Съдържание:

1. [Описание на предметната област и на заданието](#_Описание_на_предметната_1)
2. [Дефиниране на схемата на релациите](#_Дефиниране_на_схемата)
   1. [Създаване на релации](#_Създаване_на_релации)
   2. [Създаване на връзките между релациите](#_Създаване_на_връзките)
   3. [Илюстриране на схемата на релациите](#_Илюстриране_на_схемата)
3. [Добавяне на примерно съдържание](#_Добавяне_на_примерно)
   1. [Създаване на тригери](#_Създаване_на_тригери)
   2. [Добавяне на данни в релациите](#_Добавяне_на_данни)
4. [Примерни заявки](#_Примерни_заявки):
   1. [Прости заявки](#_Прости_заявки)
   2. [Заявки върху две и повече релации](#_Заявки_върху_две)
   3. [Подзаявки](#_Подзаявки)
   4. [Съединения](#_Съединения)
   5. [Групиране и аграгация](#_Групиране_и_аграгация)

# 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:

еgn 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 e 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 (codecard) 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:

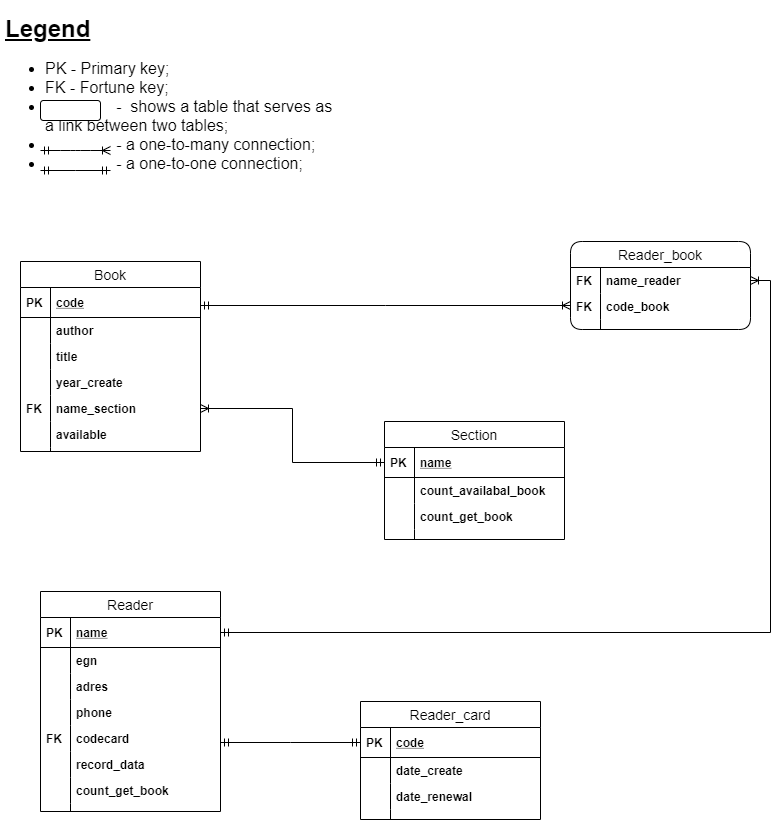
В Reader се добавя foreign key на codecard и се свързва с code от Reader\_card;

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



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 i)

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

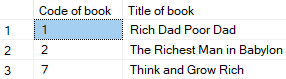
SELECT code AS 'Code of book', title AS 'Title of book'

FROM book

WHERE name\_section LIKE 'Business' AND year\_create >= 1926

ORDER BY title ASC;

Резултат:



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

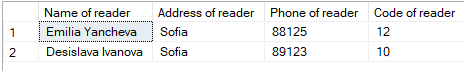
SELECT name AS 'Name of reader', adres AS 'Address of reader', phone AS 'Phone of reader', codecard AS 'Code of reader card'

FROM reader

WHERE adres = 'Sofia'

ORDER BY name DESC;

Резултат:



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

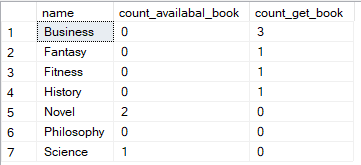
GO

SELECT \*

FROM section

WHERE name NOT LIKE 'Art and Creativity';

Резултат:



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:



Обосновка:

Заявката извежда цялата информация за книгите които са свободни.

Заявка 5:

USE Library

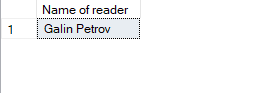
GO

SELECT name\_reader AS 'Name of reader'

FROM reader\_book

WHERE code\_book = 1;

Резултат:



Обосновка:

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

# Заявки върху две и повече релации

Заявка 1:

USE Library

GO

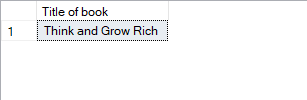
SELECT b.title AS 'Title of book'

FROM book b

JOIN reader\_book rb ON b.code = rb.code\_book

WHERE name\_reader = 'Emilia Yancheva';

Резултат:



Обосновка:

Заявката извежда заглавието на книгите които Emilia Yancheva е взела.

Заявка 2:

USE Library

GO

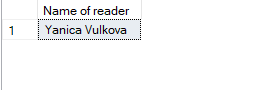
SELECT rb.name\_reader AS 'Name of reader'

FROM Reader\_book rb

JOIN book b ON rb.code\_book = b.code

WHERE b.name\_section = 'Fantasy';

Резултат:



Обосновка:

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

Заявка 3:

USE Library

GO

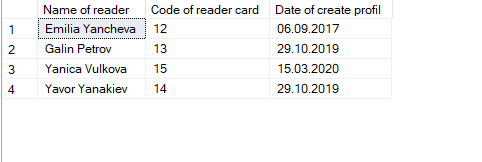
SELECT r.name AS 'Name of reader', r.codecard AS 'Code of reader card', FORMAT(rc.date\_renewal,N'dd.MM.yyyy') AS 'Date of create profil'

FROM reader r

JOIN reader\_card rc ON r.codecard = rc.code

WHERE rc.date\_renewal > '2015-06-06';

Резултат:



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

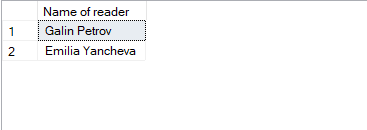
SELECT rb.name\_reader

FROM Reader\_book rb

JOIN book b ON rb.code\_book = b.code

WHERE b.author = 'Napoleon Hill';

Резултат:



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'

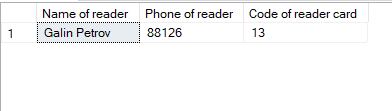
FROM reader r

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:



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

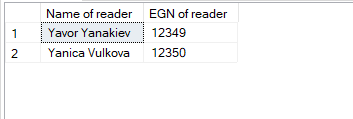
SELECT r.name AS 'Name of reader', r.egn AS 'EGN of reader'

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:



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

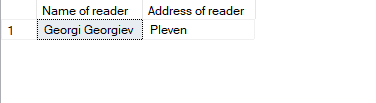
SELECT r.name AS 'Name of reader', r.adres AS 'Address of reader'

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:



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

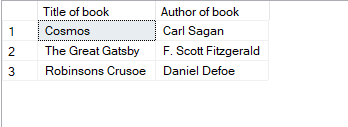
GO

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:



Обосновка:

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

Заявка 4:

USE Library

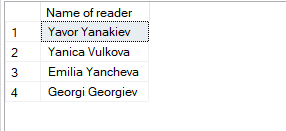
GO

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:



Обосновка:

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

Заявка 5:

USE Library

GO

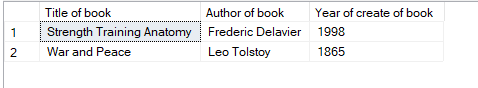
SELECT b.title AS 'Title of book', b.author AS 'Author of book', b.year\_create AS 'Year of create of book'

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:



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

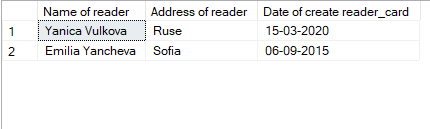
JOIN reader\_card rc ON rc.code = r.codecard

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:



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

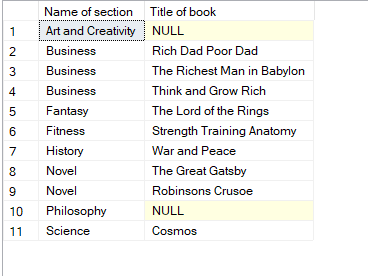
GO

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:



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

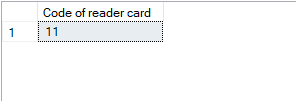
JOIN reader r ON rc.code = r.codecard

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:



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'

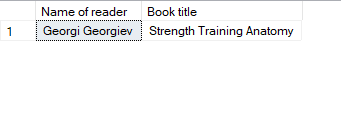
FROM reader r

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:



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 reder', 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:



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 'е'.

Заявка 1:

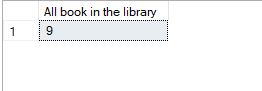
USE Library

GO

SELECT SUM(s.count\_availabal\_book + s.count\_get\_book) AS 'All book in the library'

FROM section s;

Резултат:



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

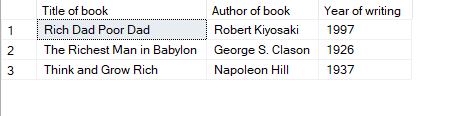
JOIN

(SELECT name\_section FROM book\_section

GROUP BY name\_section

HAVING COUNT(name\_section)>1) bs1 ON bs1.name\_section = bs.name\_section;

Резултат:



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

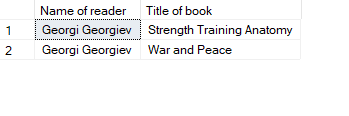
JOIN(

SELECT name\_reader FROM readerbook\_book

GROUP BY name\_reader

HAVING COUNT(name\_reader) > 1) rb1 ON rb1.name\_reader = rb.name\_reader;

Резултат:



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

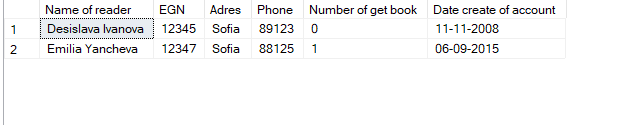
FROM reader\_readercard

GROUP BY adres

HAVING COUNT(1) > 1) rr1 ON rr.adres = rr1.adres

WHERE RIGHT(rr.phone,1) %2 <> 0;

Резултат:



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 '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

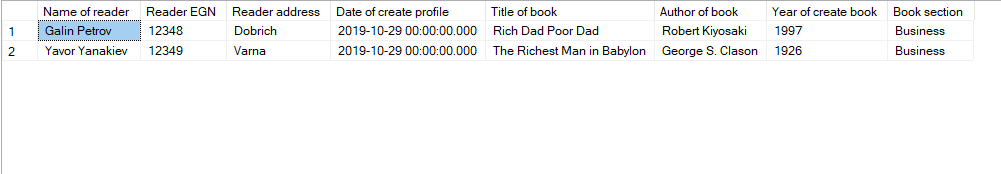
(SELECT name\_section

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:

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

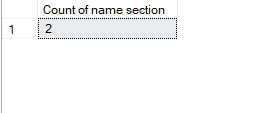
SELECT COUNT(s.name) AS 'Count of name section'

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:



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 acount', 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(

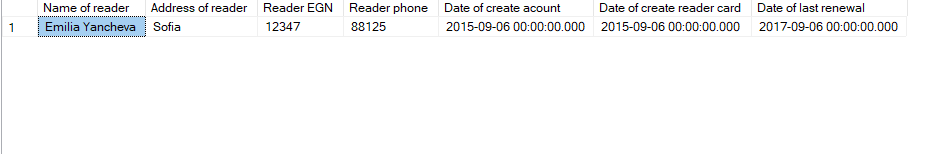
SELECT adres

FROM reader

GROUP BY adres

HAVING COUNT(1) > 1

) r1 ON r1.adres = r.adres;

Резултат:

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

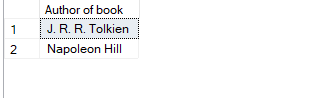
FROM book

GROUP BY year\_create

HAVING COUNT(1)>1) b1 ON b.year\_create = b1.year\_create

WHERE b.available = 'N';

Резултат:



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'

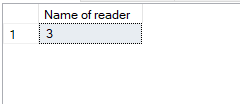
FROM reader r

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:



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

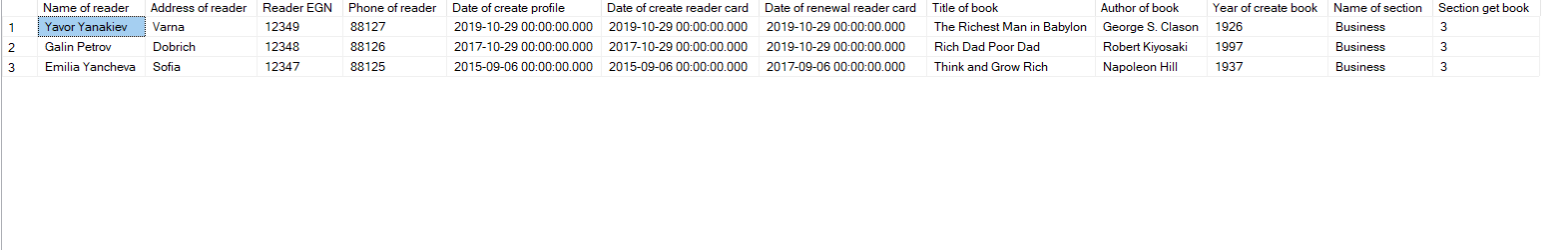
GROUP BY name\_section

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:

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.