the mapping of injected dependencies to their implementations in the inversion of control container.

```
public class BookCopyRepository : IBookCopyRepository
{
    private readonly IBookCopyAccess _access;
    private readonly IMapper<BookCopy> _mapper;

    public BookCopyRepository(IBookCopyAccess access, IMapper<BookCopy> mapper)
    {
        _access = access;
        _mapper = mapper;
    }
```

Figure 5 - Example of constructor dependency injection

```
//DAL
container.RegisterType<IBookAccess, BookAccess>();
container.RegisterType<IBookCopyAccess, BookCopyAccess>();
container.RegisterType<IMemberAccess, MemberAccess>();
container.RegisterType<IRoleAccess, RoleAccess>();
container.RegisterType<IOrderAccess, OrderAccess>();
container.RegisterType<IOrderLineAccess, OrderLineAccess>();

// repositories
container.RegisterType<IBookCopyRepository, BookCopyRepository>();
container.RegisterType<IBookRepository, BookRepository>();
container.RegisterType<IMemberRepository, MemberRepository>();
container.RegisterType<IRoleRepository, RoleRepository>();
container.RegisterType<IOrderRepository, OrderRepository>();
container.RegisterType<IOrderLineRepository, OrderLineRepository>();
```

Figure 6 - Example of mapping of dependencies and their implementations

2.2.2 Constructors and the Law of Demeter

In order to test a method of an object, the object has to be instantiated. However, if an object is difficult to construct, it is also difficult to test.

From the testing point of view, it is very hard to control the code that is executed inside a constructor. In addition, the constructor cannot be overridden - its behavior cannot be changed. Therefore a constructor should only ask for what it really needs and it should not contain any additional work. It is preferable, that a constructor only assigns dependencies to fields. This is achieved using constructor injection. Only objects that have a lifetime greater or equal to the object that is being constructed should be injected.

2.2.3 Global state

From a testing point of view, global state is a real problem. It is an unwanted external factor to the test, which can result in unstable behavior (e.g. multiple executions produce different results). Furthermore, it might cause that the test will have to be run a particular order, making it impossible to run the tests in parallel.

To be able to bypass global state issues, patters such as singleton or static methods (unless it is a leaf method) have not been used.

2.2.4 Conditionals

Methods with too many conditionals or if statements are hard to test. The reason for this is, that the cyclometric complexity is higher and there are more execution paths - more test cases needed to be created to cover them. Hence, the code is less maintainable and less extendable.

Most conditionals choose different behavior depending on the type of the object or its state. These conditionals can be replaced with polymorphism, by moving each leg of the conditional to an overriding method in a subclass. On top of that, to become "if free", paranoid programming and unnecessary null checks have to be avoided.

The benefits of polymorphism are not only easier to test code, by reducing the cognitive load, but also easier to maintain code, because all of the related items are together in the subclass.

2.3 Implementation and testing

2.3.1 Static testing

Static analysis is software analysis performed without executing or running, the software. Static analysis tools look at applications in a non-runtime environment. [7]The objective of static testing is firstly, to find and recognize mistakes as early as possible. Secondly it improves communication and has an informational and educational purpose.

2.3.1.1 Formal Reviews

Formal reviews are one part of static testing. Their purpose is to find and eliminate errors and ambiguities in documents such as requirement specification, design, test cases, etc.

There are different types of formal reviews. The least formal is the walkthrough, which is usually lead by the programmer and the author of the document that is being reviewed. The purpose of this review is to establish a common understanding, while the observations are being logged.[8] The walkthrough was used when a higher level document (such as requirement specification)was reviewed.

Technical review is, compared to a walkthrough, more formal. It is lead by a trained moderator, who is not an author of the document. This type of review does not involve management or a business team. [9] Its purpose is to reach an agreement on the technical concepts, such as UML diagrams.

Inspection review, is the most formal review and it requires high preparation. It is driven by checklists and rules. [10]

[7] Quoted from https://www.veracode.com/products/static-analysis-sast/static-analysis-tool (Retrieved in 05/2017)
[8] Adapted from https://www.tutorialspoint.com/software_testing_dictionary/code_walkthrough.htm (Retrieved in 05/2017)
[9] Adapted from https://www.tutorialspoint.com/software_testing_dictionary/technical_review.htm (Retrieved in 05/2017)
[10] Adapted from https://www.tutorialspoint.com/software_testing_dictionary/inspection.htm (Retrieved in 05/2017)

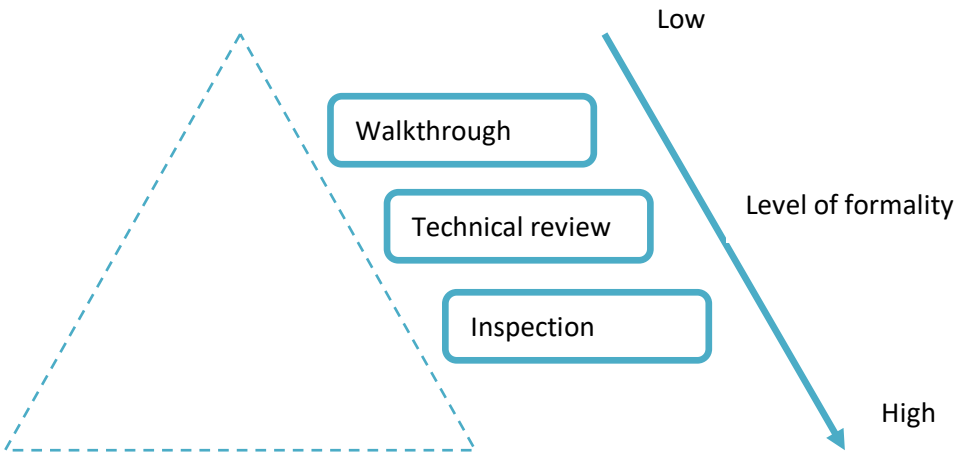


Figure 7 - Types of formal reviews

In the project, different types of reviews were used, depending on its purpose and type of the document that is being reviewed. For example, several walkthrough reviews were performed to achieve a common understanding of higher level documents, such as the requirement specification or test plan. On the other hand, technical reviews were performed on more technical documents such as database schema, ER diagrams, UML diagrams and software architecture.

2.3.1.2 Static analysis

Static analysis is a method of examining the code without executing the program. The purpose of the analysis is to find defect whether they cause failures or not. It focuses on checking, whether the code adheres company standards, validates the cyclometric complexity of the code and the control flow.

it can be either done by visual inspection alone, which is called program comprehension or by the usage of various automated tools. In the project, Resharper was commonly used in to identify a violation of established code standards.

2.3.2 Dynamic testing

Dynamic Testing is a kind of software testing technique, which is using the dynamic behavior of the code analyzed. It checks for functional behavior of software system , memory/ CPU usage and overall performance of the system.

The main objective of this testing is to confirm that the software product works in conformance with the business requirements. It is performed at all levels of testing and it can be either structure based (white box) or specification based (black box) testing.

2.3.2.1 Black box testing

Black box testing technique, views the software as a black-box with inputs and outputs. The testers have no knowledge of how the system or component is structured inside the box. The tester is concentrating on what the software does, not how it does it.

Black box testing includes both functional and non-functional testing. Functional testing is concerned with system's features or functions. Non-functional testing is concerned with examining how well the system does. It deals with requirements such as performance, usability, portability, maintainability, etc. [11]

For a black box testing techniques like Equivalence Partitioning, Boundary value analysis and All pairs were used, to be able to create a minimal number of tests and providing sufficient coverage.

Black box testing was used when test cases for book catalogue filter were created. A boundary value analysis in a combination with equivalence partitioning technique can be applied to the properties by which the catalogue will be filtered (e.g. Author). A valid author name has a length between 0 and 30 characters as shows figure number x.

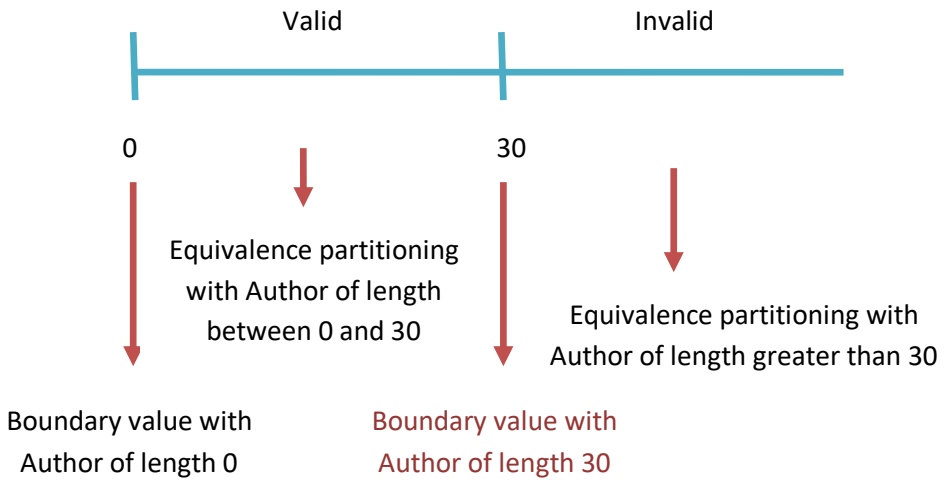
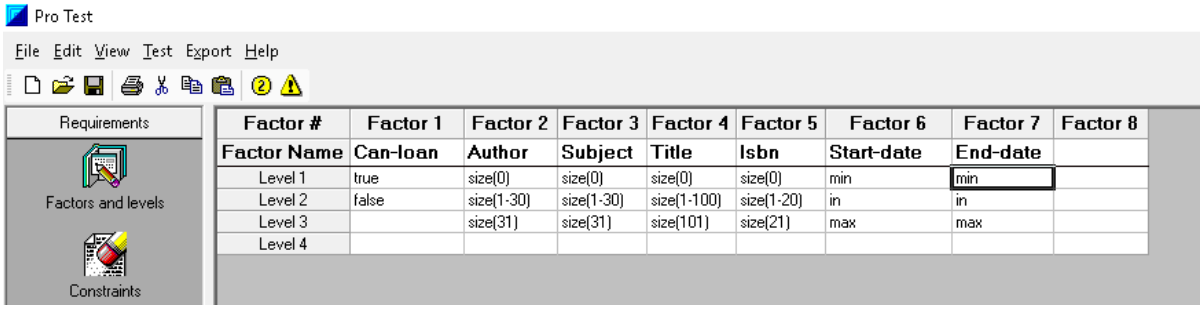


Figure 8 - Example of equivalence partitioning and boundary value analysis

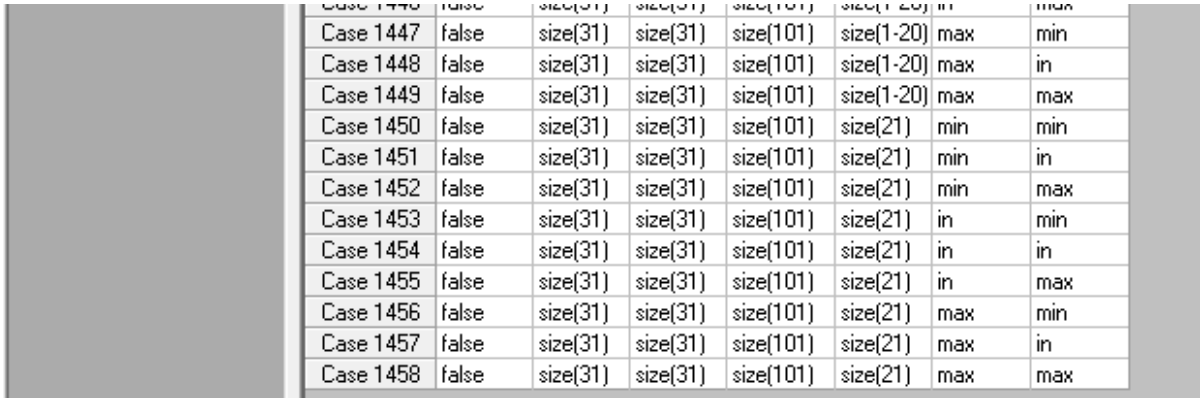
From the following cases only 3 were selected, that should provide sufficient coverage. The previously described process was applied to every attribute of the filter and afterwards All pairs method was used. At first the number of possible test cases resulted to be 1458, which was an amount impossible to test. Secondly a tool called Pro Test was used to reduce the number of test cases to sufficient amount. The result was 14 test cases, which were later implemented.

[11] Quoted from http://istqbexamcertification.com/what-is-black-box-specification-based-also-known-as-behavioral-testing-techniques (Retrieved 05/2017)



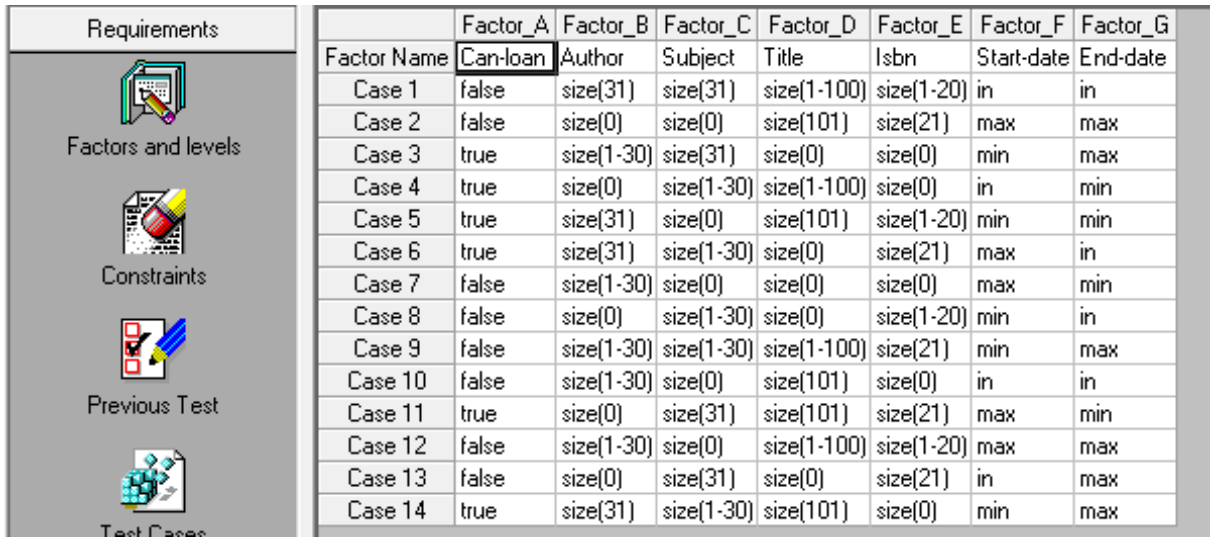
Factor #	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7	Factor 8
Factor Name	Can-loan	Author	Subject	Title	Isbn	Start-date	End-date	
Level 1	true	size(0)	size(0)	size(0)	size(0)	min	min	
Level 2	false	size(1-30)	size(1-30)	size(1-100)	size(1-20)	in	in	
Level 3		size(31)	size(31)	size(101)	size(21)	max	max	
Level 4								

Figure 9 - example of attributes in Pro Test tool



Case 1447	false	size(31)	size(31)	size(101)	size(1-20)	max	min
Case 1448	false	size(31)	size(31)	size(101)	size(1-20)	max	in
Case 1449	false	size(31)	size(31)	size(101)	size(1-20)	max	max
Case 1450	false	size(31)	size(31)	size(101)	size(21)	min	min
Case 1451	false	size(31)	size(31)	size(101)	size(21)	min	in
Case 1452	false	size(31)	size(31)	size(101)	size(21)	min	max
Case 1453	false	size(31)	size(31)	size(101)	size(21)	in	min
Case 1454	false	size(31)	size(31)	size(101)	size(21)	in	in
Case 1455	false	size(31)	size(31)	size(101)	size(21)	in	max
Case 1456	false	size(31)	size(31)	size(101)	size(21)	max	min
Case 1457	false	size(31)	size(31)	size(101)	size(21)	max	in
Case 1458	false	size(31)	size(31)	size(101)	size(21)	max	max

Figure 10 - Example combinatorial explosion of black box testing



	Factor_A	Factor_B	Factor_C	Factor_D	Factor_E	Factor_F	Factor_G
Factor Name	Can-loan	Author	Subject	Title	Isbn	Start-date	End-date
Case 1	false	size(31)	size(31)	size(1-100)	size(1-20)	in	in
Case 2	false	size(0)	size(0)	size(101)	size(21)	max	max
Case 3	true	size(1-30)	size(31)	size(0)	size(0)	min	max
Case 4	true	size(0)	size(1-30)	size(1-100)	size(0)	in	min
Case 5	true	size(31)	size(0)	size(101)	size(1-20)	min	min
Case 6	true	size(31)	size(1-30)	size(0)	size(21)	max	in
Case 7	false	size(1-30)	size(0)	size(0)	size(0)	max	min
Case 8	false	size(0)	size(1-30)	size(0)	size(1-20)	min	in
Case 9	false	size(1-30)	size(1-30)	size(1-100)	size(21)	min	max
Case 10	false	size(1-30)	size(0)	size(101)	size(0)	in	in
Case 11	true	size(0)	size(31)	size(101)	size(21)	max	min
Case 12	false	size(1-30)	size(0)	size(1-100)	size(1-20)	max	max
Case 13	false	size(0)	size(31)	size(0)	size(21)	in	max
Case 14	true	size(31)	size(1-30)	size(101)	size(0)	min	max

Figure 11 - Example of test case reduction using Pro Test

2.3.2.2 White box testing

White box testing is a testing technique, that examines the program structure and derives test data from the program logic/code. [12]

Advantages of white-box testing are:

- Efficient in finding errors and problems
- Required knowledge of internals of the software under test is beneficial for thorough testing
- Allows finding hidden errors
- Programmers introspection
- Helps to optimize the code
- Due to required internal knowledge of the software, maximum coverage is obtained

Some of the disadvantages of white-box testing are:

- Might not find unimplemented or missing features
- Requires high level knowledge of internals of the software under test
- Requires code access[13]

For calculation of all the execution paths of the method, the method was analyzed using control flow concepts. Then the cyclometric complexity was calculated. The control flow is demonstrated on the method GetPagedBooks in BookRepository class (see figure number 12).

It consists of 14 Nodes and 16 Edges, having two external method call. This values can be applied on the formula c = e - n + 2 * 1. The result is a cyclometric complexity of 6, mean a very low risk. The following are the test cases based on the cyclometric complexity.

TC	1	2	3	4	5	6
pageId	10	-1	10	10	10	10
pageSize	10	10	0	10	10	10
filter	Not null	Not null	Not null	Null	Not null	Not null
Map()	Not null	Not null	Not null	Not null	Null	Not null
SearchCatalogue()	Not null	Not null	Not null	Not null	Not null	Null
Result	Not null	Exception	Exception	Exception	Exception	Exception

Table 5 - test cases based on control flow analysis

[12] Quoted from https://www.tutorialspoint.com/software_testing_dictionary/white_box_testing.htm (Retrieved in 05/2017)

[13] Adapted from https://technologyconversations.com/2013/12/11/black-box-vs-white-box-testing (Retrieved in 05/2017)

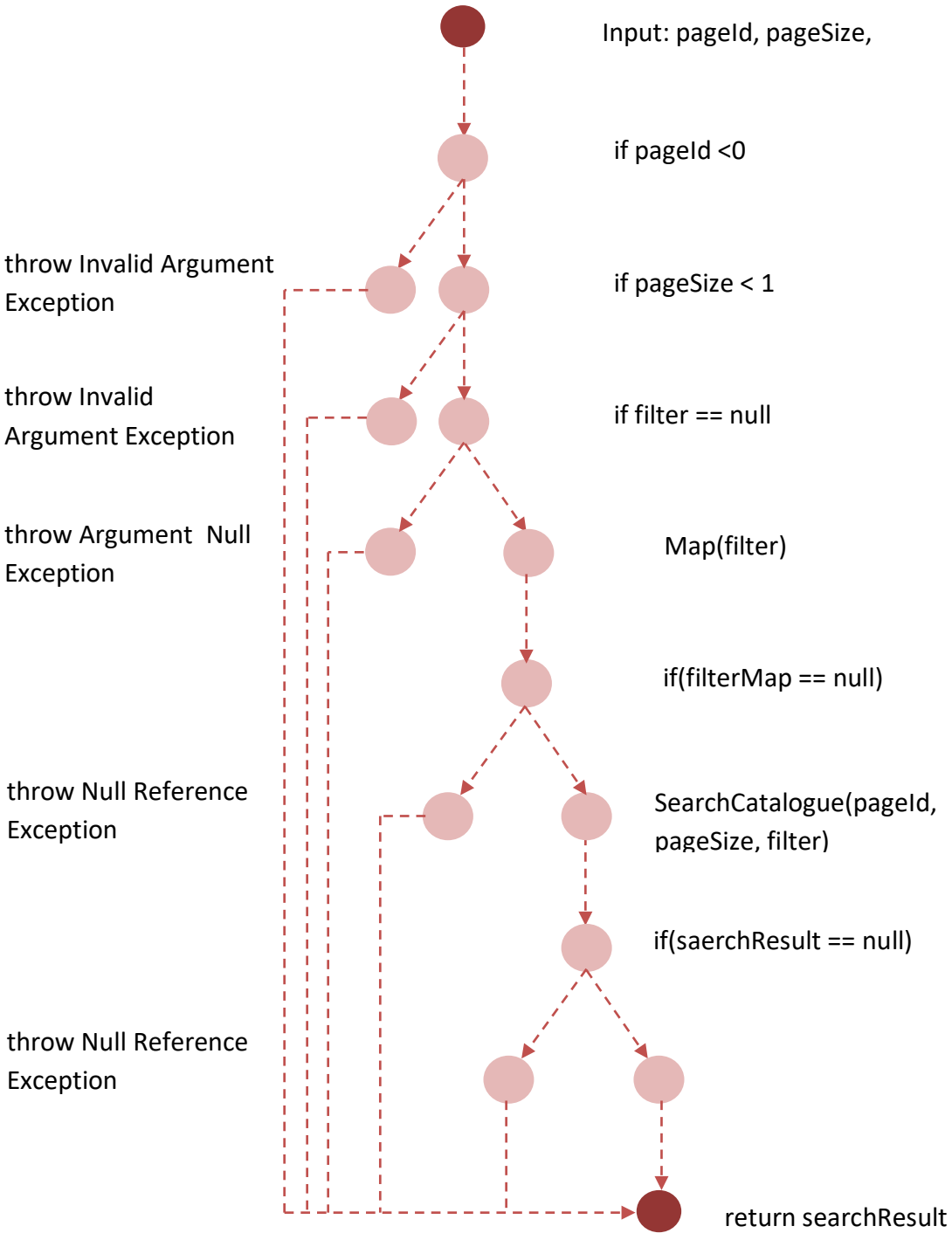


Figure 12 - example of cyclometric complexity of the GetPagedBooks method

2.3.3 Unit testing

A unit test is a piece of a code (usually a method) that invokes another piece of code and checks the correctness of some assumptions afterward. If the assumptions turn out to be wrong, the unit test has failed. A "unit" is a method or function. Unit testing is performed against a system under test (SUT).[14]

A proper unit test should have the following properties:

- It should be automated and repeatable.
- It should be easy to implement.
- Once it's written, it should remain for future use.
- Anyone should be able to run it.
- It should run quickly.[15]

2.3.3.1 Isolation

Unit test should exercise the unit in isolation. Unfortunately, the reality is that the SUT often has many dependencies, which prevent it. This is usually a sign of a bad code design described (see Testability - 2.2). Figure number 13 demonstrates the problem of dependencies of SUT. The number of dependencies can grow, up to a point that the SUT indirectly talks to a File system. In this case it is not a unit test, because it integrates with other components of the application.

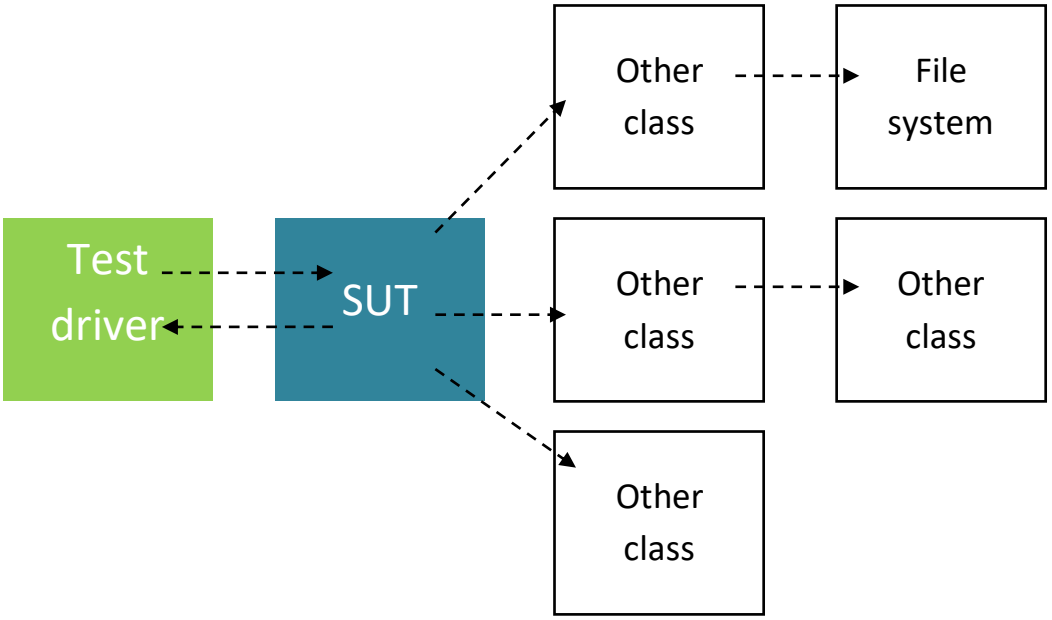


Figure 13 - Example of dependencies of SUT

[14] Quote from Roy Osherove: *The art of unit testing*, page 4
[15] Quote from Roy Osherove: *The art of unit testing*, page 6

To execute the test in isolation, it is necessary to have a "seam" and cut through the dependencies. The seam allows to replace the dependencies with friendlies, which can be seen on figure number 14. A friendly could be a mock, a stub or even a real class that is tested somewhere else and can be trusted. The important benefit is, that from a testing point of view, there is a choice and the friendly can be anything that is required for the purpose of the test, instead of it being an unwanted, uncontrollable dependency.

On the figure number 14, the arrows represent where the new operator is located and when the instantiation of the object happens. If the arrow started inside the SUT, construction of the class could not be controlled. Because the new operator was migrated into the test, the test is now responsible for constructing the object graph. Afterwards, the objects are passed through the constructor of the SUT and SUT then collaborates with the objects passed in. This allows to instantiate a subset of the application instead of testing it as a whole.

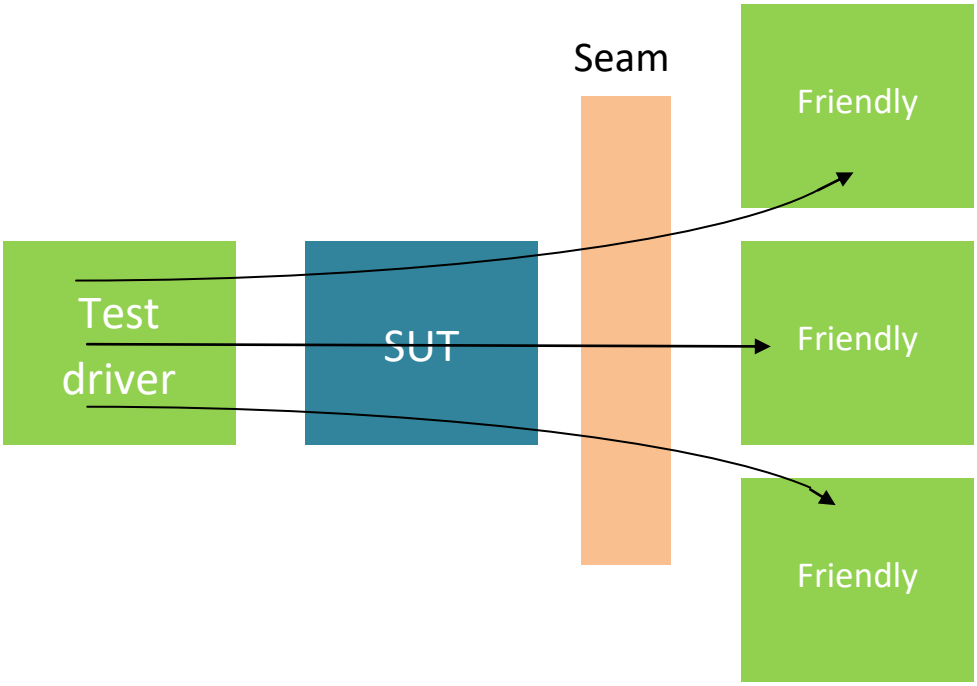


Figure 14 - Example of replacing SUT dependencies with friendlies

2.3.3.2 Mocking framework

Mock objects are instances of test-provided classes that simulate the behavior of external components. Mock objects isolate the application code under test. They create conditions that are otherwise difficult to produce[16].

[16] Quotes from https://msdn.microsoft.com/en-us/library/ff650441.aspx (Retrieved in 05/2017)

MOQ library was used to create mock objects. For the improvement of readability, mock objects are setup using extension methods, which can be seen on the figure number 15. It is important to mention, that MOQ library only creates instances of type Mock, but their role in the test can vary - it can either be a Mock or a Stub.

```
public static Mock<IMemberAccess> WithFindBySsnAndPasswordResult(this Mock<IMemberAccess> mock, IMemberMap memberMap)
{
    mock.Setup(m => m.FindBySsnAndPasswordAsync(It.IsAny<string>(), It.IsAny<string>())).ReturnsAsync(memberMap);
    return mock;
}
```

Figure 15 - Example of extension methods of mock object

Stubs replace an object so that we can test another object without problems. Figure number 16 shows the interaction between the stub and the SUT. The stub can never fail a test. The asserts the test uses are always against the SUT.[17]

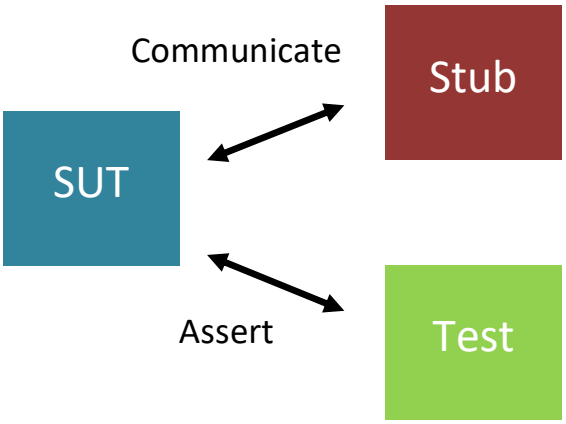


Figure 16 - Example of interaction of SUT and a stub

On the other hand, the test will use a mock object to verify whether the test failed or not. Figure number 17 shows the interaction between a test and a mock object. The assert is performed on the mock object and it is used to verify whether the test failed or not. [18]

[17] Adapted from Roy Osherove: *The art of unit testing*, page 85

[18] Adapted from Roy Osherove: *The art of unit testing*, page 85

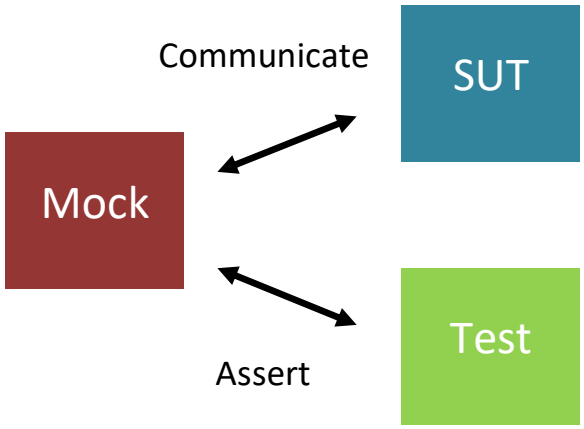


Figure 17 - Example of interaction of SUT, Mock and Test

An example of using Mock can be seen in the "InsertOneAsyncTest_ValidBook_HappyPath" test method on the figure number 18. The test verifies whether a method "InsertOneAsync" was called on the book access mock object. It is not a Stub, because there is an assert performed on it and it can affect the result of the test. On the other hand the book mapper mock object is passed to the SUT constructor, but there are no asserts performed on it, therefore it is a stub.

```
[TestMethod]
public async Task InsertOneAsyncTest_ValidBook_HappyPath()
{
    //arrange
    _mapper.WithMap(BookMapTestUtils.GenerateValidBookMap());
    _access.WithInsertResult(true);

    _sut = new BookRepository(_access.Object, _mapper.Object);

    //act
    var result = await _sut.InsertAsync(BookTestUtils.GenerateValidBook());

    //assert
    Assert.IsTrue(result);
    _access.Verify(m => m.InsertOneAsync(It.IsAny<BookMap>()), Times.Once);
}
```

Figure 18 - Example of usage of Mock object in the test method

2.3.3.3 Implementation and Standards

Unit tests were implemented using MS Test. To achieve implementation of tests in isolation (see Isolation - 2.3.3.1) Moq library was used. An example of an isolated unit test is a test that verifies that the method "FindBySsnAndPassword" in class "MemberRepository" (SUT), works as intended.

Before implementing the test itself, the test class has to be created. The name of the test class consists of the class name and the "Tests" suffix (e.g. MemberRepositoryTests). To improve readability, all the objects that will be used for the tests (SUT and friendlies) are included as private fields of the test class. Additionally, the setup and tear down methods are implemented. These methods determine, what will happen before and after each unit test and contain what all tests have in common. Usually, it is Mock object initialization and disposal.

```
16      [TestClass]
17      public class MemberRepositoryTests
18      {
19          private MemberRepository _sut;
20          private Mock<IMemberAccess> _access;
21          private Mock<IMapper<Member>> _mapper;
22
23          [TestInitialize]
24          public void SetUp()
25          {
26              _mapper = new Mock<IMapper<Member>>();
27              _access = new Mock<IMemberAccess>();
28          }
29
30          [TestCleanup]
31          public void TearDown()
32          {
33              _sut = null;
34              _mapper = null;
35              _access = null;
36          }
```

Figure 19 - Example of unit test standards

The name of the test method consists of three parts:

- Method Name—the name of the method, which is tested
- State Under Test—the conditions used to produce the expected behavior
- Expected Behavior—What it is expected of the tested method to do under the specified conditions (usually it is either a Happy path or an Exceptional case)[19]

Using this naming convention, the name of the test method for "FindBySsnAndPassword" is "FindBySsnAndPassword_ValidSsnAndPassword_HappyPath".

The structure of the Unit test is based on the Arrange-Act-Assert model. In the Arrange part, objects and variables are arranged. This includes the set up of the mock objects. The Arrange-Act-Assert model can be seen on figure number 20.

[19] Quoted from Roy Osherove: *The art of unit testing*, page 29

```
[TestMethod]
public async Task FindBySsnAndPassword_ValidSsnAndPassword_HappyPath()
{
    //arrange
    var validSsn = MemberTestUtils.GenerateValidSsn();
    var validPassword = MemberTestUtils.GenerateValidMember().Password;

    _access.WithFindBySsnAndPasswordResult(MemberMapTestUtils.GenerateValidMemberMap() as IMemberMap);
    _mapper.WithUnmap(MemberTestUtils.GenerateValidMember());

    _sut = new MemberRepository(_access.Object, _mapper.Object);

    //act
    await _sut.FindBySsnAndPassword(validSsn, validPassword);

    //assert
    _access.Verify(m => m.FindBySsnAndPasswordAsync(It.IsAny<string>(), It.IsAny<string>()), Times.Once);
}
```

Figure 20 - Example of the Arrange Act Assert model

2.3.4 Integration testing

Integration tests two or more dependent software modules as a group. It exercises many units of code that work together to evaluate one or more expected results, whereas unit test exercises one single unit in isolation.

2.3.4.1 Disadvantages of integration tests

Integration testing should not be used as a supplementary for unit testing, because it has a number of drawbacks associated with it. One of the downsides of an integration test, is that if it fails, it is not obvious what the cause of the failure is. The more components it exercises together, the more difficult it is to find the cause of the failure. For example, if an integration test tests 6 components, as shown on figure number 21, it might be difficult to find out whether the failure was caused by module number 1, 2, 3... or by an interaction between them.

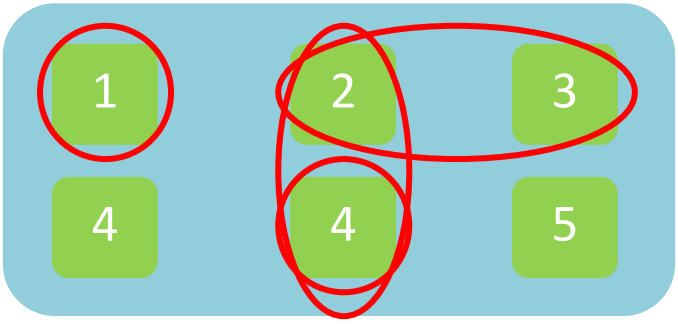


Figure 21 - Potential cause of failure of an integration test

Another problem with integration tests, is the number of them needed to test the system. On figure number 22 there is an integration test that exercises 3 different components. The number inside the component represents the number of execution paths of the component and the red arrows represent the component's interactions. In case that (due to a bad code design) it is only possible to test those components as a cluster, the number or tests necessary to reach sufficient coverage can be calculated by multiplying the number of execution paths of each component. Therefore 105 tests are needed.

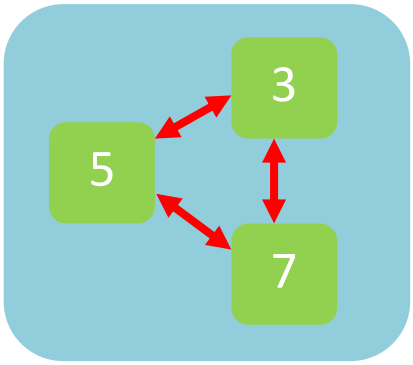


Figure 22 - Integration test exercising 3 components

Additionally, if a new component is introduced after refactoring and the number of execution paths is reduced, the number of tests needed grows. On figure number 23, there are the same components after refactoring. The number of tests required to reach sufficient coverage grew to 180.

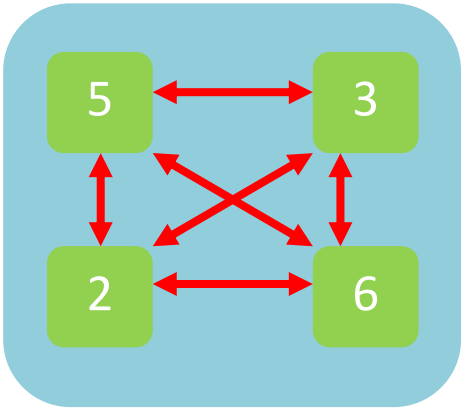


Figure 23 - Components integrations after refactoring

2.3.4.2 Necessity of integration tests

Integration tests become necessary, when integrating with external components outside the boundary of the system, such as database, libraries or other services. In the GTL project, integration tests were implemented using the black box testing technique (see Black box testing - 2.3.2.1). Figure number 24 shows, how were integration tests implemented.

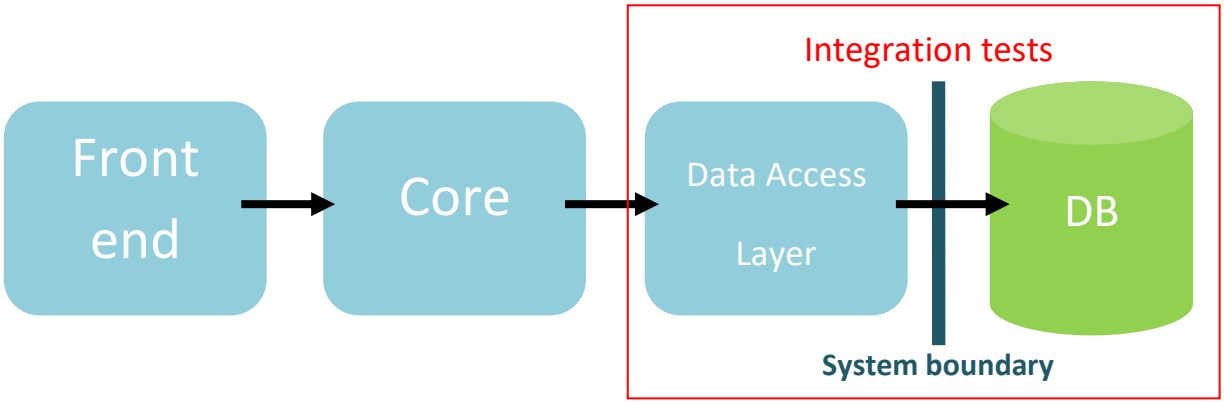


Figure 24 - Integration testing at the system boundary

An example of an integration test is searching the book catalogue as shown on figure number 25. The integration test only exercises the book access component and the database and verifies that the database returns a particular set of books.

```
//TC-I-15
[TestMethod]
public async Task SearchCatalogue_TCI15_HappyPath()
{
    //arrange
    const int pageId = 1;
    const int pageSize = 5;

    var filter = new CatalogueFilterMap()
        .WithLoanability(true)
        .WithAuthor("J.K. Rowling")
        .WithSubject("Fiction")
        .WithTitle("Harry Potter")
        .WithIsbn(string.Empty)
        .WithStartDate(DateTime.MinValue)
        .WithEndDate(DateTime.MaxValue);

    //act
    var result = await _sut.SearchCatalogue(pageId, pageSize, filter);

    //asser
    Assert.IsNotNull(result);
    Assert.IsTrue(result.Count(b => true) == 2);
}
```

Figure 25 - Example of integration test

2.3.5 System tests

System testing is the type of testing to check the behavior of a complete and fully integrated software product based on the requirements specification document (see System requirements specification document - 5.1.1) The main focus of this testing is to evaluate Business / Functional / End-user requirements. [20]

[20] Quoted from http://www.softwaretestingclass.com/system-testing-what-why-how (Retrieved in 05/2017)

2.3.5.1 Selenium

In the GTL project, system tests were implemented using the Selenium web driver. Web driver is a web automation framework that allows to executing tests against different browsers. [21] It provides several useful features that allow testing from the user's perspective (for example figure number 26 shows how selenium driver verifies if a certain element has expected value).

```
internal virtual VerificationTestStep HasValue(string value)
{
    return new VerificationTestStep(driver =>
    {
        var actualValue = FindElement(driver).GetAttribute(Attributes.Value);
        Assert.IsTrue(actualValue.Equals(value), "Element value is not equal to the expected value.");
    });
}
```

Figure 26 - Verifying element's value with selenium

2.3.5.2 Test framework

System tests are implemented by testers, rather than developers. While Selenium is a great tool for system testing, its capabilities might disassociate from the system requirements and therefore system tests might be difficult to implement. To overcome this issue, a testing framework was implemented.

A test framework serves as an abstraction to the web driver and contains a set of test steps that directly mimic the test steps in the test case specification (see Test case specification - 2.1.6.1). This abstraction allows the tester to focus on the business requirements and specification rather than development, while implementing the test. An example of test step can be demonstrated on a test case that verifies a valid user registration. In the test case, there are a procedural test steps specified, written in a form of action - verification pairs. An example of an action test step is: "*Enter value 111-00-0000 to the input field with name Ssn and defocus the input field.*". An implementation of such test step can be seen on figure number 27.

```
//Action: Find element with Name Ssn and set its value.
_testCase.AddAction(TestStep.OnPage.Register.Element.WithName("Ssn").EnterValue(ssn));
```

Figure 27 - Example of action test step implementation

[21] Quoted from http://www.guru99.com/introduction-webdriver-comparison-selenium-rc.html (Retrieved in 05/2017)

The test framework was implemented using the command pattern. Every test has exactly one test case object. A test starts with its initialization. Throughout the test, the test steps are added to a queue of the test case. Finally, every test case has to call the execution method and all test steps are executed. A system test example can be seen of figure number 28.

```
[TestClass]
public class Register : SeleniumTest
{
    private ITestCase _testCase;

    [TestInitialize]
    public void SetUp()
    {
       _testCase = new TestCase();
    }

    [TestCleanup]
    public void TearDown()
    {
        _testCase.IsValid();
        _testCase.Dispose();
    }

    /// <summary>
    /// TC-S-1
    /// Test that verifies member registration with valid ssn and password.
    /// </summary>
    [TestMethod()]
    public void Register_ValidInput_HappyPath()
    {
        const string ssn = "111-00-0000";
        const string password = "MyPassword1.";
        const string confirmPassword = "MyPassword1.";

        //Action: Navigate browser to registration page
        _testCase.AddAction(TestStep.Browser.GoToLocation.Register);

        //Verififcation: Verify that browser is on the registration page.
        _testCase.VerifyThat(TestStep.Browser.HasLocation.Register);
```

Figure 28 - Example of a system test

2.4 Test conclusion

One of the biggest challenges of this project was to efficiently combine development and testing. Ever since the beginning, the question has raised - how to combine planning aspects of testing and the agile approach of development? The path that was taken for this project can be seen on the test approach itself - it is a combination of approaches that are quite contrasting.

The very same problem raised during documentation. On one hand, there is an Agile methodology suggesting only the documentation that is necessary, on the other hand, there is a number of deliverables from the testing point of view. For example, documentation of the component test cases felt almost redundant and it was very time consuming, especially when unit tests had the biggest focus. Another issue was the nature of the scrum sprints - it was unknown what was going to be developed before the sprint planning meeting and therefore the test cases could not be prepared ahead of time.

On the other hand, documentation and risk-based testing approach have proven very useful when it comes to higher level end-to-end testing. This was done after the development phase, when all the test cases were prepared.

Finally, the testing phase went very well and almost all the exit criteria have been fulfilled. The Core module has reached the test coverage of 75% and the Data access module has reached 50%, while there are no high severity defects. The reason why the exit criteria have not been completely reached, is the fact that not all the risks with the priority factor above 15 have been end-to-end tested. End-to-end tests were very time consuming to develop, especially with the undertaken development approach.

In conlusion, it is important to say that a lot of decisions were taken mainly for the learning purposes. We have learned a lot from the planning part of the project - creating a test plan and strategy; as well as implementation - developing a testable software and implementing test properly.

3 Database

The Database section focuses on the database aspects of the GTL project. It starts with describing how Entity relationship diagram is formed and mapped to relational database schema. Database schema mapping is followed by a section called SQL statements, describing how database tables are created and populated. The second part of the Database section focuses on designing five complex queries in both relational algebra and SQL. Finally, the performance and execution plan of the queries is analyzed.

3.1 Entity relationship diagram

An Entity Relationship Diagram (ERD) is a type of flowchart that illustrates how entities such as people, objects or concepts relate to each other within a system. They are used to model and design relational databases, in terms of logic and business rules.

ER Diagrams are composed of entities, relationships and attributes. They also depict cardinality, which defines relationships in terms of numbers.[22]

Creation of an ER diagram was an initial step in the GTL project. It helped to determine business requirements and was later used to build relation database schema. The ER diagram of the GTL project can be seen on figure number 29.

[22] Adapted from https://www.lucidchart.com/pages/er-diagrams (Retrieved in 05/2017)

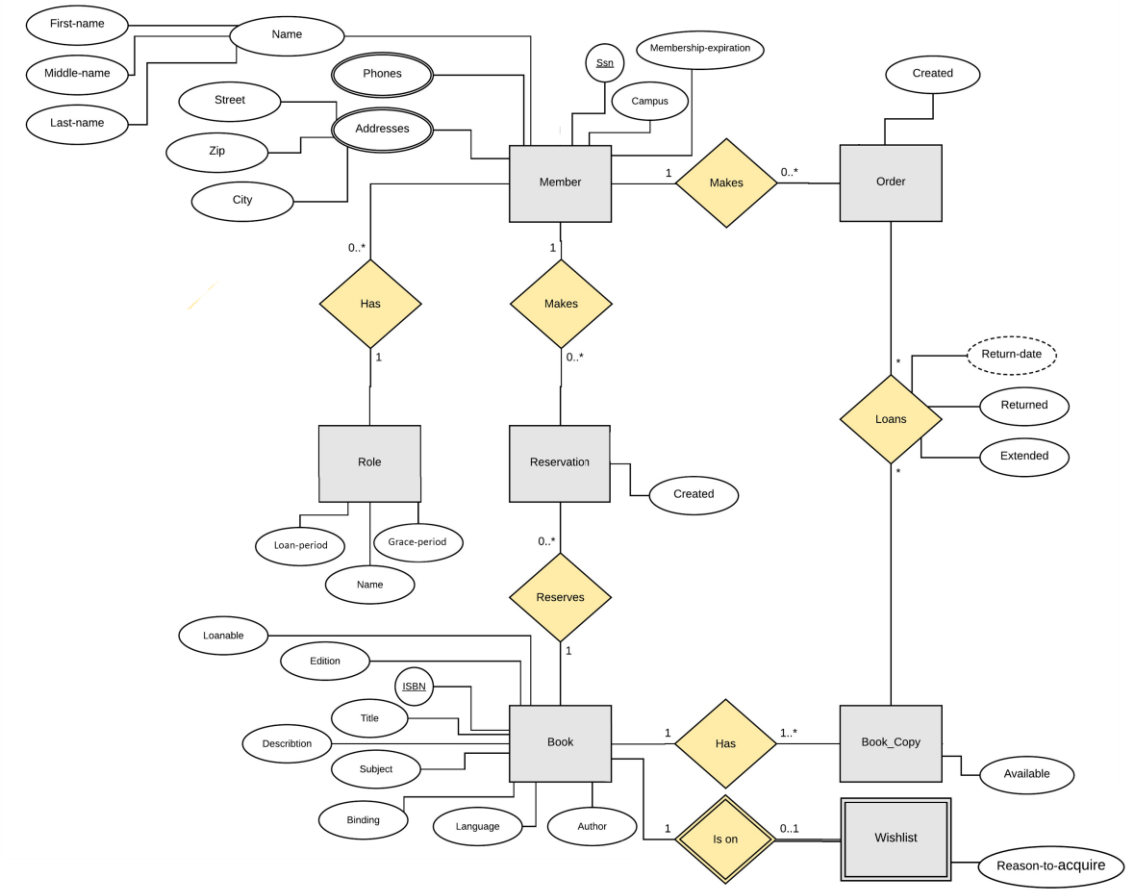


Figure 29 - Entity relationship diagram

3.1.1 Entities

An entity is a real-world object, that is easily identifiable [23] (e.g. in the GTL project it is book, member or reservation). An entity can be categorized as strong, weak or associative. A strong entity can be defined solely by its own attributes [24]. An example of a strong entity is a Book. A weak entity cannot be defined by its own attributes, for example a Wishlist - a book on wish list cannot be defined without an existing book. Associative entity is a kind of relationship about which information is stored. It can only exist between two other entities. An example of an associative entity is a Loan that exists between the Order entity and the Book_copy entity.

Entity keys refer to attributes that uniquely define an entity. Entity keys can be a super key, candidate key or primary key. Super key is a set of attributes (one or more) that together defines an entity. A candidate key is a minimal super key, meaning it has the least possible number of attributes to still be a super key. A primary key is a candidate key chosen to uniquely identify the entity set.[25]

[23] Adapted from https://www.tutorialspoint.com/dbms/er_model_basic_concepts.htm (Retrieved in 05/2017)
[24] Quote from https://www.lucidchart.com/pages/er-diagrams (Retrieved in 05/2017)
[25] Adapted from https://www.lucidchart.com/pages/er-diagrams (Retrieved in 05/2017)

3.1.2 Relationships

A relationship describes how entities act upon each other or are associated with each other. It describes the dependence between a parent and a child. A relationship can be either strong or weak. A strong relationship is used when a primary key of child entity does not contain primary key of a parent entity. An example of a strong relationship is the relationship between the Member entity and the Order entity. A weak relationship is used when a child is existentially dependent on the parent and primary key of child entity contains a primary key component of a parent. An example of a weak relationship is between the Book and Wishlist entities, because the primary key of Wishlist contains the primary key of the Book.

Degree of a relationship is the number of entities participating in that relationship. A degree can be binary (degree of 2), ternary (degree of 3) or n-ary (degree of n). In the GTL project, only binary relationships are used.

3.1.3 Attributes

An attribute is a characteristic of an entity. Each attribute has a value, while there exists a domain or range of atomic values that can be assigned to attributes. A domain consists of a name, data type and format (e.g. Book_ISBN - a unique numeric book identifier consisting of either 10 or 13 digits and additional dashes such as "99921-58-10-7").

There are different types of attributes:

- Simple attribute - has an atomic value that cannot be divided further (e.g. Author of the book).
- Composite attribute - is made of more than one simple attribute (e.g. Member name consists of first name, middle name and last name)
- Derived attribute - is calculated or otherwise derived from another attribute (e.g. Return date of the Loan)
- Multi-valued attribute - contains more than one value (e.g. Member's Addresses or Phones)[26]

[26] Adapted from https://www.lucidchart.com/pages/er-diagrams (Retrieved in 05/2017)

3.1.4 Cardinality

Cardinality defines the number of entities in one entity set, which can be associated with the number of entities of another set via relationship set. There are different types of cardinalities:

- One to one - one entity from an entity set can be associated with at most one of another entity set (e.g. cardinality between Wishlist and Book).

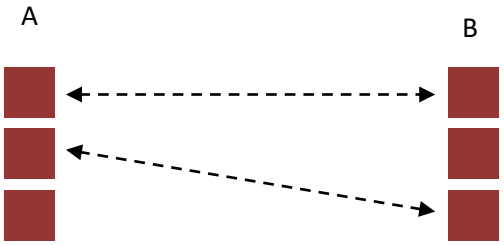


Figure 30 - One to one cardinality

- One to many - one entity from set A can be associated with more than one entities of set B, while an entity from set B can be associated with at most one entity from set A.

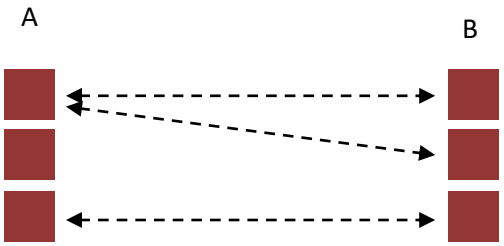


Figure 31 - One to many cardinality

- May to many - One entity from A can be associated with more than one entity from B and vice versa. [27]

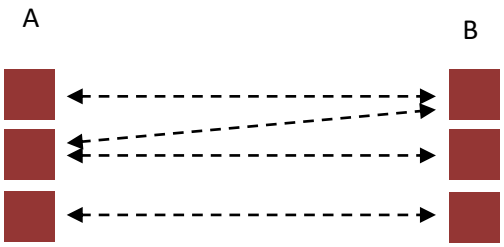


Figure 32 - Many to many cardinality

[27] Adapted from https://www.lucidchart.com/pages/er-diagrams (Retrieved in 05/2017)

3.2 ERD Mapping

Once the entity relationship diagram is satisfactory, the database design has to be implemented. To do so, relations must be decided as well as their attributes and constrains. This transition from ER diagram to relation schema is done in a number of steps, which ensure that the diagram is mapped correctly.

3.2.1 Strong entity types

The first mapping step is the mapping of strong entity types. For each strong entity type, a relation R is created. Relation R includes all simple attributes of the entity it is being mapped from. Key attributes are included and the primary key is chosen. After the first step, six relations are created:

Member					
Ssn	Campus	Membership-expiration	First-name	Middle-name	Last-name

Role	
Id	Name

Reservation	
Id	Created

Order	
Id	Created

Book								
ISBN	Title	Subject	Author	Language	Binding	Edition	Description	Can-loan

Book_Copy	
Id	Available

3.2.2 Weak entity types

Weak entity types are similarly mapped into their own relation. The primary key of the relation is a combination of owner's primary key and the possible discriminator of the weak entity. Furthermore, the primary key of the owner becomes a foreign key of the relation.

An example of Weak entity mapping is the Wishlist. The relation created from the Wishlist entity can be seen below. Its primary key is Isbn (the primary key of the Book relation), while it is also a foreign key. No additional discriminators were necessary, because of the 1 to 1 relationship between the Book and Wishilist entities.

Wishlist	
Isbn	Reason-to-acquire

3.2.3 Binary 1:1 relationships

The implementation of relationships involve the use of foreign keys. It is important to consider both the degree and the cardinality of a relationship.

When it comes to 1 to 1 relationship, there is a choice of how is the relationship going to be implemented. The foreign key can be in one of the two relations or in both. The attributes defined for the relationship are placed to the same table than foreign key.

3.2.4 Binary 1:N relationships

When 1 to many relationships are mapped, the foreign key as well as the relationship attributes are placed to the side with cardinality of many. The following are the modified relations after the one to many relationship mapping:

Member					
Ssn	Campus	Membership-expiration	First-name	Last-name	Role-id

Reservation			
Id	Member-ssn	Book-isbn	Created

Order		
Id	Member-ssn	Created

Book_Copy		
Id	Book-isbn	Available

3.2.5 Binary M:N relationships

Every many to many relationship is implemented as a separate relation. The new relation has a composite primary key corresponding to primary keys of the relations implementing the participating entity types together with possible discriminator attribute. Any attributes defined for the relationship are included in a new relation.

An example of mapping many to many relationship is the "Loans" relationship between the Order and Book_copy entities. The new relation "Order_line" was created with a composite primary key Order-id and Book-copy-id. Additionally, Loans relationship attributes were included in the Order_line relation. Order_line relation can be seen below.

Order_line			
Order-id	Book-copy-id	Returned	Extended

3.2.6 Multi-valued attributes

All attributes except for the derived and composite attributes, must appear in relations. Derived attributes are included if their presence improves performance.

Each multi-valued attribute is implemented using a new relation. The primary key of the relation is a composite primary key between the primary key of the entity it was assigned to and the multi-valued attribute itself. In the new relation, the attribute is no longer considered multi-valued.

An example of mapping of a multi-valued attribute is a multi-valued attribute of the Member entity - Phones. A new relation Phone was created, with the composite primary key Member-ssn and phone. Member entity had another multi-valued attribute - Addresses. In this case the attribute was also composite and consisted of Street, Zip and City attributes. Similarly, a new relation Address was created, with primary key Member-ssn and Street. The rest of the composite attributes were included in the relation. Both Phone and Address relations can be below.

Phone	
Member-ssn	Phone

Address			
Member-ssn	Street	Zip	City

3.3 Database normalization

Database Normalization is a technique of organizing the data in the database. Normalization is a systematic approach of decomposing tables to eliminate data redundancy and undesirable characteristics like Insertion, Update and Deletion anomalies. It is a multi-step process that puts data into tabular form by removing duplicated data from the relation tables. [28]

3.3.1 First normal form

The first normal form (1NF) says, that the information is stored in a relational table and each column contains atomic values, and there are not repeating groups of columns.

3.3.2 Second normal form

The second normal form says, that for a table to be in a second normal form, it must first satisfy all rules specified by the first normal form. Additionally, it says, that any partial dependencies between a prime and nonprime attributes are forbidden. The prime attribute is an attribute that occurs in a candidate key, on the other hand non-prime attribute is an attribute that does not occur in any candidate key. Figure number 33 demonstrates the violation of the second normal form.

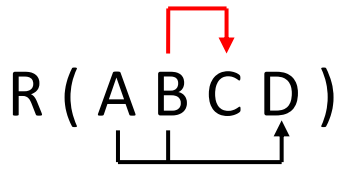


Figure 33 - Violation of the second normal form

3.3.3 Third normal form

The third normal form says, that in order for a table to be in a third normal form, it must first satisfy all rules specified by the second normal form. Additionally, it says, that every non-prime attribute of the table must be dependent on the primary key, or in other words, there should not be the case that a non-prime attribute is determined by another non-prime attribute. Figure number 34 demonstrates a violation of the third normal form.

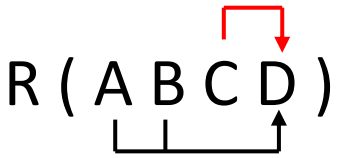


Figure 34 - Violation of the third normal form

[28] Quote from http://www.studytonight.com/dbms/database-normalization.php (Retrieved in 05/2017)

3.3.4 Boyce and Codd Normal Form

Boyce and Codd Normal Form is a higher version of the Third Normal form. This form deals with a certain type of anomaly that is not handled by 3NF. A 3NF table which does not have multiple overlapping candidate keys is said to be in BCNF. For a table to be in BCNF, following conditions must be satisfied: Table must be in third normal form and additionally, each functional dependency (X -> Y), X should be a super Key. A super key is a set of attributes within a table whose values can be used to uniquely identify a tuple. Figure number 35 demonstrates violation of the BCNF.

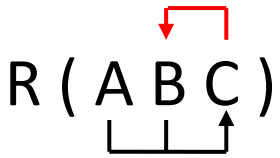


Figure 35 - Violation of the Boyce and Codd normal form

3.3.5 Verifying normal forms

According to the rules described in chapters 3.3.1 - 3.3.4 Tables of the GTL projects have been verified. An example is the Member table. Its relation schema together with functional dependencies can be seen on figure number 36. Its candidate key is Ssn and its functional dependencies are: Ssn -> Campus, Ssn -> Membership-expiration, Ssn -> First-name, Ssn -> Last-name, Ssn -> Role-id.

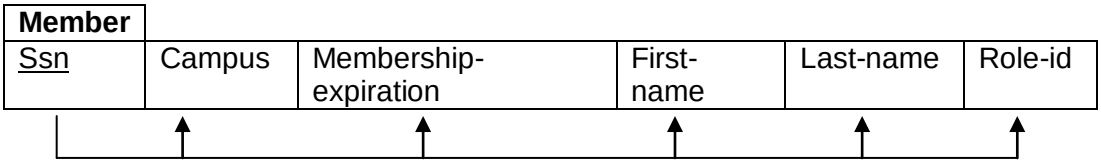


Figure 36 - Functional dependencies of the Member relation

Since there are no partial dependencies, it can be assumed that the table is in second normal form. There are no transitive dependencies - there is not a non-prime attribute depending on a non-prime attribute, it can be assumed that the table is in third normal form. Finally, all determinants are super keys, therefore it can be concluded that the table is in Boyce and Codd Normal Form.

Using similar approach, it was concluded that all except one table are in BCNF. The only table that is not in BCNF is the table Address, which can be seen on figure number 37. This table is in second normal form, because it has no partial dependencies. However, it is not in third normal form, because a non-prime attribute Zip is functionally dependent on a non-prime attribute City.

A way to fix this issue, would be to remove City from the relation and put it to a new relation, where Zip would be a primary key. Since the candidate key Member-ssn has a functional dependency with City it was concluded that the functional dependency between Zip and City does not oppose a thread and such solution was considered an unnecessary optimization.

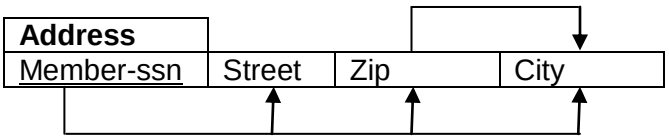


Figure 37 - Functional dependencies of the Address relation

3.4 Relational Algebra

Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output. It uses operators to perform queries. The fundamental operations of relational algebra are:

- **Selection** - selects tuples that satisfy the given predicate from a relation. It has notation σ p(r), where σ selection predicate, r stands for relation and p is a propositional logic formula.
- **Projection** - projects column(s) that satisfy a given predicate. It has notation ∏A1, A2, An (r), where A1, A2 and An are attribute names of relation r. Since the relation is a set, duplicate rows are eliminated.
- **Union** - performs a binary union between two given relations. It has notation r U s.
- **Set difference** - a result of set difference operation is tuples, which are present in one relation but are not in the second relation. It has notation r - s.
- **Cartesian product** - combines information of two different relations into one. It has notation r X s.
- **Rename** - The rename operation allows to rename the output relation. It has notation ρ x (E), where the result of expression E is saved with name of x.[29]
- **Natural join** - The result of the natural join is the set of all combinations of tuples in R and S that are equal on their common attribute names. It has notation r ⋈ s, where r and s are relations.

[29] Adapted from https://www.tutorialspoint.com/dbms/relational_algebra.htm (Retrieved in 05/2071)

3.5 SQL Statements

The following section describes different types of SQL statements as well as stored procedures and views, their advantages and usage.

3.5.1 Data Definition Language

Data Definition Language (DDL) statements are used to build and modify the structure of tables and other objects in the database. When DDL statement is executed, it takes effect immediately. They can perform tasks, such as:

- Create, alter, and drop schema objects
- Grant and revoke privileges and roles
- Analyze information on a table, index, or cluster
- Establish auditing options
- Add comments to the data dictionary [30]

An example of a DDL statement is a command that creates a reservation table. The statement can be seen on figure number 38. The create table reservation statement is a part of a SQL script that creates the whole database according to the database schema (see Create Database script - 5.3.1).

```
Create table [Reservation](
    [Id]                    int             not null    identity(1,1),
    [Member-ssn]            varchar(20)     not null,
    [Book-isbn]             varchar(20)     not null,
    [Created]               date            null        default(GETDATE()),

    primary key ([Id]),
    foreign key ([Member-ssn]) references [Member]([Ssn]),
    foreign key ([Book-isbn]) references [Book]([Isbn])
);
```

Figure 38 - Example of SQL DDL statement

The "CREATE", "ALTER", and "DROP" commands require exclusive access to the specified object. For example, a "DROP TABLE" statement fails if another user has an open transaction on the specified table. [31] This failure can be observed, for example, when the create table scripts attempts to drop possibly existing tables, but another thread has an open transaction on any of the tables the script attempts to drop. Due to this reason, any executing threads on the Library_case database are killed at the beginning of the script.

[30] Adapted from https://docs.oracle.com/cd/B14117_01/server.101/b10759/statements_1001.htm (Retrieved in 05/2017)
[31] Quoted from https://docs.oracle.com/cd/B14117_01/server.101/b10759/statements_1001.htm (Retrieved in 05/2017)

```
Declare @id int
Declare @kill varchar(1000);
Declare @getid cursor

set @getid = cursor for
select [SPID] from #sp_who where [DbName] = 'Library_case';

open @getid
fetch next from @getid into @id

while @@FETCH_STATUS = 0
begin
    print ('Deleting process' + CAST(@id as varchar(4)));
    set @kill = 'KILL ' + CAST(@id as varchar(4))
    exec (@kill);
    fetch next from @getid into @id
end

close @getid
deallocate @getid
--End kill process
```

Figure 39 - Example of thread killing script, before DB can be created (recreated)

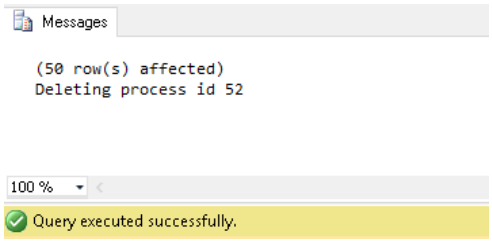


Figure 40 - Output of the thread killing script

3.5.1.1 Integrity constrains

Constraints can be created on a single or multiple columns of any table. They maintain the data integrity of the table.

Integrity constrains can be used to restrict values in the database. Constraints are part of a database schema definition and help to maintain data integrity of the table. There are six types of integrity constrains:

- NOT NULL - prohibits a database value from being null
- UNIQUE - prohibits multiple rows from having the same value in the same column or combination of columns but allows some values to be null
- PRIMARY KEY - combines a NOT NULL constraint and a unique constraint in a single declaration
- FOREIGN KEY - requires values in one table to match values in another table
- CHECK - requires a value in the database to comply with a specified condition
- DEFAULT - assigns a default value if not value is specified. [32]

[32] Adapted from https://www.tutorialspoint.com/sql/sql-constraints.htm (Retrieved in 05/2017)

3.5.1.2 Not null constraint

A NOT NULL constraint prohibits a column from containing nulls. The NULL keyword does not actually define an integrity constraint, but it can be specified to explicitly permit a column to contain nulls. NOT NULL and NULL must be defined using inline specification. If neither NOT NULL nor NULL is specified, then the default is NULL. [33]

3.5.1.3 Unique constraint

A unique constraint designates a column as a unique key. A composite unique key designates a combination of columns as the unique key. To satisfy a unique constraint, no two rows in the table can have the same value for the unique key. However, the unique key made up of a single column can contain nulls. To satisfy a composite unique key, no two rows in the table or view can have the same combination of values in the key columns. Any row that contains nulls in all key columns automatically satisfies the constraint. However, two rows that contain nulls for one or more key columns and the same combination of values for the other key columns violate the constraint. [34]

3.5.1.4 Primary key constraint

A primary key constraint designates a column as the primary key of a table or view. A composite primary key designates a combination of columns as the primary key. A primary key constraint combines a NOT NULL and unique constraint in one declaration. Therefore, to satisfy a primary key constraint, no primary key value can appear in more than one row in the table, as well as no column that is part of the primary key can contain a null. Furthermore, there can only be one primary key for a table and it is not possible to designate the same column (or a combination of columns) as both primary key and unique key. [35]

3.5.1.5 Foreign key constraint

A foreign key constraint (also called a referential integrity constraint) designates a column as the foreign key and establishes a relationship between that foreign key and a specified primary or unique key, called the referenced key. A composite foreign key designates a combination of columns as the foreign key. The table or view containing the foreign key is called the child object, and the table or view containing the referenced key is called the parent object.

[33] Quoted from https://www.tutorialspoint.com/sql/sql-constraints.htm (Retrieved in 05/2017)
[34] Quoted from https://www.tutorialspoint.com/sql/sql-constraints.htm (Retrieved in 05/2017)
[35] Quoted from https://www.tutorialspoint.com/sql/sql-constraints.htm (Retrieved in 05/2017)

To satisfy a composite foreign key constraint, the composite foreign key must refer to a composite unique key or a composite primary key in the parent table or view. The same column (or a combination of columns) can be both a foreign key and a primary key.[36] An example of this is Isbn column in the Wishlist table on figure number 43. It is also possible to define multiple foreign keys on the same table or view.

```
Create table [Wishlist](
    [Isbn]                  varchar(20)     not null,
    [Reason-to-acquire]     varchar(200)    not null,

    primary key ([Isbn]),
    foreign key ([Isbn]) references [Book]([Isbn])
);
```

Figure 41 - Example of DDL statement for creation of the Wishlist table

3.5.1.6 Check constraint

A check constraint specifies a condition that each row in the table must satisfy. To satisfy the constraint, each row in the table must make the condition either TRUE or unknown (due to a null). The condition of a check constraint can refer to any column in the table, but it cannot refer to columns of other tables.

3.5.1.7 Default constraint

The default constraint provides a default value to a column when the INSERT INTO statement does not provide a specific value. An example of the default constraint can be seen on figure number 42. When a user attempts to insert new Reservation record and does not specify a date, SQL inserts the result of Getdate() function as a default value.

[36] Quoted from https://www.tutorialspoint.com/sql/sql-constraints.htm (Retrieved in 05/2017)

3.5.2 Data Manipulation Language

Data Manipulation Language (DML) statements to access and manipulate data in existing schema objects. These statements do not implicitly commit the current transaction. The DML statements are - CALL, DELETE, EXPLAIN PLAN, INSERT, LOCK TABLE, MERGE, SELECT, UPDATE. The SELECT statement is a limited form of DML statement, because it can only access data in the database and cannot manipulate them. However, it can operate on the accessed data before returning the results of the query. [37]

An example of a simple DML statement on figure number x. This statement is a part of the "PopulateDb" script, used to populate the database with test data.

```
83   Insert into [Book_Copy] ([Available], [Book-isbn]) values ('true', '0439139603');
84   Insert into [Book_Copy] ([Available], [Book-isbn]) values ('false', '0439139605');
```

Figure 42 - Example of SQL DML statement

3.5.3 Stored procedures

A stored procedure is a set of SQL statements with an assigned name, which are stored in RDBMS as a group, so it can be reused and shared by multiple programs.[38] Using stored procedures over ad-hoc queries has a number of advantages, which are described in the following chapters.

An example of a stored procedure that returns most popular books of the most popular subject, taking the start date, end date and amount as parameters (see Query 3 - 3.8.1.3).

3.5.3.1 Reusability of the execution plan

When a query is issued to SQL server, the syntax of the query is checked, the query is compiled and finally, an execution plan is generated. The execution plan is the query optimizer's attempt to calculate the most efficient way to implement submitted query. After the execution plan is generated, query is executed. When a query is run again, execution plan is reused.

While ad-hoc queries are able to reuse execution plan, even a slight change of query (like changing the value of the parameter) will cause the need to generate a new execution plan. On the other hand, stored procedures will reuse the execution plan even when different parameters are passed to them.

[37] Quoted from https://docs.oracle.com/cd/B14117_01/server.101/b10759/statements_1001.htm (Retrieved in 05/2017)
[38] Quoted from http://searchoracle.techtarget.com/definition/stored-procedure (Retrieved in 05/2017)

3.5.3.2 Reduction of network traffic

Stored procedures can be incredibly long and can contain a lot of logic within them. However, to execute them, only three things are necessary - the "execute" key work, a name of the stored procedure and the parameters. This is possible, because stored procedure resides on SQL server itself. If the stored procedure was not used, all the ad-hoc SQL statements would have to be sent over the network to SQL server, which could have a negative impact on the network traffic.

3.5.3.3 Maintainability

Several applications can reuse a particular stored procedure. If there is a bug in the stored procedure or the requirements have changed, the stored procedure is the only one place the has to be changed. While if the bug is in the inline SQL statement, the change would most likely have to be done in multiple places, which makes maintainability difficult.

3.5.3.4 Security

Stored procedures can be used to grain control. If a user requires access to a specific table, instead of granting him access to the entire table, where he can access all records, stored procedure can be created. The user can have access to execute the stored procedure, while he/ she does not have access to the underlying table itself. Another approach, would be the usage of Views, which provide a similar benefit (see Views - 3.5.4). Additionally, stored procedure provide security by avoiding SQL injection attacks.

3.5.4 Views

A view is a SQL statement that is stored in the database with an associated name. It is a composition of a table in the form of a predefined SQL query. A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depend on the written SQL query to create a view. Views, which are a type of virtual tables allow users to do the following:

- Structure data in a way that users or classes of users find natural or intuitive.
- Restrict access to the data in such a way that a user can see and (sometimes) modify exactly what they need and no more.
- Summarize data from various tables which can be used to generate reports.

An example is a view that returns the number of books that were published this year, grouped by subject (see figure number 45). In order to execute the query, only the view name needs to be specified. The disadvantage of the view is that it cannot accept parameters. If the requirement changes and the user would want to specify the year by which books are selected, stored procedure would have to be used.

```
1  create view [vwNoBooksPublishedThisYearBySubject] as
2
3  select Count(Isbn) as NoBooks, [Subject]
4  from Book
5  where   Published >= Cast(Cast(Year(GetDate()) AS varchar) + '-01-01' AS datetime) and
6          Published <= Cast(Cast(Year(GetDate()) AS varchar) + '-12-31' AS datetime)
7
8  group by [Subject];
```

Figure 43 - Example of View

3.6 Query processing

3.6.1 Indexes

An index is a distinct structure in the database that is built using the create index statement. It requires its own disk space and holds a copy of the indexed table data. Creating an index does not change the table data; it just creates new data structure that refers to the table. The primary purpose of an index is to provide an ordered representation of the indexed data.

A database index undergoes constant change - An SQL database must process insert, delete and update statements immediately and keep the index order without moving large amounts of data.[39] The solution to the problem is to establish a logical order that is independent of the physical order in memory - doubly linked list. Every node of a doubly linked list has links to two neighboring entries. New nodes are inserted between two existing nodes by updating their links to refer to the new node - therefore it is possible to insert new entries without moving large amounts of data. [40]

Indexes are automatically created when PRIMARY KEY and UNIQUE constraints are defined on table columns. For example, when a Book table was created, with a primary key Isbn, the database engine automatically created a primary key constraint (see figure number 46) and index on that column.

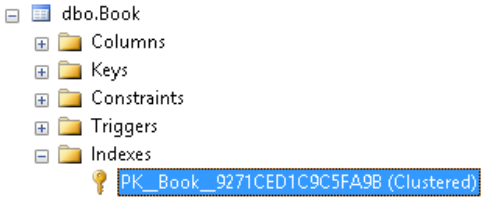


Figure 44 - Example of an index on the Book table

[39] Adapted from http://use-the-index-luke.com/sql/anatomy (Retrieved in 05/2017)
[40] Adapted from http://use-the-index-luke.com/sql/anatomy/the-leaf-nodes (Retrieved in 05/2017)

3.6.1.1 Clustered Index

Clustered indexes sort and store the data rows in the table or view based on their key values. These are the columns included in the index definition. There can be only one clustered index per table, because the data rows themselves can be sorted in only one order.

The only time the data rows in a table are stored in sorted order is when the table contains a clustered index. When a table has a clustered index, the table is called a clustered table. If a table has no clustered index, its data rows are stored in an unordered structure called a heap.[41] An example of a clustered index can be seen on figure number 46.

3.6.1.2 Non-clustered Index

Non-clustered indexes have a structure separate from the data rows. A non-clustered index contains the non-clustered index key values and each key value entry has a pointer to the data row that contains the key value.

The pointer from an index row in a non-clustered index to a data row is called a row locator. The structure of the row locator depends on whether the data pages are stored in a heap or a clustered table. For a heap, a row locator is a pointer to the row. For a clustered table, the row locator is the clustered index key.[42]

An example of a non-clustered index is an index on the Ssn and RoleId column in the Member table. If the Ssns of the members with role id 3 are selected, the query optimizer will most likely perform a non cluster index scan. More about the implementation of non-clustered indexes in chapter Query performance analysis - 3.8.2.

3.6.1.3 Indexes and query optimizer

Well-designed indexes can improve query performance. Indexes can be helpful for a variety of queries that contain SELECT, UPDATE, DELETE, or MERGE statements. When the query optimizer uses an index, it searches the index key columns, finds the storage location of the rows needed by the query and extracts the matching rows from that location. Generally, searching the index is much faster than searching the table because unlike a table, an index frequently contains very few columns per row and the rows are in sorted order.

[41] Quoted from https://docs.microsoft.com/en-us/sql/relational-databases/indexes/clustered-and-nonclustered-indexes-described (Retrieved in 05/2017)
[42] Quote from https://docs.microsoft.com/en-us/sql/relational-databases/indexes/clustered-and-nonclustered-indexes-described (Retrieved in 05/2017)

3.6.2 Execution plan

Before the database can execute an SQL statement, the optimizer must create an execution plan for it. The database then executes this plan in a step-by-step manner. In this respect, the optimizer is very similar to a compiler because it translates the source code (SQL statement) into an executable program (execution plan).[43] An execution plan can be seen in chapter Query performance analysis - 3.8.2. Execution plan contains several different operations that are described in the following sub-chapters.

3.6.2.1 Index and table access

In SQL Server "Scan" refers to reading the entire index or table, while "Seek" operations use B-tree or physical address to access a specific part of the index or table. The following are the possible table access operations:

- Index seek / Clustered index seek - performs a B-tree traversal and goes through leaf nodes to find matching entries.
- Index scan/ Clustered index scan - Read all the rows of the index in the index order. The database might perform this operation if it needs all rows in the index order, for example because of the "order by" clause.
- Key Lookup (clustered) - retrieves a single row from a clustered index.
- RID Lookup (heap) - retrieves a single row from a table
- Table scan / Full table scan - one of the most expensive operations, because the entire table is scanned. [44]

3.6.2.2 Join operations

Generally, join operations process only two tables at a time. In case a query has more joins, they are executed sequentially. The following are the possible join operations:

- Nested loops - joins two tables, by fetching the result from one table and querying the other table for each row from the first.
- Hash match - the hash match join loads the candidate records from one side of the join into a hash table which is then probed for each row from the other side of the join.
- Merge join - The merge join combines two sorted lists. Both sides of the join must be presorted.[45]

[43] Quote from http://use-the-index-luke.com/sql/explain-plan (Retrieved in 05/2017)
[44] Adapted from http://use-the-index-luke.com/sql/explain-plan/sql-server/operations (Retrieved in 05/2017)
[45] Adapted from http://use-the-index-luke.com/sql/explain-plan/sql-server/operations (Retrieved in 05/2017)

3.6.2.3 Sorting and grouping

The following are the possible sorting and grouping operations:

- Sort - Sorts the result according to the order by clause. T
- Sort (Top N Sort) - Sorts a subset of the result according to the order by clause.
- Stream Aggregate - Aggregates a presorted set according to the group by clause.
- Hash match - Groups the result using a hash table. The output is not ordered in any meaningful way.[46]

3.7 Concurrency and Transactions

3.7.1 SQL Transactions

A SQL transaction is a group of command that change the data stored in the database. It is treated as a single unit and it ensures that either all or none of the commands succeed. If any of the commands in the transaction fails, all the commands fail and any data that was modified in the database is rolled back. Transaction maintains the integrity of data in the database.

Transactions have the following four standard properties, called the ACID properties:

- **Atomicity** − ensures that all operations within the work unit are completed successfully. Otherwise, the transaction is aborted at the point of failure and all the previous operations are rolled back to their former state.
- **Consistency** − ensures that the database properly changes states upon a successfully committed transaction.
- **Isolation** − enables transactions to operate independently of and transparent to each other.
- **Durability** − ensures that the result or effect of a committed transaction persists in case of a system failure.[47]

Transactional control commands are only used with the DML commands (see Data manipulation language - 3.5.2). They cannot be used while creating tables or dropping them because these operations are automatically committed in the database.

In SQL, transaction processing has following steps:

1. Begin transaction
2. Process database commands
3. Check for errors (if errors occurred rollback transaction, otherwise commit the transaction)

[46] Adapted from http://use-the-index-luke.com/sql/explain-plan/sql-server/operations (Retrieved in 05/2017)
[47] Quote from https://www.tutorialspoint.com/sql/sql-transactions.htm (Retrieved in 05/2017)

3.7.2 Concurrency

Databases are potentially used by many users at the same time - they run concurrent transactions all the time. While allowing concurrent transaction is essential for performance, they might introduce different concurrency issues.

One way of solving the concurrency issues would be to allow only one user to execute only one transaction at the same time. Unfortunately, this might cause performance throwbacks, because all the transactions get queued. One the other had if all transactions are allowed to execute at the same time the following concurrency problems may occur:

- Dirty read
- Lost update
- Non-repeatable read
- Phantom read

To be able to balance the concurrency and performance, SQL server provides different transaction isolation levels:

- Read uncommitted
- read committed
- Repeatable read
- Snapshot
- Serializable

Depending on the chosen isolation level, different degrees of performance can be reached and different isolation problems can occur. This relationship can be seen on table number 6.

Isolation level	Dirty read	Lost update	Non-repeatable read	Phantom read
Read uncommitted	Yes	Yes	Yes	Yes
Read committed	No	Yes	Yes	Yes
Repeatable read	No	No	No	Yes
Snapshot	No	No	No	No
Serializable	No	No	No	No

Table 6 - Relationship between Isolation levels and concurrency issues

3.7.3 Dirty read

The dirty read happens when it is permitted to read data modified by another transaction, which has not yet been committed. In most cases this does not oppose a threat, however if the first transaction roles back, the second transaction has dirty data that does not exist anymore. This concurrency issue occurs if the transaction isolation level is set to read uncommitted (as demonstrates table number 6). However, the default transaction isolation level of SQL Server is Read committed.

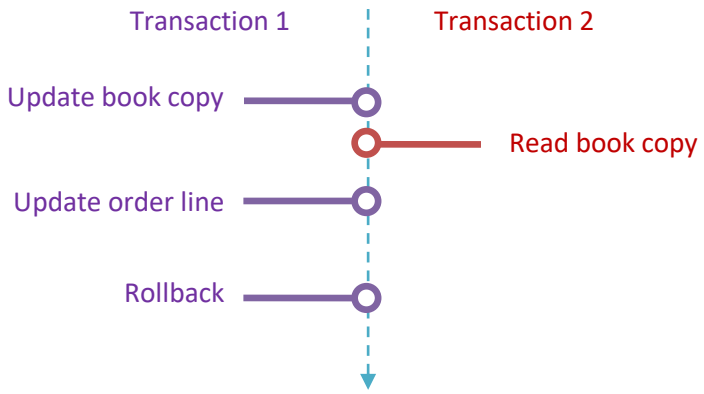


Figure 45 - Example of dirty read

An example of dirty read can be seen of figure number 47 and figure number 48. In this example, first transaction attempts to execute two commands - update the "Available" attribute of the book copy to true and update the "Returned" attribute of an order line record to true. However, the second statement will cause the transaction to fail (because of the unexpected varchar in the where clause) and the transaction will be rolled back. In between these two statements a second transaction attempts to read the Availability of the same book copy. The result of the select statement of the second transaction show that the book copy is available, however this state will change when the first transaction is rolled back.

```
--1st Transaction
Begin transaction
    Begin try
        Update [Book_copy] set [Available] = 'true' where [Id] = 25;
        Waitfor delay '00:00:10';
        Update [Order_line] set [Returned] = 'true' where [Order-id] = 4 and [Book-copy-id] = '25.';

        Commit transaction;
        Print 'Transaction has been commited.';
    End try
    Begin catch
        Rollback transaction
        Print 'Transaction has been Rolled back';
    End catch

--2nd Transaction
Set transaction isolation level read uncommitted;
Select [Available] from [Book_copy] where [Id] = 25;
```

Figure 46 - Example of dirty read in sql

3.7.4 Lost update

The lost problem happens when two or more transactions read and update the same data. This might possibly result in one transaction overriding the update of another transaction and therefore the update will be lost. To prevent this issue from happening, isolation level has to be set to Repeatable read or higher.

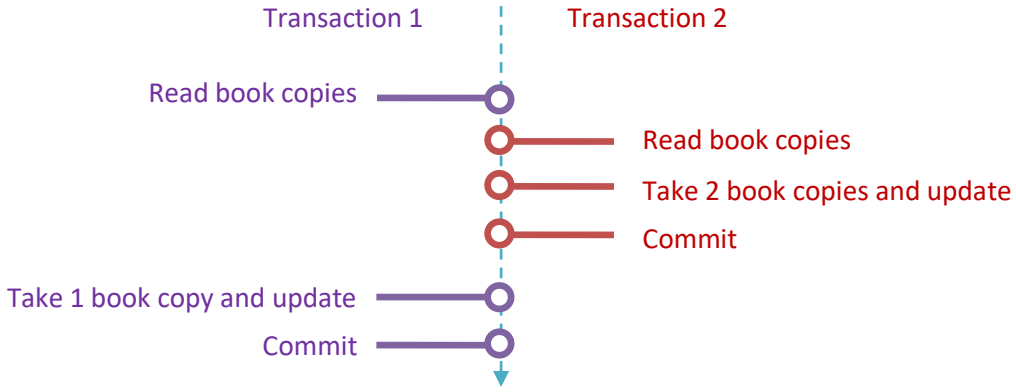


Figure 47 - Example of lost update

To demonstrate an example of lost update, database schema had to be adjusted. Now Book table contains a column "NoBookCopies" instead of referring to another table. First transaction firstly selects the number of available book copies and finds out that 20 book copies of particular book are available. In a meanwhile, second transaction does the same - it read the number of available book copies and finds 20 available books. Second transaction takes two book copies, updates the changes and commits. Afterwards, a slower first transaction takes one book copy, updates the changes and commits. Thus, first transaction has overridden the update of second transaction and the book has now 19 available book copies, even though 3 books have been borrowed. This example is demonstrated on figures number 49 and 50.

```
--1st Transaction
Begin transaction
Declare @avilableBookCopies int;

Select @avilableBookCopies = [NoBookCopies] from [Book] where [Isbn] = '0000001';

Waitfor delay '00:00:10';
Set @avilableBookCopies = @avilableBookCopies - 1;

Update [Book] Set [NoBookCopies] = @avilableBookCopies where Isbn = '0000001';

Commit transaction;

--2nd Transaction
Begin transaction
Declare @avilableBookCopies int;

Select @avilableBookCopies = [NoBookCopies] from [Book] where [Isbn] = '0000001';

Waitfor delay '00:00:01';
Set @avilableBookCopies = @avilableBookCopies - 2;

Update [Book] Set [NoBookCopies] = @avilableBookCopies where Isbn = '0000001';

Commit transaction;
```

Figure 48 - Example of lost update in sql

3.7.5 Non-repeatable read

Non-repeatable read happens when one transaction reads the same data twice and another transaction updates the data in between the first and second read of the first transaction. To prevent this issue from happening, isolation level has to be set to Repeatable read or higher.

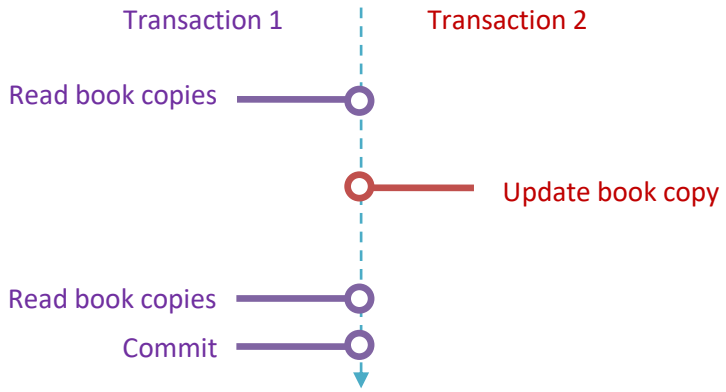


Figure 49 - Example of non-repeatable read

An example of a non-repeatable read can be seen on figures x, y and z. First transaction reads the number of available book copies of book with Isbn equal to "0439139600". Second transaction then updates additional book copy by setting it as available. Finally, first transaction performs the same read again, but the number of available book copies does not match - the first read could not be repeated.

```
--1st Transaction
Begin transaction
declare @availableCopies1 int;
declare @availableCopies2 int;

Select @availableCopies1 = Count(Id) from [Book_copy] where [Book-isbn] = '0439139600' and [Available] = 'true';
Print @availableCopies1;

waitfor delay '00:00:10';

Select @availableCopies2 = Count(Id) from [Book_copy] where [Book-isbn] = '0439139600' and [Available] = 'true';
Print @availableCopies2;

Commit transaction
```

```
--2nd Transaction
Begin transaction

Update [Book_copy] set [Available] = 'true' where [Id] = 17;

Commit transaction
```

Figure 50 - Example of non-repeatable read in sql

3.7.6 Phantom read

Phantom read happens when a transaction executes a query twice and it results with a different number of rows each time. This happens when a second transaction inserts a new row that matches the where clause of the query executed by the first transaction. To prevent Phantom reads, transaction isolation level has to be set to either snapshot or serializable.

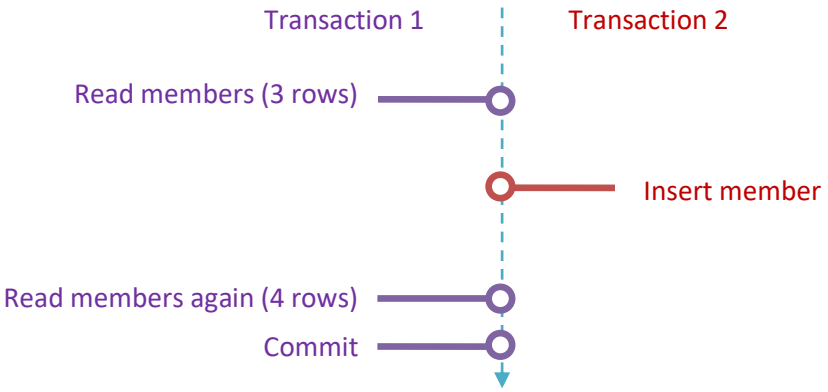


Figure 51 - Example of phantom read

An example of a phantom read can be seen on figure number 53 and 54. The first transaction begins and selects Ssn number of members based on their membership expiration date. The select statement produces a result set of three records. In a meanwhile another transaction inserts new member that satisfies the condition of the first transaction's select statement. When the first transaction performs the read again, the result set contains four rows instead of three.

```
--1st Transaction
Begin transaction

Select [Ssn] from Member where [Membership-expiration] < '2018-03-31';

waitfor delay '00:00:05';

Select [Ssn] from Member where [Membership-expiration] < '2018-03-31';

Commit transaction
--2nd Transaction
Begin transaction

Insert into [Member] values(
'000-00-0011', 1, 'Ucn Sofiendalsvej', '2018-01-01', 'Firstname', 'Lastname', 'password1');

Commit transaction
```

	Ssn
1	000-00-0002
2	000-00-0004
3	000-00-0005

	Ssn
1	000-00-0002
2	000-00-0004
3	000-00-0005
4	000-00-0011

Figure 52 - Example of phantom read in sql

3.8 Database Implementation

This section focuses on the implementation of the five complex queries, their optimization and performance analysis. Additionally, it examines the security consideration of the GTL database.

3.8.1 Query implementation

3.8.1.1 Query 1

The first query is designed to show what cities have the highest number of extended orders in a certain time interval. The result set contains only two attributes - "No-extended-orders" and "City". To be able to produce such a result, the query must join multiple tables together - Order, Order_Line, Member and Address.

According to the chapter 3.4 - Relational algebra, the query can be represented as following:

$$\pi_{\text{No-Extended-orders, a.City}}\ \gamma_{\text{a.City; COUNT(o.Id)}\rightarrow\text{No-Extended-orders}}$$

$$\rho_{o}\left(\pi_{\text{Id, Member-ssn}}\ \sigma_{\text{Created > '2016-12-31' and Created < '2018-01-01'}}\ \textbf{Order}\right) \bowtie_{\text{o.Id = ol.Order-id}}$$

$$\rho_{ol}\left(\pi_{\text{OrderId}}\ \sigma_{\text{Extended = true}}\ \textbf{Order\_Line}\right) \bowtie_{\text{o.Member-ssn = m.Ssn}}$$

$$\rho_{m}\left(\pi_{\text{Ssn}}\ \sigma_{\text{Role-id = '2'}}\ \textbf{Member}\right) \bowtie_{\text{a.Member-ssn = m.Ssn}}$$

$$\rho_{a}\left(\pi_{\text{Member-ssn, City}}\ \textbf{Address}\right)$$

Additionally, it can be represented in a form of a query tree. Figure number 55 shows the result of a Heuristics optimization. Following steps were made during the Heuristics optimization:

- Moving Select statements down in the query tree
- Applying more restrictive select statements first
- Replacing Cartesian products and selects with join operations
- Moving Project operation down the query tree

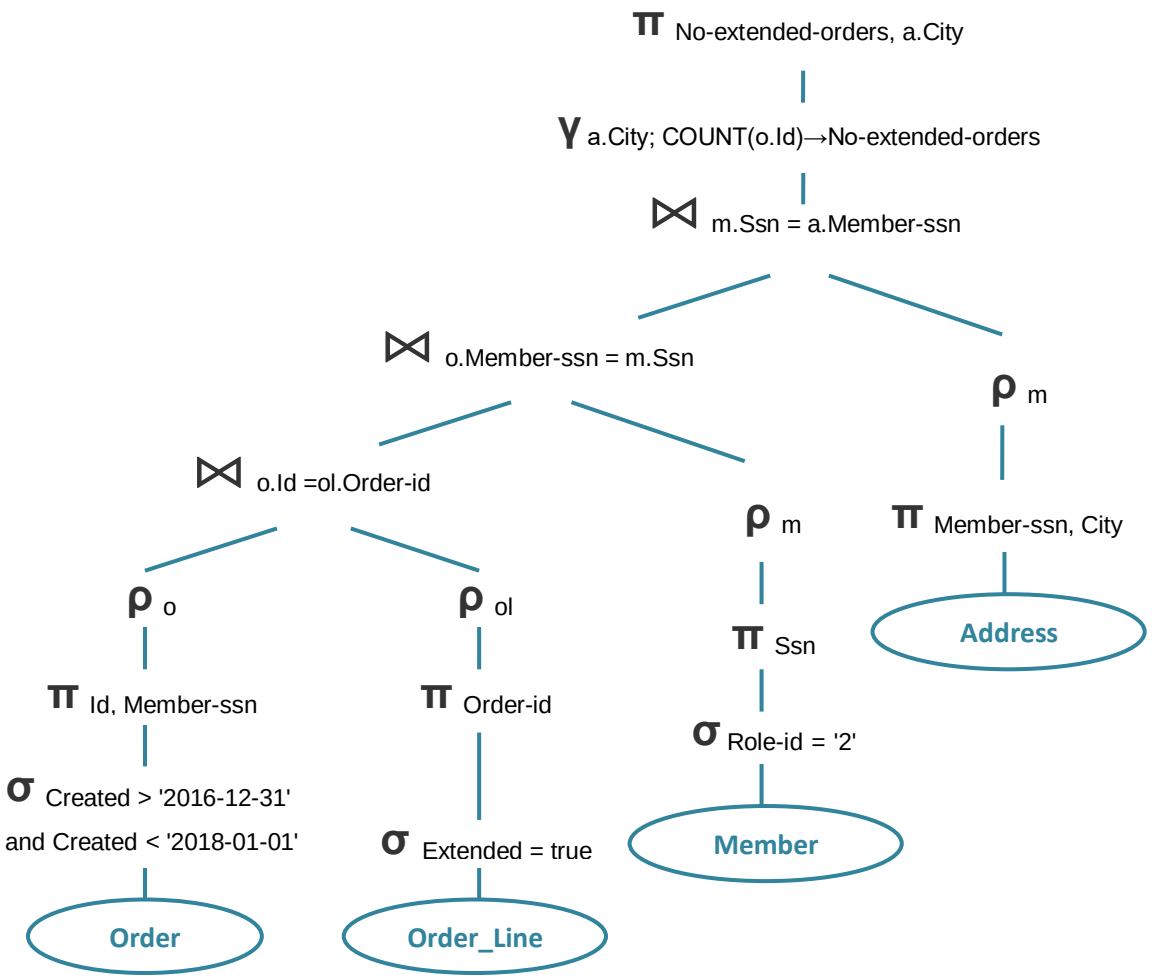


Figure 53 - Query tree of query number 1 after Heuristics optimization

Finally, the SQL representation of the query is shown on figure number 56.

```
select Count(o.Id) as [No-extended-orders], a.City
from (  Select Id, [Member-ssn]
        from [Order]
        where Created > '2016-12-31' and Created < '2018-01-01') o

inner join (select [Order-id]
            from [Order_Line]
            where Extended = 'true') ol
on o.Id = ol.[Order-id]

inner join (
    select [Ssn]
    from Member
    where RoleId = '2'
) m
on o.[Member-ssn] = m.Ssn

inner join (
    select [City], [Member-ssn]
    from [Address]
) a
on a.[Member-ssn] = m.Ssn

group by a.City
order by [No-extended-orders] desc
```

Figure 54 - Sql representation of query number 1

3.8.1.2 Query 2

The second query is designed to show data regarding orders that have not returned book on time. The query looks at orders in a particular date interval, based on when the order was created. Additionally, the query takes into consideration the role of the member who did not return the book, more particularly the loan period and the grace period of the particular role. This is due to the fact that these periods vary according to the role(e.g. Member can loan a book for 21 days, while Teacher can loan a book for 3 months).

The query joins tables - Order, Order_Line, Member and Role, and returns a data set with attributes Order-id, Order-created, Expected-return-date, Member-ssn and Member-role.

The SQL Statement of the second query can be seen on the figure number 57.

```
6  select  o.Id as [Order-id], o.Created as [Order-created],
7          dateadd(day, r.Period, o.Created) as [Expected-return-date],
8          o.[Member-ssn], r.Name as [Member-role]
9
10 from (
11     Select Id, [Member-ssn], Created
12     from [Order]
13     where Created > '2016-12-31' and Created < '2018-01-01') o
14
15 inner join (
16     select [Order-id]
17     from [Order_Line]
18     where Returned = 'false') ol
19 on o.Id = ol.[Order-id]
20
21 inner join (
22     select Ssn, RoleId
23     from Member) m
24 on o.[Member-ssn] = m.Ssn
25
26 inner join (
27     select Id, Name, ([Loan-period] + [Grace-period]) as Period
28     from [Role]) r
29 on r.Id = m.RoleId
30
31 where GetDate() > dateadd(day, r.Period, o.Created)
```

Figure 55 - Sql representation of query number 2

3.8.1.3 Query 3

The third query is designed to show the most popular books of the most popular subject. In fact, it can be separated into two queries. The first part focuses on getting the most popular subject of the book (e.g. Science fiction, drama) in the particular time interval. The first query joins tables Order, Order_Line, Book_Copy and Book and returns the book subject with the highest number of ordered books.

The query returning the most popular subject can be written in relational algebra as follows:

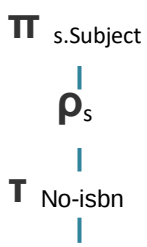
$\pi_{\text{Subject}}\ \rho_{s}\ (\ \tau_{\text{NoIsbn asc}}\ \pi_{\text{b.Subject, No-isbn}}\ \gamma_{\text{b.Subject; COUNT(b.Isbn)}\rightarrow\text{No-isbn}}$

$\rho_{o}\ (\ \pi_{\text{Id}}\ \sigma_{\text{Created > '2016-12-31' and Created < '2018-01-01'}}\ \textbf{Order})\ \bowtie_{\text{o.Id = ol.OrderId}}$

$\rho_{ol}\ (\ \pi_{\text{Order-id, Book-copy-id}}\ \textbf{Order\_Line})\ \bowtie_{\text{ol. Book-copy-id = bc.Id}}$

$\rho_{bc}\ (\ \pi_{\text{Id, Book-isbn}}\ \textbf{Book\_Copy})\ \bowtie_{\text{bc.Boo-isbn = b.Isbn}}$

$\rho_{b}\ (\ \pi_{\text{Isbn, Subject}}\ \textbf{Book})$



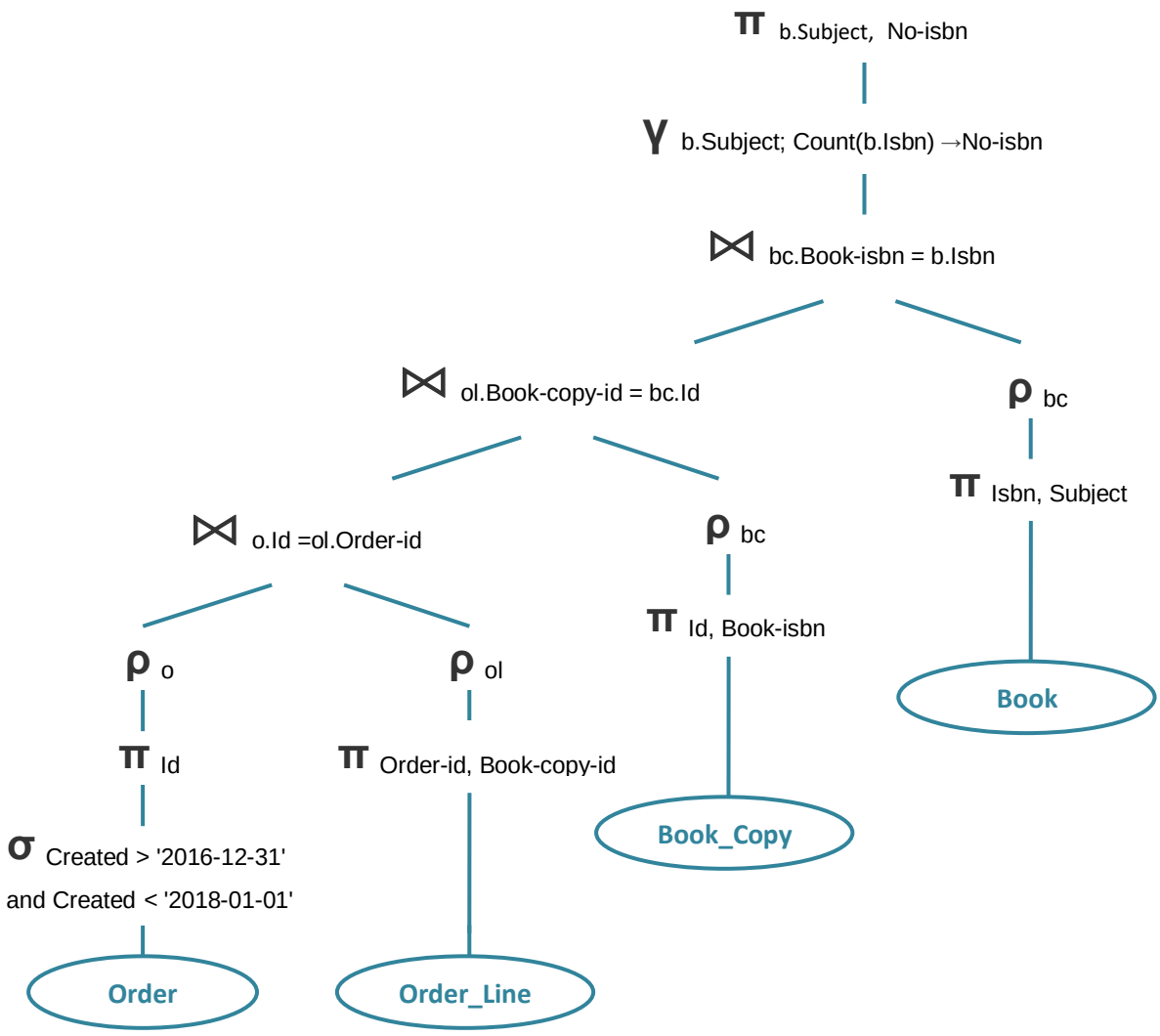


Figure 56 - Query tree of the first part of the query number 3

Figure number 58 shows the query tree representation of the first part of the query. The second part of the query return a result set of the most popular books (the ones that are being ordered the most) within the specified subject. The second part of the query has a very similar approach, when it comes to the tables it joins, however the data are after all grouped differently and it returns a set of books, their Isbns, Titles, Authors, Subjects as well as number of orders. Both queries were implemented as stored procedures, while one stored procedure executes the other. The implementation of both queries can be seen on figure number 59 and 60.

```
198 Create procedure [spGetMostPopularSubject]
199 @startDate date,
200 @endDate date,
201 @subject varchar(20) output
202 as
203 begin
204             select top 1 @subject =  b.[Subject]
205             from (  select Id
206                     from [Order]
207                     where Created > @startDate and Created < @endDate ) o
208             inner join (
209                     select [Order-id], [Book-copy-id]
210                     from Order_Line) ol
211             on o.Id = ol.[Order-id]
212
213             inner join (
214                 select Id, [Book-isbn]
215                 from Book_Copy) bc
216             on ol.[Book-copy-id] = bc.[Id]
217
218             inner join (
219                 select Isbn, [Subject]
220                 from Book) b
221             on bc.[Book-isbn] = b.Isbn
222
223             group by b.[Subject]
224             order by Count(b.Isbn) desc
225 end
```

Figure 57 - Sql representation of the first part of the query number 3

```
157 create procedure [spGetTopBooksInTopSubject]
158
159 @startDate date,
160 @endDate date,
161 @amount int
162 as
163 begin
164     declare @subject varchar(20);
165     execute [spGetMostPopularSubject] @startDate, @endDate, @subject out;
166
167     select top (@amount) b.Title, b.Isbn, b.Author, b.[Subject], Count(o.Id) as [No-orders]
168     from (
169             select Id
170             from [Order]
171             where Created > @startDate and Created < @endDate ) o
172
173     inner join (
174         select [Order-id], [Book-copy-id]
175         from Order_Line) ol
176     on o.Id = ol.[Order-id]
177
178     inner join (
179         select Id, [Book-isbn]
180         from Book_Copy) bc
181     on ol.[Book-copy-id] = bc.[Id]
182
183     inner join (
184         select Isbn, Title, Author, [Subject]
185         from Book) b
186     on bc.[Book-isbn] = b.Isbn
187
188     where b.[Subject] = @subject
189     group by b.Isbn, b.Title, b.Isbn, b.Author, b.[Subject]
190     order by [No-orders] desc
191 end
```

Figure 58 - Sql representation of the second part of the query number 3

3.8.1.4 Query 4

The fourth query is designed to get number of orders of a particular book in individual months, given the book isbn and year (e.g. in year 2017 book with isbn 1010-1010 had 25 orders in may and 20 orders in June). The query join tables Order, Order_Line, and Book_Copy to produce

desired result. Once again it is implemented as a stored procedure, with input parameters of Isbn and year. The sql implementation can be seen on figure number 61.

```
300 create procedure [spGetMonthlyNoORdersWithIsbn]
301 @year int,
302 @isbn varchar(30)
303 as
304 begin
305
306     declare @startDate date;
307     declare @endDate date;
308
309     set @startDate = CAST(CAST(@year AS varchar) + '-01-01' AS DATETIME);
310     set @endDate = CAST(CAST(@year AS varchar) + '-12-31' AS DATETIME);
311
312     select
313         Count(o.Id) as [No-orders], DateName( month , DateAdd( month , DATEPART(Month, Created) , -1 )) as [Month]
314     from (
315         select Id, Created
316         from [Order]
317         where Created > @startDate and Created < @endDate ) o
318
319     inner join (
320         select [Order-id], [Book-copy-id]
321         from [Order_Line]
322     ) ol
323     on o.Id = ol.[Order-id]
324
325     inner join (
326         select Id
327         from [Book_Copy]
328         where [Book-isbn] = @isbn
329     ) bc
330     on bc.Id = ol.[Book-copy-id]
331
332     GROUP BY DateName( month , DateAdd( month , DATEPART(Month, Created) , -1 ))
333     Order by [No-orders] desc
334 end
```

Figure 59 - Sql representation of query number 4

3.9.1.5 Query 5

The fifth query was designed to get the most popular author, based on the number of orders given the date interval and the city - in other words it answers the question: "Who is the most popular author in Aalborg?". This query is most likely very heavy performance wise, simple because of its design - a lot of tables have to be joined to produce such results(Order, Member, Address, Order_Line, Book_Copy and Book tables have to be joined). On top of that, generally very little rows are filtered, because the where clause is used only on the Order table and the Address table. The implementation of this query can be seen on figure number 62.

```sql
select Count(o.Id) as [No-orders], b.Author
from(
    select Id, [Member-ssn]
    from [Order]
    where Created > '2016-12-31' and Created < '2019-01-01') o

inner join (
    select Ssn
    from Member) m
on o.[Member-ssn] = m.Ssn

inner join (
    select [Member-ssn]
    from [Address]
    where Zip = '9000') a
on m.Ssn = a.[Member-ssn]

inner join (
    select [Order-Id], [Book-copy-id]
    from [Order_Line]) ol
on o.Id = ol.[Order-id]

inner join (
    select Id, [Book-isbn]
    from Book_Copy) bc
on ol.[Book-copy-id] = bc.[Id]

inner join (
    select Isbn, Author
    from Book) b
on bc.[Book-isbn] = b.Isbn

group by Author
order by [No-orders] desc
```

Figure 60 - Sql representation of query number 5

3.8.2 Query performance analysis

Database performance is one of the most crucial properties of any system. There are many aspects that can cause a deceleration of performance. In the previous chapter, queries were designed, while performance was taken into consideration. In the following chapter, performance will be measured, analyzed and improved with indexes.

For measurement of the performance of the queries, database had to be populated with a test data. Only the tables that are involved in queries described in the previous chapter were populated. To perform the measurements accurately, the database was populated with approximately 100 000 books, each having 3 book copies on average. Additionally it was populated with approximately 300 000 members, orders, order lines and addresses.

The query execution duration was measured in 3 different stages - without indexes, only with clustered indexes (created by default) and with both clustered and non-clustered indexes. On top of that, measurement was taken twice, once with execution of the "dropcleanbuffers" command, which cleans the buffer pool. The duration of the query execution was measured using the built-in tool SQL profiler. It was measured 5 times under the same conditions and the average duration was taken into a count.

Table number 7 as well as figure number 63 show the results of the first measurement - without cleaning the buffer pool. Second and forth query performed significantly better than others. The reason behind why the second query performed so well, is the selection of Order_Lines that were not yet returned. This selection reduced the amount of rows significantly and cause fast query execution. Similarly query number four was selecting the book copies by isbn, which resulted in small number of records.

Without cleaning buffer pool			
Query	No indexes	Clustered indexes	Non-clustered indexes
1	1015,2	893,8	840
2	17,8	12,2	12
3	1349,6	1015,4	1219
4	106,4	58,6	58,8
5	1130,8	1004,4	999,6

Table 7 - Query performance measurement without cleaning the buffer pool

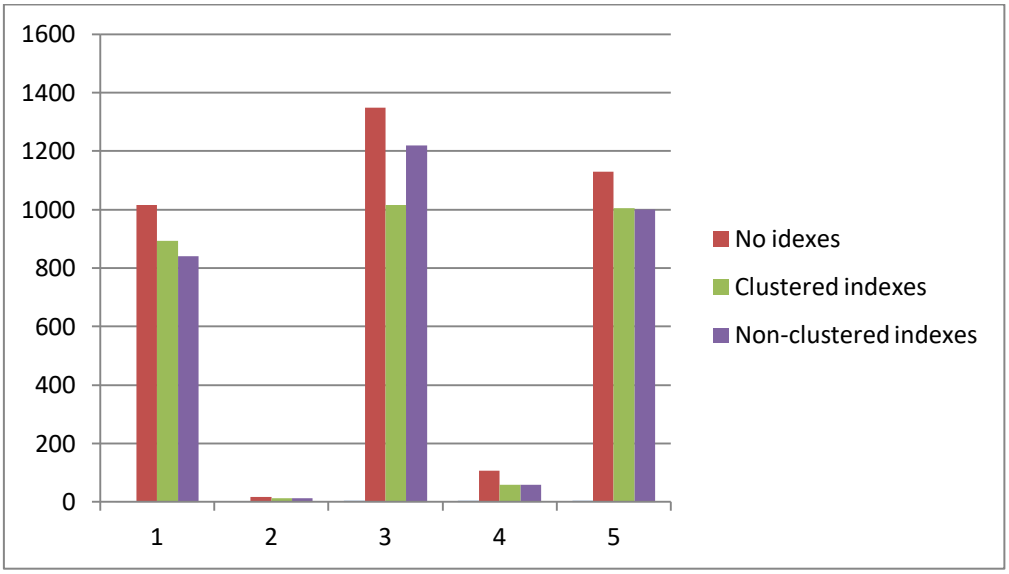


Figure 61 - Graphical representation of query performance measurement

Surprisingly, adding a non-clustered indexes has improved performance just slightly. In a case of query number three, it even resulted in slower execution. Nevertheless, the query optimizer did pick up the non-clustered indexes and the execution plan was changed. This can be seen on the second query. The execution plan before and after the addition of non-clustered indexes can be seen on figure number 64.

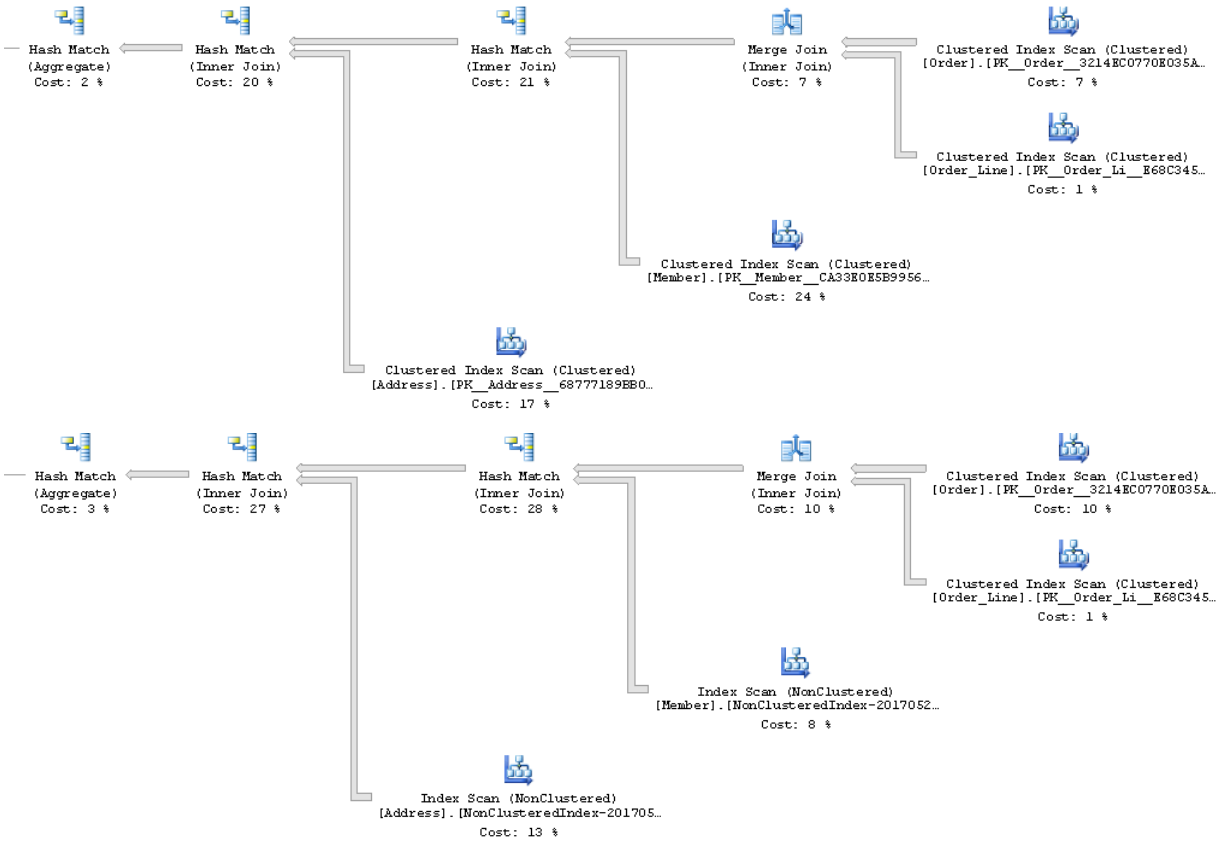


Figure 62 - Execution plan before and after non-clustered index was added

Table number 8 as well as figure number 65 shows the results of the second measurement - with cleaning the buffer pool. This measurement really highlights the importance of non-clustered indexes, because it significantly reduced duration - even over 65%. Surprisingly, when a buffer pool is cleaned, a table without a cluster index performed faster that when it has one.

With cleaning buffer pool			
Query	No indexes	Clustered indexes	Non-clustered indexes
1	4153	4363	2005,8
2	422,8	431	411,4
3	4118,6	6353,4	2209,4
4	2157	1195,4	377,2
5	5050,2	7637,2	2035

Table 8 - Query performance measurement after cleaning the buffer pool

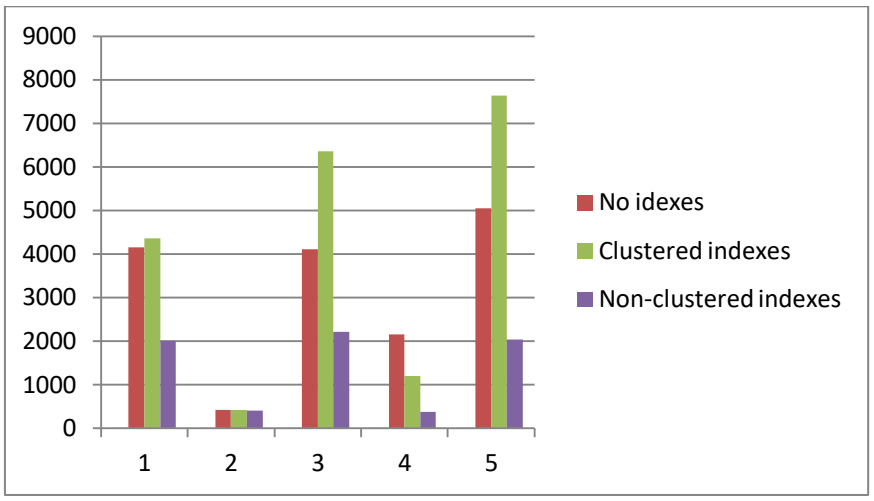


Figure 63 - Graphical representation of query performance measurement

3.8.3 Security considerations

Security is a very important part of the database systems. It is necessary to realize who is going to use the database. In the GTL project, the only user types are either database administrator or an application user that will access the database via the web application.

In order to ensure that the application user has only access to the databases and actions he/she needs, new user called GTL_User was created. HTL_User has only access to the Library_case database (see figure number 66) and he has been given database level roles db_datareader and db_datawriter (see figure number 67).

Users mapped to this login:

Map	Database	User
☑	Library_case	GTL_User
☐	master	
☐	msdb	
☐	tempdb	

Figure 64 - Example of database access control

Db_datareader database role allows a user to be able to issue a SELECT statement against all tables and views in the database.

Db_datawriter gives implicit access to tables and views within a database. It can be blocked by an explicit deny for the user or for a role the user is a member of. Unlike db_datareader, however, db_datawriter gives INSERT, UPDATE, and DELETE permissions.[48]

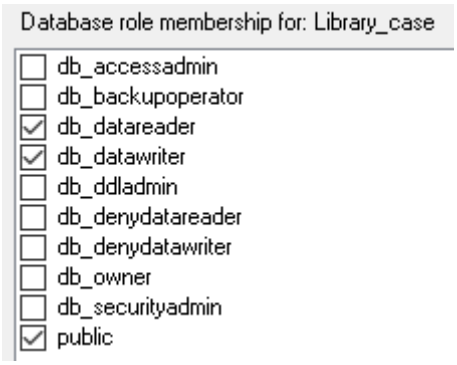


Figure 65 - Database level roles of GTL_User

3.9 Database conclusion

One of the things that went very well in the GTL project was the design of the database itself. While it is not the first time we worked with both ERDs and database schemas, it was completely different to start with clearly defined customer requirements. Proper design of ERD and its mapping to relation schema were the biggest learning points of the database creation process.

Tuning of the performance of the queries was another very interesting assignment. It was surprising to see that a lot of design mistakes come up when the sql query is written in the query tree and relational algebra. In the tree, mistakes can be easily spotted as well as fixed by following few simple rules. On top of that the query performance analysis was an interesting process as well. We have learned a lot about what is happening under the hood of sql server, how indexes can cause impact on performance and how to read the execution plan.

[48] Quote from https://www.mssqltips.com/sqlservertip/1900/understanding-sql-server-fixed-database-roles (Retrieved in 05/2017)

4. References

[1] Dorothy Graham, E. v. (2008). *Foundation of Software Testing .* Intl Thomson Business.

[2] Osherove, R. (2009). *The Art of Unit Testing.* Manning.

[3] Ramez Elmasri, S. B. (2010). *Fundamentals of Database Systems (6th Edition).* Pearson.

- http://www.coleyconsulting.co.uk/testing-entry-criteria.htm
- https://dzone.com/articles/how-to-define-your-testing-scope
- http://www.softwaretestingmentor.com/what-are-test-deliverables
- http://www.guru99.com/what-everybody-ought-to-know-about-test-planing.html
- http://www.guru99.com/risk-based-testing.html
- http://www.requirementdriventesting.com/how-to-test-strategy/ (test levels)
- http://www.mytipsfor.com/writing/testobjectives/crash-course
- http://www.defectmanagement.com/defectmanagement/defresolution.htm
- http://www.defectmanagement.com/defectmanagement/pririsk.htm
- http://www.defectmanagement.com/defectmanagement/schedfix.htm
- http://www.defectmanagement.com/defectmanagement/represolution.htm
- https://www.youtube.com/watch?v=X5OaU4zli44&list=PLCYqwbyDA33Q8qDxpzorClu5bwX3wHkiF
- https://www.youtube.com/watch?v=tN1xp76pwHM
- https://www.veracode.com/products/static-analysis-sast/static-analysis-tool
- https://www.tutorialspoint.com/software_testing_dictionary/technical_review.htm
- https://www.tutorialspoint.com/software_testing_dictionary/code_walkthrough.htm
- https://www.tutorialspoint.com/software_testing_dictionary/structured_walkthrough.htm
- https://www.tutorialspoint.com/software_testing_dictionary/static_testing.htm
- https://www.tutorialspoint.com/software_testing_dictionary/inspection.htm
- https://www.tutorialspoint.com/software_testing_dictionary/dynamic_testing.htm
- http://www.guru99.com/static-dynamic-testing.html
- http://istqbexamcertification.com/what-is-black-box-specification-based-also-known-as-behavioral-testing-techniques
- https://timsdevblog.wordpress.com/2015/03/30/kent-beck-on-tdd
- https://technologyconversations.com/2013/12/11/black-box-vs-white-box-testing
- https://www.tutorialspoint.com/software_testing_dictionary/white_box_testing.htm
- https://www.youtube.com/watch?v=VDfX44fZoMc
- http://www.softwaretestingclass.com/system-testing-what-why-how
- https://ops.fhwa.dot.gov/publications/fhwahop13046/sec6.htm
- http://reqtest.com/testing-blog/sample-test-plan
- http://www.conceptdraw.com/How-To-Guide/erd-entity-relationship-diagram-symbols
- https://www.lucidchart.com/pages/er-diagrams
- https://www.tutorialspoint.com/dbms/er_model_basic_concepts.htm

- https://www.smartdraw.com/entity-relationship-diagram
- http://www.studytonight.com/dbms/database-normalization.php
- https://www.thoughtco.com/database-normalization-basics-1019735
- https://www.youtube.com/watch?v=9qWpfTW_c5Y
- https://www.youtube.com/watch?v=cm1DYCQhFZw
- https://www.youtube.com/watch?v=80CcB9_HSxU
- https://www.youtube.com/watch?v=CedOasDoe-w
- https://docs.oracle.com/cd/A97630_01/server.920/a96540/clauses3a.htm
- http://searchoracle.techtarget.com/definition/stored-procedure
- https://www.simple-talk.com/sql/performance/execution-plan-basics
- https://www.youtube.com/watch?v=uDcVd4vUU3s&t=364s
- https://www.youtube.com/watch?v=5ZEchu2WnD4&list=PL08903FB7ACA1C2FB&index=71
- https://www.youtube.com/watch?v=jD0c4X0tSc8&index=72&list=PL08903FB7ACA1C2FB
- http://use-the-index-luke.com/sql/anatomy
- https://docs.microsoft.com/en-us/sql/relational-databases/indexes/clustered-and-nonclustered-indexes-described
- https://www.tutorialspoint.com/dbms/relational_algebra.htm
- https://www.mssqltips.com/sqlservertip/1900/understanding-sql-server-fixed-database-roles

5. Appendix

5.1 Test deliverables

5.1.1 System requirements specification document

Functional requirements of the GTL project:

ID	Description
SR-1	Librarian is able to add a book with valid ISBN.
SR-2	Librarian and Member is able to see book details.
SR-3	Librarian is able to see book description.
SR-4	Every user is able to see a catalogue of books.
SR-5	Book catalogue can be filtered with different options (e.g. search by ISBN, Author, ...)
SR-6	User can access book catalogue of other libraries.
SR-7	Librarian can add/ remove book copies of the book that already exists.
SR-8	User can be registered with email, first name, last name, address and password.
SR-9	Registered user can log in the system.
SR-10	Newly registered member have to be verified by a librarian.
SR-11	Standard user can loan a book copy for 28 days.
SR-12	Registered user can view their profile information on the profile page.
SR-13	Members can reserve a book that is not in the library.
SR-14	Librarian can update information about an existing book.
SR-15	Librarian can delete/ remove book (book cannot have any book copies loaned).
SR-16	Librarian can add a book to the wish list.
SR-17	Member have right to have an extra week of renting time in case the book is not returned.
SR-18	Teachers are automatically registered when they are hired at the university.
SR-19	Teachers have extended renting period.
SR-20	Teacher's notifications are sent to the campus address.
SR-21	Librarians can update member's profile information.
SR-22	Librarian can remove/ delete member's profile.
SR-23	Members are notified, when their reserved book arrives to the library.
SR-24	Loan notice is sent to the member's address, in case book was not returned in time (24d).
SR-25	Member is issued a card after profile confirmation. Card expires in 4 years.
SR-26	Card expiration notice is sent to the member's address before it expires.
SR-27	Member profile is automatically deleted after membership expiration.

5.1.2 Defect log

Example of defect tracking matrix:

ID	SR-ID	TC	Description	Reproduction steps	Expected result	Actial result	Status	Created	Resolved date
1	13	TC-U-55	Gets a paged model for a valid role, when it exists.	-	Pass	Fail	Resolved	24.04.2017	28.04.2017
2	13	TC-U-58	Verification of an existing valid member.	-	Pass	Fail	Resolved	24.04.2017	28.04.2017
3	13	TC-U-64	Remove a member from a role. e.g: when user gets verified	-	Pass	Fail	Resolved	24.04.2017	28.04.2017
4	16	TC-U-67	Updating the details of a role. e.g: the name of the role.	-	Pass	Fail	Pending	24.04.2017	-
5	13	TC-U-100	Repository part of remove from role	-	Pass	Fail	Pending	29.04.2017	-

5.1.3 Test case specification document

Unit tests specification

TC-ID	TC-Title	SR-ID	Input	Output	Environment	Pass/Fail
TC-U-1	Search catalogue with valid input data	4	PageId (int) = 10; PageSize(int) = 10; Filter = not null;	Not null	-list of books is found in the database	Pass
TC-U-2		4				
TC-U-3		4				
TC-U-4		4				
TC-U-5		4				
TC-U-6	Register	8	Isbn (string) = "ISBN";	False		Pass
TC-U-7	Register	8	Book (obj) = null;	False		Pass
TC-U-8	Rent a book	11	Book (obj) = null;	False		Pass
TC-U-9	Rent a book	11	Author (string) = "J.K. Rowling"; Binding (string) = "2"; Description (string) = "Harry Potter";	True	-a book is found in the database	Pass
TC-U-10	Find book by isbn	5	Isbn (string) = "ISBN";	False		Pass
TC-U-11	Find book by isbn	5	Author (string) = "J.K. Rowling"; Binding (string) = "2"; Description (string) = "Harry Potter";	True	-a book is found in the database	Pass
TC-U-12	Find member by Id	22	Ssn = "chr-no-alwd”; Id = “”;	Null		Pass
TC-U-13	-//-	22	Id = “1”;	Not null		Pass
TC-U-14	Find member by Id	22	Id = null;	Null		Pass

TC-U-15	Remove from role	21	FirstName = "Bob", LastName = "Bobonson", RoleId = “1”, Role.Name = “Member”	True		Pass
TC-U-16	Remove from role	21	FirstName = "", LastName = "Bobonson", RoleId = “1”, Role.Name = “Member”	False		Pass
TC-U-17	Remove from role	21	FirstName = "", LastName = "Bobonson", RoleId = “1”, Role.Name = null;	False		Pass
TC-U-18	Add a book	1	Author = "J.K. Rowling", Edition = "1st", Isbn = "0439139600",	True		Pass
TC-U-19	Add a book	1	Author = "J.K. Rowling", Edition = "1st", Isbn = "ISBN",	False		Pass
TC-U-20	Add a book	1	Book = null;	False		Pass
TC-U-21	Delete a book	7	Author = "J.K. Rowling", Edition = "1st", Isbn = "0439139600",	True		Pass
TC-U-22	Delete a book	7	Author = "J.K. Rowling", Edition = "1st", Isbn = "ISBN",	False		Pass
TC-U-23	Delete a book	7	Book = null;	False		Pass
TC-U-24	Update a book	14	Author = "J.K. Rowling", Edition = "1st", Isbn = "0439139600",	True		Pass
TC-U-25	Update a book	14	Author = "J.K. Rowling", Edition = "1st", Isbn = "ISBN",	False		Pass
TC-U-26	Update a book	14	Book = null;	False		Pass
TC-U-27	Find book by Isbn	4	Isbn = "978-3-16-148410-0";	True		Pass
TC-U-28	Find book by Isbn	4	Isbn = "111-1-11-111111-1";	False		Pass
TC-U-29	Find book by Isbn	4	Isbn = null	False		Pass
TC-U-30	Find book by description	4	Description = "Harry Potter is midway through his training as a wizard and his coming of age";	True		Pass
TC-U-31	Find book by description	4	Description = “";	False		Pass
TC-U-32	Find book by description	4	Description = null	False		Pass

TC-U-33	Get book description	3	Isbn = "978-3-16-148410-0";	Not null		Pass
TC-U-34	Get book description	3	Isbn = "111-1-11-111111-1";	Null		Pass
TC-U-35	Get book description	3	Isbn = null	Null		Pass
TC-U-36	Get paged catalog model	4	id = 1; size = 10;	True		Pass
TC-U-37	Get paged catalog model	4	id = -1; size = 10;	False		Pass
TC-U-38	Get paged catalog model	4	id = 1; size = -10;	False		Pass
TC-U-39	Rent books	11	Ssn = "00a0-b0-a00a";	False		Pass
TC-U-40	Rent books	11	Ssn = "000-00-0000"; List = new List();	False		Pass
TC-U-41	Rent books	11	Ssn = "000-00-0000"; List = null;	False		Pass
TC-U-42	Rent books	11	Role = “Unverified”	False		Pass
TC-U-43	Rent books	11	Book.Loanable = “False”	False		Pass
TC-U-44	Rent books	11	Book.copies = 0;	False		Pass
TC-U-45	Rent books	11		True		Pass
TC-U-46	Update member	21	Ssn = "000-00-0000", Campus = "Sofiendalsvej", FirstName = "Bob",	True		Pass
TC-U-47	Update member	21	Ssn = "chr-no-alwd", Campus = "Doesn't matter", FirstName = "Doesn't",	False		Pass
TC-U-48	Update member	21	Member = null	False		True
TC-U-49	Add Member	10	Ssn = "000-00-0000", Campus = "Sofiendalsvej", FirstName = "Bob",	True		Pass
TC-U-50	Add Member	10	Ssn = "chr-no-alwd", Campus = "Doesn't matter", FirstName = "Doesn't",	False		Pass
TC-U-51	Add Member	10	Member = null;	False		Pass
TC-U-52	Delete Member	22	Ssn = "000-00-0000", Campus = "Sofiendalsvej", FirstName = "Bob",	True		Pass

TC-U-53	Delete Member	22	Ssn = "chr-no-alwd", Campus = "Doesn't matter", FirstName = "Doesn't",	False		Pass
TC-U-54	Delete Member	22	Member = null;	False		Pass
TC-U-55	Find with role paged model	10	roleName = “Member”;	Not null		Fail
TC-U-56	Find with role paged model	10	roleName = “”;	Null		Pass
TC-U-57	Find with role paged model	10	roleName = null;	Null		Pass
TC-U-58	Verify member	10	Ssn = "000-00-0000",	True		Fail
TC-U-59	Verify member	10	Ssn = "chr-no-alwd",	False		Pass
TC-U-60	Verify member	10	Ssn = null;	False		Pass
TC-U-61	Find by name	21	UserName = ssn;	True		Pass
TC-U-62	Find by name	21	UserName = “”;	False		Pass
TC-U-63	Find by name	21	UserName = null;	False		Pass
TC-U-64	Remove from role	21	FirstName = "Bob", LastName = "Bobonson", RoleId = “1”, Role.Name = “Member”	True		Fail
TC-U-65	Remove from role	10	FirstName = "", LastName = "Bobonson", RoleId = “1”, Role.Name = “Member”	False		Pass
TC-U-66	Remove from role	10	FirstName = "", LastName = "Bobonson", RoleId = “1”, Role.Name = null;	False		Pass
TC-U-67	Update a role	21	FirstName = "Bob", LastName = "Bobonson", RoleId = “1”, Role.Name = “Member”	True		Fail
TC-U-68	Update a role	21	FirstName = "", LastName = "Bobonson", RoleId = “1”, Role.Name = “Member”	False		Pass
TC-U-69	Update a role	21	FirstName = "", LastName = "Bobonson", RoleId = “1”, Role.Name = null;	False		Pass

TC-U-70	Get Role Name	21	Valid member	Not Null		Pass
TC-U-71	Get Role Name	21	Invalid member	Null		Pass
TC-U-72	Get Role Name	21	Null member	Null		Pass
TC-U-73	Get member in role count	21	RoleName = “Member”;	int		Pass
TC-U-74	Get member in role count	21	RoleName = “”;	0		Pass
TC-U-75	Get member in role count	21	RoleName = null;	0		Pass
TC-U-76	Add book to wishlist	16	Id = 1, Reason = "Because I want to"	True		Pass
TC-U-77	Add book to wishlist	16	Id = -1, Reason = "Because I want to"	False		Pass
TC-U-78	Add book to wishlist	16	Book = null;	False		Pass
TC-U-79	Get wishlist book	16	Id = 1;	True		Pass
TC-U-80	Get wishlist book	16	Id = -1;	False		Pass
TC-U-81	Update wishlist book	16	Id = 1, Reason = "Because I want to"	True		Pass
TC-U-82	Update wishlist book	16	Id = -1, Reason = "Because I want to"	False		Pass
TC-U-83	Update wishlist book	16	WishlistBook = null;	False		Pass
TC-U-84	Remove wishlist book by book object	16	Id = 1, Reason = "Because I want to"	True		Pass
TC-U-85	Remove wishlist book by book object	16	Id = -1, Reason = "Because I want to"	False		Pass
TC-U-86	Remove wishlist book by book object	16	WishlistBook = null;	False		Pass
TC-U-87	Remove wishlist book by id	16	Id = 1;	True		Pass
TC-U-88	Remove wishlist book by id	16	Id = -1;	False		Pass
TC-U-89	Find wishlist book by id	16	Id = 1;	True		Pass
TC-U-90	Find wishlist book by id	16	Id = -1;	False		Pass

TC-U-91	Create role	10	Id = "1", Name = "Member"	True		Pass
TC-U-92	Create role	10	Role = null;	False		Pass
TC-U-93	Update role	10	Id = "1", Name = "Member"	True		Pass
TC-U-94	Update role	10	Role = null;	False		Pass
TC-U-95	Delete Role	10	Id = "1", Name = "Member"	True		Pass
TC-U-96	Delete Role	10	Role = null;	False		Pass
TC-U-97	Find role by id	10	Id = “1234”;	True		Pass
TC-U-98	Find role by id	10	Id = null;	False		Pass
TC-U-99	Find role by id	10	Id = “-2”;	False		Pass
TC-U-100	Remove from role	10	FirstName = "Bob", LastName = "Bobonson", RoleId = “1”, Role.Name = “Member”	True		Fails

Integration tests specfication

ID	R-ID	Input	Output	Environment	Pass/Fail	Preconditions
TC-I-1	4	TCIS-2	IEnumerable<CatalogueItemMap> of size 0	- database connection is established	Pass	- script " populateDb" has been ran on SQL server
TC-I-2	4	TCIS-3	IEnumerable<CatalogueItemMap> of size 0	- database connection is established	Pass	- script " populateDb" has been ran on SQL server
TC-I-3	4	TCIS-4	IEnumerable<CatalogueItemMap> of size 0	- database connection is established	Pass	- script " populateDb" has been ran on SQL server
TC-I-4	4	TCIS-5	IEnumerable<CatalogueItemMap> of size 5	- database connection is established	Pass	- script " populateDb" has been ran on SQL server
TC-I-5	4	TCIS-6	IEnumerable<CatalogueItemMap> of size 0	- database connection is established	Pass	- script " populateDb" has been ran on SQL server
TC-I-6	4	TCIS-7	IEnumerable<CatalogueItemMap> of size 0	- database connection is established	Pass	- script " populateDb" has been ran on SQL server
TC-I-7	4	TCIS-8	IEnumerable<CatalogueItemMap> of size 0	- database connection is established	Pass	- script " populateDb" has been ran on SQL server
TC-I-8	4	TCIS-9	IEnumerable<CatalogueItemMap> of size 1	- database connection is established	Pass	- script " populateDb" has been ran on SQL server
TC-I-9	4	TCIS-10	IEnumerable<CatalogueItemMap> of size 0	- database connection is established	Pass	- script " populateDb" has been ran on SQL server

TC-I-10	4	TCIS-11	IEnumerable<CatalogueItemMap> of size 0	- database connection is established	Pass	- script " populateDb" has been ran on SQL server
TC-I-11	4	TCIS-12	IEnumerable<CatalogueItemMap> of size 0	- database connection is established	Pass	- script " populateDb" has been ran on SQL server
TC-I-12	4	TCIS-13	IEnumerable<CatalogueItemMap> of size 1	- database connection is established	Pass	- script " populateDb" has been ran on SQL server
TC-I-13	4	TCIS-14	IEnumerable<CatalogueItemMap> of size 0	- database connection is established	Pass	- script " populateDb" has been ran on SQL server
TC-I-14	4	TCIS-15	IEnumerable<CatalogueItemMap> of size 5	- database connection is established	Pass	- script " populateDb" has been ran on SQL server
TC-I-15	4	TCIS-16	IEnumerable<CatalogueItemMap> of size 2	- database connection is established	Pass	- script " populateDb" has been ran on SQL server
TC-I-16	4	TCIS-17	Exception("Unexpected Database Error")	- database connection is established	Fail	- script " populateDb" has been ran on SQL server
TC-I-17	4	TCIS-18	Exception("Unexpected Database Error")	- database connection is established	Fail	- script " populateDb" has been ran on SQL server

System tests specification

Test case ID: TC-S-1
Description: Test that user can be registered with valid ssn and password and non-verified member role is assigned to him. User is automatically logged in and is redirected to the home page.

Test items: SR-8, SR-9, SR-10
Risk IDs (R-ID): 7,
Input specification: TCIS-1
Expected result: User is registered successfully.
Environmental needs:

- Google chrome browser is installed on the machine
- Connection to the database is established.
- AppConfig file of SystemTest project has valid database configuration.
- Member with ssn "111-00-0000" is not present in the database.
- Firewall does not block chromedriver.exe

Precondition steps: -

Test procedure steps

Action test step	Verification test step
Navigate Google Chrome browser to URL: "localhost:60764/Account/Register"	Browser is on URL: "localhost:60764/Account/Register"
Enter value "111-00-0000" to the input field with name "Ssn" and defocus the input field.	No validation error message is displayed.
Enter value "MyPassword1." to the input field with name "Password" and defocus the input field.	No validation error message is displayed.

Enter value “MyPassword1.” to the input field with name “Confirm Password” and defocus the input field.	No validation error message is displayed.
Left click on the submit button.	No validation error message is displayed. Browser is redirected to the URL: "http://localhost:60764/".
	On the right side of the top navbar is a message: “Hello 111-00-0000!” indicating that user with Ssn “111-00-0000” is successfully logged in.
	Verify that in the database, user with Ssn “111-00-0000” has assigned a role with ID: 2 and his password is hashed.

Postcondition steps:

- Remove user with Ssn: “111-00-0000” from the database.

Pass/ Fails:

- Pass

Input specification

ID: TCIS-1
Variables:

Variable name	Value domain	Pass/ Fail
Ssn	String("111-00-0000")	Pass
Password	String(“MyPassword1.”)	Pass
Confirm password	String(“MyPassword1.”)	Pass

ID: TCIS-2
Variables:

Variable name	Value domain	Pass/ Fail
Page index	Integer(0)	Pass
Page size	Integer(5)	Pass
Can loan	Boolean(false)	Pass
Author	String(Size (0))	Pass
Subject	String(Size (31))	Pass
Title	String(Size (101))	Pass
Isbn	String(“0439139602”)	Pass

Start date	Datetime(“2000-01-01”)	Pass
End date	Datetime(“2020-01-01”)	Pass

ID: TCIS-3
Variables:

Variable name	**Value domain**	**Pass/ Fail**
Page index	Integer(0)	Pass
Page size	Integer(5)	Pass
Can loan	Boolean(false)	Pass
Author	String(Size (0))	Pass
Subject	String(Size (0))	Pass
Title	String(Size (101))	Pass
Isbn	String(Size(21))	Pass
Start date	Datetime(max)	Pass
End date	Datetime(max)	Pass

ID: TCIS-4
Variables:

Variable name	**Value domain**	**Pass/ Fail**
Page index	Integer(0)	Pass
Page size	Integer(5)	Pass
Can loan	Boolean(true)	Pass
Author	String(“J.K. Rowling”)	Pass
Subject	String(Size (31))	Pass
Title	String(Size (0))	Pass
Isbn	String(Size(0))	Pass
Start date	Datetime(min)	Pass
End date	Datetime(max)	Pass

ID: TCIS-5
Variables:

Variable name	Value domain	Pass/ Fail
Page index	Integer(0)	Pass
Page size	Integer(5)	Pass
Can loan	Boolean(true)	Pass
Author	String(“J.K. Rowling”)	Pass
Subject	String(“Fiction”)	Pass
Title	String(“Harry Potter”)	Pass
Isbn	String(Size(0))	Pass
Start date	Datetime(“2000-01-01”)	Pass
End date	Datetime(min)	Pass

ID: TCIS-6
Variables:

Variable name	Value domain	Pass/ Fail
Page index	Integer(0)	Pass
Page size	Integer(5)	Pass
Can loan	Boolean(true)	Pass
Author	String(Size(31))	Pass
Subject	String(Size(0))	Pass
Title	String(Size(101))	Pass
Isbn	String(“0439139602”)	Pass
Start date	Datetime(min)	Pass
End date	Datetime(min)	Pass

ID: TCIS-7
Variables:

Variable name	Value domain	Pass/ Fail
Page index	Integer(0)	Pass
Page size	Integer(5)	Pass
Can loan	Boolean(true)	Pass
Author	String(Size(31))	Pass
Subject	String(“Fiction”)	Pass
Title	String(Size(0))	Pass
Isbn	String(Size(21))	Pass
Start date	Datetime(max)	Pass
End date	Datetime(“2000-01-01”)	Pass

ID: TCIS-8
Variables:

Variable name	Value domain	Pass/ Fail
Page index	Integer(0)	Pass
Page size	Integer(5)	Pass
Can loan	Boolean(false)	Pass
Author	String(“J.K. Rowling”)	Pass
Subject	String(Size(0))	Pass
Title	String(Size(0))	Pass
Isbn	String(Size(0))	Pass
Start date	Datetime(max)	Pass
End date	Datetime(min)	Pass

ID: TCIS-9
Variables:

Variable name	Value domain	Pass/ Fail
Page index	Integer(0)	Pass
Page size	Integer(5)	Pass
Can loan	Boolean(false)	Pass
Author	String(Size(0))	Pass
Subject	String(“Nonfiction”)	Pass
Title	String(Size(0))	Pass
Isbn	String(“0439139610”)	Pass
Start date	Datetime(min)	Pass
End date	Datetime(“2030-01-01”)	Pass

ID: TCIS-10
Variables:

Variable name	Value domain	Pass/ Fail
Page index	Integer(0)	Pass
Page size	Integer(5)	Pass
Can loan	Boolean(false)	Pass
Author	String(“J. K. Rowling”)	Pass
Subject	String(“Fiction”)	Pass
Title	String(“Harry Potter”)	Pass
Isbn	String(Size(21))	Pass
Start date	Datetime(min)	Pass
End date	Datetime(max)	Pass

ID: TCIS-11
Variables:

Variable name	Value domain	Pass/ Fail
Page index	Integer(0)	Pass
Page size	Integer(5)	Pass
Can loan	Boolean(false)	Pass
Author	String("J. K. Rowling")	Pass
Subject	String(Size(0))	Pass
Title	String(Size(101))	Pass
Isbn	String(Size(0))	Pass
Start date	Datetime(min)	Pass
End date	Datetime(max)	Pass

ID: TCIS-12
Variables:

Variable name	Value domain	Pass/ Fail
Page index	Integer(0)	Pass
Page size	Integer(5)	Pass
Can loan	Boolean(true)	Pass
Author	String(Size(0))	Pass
Subject	String(Size(31))	Pass
Title	String(Size(101))	Pass
Isbn	String(Size(21))	Pass
Start date	Datetime(max)	Pass
End date	Datetime(min)	Pass

ID: TCIS-13
Variables:

Variable name	Value domain	Pass/ Fail
Page index	Integer(0)	Pass
Page size	Integer(5)	Pass
Can loan	Boolean(false)	Pass
Author	String(“Simon Sinek”)	Pass
Subject	String(Size(0))	Pass
Title	String("Leaders eat last")	Pass
Isbn	String(“0439139610”)	Pass
Start date	Datetime(max)	Pass
End date	Datetime(max)	Pass

ID: TCIS-14
Variables:

Variable name	Value domain	Pass/ Fail
Page index	Integer(0)	Pass
Page size	Integer(5)	Pass
Can loan	Boolean(false)	Pass
Author	String(Size(0))	Pass
Subject	String(Size(31))	Pass
Title	String(Size(0))	Pass
Isbn	String(Size(21))	Pass
Start date	Datetime(“2000-01-01”)	Pass
End date	Datetime(max)	Pass

ID: TCIS-15
Variables:

Variable name	Value domain	Pass/ Fail
Page index	Integer(0)	Pass
Page size	Integer(5)	Pass
Can loan	Boolean(true)	Pass
Author	String(“J.K. Rowling”)	Pass
Subject	String(“Fiction”)	Pass
Title	String(“Harry Potter”)	Pass
Isbn	String(Size(0))	Pass
Start date	Datetime(min)	Pass
End date	Datetime(max)	Pass

ID: TCIS-16
Variables:

Variable name	Value domain	Pass/ Fail
Page index	Integer(1)	Pass
Page size	Integer(5)	Pass
Can loan	Boolean(true)	Pass
Author	String(“J.K. Rowling”)	Pass
Subject	String(“Fiction”)	Pass
Title	String(“Harry Potter”)	Pass
Isbn	String(Size(0))	Pass
Start date	Datetime(min)	Pass
End date	Datetime(max)	Pass

ID: TCIS-17
Variables:

Variable name	Value domain	Pass/ Fail
Page index	Integer(-1)	Pass
Page size	Integer(5)	Pass
Can loan	Boolean(true)	Pass
Author	String(“J.K. Rowling”)	Pass
Subject	String(“Fiction”)	Pass
Title	String(“Harry Potter”)	Pass
Isbn	String(Size(0))	Pass
Start date	Datetime(min)	Pass
End date	Datetime(max)	Pass

ID: TCIS-18
Variables:

Variable name	Value domain	Pass/ Fail
Page index	Integer(0)	Pass
Page size	Integer(0)	Pass
Can loan	Boolean(true)	Pass
Author	String(“J.K. Rowling”)	Pass
Subject	String(“Fiction”)	Pass
Title	String(“Harry Potter”)	Pass
Isbn	String(Size(0))	Pass
Start date	Datetime(min)	Pass
End date	Datetime(max)	Pass

Output specification

ID: Example
Variables:

Variable name		
ISBN	String(Size(x..y))	Pass
Author	String("SomeName")	Pass
...	...	...

5.1.4 Requirement to test case traceability matrix

ID	SR-ID	Requirement description	Status	System components	Software Module	TC-ID	Implemented	Tested
1	1	Librarian is able to add a book with valid ISBN.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-18	27.03.2017	27.03.2017
2	1	Librarian is able to add a book with valid ISBN.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-19	27.03.2017	27.03.2017
3	1	Librarian is able to add a book with valid ISBN.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-20	27.03.2017	27.03.2017
4	3	Librarian is able to see book description.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-33	27.03.2017	27.03.2017
5	3	Librarian is able to see book description.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-34	27.03.2017	27.03.2017
6	3	Librarian is able to see book description.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-35	27.03.2017	27.03.2017
7	4	Every user is able to see a catalogue of books.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-1	25.04.2017	26.04.2017
8	4	Every user is able to see a catalogue of books.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-2	25.04.2017	26.04.2017
9	4	Every user is able to see a catalogue of books.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-3	25.04.2017	26.04.2017
10	4	Every user is able to see a catalogue of books.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-4	25.04.2017	26.04.2017
11	4	Every user is able to see a catalogue of books.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-5	25.04.2017	26.04.2017
12	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-I-1	25.04.2017	08.05.2017

13	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-I-2	25.04.2017	08.05.2017
14	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-I-3	25.04.2017	08.05.2017
15	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-I-4	25.04.2017	08.05.2017
16	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-I-5	25.04.2017	08.05.2017
17	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-I-6	25.04.2017	08.05.2017
18	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-I-7	25.04.2017	08.05.2017
19	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-I-8	25.04.2017	08.05.2017
20	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-I-9	25.04.2017	08.05.2017
21	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-I-10	25.04.2017	08.05.2017
22	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-I-11	25.04.2017	08.05.2017
23	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-I-12	25.04.2017	08.05.2017
24	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-I-13	25.04.2017	08.05.2017
25	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-I-14	25.04.2017	08.05.2017
26	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-I-15	25.04.2017	08.05.2017
27	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-I-16	25.04.2017	08.05.2017
28	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-I-17	25.04.2017	08.05.2017
29	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-U-27	25.04.2017	08.05.2017
30	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-U-28	25.04.2017	08.05.2017
31	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-U-29	25.04.2017	08.05.2017
32	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-U-30	25.04.2017	08.05.2017

33	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-U-31	25.04.2017	08.05.2017
34	4	Every user is able to see a catalogue of books.	Completed	Book Access, Catalogue Filter Map, Sql Server Database	DataAccess Layer	TC-U-32	25.04.2017	08.05.2017
35	5	Book catalogue can be filtered with different options (e.g. search by ISBN, Author, ...)	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-10	27.03.2017	27.03.2017
36	5	Book catalogue can be filtered with different options (e.g. search by ISBN, Author, ...)	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-11	27.03.2017	27.03.2017
37	7	Librarian can add/ remove book copies of the book that already exists.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-21	27.03.2017	27.03.2017
38	11	Standard user can loan a book copy for 28 days.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-39	13.05.2017	13.05.2017
39	11	Standard user can loan a book copy for 28 days.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-40	13.05.2017	13.05.2017
40	11	Standard user can loan a book copy for 28 days.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-41	13.05.2017	13.05.2017
41	11	Standard user can loan a book copy for 28 days.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-42	13.05.2017	13.05.2017
42	11	Standard user can loan a book copy for 28 days.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-43	13.05.2017	13.05.2017
43	11	Standard user can loan a book copy for 28 days.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-44	13.05.2017	13.05.2017
44	11	Standard user can loan a book copy for 28 days.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-45	13.05.2017	13.05.2017
45	14	Librarian can update information about an existing book.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-24	27.03.2017	27.03.2017
46	14	Librarian can update information about an existing book.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-25	27.03.2017	27.03.2017
47	14	Librarian can update information about an existing book.	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-26	27.03.2017	27.03.2017
48	15	Librarian can delete/ remove book (book cannot have any book copies loaned).	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-21	27.03.2017	27.03.2017
49	15	Librarian can delete/ remove book (book cannot have any book copies loaned).	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-22	27.03.2017	27.03.2017
50	15	Librarian can delete/ remove book (book cannot have any book copies loaned).	Completed	Book Repository, Book Map, Book Access, Book	Core	TC-U-23	27.03.2017	27.03.2017

51	16	Librarian can add a book to the wish list.	Completed	Wish List Repository, Wish List Map, Wish List Access, WishListBook	Core	TC-U-76	29.04.2017	29.04.2017
52	16	Librarian can add a book to the wish list.	Completed	Wish List Repository, Wish List Map, Wish List Access, WishListBook	Core	TC-U-77	29.04.2017	29.04.2017
53	16	Librarian can add a book to the wish list.	Completed	Wish List Repository, Wish List Map, Wish List Access, WishListBook	Core	TC-U-78	29.04.2017	29.04.2017
54	21	Librarians can update member's profile information.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-16	13.05.2017	12.05.2017
55	21	Librarians can update member's profile information.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-17	13.05.2017	12.05.2017
56	21	Librarians can update member's profile information.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-46	13.05.2017	12.05.2017
57	21	Librarians can update member's profile information.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-47	13.05.2017	12.05.2017
58	21	Librarians can update member's profile information.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-48	13.05.2017	12.05.2017
59	21	Librarians can update member's profile information.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-61	13.05.2017	12.05.2017
60	21	Librarians can update member's profile information.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-62	13.05.2017	12.05.2017
61	21	Librarians can update member's profile information.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-63	13.05.2017	12.05.2017
62	21	Librarians can update member's profile information.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-64	13.05.2017	12.05.2017
63	21	Librarians can update member's profile information.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-67	13.05.2017	12.05.2017

64	21	Librarians can update member's profile information.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-68	13.05.2017	12.05.2017
65	21	Librarians can update member's profile information.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-69	13.05.2017	12.05.2017
66	21	Librarians can update member's profile information.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-70	13.05.2017	12.05.2017
67	21	Librarians can update member's profile information.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-71	13.05.2017	12.05.2017
68	21	Librarians can update member's profile information.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-72	13.05.2017	12.05.2017
69	21	Librarians can update member's profile information.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-73	13.05.2017	12.05.2017
70	21	Librarians can update member's profile information.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-74	13.05.2017	12.05.2017
71	21	Librarians can update member's profile information.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-75	13.05.2017	12.05.2017
72	22	Librarian can remove/ delete member's profile.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-12	13.05.2017	12.05.2017
73	22	Librarian can remove/ delete member's profile.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-13	13.05.2017	12.05.2017
74	22	Librarian can remove/ delete member's profile.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-14	13.05.2017	12.05.2017
75	22	Librarian can remove/ delete member's profile.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-52	13.05.2017	12.05.2017
76	22	Librarian can remove/ delete member's profile.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-53	13.05.2017	12.05.2017

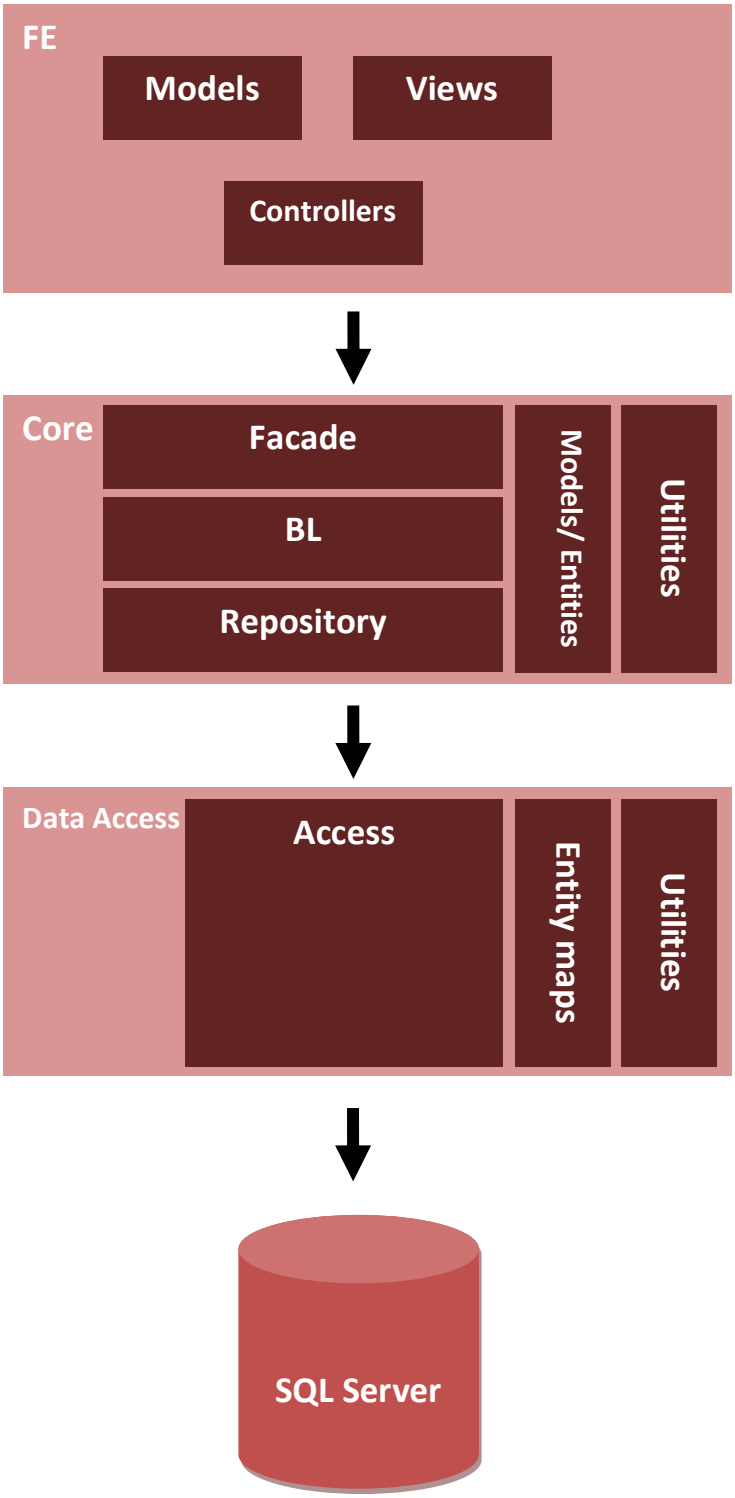
77	22	Librarian can remove/ delete member's profile.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-54	13.05.2017	12.05.2017
78	22	Librarian can remove/ delete member's profile.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-12	13.05.2017	12.05.2017
79	22	Librarian can remove/ delete member's profile.	Completed	Member Repository, Role Repository Member Map, Role Map, Member Access, Role Access Member, Role	Core	TC-U-12	13.05.2017	12.05.2017
80	8, 9, 10	User can be registered with ssn and password. Registered user can log in the system. Newly registered member have to be verified by a librarian.	Completed	-	Case.Front End, Case.Core, DataAccess Layer	TC-S-1	20.04.2017	10.05.2017

5.1.5 Risk analysis

R-ID	SR	Product risk	Likelihood	Impact	Priority
1	14	Number of book copies of a book is updated (decreased) while they are loaned and they get lost.	5	5	25
2	15	Librarian removes a book that has loaned book copies.	5	5	25
3	11	Standard user loans a non existing book copy.	4	5	20
4	11	Standard user loans a book that cannot be loaned (is not available).	4	5	20
5	11	Standard user loans a book, although he already has 5 active loans.	4	5	20
6	11	Standard user loans a non existing book.	4	5	20
7	8	User registration succeeds, but user with the same email already exists.	4	4	16
8	9	Log in succeeds if user does not exist (is not registered)	3	5	15
9	14	Member has access to update information of a book.	3	5	15
10	14	Book update succeeds with incorrect data.	3	5	15
11	15	Member can remove a book.	3	5	15
12	1	Book with invalid ISBN is added to the database	3	4	12
13	5	Filter produces null result and raises an exception	4	3	12
14	6	Connection to the catalog of other library fails	4	3	12
15	7	Librarian adds a book copy of book with invalid ISBN.	3	4	12
16	8	User registration succeeds with invalid data	4	3	12
17	11	Standard user loans a book for longer than 28 days without notice.	4	3	12
18	13	User reserves a book that is in the library.	3	4	12
19	13	User reserves a book with invalid ISBN	3	4	12
20	16	Librarian adds a book with invalid ISBN to wish list	3	4	12
21	16	Book that already exists in the system is added to wish list.	3	4	12
22	11	User loan is not registered.	2	5	10

23	1	Valid book is not added to the database.	3	3	9
24	5	Filter by incorrect ISBN raises and exception	3	3	9
25	6	Other library changes their API and request fails	3	3	9
26	10	User is verified and cannot log in.	3	3	9
27	13	User cannot reserve new book.	3	3	9
28	7	Librarian adds a book copy of book that is not registered	2	4	8
29	7	Member can remove a book copy	2	4	8
30	7	Member can add a book copy	2	4	8
31	12	User profiles are accessible by not logged in users.	2	4	8
32	16	Book that already exists on wish list is added to with list.	2	4	8
33	2	Book details are not valid and exception is thrown	2	3	6
34	9	Log in with valid information fails	3	2	6
35	11	Standard user cannot loan a book.	2	3	6
36	1	A book is added to the database by unauthorized user	1	4	4
37	7	Librarian is not able to remove a book copy	2	2	4
38	8	Password is not hashed.	1	4	4
39	13	User reserves a book and is not notified when it arrives	4	1	4
40	16	Member has access to add a book to wish list.	2	2	4
41	10	Registered user can log in without verification from librarian.	3	1	3
42	14	Book update fails and librarian is not notified.	3	1	3
43	3	Librarian is unable to access book description	1	2	2
44	3	Member can access book description	1	2	2
45	7	Librarian is not notified if there is no book copy to be removed	2	1	2
46	8	Member is registered with invalid membership expiration date (longer/ shorter than 4 years).	2	1	2
47	12	User profiles is not accessible by logged in user.	1	2	2

5.2 System architecture



5.3 Sql statements

5.3.1 Create database script

```
use [master]

--Kill all processes related to library case
create  table #sp_who(
        [SPID] int,
        [status] varchar(MAX),
        [login] varchar(MAX),
        [HostName] varchar(MAX),
        [BlkBy] varchar(MAX),
        [DBName] varchar(MAX),
        [Command] varchar(MAX),
        [CPUTime] int,
        [DiskIO] int,
        [LastBatch] varchar(MAX),
        [ProgramName] varchar(MAX),
        [SPID_1] int,
        [REQUESTID] int
)
Go

Insert into #sp_who exec sp_who2
Go

Declare @id int
Declare @kill varchar(1000);
Declare @getid cursor

set @getid = cursor for
select [SPID] from #sp_who where [DbName] = 'Library_case';

open @getid
fetch next from @getid into @id

while @@FETCH_STATUS = 0
begin
       print ('Deleting process id ' + CAST(@id as varchar(4)));
       set @kill = 'KILL ' + CAST(@id as varchar(4))
       exec (@kill);
       fetch next from @getid into @id
end

close @getid
deallocate @getid
--End kill process

Go
drop table #sp_who



Drop database [Library_case];
Go
```

```
Create database [Library_case];
Go

use [Library_case]

Create table [Role](
	[Id]                            varchar(50)         not null,
	[Name]                          varchar(20)         not null,
	[Loan-period]                   int                 not null default 21,
	[Grace-period]                  int                 not null default 7

	primary key (Id)
);

Create table [Book](
	[Isbn]                          varchar(20)         not null,
	[Title]                         varchar(100)        not null,
	[Description]                   varchar(500)        not null,
	[Subject]                       varchar(20)         not null,
	[Language]                      varchar(20)         not null,
	[Binding]                       varchar(20)         not null,
	--Added
	[Published]                     date                null   default GetDate(),
	[Edition]                       varchar(3)          null   default('1st'),
	[Author]                        varchar(40)         not null,
	[Can-loan]                      bit                 null   default('1'),

	primary key ([Isbn])
);

Create table [Wishlist](
	[Isbn]                          varchar(20)         not null,
	[Reason-to-acquire]             varchar(200)        not null,

	primary key ([Isbn]),
	foreign key ([Isbn]) references [Book]([Isbn])
);

Go

Create table [Member](
   [Ssn]                            varchar(20)         not null,
   [RoleId]                         varchar(50)         not null,
   [Campus]                         varchar(30)         not null,
   [Membership-expiration]          date                not null,
   [FirstName]                      varchar(20)         not null,
   [LastName]                       varchar(20)         not null,
   [Password]                       nvarchar(128)       not null,

   primary key ([Ssn]),
   foreign key ([RoleId]) references [Role]([id]),
);

Create table [Book_Copy](
	[Id]                            int          not null     identity(1,1),
	[Available]                     bit          null         default('1'),
	[Book-isbn]                     varchar(20)  not null,

	primary key ([Id]),
	foreign key ([Book-isbn]) references [Book]([Isbn])
);
```

```
Go

Create table [Address](
	[Member-ssn]		varchar(20)		not null,
	[Street]		varchar(20)		not null,
	[Zip]			varchar(10)		not null,
	[City]			varchar(20)		not null,

	primary key ([Member-ssn], [Street]),
	foreign key ([Member-ssn]) references [Member]([Ssn])
);

Create table [Phone](
	[Member-ssn]		varchar(20)		not null,
	[Phone]			varchar(30)		not null,

	primary key ([Member-ssn], [Phone]),
	foreign key ([Member-ssn]) references [Member]([Ssn])
);

Create table [Order](
	[Id]			int			not null	identity(1,1),
	[Member-ssn]		varchar(20)		not null,
	[Created]		date			null		default(GETDATE()),

	primary key  ([Id]),
	foreign key ([Member-ssn]) references [Member]([Ssn])
);

Create table [Reservation](
	[Id]			int			not null	identity(1,1),
	[Member-ssn]		varchar(20)		not null,
	[Book-isbn]		varchar(20)		not null,
	[Created]		date			null		default(GETDATE()),

	primary key  ([Id]),
	foreign key ([Member-ssn]) references [Member]([Ssn]),
	foreign key ([Book-isbn]) references [Book]([Isbn])
);

Go

Create table [Order_Line](
	[Order-id]			int		not null,
	[Book-copy-id]			int		not null,
	Returned			bit		null		default('0'),
	Extended			bit		null		default('0'),

	primary key([Order-id], [Book-copy-id]),
	foreign key ([Order-id]) references [Order]([Id]),
	foreign key ([Book-copy-id]) references [Book_Copy]([Id])
);
```