

Съдържание:

1. Описание на предметната област и на заданието
2. Дефиниране на схемата на релациите
 - a. Създаване на релации
 - b. Създаване на връзките между релациите
 - c. Илюстриране на схемата на релациите
3. Добавяне на примерно съдържание
 - a. Създаване на тригери
 - b. Добавяне на данни в релациите
4. Примерни заявки:
 - a. Прости заявки
 - b. Заявки върху две и повече релации
 - c. Подзаявки
 - d. Съединения
 - e. Групиране и агрегация

Описание на предметната област и на заданието

Разработената база от данни в този документ показва как една библиотека лесно и достъпно може да следи своите налично данни. Базата от данни на библиотеката предоставя цялостна информация за книгите, които притежава тя, както и информация за нейните читатели. Базата още показва кой читател коя книга е взел за по лесна справка от страна на служителите. Също така всеки читател му се предоставя читателска карта за която данните се намират в базата от данни за справка. Всяка библиотека има различни секции които разделят различните видове книги и трябва да се следи за наличието на книгите в съответна секция. За тази цел базата предоставя точно тези данни за по лесни и достъпни справки.

Цялостната идея за разработването на тази база е улесняването на служителите на библиотеката с цел бърз достъп до данните на книгите и читателите и по дълъг период на запазване на информацията за тях.

Дефиниране на схемата на релациите

- Създаване на релации

Създаване на база от данни Library

GO

CREATE DATABASE Library

В тази база от данни дефинираме релациите:

Reader която съдържа данните за читателите в библиотеката:

USE Library

GO

CREATE TABLE Reader

```
(
    egn INT NOT NULL,
    name VARCHAR(256) PRIMARY KEY NOT NULL,
    adres VARCHAR(256),
    phone INT,
    codecard INT NOT NULL,
    record_data DATETIME NOT NULL
);
```

Значение на колоните:

egn е от тип integer и съдържа ЕГН на читателя (не може да съдържа празна стойност);

name е от тип varchar от 256 знака, primary key и съдържа данни за името на читателите (не може да съдържа празна стойност);

adres е от тип varchar от 256 знака и съдържа данни за адреса на читателите (може и да съдържа празна стойност);

phone е от тип integer и съдържа данни за телефоните на читателите (може и да съдържа празна стойност);

codecard е от тип integer и съдържа специфичния код на читателската карта на всеки читател (не може да съдържа празна стойност);

record_data е от тип datetime и съдържа датата на вкарване на данните за читателя (не може да съдържа празна стойност);

Book която съдържа данни за книгите в библиотеката:

USE Library

GO

CREATE TABLE Book

```
(
    code INT PRIMARY KEY NOT NULL,
    author VARCHAR(256) NOT NULL,
    title VARCHAR(256) NOT NULL,
    year_create INT NOT NULL,
    name_section VARCHAR(256) NOT NULL,
    available CHAR(1) NOT NULL,
    CHECK ((available = 'Y' OR available = 'N') AND (year_create >= 1000 AND
year_create <= 9999))
);
```

Значение на колоните:

code е от тип integer, primary key и съдържа данните за кода на книгите (не може да съдържа празна стойност);

author е от тип varchar от 256 знака и съдържа данните за автора на книгите (не може да съдържа празна стойност);

title е от тип varchar от 256 знака и съдържа данни за заглавието на книгите (не може да съдържа празна стойност);

year_create е от тип integer и съдържа данни за годината на създаване(написване) на книгата (не може да съдържа празна стойност);

available е от тип char от 1 знак и показва дали книгата е налична или не, за тази цел тя съдържа само Y/N данни (не може да съдържа празна стойност);

Забележка: В таблица Book има проверка на ниво таблица която проверява дали available съдържа само Y/N стойности и дали годината на създаване (написване) на книгата има валидна стойност.

Section която показва наличието на книгите във всяка една секция в библиотеката (с цел справка):

USE Library

GO

CREATE TABLE Section

```
(  
    name VARCHAR(256) PRIMARY KEY NOT NULL  
);
```

Значение на колоните:

name е от тип varchar от 256 знака, primary key и съдържа данни за името на секцията (не може да съдържа празна стойност);

Reader_card която дава данни за картата на читателя:

USE Library

GO

CREATE TABLE Reader_card

```
(  
    code INT PRIMARY KEY NOT NULL,  
    date_create DATETIME NOT NULL DEFAULT GETDATE(),  
    date_renewal DATETIME NOT NULL  
);
```

Значение на колоните:

code е от тип integer, primary key и съдържа данни за кода на читателската карта (не може да съдържа празна стойност);

date_create е от тип datetime и съдържа данни за датата на създаване на читателската карта (не може да съдържа празна стойност);

date_renewal е от тип datetime и съдържа данни за датата на последното подновяване на читателската карта (не може да съдържа празна стойност);

Забележка: Ако не се въведат данни в date_create автоматично се генерира (по подразбиране) датата на настоящият ден.

Reader_book която дава данни за читателите които са взели книга от библиотеката и коя книга са взели:

USE Library

GO

```
CREATE TABLE Reader_book  
(  
    name_reader VARCHAR(256),  
    code_book INT  
);
```

Значение на колоните:

name_reader е от тип varchar от 256 знака и съдържа данни за името на читателя който е взел книгата (може и да съдържа празна стойност);

code_book е от тип integer и съдържа данни за кода на взетата книга (може и да съдържа празна стойност);

- Създаване на връзките между релациите

Добавят се foreign key и по този начин се осъществява връзката между таблиците:

```
USE Library
```

```
GO
```

```
ALTER TABLE Reader
```

```
ADD FOREIGN KEY (cardcode) REFERENCES Reader_card(code);
```

```
GO
```

```
ALTER TABLE Reader_book
```

```
ADD FOREIGN KEY (code_book) REFERENCES Book(code);
```

```
GO
```

```
ALTER TABLE Reader_book
```

```
ADD FOREIGN KEY (name_reader) REFERENCES Reader(name);
```

```
GO
```

```
ALTER TABLE Book
```

```
ADD FOREIGN KEY (name_section) REFERENCES Section(name);
```

Както следва:

В Reader се добавя foreign key на cardcode и се свързва с code от Reader_card;

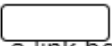
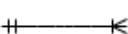

В Reader_book се добавя foreign key на code_book и се свързва с code от Book;

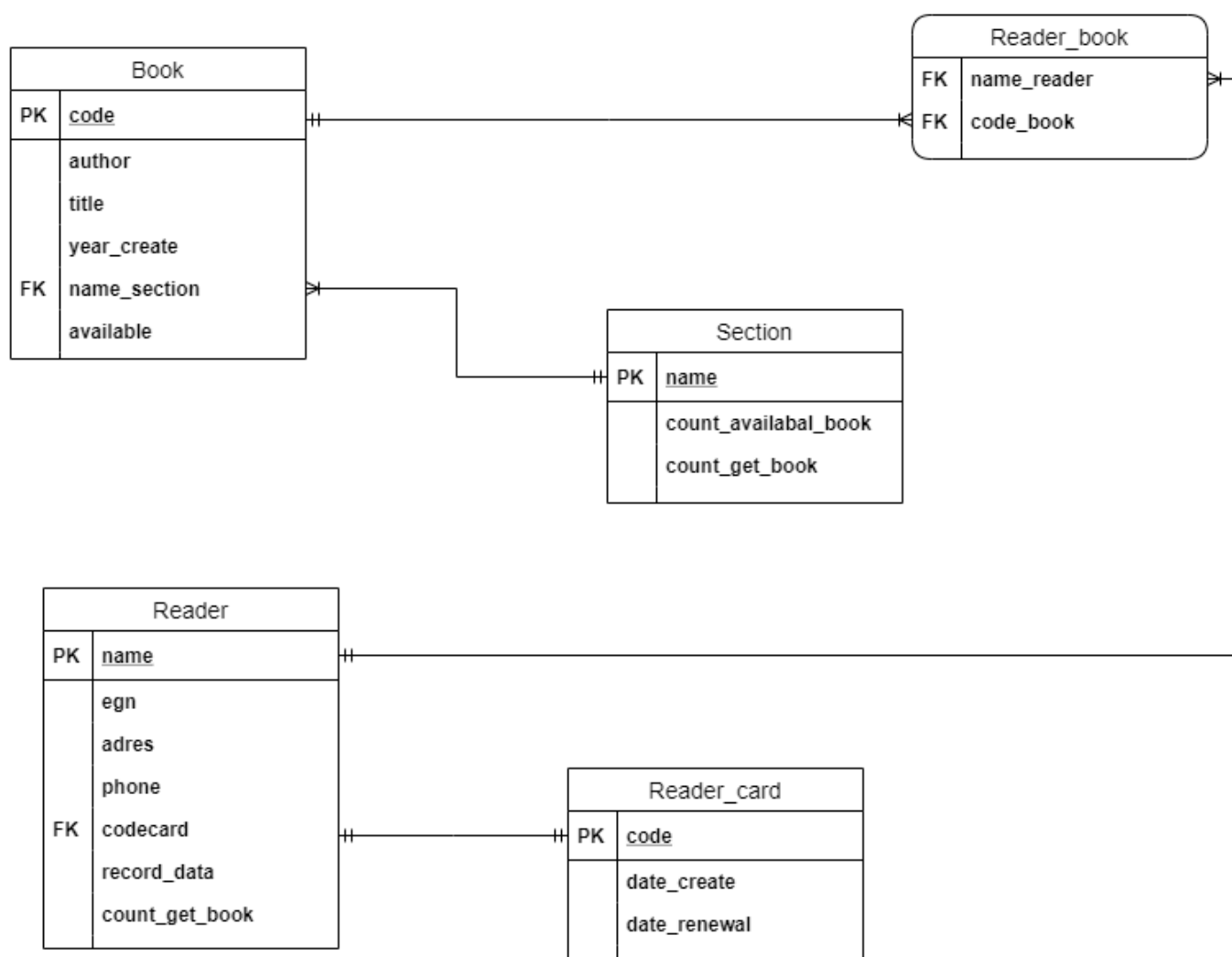
В Reader_book се добавя foreign key на name_reader и се свързва с name от Reader;

В Book се добавя foreign key на name_section и се свързва с name от Section.

- Илюстриране на схемата на релациите

Legend

- PK - Primary key;
- FK - Foreign key;
-  - shows a table that serves as a link between two tables;
-  - a one-to-many connection;
-  - a one-to-one connection;



Схемата представя по какъв начин си комуникират отделните колони в всяка една релация. Към схемата има легенда за по-голяма пълнота.

Добавяне на примерно съдържание

- Създаване на тригери

Преди въвеждането на данните в релациите ще се създадат тригерите с цел по ефикасно функциониране на базата.

```
USE Library
```

```
GO
```

```
ALTER TABLE Reader
```

```
ADD count_get_book INT;
```

```
GO
```

```
CREATE TRIGGER trg_readerbook_insert
```

```
ON reader_book
```

```
AFTER INSERT
```

```
AS
```

```
BEGIN
```

```
    UPDATE Reader SET count_get_book = count_get_book + 1
```

```
    WHERE name IN (SELECT i.name_reader FROM inserted i)
```

```
    UPDATE Book SET available = 'N'
```

```
    WHERE code IN (SELECT i.code_book FROM inserted i)
```

```
END;
```

Добавя се нова колона в Reader наречен count_get_book от тип integer който показва колко книги е взел съответният читател и се създава тригер в reader_book който при добавяне на данни в тази релация да добавя 1 в count_get_book от Reader и да сменя available от Book на 'N'.

```
USE Library
```

```
GO
```

```
ALTER TABLE Section
```

```
ADD count_availabal_book INT, count_get_book INT;
```

```
GO
```

```
CREATE TRIGGER trg_book_insert
```

```
ON Book
```

```
AFTER INSERT
```

```
AS
```

```
BEGIN
```

```
    UPDATE Section SET count_availabal_book = count_availabal_book + 1  
    WHERE name IN (SELECT i.name_section FROM inserted i WHERE
```

```
i.available = 'Y');
```

```
    UPDATE Section SET count_get_book = count_get_book + 1  
    WHERE name IN (SELECT i.name_section FROM inserted i WHERE
```

```
i.available = 'N');
```

```
END;
```

Добавя се нови колона count_availabal_book от тип integer който следи броя на наличните книги в отделните секции и count_get_book от тип integer който следи броя на взетите книги от съответната секция в Section.

Създава се тригер в Book който при добавяне на данни в тази релация да добавя 1 към count_availabal_book от Section са когато available от Book е 'Y' и добавя 1 към count_get_book от Section само ако available от Book е 'N'.

USE Library

GO

CREATE TRIGGER trg_readerbook_delete

ON reader_book

AFTER DELETE

AS

BEGIN

UPDATE Reader SET count_get_book = count_get_book - 1
WHERE name IN (SELECT d.name_reader FROM deleted d);

UPDATE Book SET available = 'Y'
WHERE code IN (SELECT d.code_book FROM deleted d)

END;

Създава се тригер в reader_book който при изтриване на данни от тази релация намалява с 1 count_get_book от Reader и променя available от Book на ,Y'.

USE Library

GO

CREATE TRIGGER trg_book_delete

ON book

AFTER DELETE

AS

BEGIN

UPDATE Section SET count_availabal_book = count_availabal_book - 1
WHERE name IN (SELECT i.name_section FROM deleted i WHERE

i.available = 'Y');

UPDATE Section SET count_get_book = count_get_book - 1
WHERE name IN (SELECT i.name_section FROM deleted i WHERE

i.available = 'N');

END;

Създава се тригер в book който при изтриване на данни от тази релация премахва 1 от count_availabal_book в Section само ако available в book е ,Y‘ и премахва 1 от count_get_book само ако available от book е ,N‘.

USE Library

GO

CREATE TRIGGER trg_reader_update

ON reader

AFTER UPDATE

AS

BEGIN

UPDATE reader_book SET name_reader = (SELECT i.name FROM inserted

WHERE name_reader IN (SELECT d.name FROM deleted d);

END;

Създава тригер в reader който при обновяване на данните в тази релация променя данните в name_reader от reader_book.

USE Library

GO

CREATE TRIGGER trg_book_update

ON book

AFTER UPDATE

AS

BEGIN

UPDATE reader_book SET code_book = (SELECT i.code FROM inserted i)

WHERE code_book IN (SELECT d.code FROM deleted d);

UPDATE Section SET count_availabal_book = count_availabal_book + 1

WHERE name IN (SELECT i.name_section FROM inserted i

JOIN deleted d ON i.code = d.code

WHERE i.available = 'Y' AND d.available = 'N');

UPDATE Section SET count_get_book = count_get_book + 1

WHERE name IN (SELECT i.name_section FROM inserted i

JOIN deleted d ON i.code = d.code

WHERE i.available = 'N' AND d.available = 'Y');

```

UPDATE Section SET count_availabal_book = count_availabal_book - 1
WHERE name IN (SELECT d.name_section FROM deleted d
                JOIN inserted i ON d.code = i.code
                WHERE i.available = 'N' AND d.available = 'Y');

UPDATE Section SET count_get_book = count_get_book - 1
WHERE name IN (SELECT d.name_section FROM deleted d
                JOIN inserted i ON i.code = d.code
                WHERE i.available = 'Y' AND d.available = 'N');

END;

```

Създава тригер в book който при обновяване на данните в тази релация променя:

- code_book от reader_book;
- увеличава count_availabal_book от Section, ако новата стойност в available от book е ,Y‘ и старата е била ,N‘;
- увеличава count_get_book, ако новата стойност в available от book е ,N‘ и старата е била ,Y‘;
- намалява count_availabal_book, ако новата стойност в available от book е ,N‘ и старата е била ,Y‘;
- намалява count_get_book, ако новата стойност в available от book е ,Y‘ и старата е била ,N‘;

- Добавяне на данни в релациите

Добавяне на данни в релацията Section:

USE Library

GO

INSERT INTO Section VALUES ('Art and Creativity',0,0);

INSERT INTO Section VALUES ('Business',0,0);

INSERT INTO Section VALUES ('Fantasy',0,0);

INSERT INTO Section VALUES ('Fitness',0,0);

INSERT INTO Section VALUES ('History',0,0);

INSERT INTO Section VALUES ('Novel',0,0);

INSERT INTO Section VALUES ('Philosophy',0,0);

INSERT INTO Section VALUES ('Science',0,0);

Добавяне на данни в релацията Book:

USE Library

GO

INSERT INTO Book VALUES (1,'Robert Kiyosaki','Rich Dad Poor Dad',1997,'Business','Y');

INSERT INTO Book VALUES (2,'George S. Clason','The Richest Man in Babylon ',1926,'Business','N');

INSERT INTO Book VALUES (3,'J. R. R. Tolkien','The Lord of the Rings',1937,'Fantasy','N');

INSERT INTO Book VALUES (4,'Frederic Delavier','Strength Training Anatomy',1998,'Fitness','Y');

INSERT INTO Book VALUES (5,'Leo Tolstoy','War and Peace',1865,'History','N');

INSERT INTO Book VALUES (6,'Carl Sagan','Cosmos',1980,'Science','Y');

INSERT INTO Book VALUES (7,'Napoleon Hill','Think and Grow Rich',1937,'Business','Y');

INSERT INTO Book VALUES (8,'F. Scott Fitzgerald','The Great Gatsby',1925,'Novel','Y');

INSERT INTO Book VALUES (9,'Daniel Defoe','Robinsons Crusoe',1719,'Novel','Y');

Добавяне на данни в релацията Reader:

USE Library

GO

```
INSERT INTO Reader VALUES (12345,'Desislava Ivanova','Sofia',89123,10,'2008-11-11 00:00:00.000',0);
```

```
INSERT INTO Reader VALUES (12346,'Georgi Georgiev','Pleven','088124',11,'2011-5-11',0);
```

```
INSERT INTO Reader VALUES (12347,'Emilia Yancheva','Sofia','088125',12,'2015-9-6',0);
```

```
INSERT INTO Reader VALUES (12348,'Galin Petrov','Dobrich','088126',13,'2017-10-29',0);
```

```
INSERT INTO Reader VALUES (12349,'Yavor Yanakiev','Varna','088127',14,'2019-10-29',0);
```

```
INSERT INTO Reader VALUES (12350,'Yanica Vulkova','Ruse','088128',15,'2020-3-15',0);
```

Добавяне на данни в релацията Reader_card:

USE Library

GO

```
INSERT INTO Reader_card VALUES (10,'2008-11-11 00:00:00.000','2010-11-11 00:00:00.000');
```

```
INSERT INTO Reader_card VALUES (11,'2011-05-11 00:00:00.000','2014-05-11 00:00:00.000');
```

```
INSERT INTO Reader_card VALUES (12,'2015-09-06 00:00:00.000','2017-09-06 00:00:00.000');
```

```
INSERT INTO Reader_card VALUES (13,'2017-10-29 00:00:00.000','2019-10-29 00:00:00.000');
```

```
INSERT INTO Reader_card VALUES (14,'2019-10-29 00:00:00.000','2019-10-29 00:00:00.000');
```

```
INSERT INTO Reader_card VALUES (15,'2020-03-15 00:00:00.000','2020-03-15 00:00:00.000');
```

Добавяне на данни в релацията Reader_book:

USE Library

GO

```
INSERT INTO Reader_book VALUES ('Galin Petrov',1);  
INSERT INTO Reader_book VALUES ('Yavor Yanakiev',2);  
INSERT INTO Reader_book VALUES ('Yanica Vulkova',3);  
INSERT INTO Reader_book VALUES ('Emilia Yancheva',7);  
INSERT INTO Reader_book VALUES ('Georgi Georgiev',4);  
INSERT INTO Reader_book VALUES ('Georgi Georgiev',5);
```

Забележка: Данните в релациите са въвеждани в последователността в която са в документа с цел избягване на ненужни усложнения, също така връзките са създадени след създаването на тригерите и въвеждането на данните в релациите.

Примерни заявки

- Прости заявки

Заявка 1:

USE Library

GO

```
SELECT code AS 'Code of book', title AS 'Title of book'
FROM book
WHERE name_section LIKE 'Business' AND year_create >= 1926
ORDER BY title ASC;
```

Резултат:

	Code of book	Title of book
1	1	Rich Dad Poor Dad
2	2	The Richest Man in Babylon
3	7	Think and Grow Rich

Обосновка:

Заявката извежда кода и заглавието на книгите които са от секция Business и са написани след 1926 сортирани низходящо по заглавието на книгата.

Заявка 2:

USE Library

GO

```
SELECT name AS 'Name of reader', adres AS 'Address of reader', phone AS
'Phone of reader', codecard AS 'Code of reader card'
FROM reader
WHERE adres = 'Sofia'
ORDER BY name DESC;
```

Резултат:

	Name of reader	Address of reader	Phone of reader	Code of reader
1	Emilia Yancheva	Sofia	88125	12
2	Desislava Ivanova	Sofia	89123	10

Обосновка:

Заявката извежда имената, адресите, телефоните и кода на читателските карти на тези читатели които имат адрес Sofia сортирани възходящо по името на читателя.

Заявка 3:

USE Library

GO

SELECT *

FROM section

WHERE name NOT LIKE 'Art and Creativity';

Резултат:

	name	count_availabal_book	count_get_book
1	Business	0	3
2	Fantasy	0	1
3	Fitness	0	1
4	History	0	1
5	Novel	2	0
6	Philosophy	0	0
7	Science	1	0

Обосновка:

Заявката извежда цялата информация за секциите без тази на Art and Creativity.

Заявка 4:

USE Library

GO

SELECT *

FROM book

WHERE available = 'Y';

Резултат:

	code	author	title	year_create	name_section	available
1	6	Carl Sagan	Cosmos	1980	Science	Y
2	8	F. Scott Fitzgerald	The Great Gatsby	1925	Novel	Y
3	9	Daniel Defoe	Robinsons Crusoe	1719	Novel	Y

Обосновка:

Заявката извежда цялата информация за книгите които са свободни.

Заявка 5:

USE Library

GO

SELECT name_reader AS 'Name of reader'

FROM reader_book

WHERE code_book = 1;

Резултат:

	Name of reader
1	Galin Petrov

Обосновка:

Заявката извежда имената на читателите които са взели книга с код 1.

- Заявки върху две и повече релации

Заявка 1:

USE Library

GO

```
SELECT b.title AS 'Title of book'  
FROM book b  
JOIN reader_book rb ON b.code = rb.code_book  
WHERE name_reader = 'Emilia Yancheva';
```

Резултат:

	Title of book
1	Think and Grow Rich

Обосновка:

Заявката извежда заглавието на книгите които Emilia Yancheva е взела.

Заявка 2:

USE Library

GO

```
SELECT rb.name_reader AS 'Name of reader'  
FROM Reader_book rb  
JOIN book b ON rb.code_book = b.code  
WHERE b.name_section = 'Fantasy';
```

Резултат:

	Name of reader
1	Yanica Vulkova

Обосновка:

Заявката извежда имената на читателите които са взели книги само от секция Fantasy.

Заявка 3:

USE Library

GO

```
SELECT r.name AS 'Name of reader', r.codecard AS 'Code of reader card',
FORMAT(rc.date_renewal,N'dd.MM.yyyy') AS 'Date of create profil'
FROM reader r
JOIN reader_card rc ON r.codecard = rc.code
WHERE rc.date_renewal > '2015-06-06';
```

Резултат:

	Name of reader	Code of reader card	Date of create profil
1	Emilia Yancheva	12	06.09.2017
2	Galin Petrov	13	29.10.2019
3	Yanica Vulkova	15	15.03.2020
4	Yavor Yanakiev	14	29.10.2019

Обосновка:

Заявката извежда имената на читателите, техният код на читателската им карта и дата на създаване на профила им която да е след 2015-06-06.

Забележка:

FORMAT() извежда датата на създаване на профила на читателя в формат „[ден].[месец].[година]“.

Заявка 4:

USE Library

GO

```
SELECT rb.name_reader AS 'Name of reader'
FROM Reader_book rb
JOIN book b ON rb.code_book = b.code
WHERE b.author = 'Robert Kiyosaki'
UNION ALL
SELECT rb.name_reader
FROM Reader_book rb
JOIN book b ON rb.code_book = b.code
WHERE b.author = 'Napoleon Hill';
```

Резултат:

	Name of reader
1	Galin Petrov
2	Emilia Yancheva

Обосновка:

Заявката извежда имената на читателите които са взели книги с автори Robert Kiyosaki или Napoleon Hill.

Заявка 5:

USE Library

GO

```
SELECT r.name AS 'Name of reader', r.phone AS 'Phone of reader', r.codecard
AS 'Code of reader card'
FROM reader r
JOIN Reader_book rb ON r.name = rb.name_reader
JOIN book b ON rb.code_book = b.code
WHERE LEFT(b.author,1) = LEFT(b.title,1);
```

Резултат:

	Name of reader	Phone of reader	Code of reader card
1	Galin Petrov	88126	13

Обосновка:

Заявката извежда имената на читателите, телефоните им и кода на читателската им карта само на тези които са взели книга чиито автор и заглавие започват с еднакви букви

Забележка:

LEFT() връща определен брой елементи от променливата която му е подадена като този брой е точно дефиниран и броенето е от ляво на дясно.

- Подзаявки

Заявка 1:

USE Library

GO

```
SELECT r.name AS 'Name of reader', r.egn AS 'EGN of reader'  
FROM reader r  
WHERE r.codecard IN (SELECT rc.code FROM reader_card rc WHERE  
YEAR(rc.date_create) = FORMAT(rc.date_renewal, N'yyyy'))  
ORDER BY name DESC;
```

Резултат:

	Name of reader	EGN of reader
1	Yavor Yanakiev	12349
2	Yanica Vulkova	12350

Обосновка:

Заявката извежда името и егенето на читателите на които годината на създаване съвпада с годината на подновяване на читателските им карти.

Забележка:

YEAR() връща единствено годината на подадената му дата.

Заявка 2:

USE Library

GO

```
SELECT r.name AS 'Name of reader', r.adres AS 'Address of reader'  
FROM reader r  
WHERE r.codecard IN (SELECT rc.code FROM reader_card rc WHERE  
FORMAT(rc.date_create, N'dd') = 11)  
AND r.count_get_book > 1;
```


Резултат:

	Name of reader	Address of reader
1	Georgi Georgiev	Pleven

Обосновка:

Заявката извежда имената и адресите на читателите чийто читателски карти са направени на 11 (стойности е ден) и броя на взетите от него книги да е повече от 1.

Заявка 3:

USE Library

GO

```
SELECT b.title AS 'Title of book', b.author AS 'Author of book'
FROM book b
WHERE b.name_section IN (SELECT s.name FROM section s WHERE
s.count_availabal_book >= 1);
```

Резултат:

	Title of book	Author of book
1	Cosmos	Carl Sagan
2	The Great Gatsby	F. Scott Fitzgerald
3	Robinsons Crusoe	Daniel Defoe

Обосновка:

Заявката извежда заглавието и автора на книгите чиито секции имат поне една свободна книга.

Заявка 4:

USE Library

GO

```
SELECT rb.name_reader AS 'Name of reader'  
FROM reader_book rb  
WHERE rb.code_book IN (SELECT b.code FROM book b WHERE b.year_create  
<= 1937);
```

Резултат:

	Name of reader
1	Yavor Yanakiev
2	Yanica Vulkova
3	Emilia Yancheva
4	Georgi Georgiev

Обосновка:

Заявката извежда имената на читателите които са взели книга която е написана през или преди 1937.

Заявка 5:

USE Library

GO

```
SELECT b.title AS 'Title of book', b.author AS 'Author of book', b.year_create AS  
'Year of create of book'  
FROM book b  
WHERE b.code IN (SELECT rb.code_book FROM reader_book rb WHERE  
rb.name_reader = 'Georgi Georgiev')  
AND b.name_section IN (SELECT s.name FROM section s WHERE  
s.count_availabal_book >= 1 OR s.count_get_book >= 1);
```

Резултат:

	Title of book	Author of book	Year of create of book
1	Strength Training Anatomy	Frederic Delavier	1998
2	War and Peace	Leo Tolstoy	1865

Обосновка:

Заявката изважда заглавието, автора, годината на създаване (написване) на книгите, които са взети от Georgi Georgiev и секцията им има поне 1 взета или невзета книга.

- Съединения

Заявка 1:

USE Library

GO

```
CREATE INDEX idx_book_year  
ON book(code,year_create);
```

GO

```
SELECT r.name AS 'Name of reader', r.adres AS 'Address of reader',  
FORMAT(rc.date_create,'dd-MM-yyyy') AS 'Date of create reader_card'  
FROM reader r  
JOIN reader_card rc ON rc.code = r.codecard  
JOIN reader_book rb ON r.name = rb.name_reader  
JOIN book b ON b.code = rb.code_book  
WHERE b.year_create = 1937;
```

Резултат:

	Name of reader	Address of reader	Date of create reader_card
1	Yanica Vulkova	Ruse	15-03-2020
2	Emilia Yancheva	Sofia	06-09-2015

Обосновка:

Създаваме индекс в book с цел по бърз достъп до годината на създаване (написване) на книгата. Заявката извежда името, адреса и датата на създаване на читателската карта на читателите които са взели книга написана през 1937.

Заявка 2:

USE Library

GO

```
SELECT s.name AS 'Name of section', b.title AS 'Title of book'  
FROM section s  
FULL OUTER JOIN book b ON s.name = b.name_section;
```

Резултат:

	Name of section	Title of book
1	Art and Creativity	NULL
2	Business	Rich Dad Poor Dad
3	Business	The Richest Man in Babylon
4	Business	Think and Grow Rich
5	Fantasy	The Lord of the Rings
6	Fitness	Strength Training Anatomy
7	History	War and Peace
8	Novel	The Great Gatsby
9	Novel	Robinsons Crusoe
10	Philosophy	NULL
11	Science	Cosmos

Обосновка:

Заявката извежда всички секции и заглавията на всички книги към тях дори и секциите да не съдържат нито една книга книга.

Заявка 3:

USE Library

GO

```
CREATE INDEX idx_book_aval_ns  
ON book(code,available,name_section);
```

GO

```
CREATE INDEX idx_readercard_dr  
ON reader_card(code,date_renewal);
```

GO

```
SELECT rc.code AS 'Code of reader card'  
FROM reader_card rc  
JOIN reader r ON rc.code = r.codecard  
JOIN reader_book rb ON rb.name_reader = r.name  
JOIN book b ON b.code = rb.code_book  
WHERE b.available = 'N' AND b.name_section = 'History' AND  
YEAR(rc.date_renewal) >= 2010;
```

Резултат:

	Code of reader card
1	11

Обосновка:

Създават се 2 индекса първият в book е с цел по бърз достъп до наличните книги и техните секции, а вторият в reader_card е с цел по бърз достъп до датата на последно подновяване на читателска карта. Заявката извежда кодовете на читателските карти на тези читатели които са взели книга от секция History и картата е последно подновена през или след 2010 год.

Заявка 4:

USE Library

GO

CREATE INDEX idx_reader_cgb

ON reader(name,count_get_book);

GO

CREATE INDEX idx_book_aut

ON book(code,author);

GO

SELECT r.name AS 'Name of reader', b.title AS 'Book title'

FROM reader r

JOIN reader_book rb ON rb.name_reader = r.name

JOIN book b ON b.code = rb.code_book

WHERE r.count_get_book <= 2 AND b.author = 'Frederic Delavier' OR b.author = 'Daniel Defoe';

Резултат:

	Name of reader	Book title
1	Georgi Georgiev	Strength Training Anatomy

Обосновка:

Създаваме 2 индекса, първият в reader с цел по бърз достъп до броя на книгите които е взел всеки един читател, а вторият в book с цел по бърз достъп до авторите на всяка една книга книгите. Заявката извежда заглавието на книгите и имената на читателите, които имат не повече от 2 взети книги, които са написани от Frederic Delavier или Daniel Defoe.

Заявка 5:

USE Library

GO

CREATE INDEX idx_section_cd

ON section(name,count_get_book);

GO

SELECT s.name AS 'Name of section', b.title AS 'Title of book', rb.name_reader
AS 'Name of reader', r.egn AS 'Reader EGN', FORMAT(rc.date_create,'MMMM')
AS 'Month of create of profile'

FROM section s

JOIN book b ON b.name_section = s.name

JOIN reader_book rb ON rb.code_book = b.code

JOIN reader r ON r.name = rb.name_reader

JOIN reader_card rc ON rc.code = r.codecard

WHERE s.count_get_book >= 1 AND b.year_create > 1865 AND b.available =
'N';

Резултат:

	Name of section	Title of book	Name of reader	Reader EGN	Month of create of profile
1	Business	Rich Dad Poor Dad	Galin Petrov	12348	October
2	Business	The Richest Man in Babylon	Yavor Yanakiev	12349	October
3	Fantasy	The Lord of the Rings	Yanica Vulkova	12350	March
4	Business	Think and Grow Rich	Emilia Yancheva	12347	September
5	Fitness	Strength Training Anatomy	Georgi Georgiev	12346	May

Обосновка:

Създава се индекс в section с цел по бърз достъп до броя на книгите които са взети в момента за всяка една секция. Заявката извежда имената на секциите, които имат поне една взета книга и взетите книги от тази секция да са написани след 1865, извежда и заглавието на тези книги, имената на читателите, които са ги взели и тяхното ЕГН и месеца на създаване на читателската им карта изписана с думи.

Забележка:

FORMAT() позволява и изписването на дните и месеците с думи (зависи от зададения формат на датата).

- Групиране и аграгация

Преди заявките създаваме изгледите с цел по голяма пълнота:

Изглед 1

USE Library

```
GO
CREATE VIEW readerbook_book
AS
SELECT rb.name_reader, b.title
FROM reader_book rb
JOIN book b ON b.code = rb.code_book;
```

Обосновка:

Създава изглед който показва името на читателите които са взели книга и заглавията на съответните книги.

Изглед 2:

USE Library

```
GO
CREATE VIEW reader_readercard_readerbook
AS
SELECT rb.name_reader, r.egn, r.adres, rc.date_renewal
FROM reader r
JOIN reader_book rb ON rb.name_reader = r.name
JOIN reader_card rc ON rc.code = r.codecard;
```

Обосновка:

Създава изглед който включва имената на читателите, които са взели поне една книга, ЕГН-то им, адреса им и датата на последното подновяване на картата им.

Изглед 3:

USE Library

GO

CREATE VIEW reader_readercard

AS

SELECT r.name, r.egn, r.adres, r.phone,

r.count_get_book, FORMAT(r.record_data, 'dd-MM-yyyy') AS date

FROM reader r

JOIN reader_card rc ON rc.code = r.codecard

WHERE MONTH(rc.date_create) %2 <> 0 AND MONTH(rc.date_renewal) %2 <>

0;

Обосновка:

Създава изглед който е за данните на читателя, чиито месеци на създаване и подновяване на картите е нечетно число

Изглед 4:

USE Library

GO

CREATE VIEW book_section

AS

SELECT b.title, b.author, b.year_create, b.name_section

FROM book b

JOIN section s ON s.name = b.name_section

WHERE s.count_get_book >= 1 AND s.name LIKE '%e%';

Обосновка:

Създава изглед който е за данните на книгите чиито секции имат поне 1 взета книга и в името си съдържат 'e'.

Заявка 1:

USE Library

GO

```
SELECT SUM(s.count_availabal_book + s.count_get_book) AS 'All book in the library'  
FROM section s;
```

Резултат:

	All book in the library
1	9

Обосновка:

Заявката извежда колко е общият брой на всички книги в библиотеката

Заявка 2:

USE Library

GO

```
SELECT bs.title AS 'Title of book', bs.author AS 'Author of book', bs.year_create  
AS 'Year of writing'  
FROM book_section bs  
JOIN  
(SELECT name_section FROM book_section  
GROUP BY name_section  
HAVING COUNT(name_section)>1) bs1 ON bs1.name_section =  
bs.name_section;
```

Резултат:

	Title of book	Author of book	Year of writing
1	Rich Dad Poor Dad	Robert Kiyosaki	1997
2	The Richest Man in Babylon	George S. Clason	1926
3	Think and Grow Rich	Napoleon Hill	1937

Обосновка:

Заявката извежда информация за заглавието, автора, годината на написване на книгите чиито секции имат поне 1 взета книга и в името си съдържат 'е', също така извежда само тези книги които се намират в секция която се среща повече от един път.

Заявка 3:

USE Library

GO

```
SELECT rb.name_reader AS 'Name of reader', rb.title AS 'Title of book'
FROM readerbook_book rb
JOIN(
SELECT name_reader FROM readerbook_book
GROUP BY name_reader
HAVING COUNT(name_reader) > 1) rb1 ON rb1.name_reader =
rb.name_reader;
```

Резултат:

	Name of reader	Title of book
1	Georgi Georgiev	Strength Training Anatomy
2	Georgi Georgiev	War and Peace

Обосновка:

Заявката извежда имената на читателите които са взели повече от една книга и заглавията им.

Заявка 4:

USE Library

GO

```
SELECT rr.name AS 'Name of reader', rr.egn AS 'EGN', rr.adres AS 'Adres',  
rr.phone AS 'Phone', rr.count_get_book AS 'Number of get book', rr.date AS  
'Date create of account'  
FROM reader_readercard rr  
JOIN  
(SELECT adres  
FROM reader_readercard  
GROUP BY adres  
HAVING COUNT(1) > 1) rr1 ON rr.adres = rr1.adres  
WHERE RIGHT(rr.phone,1) %2 <> 0;
```

Резултат:

	Name of reader	EGN	Adres	Phone	Number of get book	Date create of account
1	Desislava Ivanova	12345	Sofia	89123	0	11-11-2008
2	Emilia Yancheva	12347	Sofia	88125	1	06-09-2015

Обосновка:

Заявката извежда данните на читателите, чиито месеци на създаване и подновяване на читателската им картите е нечетно число с еднакви адреси и телефоните им завършват на нечетна цифра

Забележка:

RIGHT() връща определен брой елементи от променливата която му е подадена като този брой е точно дефиниран и броенето е от дясно на ляво.

Заявка 5:

USE Library

GO

```
SELECT DISTINCT rrr.name_reader AS 'Name of reader', rrr.egn AS 'Reader EGN', rrr.adres AS 'Reader address', rrr.date_renewal AS 'Date of create profile', b.title AS 'Title of book', b.author AS 'Author of book', b.year_create AS 'Year of create book', b.name_section AS 'Book section'
FROM reader_readercard_readerbook rrr
JOIN reader_book rb ON rb.name_reader = rrr.name_reader
JOIN book b ON b.code = rb.code_book
JOIN
(SELECT name_section
FROM book
GROUP BY name_section
HAVING COUNT(name_section) > 1) b1 ON b1.name_section = b.name_section
WHERE LEN(b.author) > (SELECT RIGHT(MIN(YEAR(date_renewal)),2) FROM reader_readercard_readerbook);
```

Резултат:

	Name of reader	Reader EGN	Reader address	Date of create profile	Title of book	Author of book	Year of create book	Book section
1	Galin Petrov	12348	Dobrich	2019-10-29 00:00:00.000	Rich Dad Poor Dad	Robert Kiyosaki	1997	Business
2	Yavor Yanakiev	12349	Varna	2019-10-29 00:00:00.000	The Richest Man in Babylon	George S. Clason	1926	Business

Обосновка:

Заявката извежда имената на читателите, които са взели поне една книга, ЕГН-то им, адреса им и датата на последното подновяване на картата им и информация за книгата която са взели, като извежда само тези които дължината на името на автора е по-голяма от последните 2 цифри на най-малката година на създаване на профила на читателя, също така само информацията за книгите и читателите от секция с повече от 1 взета книга.

Забележка:

LEN() връща дължината на посочената променлива.

Заявка 6:

USE Library

GO

SELECT COUNT(s.name) AS 'Count of name section'

FROM book b

JOIN section s ON s.name= b.name_section

WHERE s.count_availabal_book >= 1 AND LEFT(b.year_create,2) = 19;

Резултат:

	Count of name section
1	2

Обосновка:

Заявката извежда броя на секциите които имат поне 1 налична книга и годината на написване на книга от тази секция да са от 20 век.

Заявка 7:

USE Library

GO

```
SELECT r.name AS 'Name of reader', r.adres AS 'Address of reader', r.egn AS 'Reader EGN', r.phone AS 'Reader phone', r.record_data AS 'Date of create account', rc.date_create AS 'Date of create reader card', rc.date_renewal AS 'Date of last renewal'
FROM reader r
JOIN reader_card rc ON r.codecard = rc.code
JOIN
(SELECT r.record_data
FROM reader r
JOIN reader_card rc ON rc.code = r.codecard
WHERE YEAR(r.record_data) BETWEEN (SELECT MIN(YEAR(date_renewal))
FROM reader_card) AND (SELECT MAX(YEAR(date_create)) FROM
Reader_card)
) r2 ON r2.record_data = r.record_data
JOIN(
SELECT adres
FROM reader
GROUP BY adres
HAVING COUNT(1) > 1
) r1 ON r1.adres = r.adres;
```

Резултат:

	Name of reader	Address of reader	Reader EGN	Reader phone	Date of create account	Date of create reader card	Date of last renewal
1	Emilia Yancheva	Sofia	12347	88125	2015-09-06 00:00:00.000	2015-09-06 00:00:00.000	2017-09-06 00:00:00.000

Обосновка:

Заявката извежда информация за читателите и читателските им карти като извежда единствено тези на които годината на създаване на профила е между най-малката година на подновяване и най-голямата на създаване на читателската карта и читателите с еднакви адреси.

Забележка:

BETWEEN [val1] AND [val2] указва в какъв диапазон трябва да варира дадена стойност.

Заявка 8:

USE Library

GO

```
SELECT b.author AS 'Author of book'
FROM book b
JOIN
(SELECT year_create
FROM book
GROUP BY year_create
HAVING COUNT(1)>1) b1 ON b.year_create = b1.year_create
WHERE b.available = 'N';
```

Резултат:

	Author of book
1	J. R. R. Tolkien
2	Napoleon Hill

Обосновка:

Заявката извежда имената на авторите на книгите написани през една и съща година и не са налични.

Заявка 9:

USE Library

GO

```
SELECT COUNT(r.name) AS 'Name of reader'
FROM reader r
JOIN reader_book rb ON rb.name_reader = r.name
JOIN book b ON b.code = rb.code_book
WHERE r.count_get_book = 1 AND b.name_section = 'Business';
```

Резултат:

	Name of reader
1	3

Обосновка:

Заявката извежда броя на читателите взели по 1 книга от секция Business.

Заявка 10:

USE Library

GO

```
SELECT r.name AS 'Name of reader', r.adres AS 'Address of reader', r.egn AS
'Reader EGN', r.phone AS 'Phone of reader', r.record_data AS 'Date of create
profile', rc.date_create AS 'Date of create reader card', rc.date_renewal AS
'Date of renewal reader card', b.title AS 'Title of book', b.author AS 'Author of
book', b.year_create AS 'Year of create book', b.name_section AS 'Name of
section', s.count_get_book AS 'Section get book'
FROM reader r
JOIN reader_card rc ON rc.code = r.codecard
JOIN reader_book rb ON rb.name_reader = r.name
JOIN book b ON b.code = rb.code_book
JOIN section s ON s.name = b.name_section
JOIN(
SELECT name_section
FROM book
GROUP BY name_section
HAVING COUNT(1) > 1
) b1 ON b1.name_section = b.name_section
WHERE b.year_create >= (SELECT AVG(year_create) FROM book)
ORDER BY r.name DESC;
```

Резултат:

	Name of reader	Address of reader	Reader EGN	Phone of reader	Date of create profile	Date of create reader card	Date of renewal reader card	Title of book	Author of book	Year of create book	Name of section	Section get book
1	Yavor Yanakiev	Varna	12349	88127	2019-10-29 00:00:00.000	2019-10-29 00:00:00.000	2019-10-29 00:00:00.000	The Richest Man in Babylon	George S. Clason	1926	Business	3
2	Galin Petrov	Dobrich	12348	88126	2017-10-29 00:00:00.000	2017-10-29 00:00:00.000	2019-10-29 00:00:00.000	Rich Dad Poor Dad	Robert Kiyosaki	1997	Business	3
3	Emilia Yancheva	Sofia	12347	88125	2015-09-06 00:00:00.000	2015-09-06 00:00:00.000	2017-09-06 00:00:00.000	Think and Grow Rich	Napoleon Hill	1937	Business	3

Обосновка:

Заявката извежда информация за читателите и книгите които са взели на които секциите се повтарят, годините на написване на книгите да надвишават средната им годишна стойност и са сортирани по името на читателя възходящо.