Vizsgaremek

# Hallgató neve:Tóth Zoltán László

# Hallgató e-mail címe:zltoth@libero.it

# A választott téma rövid elnevezése:Egyetemi könyvtár adatbázisa

# A téma szöveges leírása

*Itt a téma főbb funkciót kell leírni röviden (max. 1 oldal).*

Az *UniLibrary* fantázianévvel ellátott rendszer egy egyetemi könyvtár működését modellezi.

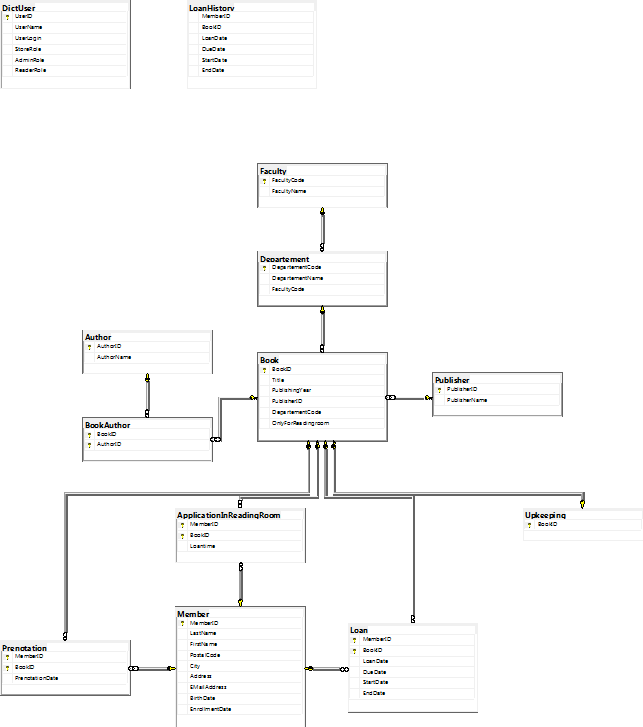
Az *UniLibrary* adatbázis az alábbi adatokat tárolja:

* Könyvek (Book tábla)
* Egyetemi karok (Faculty tábla), Tanszékek (Departement tábla), ezek tulajdonképpen mint könyvkategóriák funkcionálnak
* Olvasók vagy tagok (a két szót rokonértelműként használom hol így, hol úgy, ez nem túl zavaró, ezért az egységesítést idő hiányában későbbre halasztottam)(Member tábla)
* Szerzők (author tábla), kapcsolat a könyvvel (BookAuthor tábla), Kiadók (Publisher tábla)
* Kölcsönzések (Loan és LoanHistory táblák).
* Felhasználók és jogosultságok(beszerző és raktáros, könyvkiadó adminisztrátor, olvasó) (DictUser tábla)
* Előjegyzések (Prenotation tábla)
* Olvasótermi könyvhasználat (ApplicationInReadingRoom tábla)
* Könyvek a Kötészeten (Upkeeping tábla)
* Lehet könyveket felvenni az adatbázisba, kiselejtezni, keresni az adatbázisban és kikölcsönözni, vagy előjegyezni. Kölcsönzés max. Két hétre, kétszer egymás után lehet hosszabbítani, ha valaki előjegyezte, akkor már nem lehet hosszabbítani.

Az *UniLibrary* adatbázis az alábbi funkciók ellátását biztosítja:

* Az adatbázisban tárolt adatok integritásának biztosítása különféle megszorítások és idegen kulcsok révén.
* Lehetővé teszi a kölcsönzések és előjegyzések ellenőrzött nyilvántartását, a kölcsönzési és előjegyzési szabályok betartását. Ez tárolt eljárásokkal történik
* Lehetővé teszi, hogy a tagok megtalálják a nekik megfelelő könyvet. Ezt egy view biztosítja
* A raktáros látja a könyvkészletet és új könyveket szerezhet be és vehet fel a nyilvántartásba egy update-elhető view segítségével.
* A rendszr nyilvántartja, melyek a kölcsönözhető, melyek a csak olvasóteremben használható könyvek, melyek vannak karbantartás alatt. Ez a kölcsönzések felvételénél is figyelembe van véve.
* Az aznapi olvasótermi használatok és a lejárt előjegyzések automatikusan, jobok segítségével törlődnek.

# Adatbázis diagramm



# Sémák

dbo

# Táblák (Az elsőleges kulcsok, idegen kulcsok, nemklaszteres indexek pontos nevét mem imdenhol tüntettem fel, mert ezek nem túl olvasmányosak és nem adnak többlet információt. Az szerepel a leírásban, hogy mely táblák, mely mezőit érintik)

## Tábla neve és Mezőleírás

TABLE [dbo].[ApplicationInReadingRoom](

[MemberID] [int] NOT NULL,

[BookID] [int] NOT NULL,

[Loantime] [time](7) NOT NULL Olvasótermi könyvhasználat kezdete

### Indexek

PRIMARY KEY CLUSTERED ( [MemberID] ASC,[BookID] ASC) )

### Táblakapcsolatok

FOREIGN KEY([BookID]) REFERENCES [dbo].[Book] ([BookID])

FOREIGN KEY([MemberID]) REFERENCES [dbo].[Member] ([MemberID])

## Tábla neve és Mezőleírás

TABLE [dbo].[Author](

[AuthorID] [smallint] IDENTITY(1,1) NOT NULL,

[AuthorName] [varchar](30) NOT NULL

### Indexek

PRIMARY KEY CLUSTERED ([AuthorID] ASC)

NONCLUSTERED INDEX [IX\_Author\_Name] ON [dbo].[Author] ([AuthorName] ASC)

### Tábla neve és Mezőleírás

TABLE [dbo].[Book](

[BookID] [int] IDENTITY(1,1) NOT NULL,

[Title] [varchar](30) NOT NULL,

[PublishingYear] [smallint] NOT NULL,

[PublisherID] [tinyint] NOT NULL,

[DepartementCode] [varchar](6) NOT NULL,

[OnlyForReadingroom] [bit] NOT NULL,

### Indexek

PRIMARY KEY CLUSTERED ([BookID] ASC)

NONCLUSTERED INDEX [IX\_Book\_Title] ON [dbo].[Book] ([Title] ASC)

### Táblakapcsolatok

CONSTRAINT [FK\_\_Book\_\_Departemen\_\_412EB0B6] FOREIGN KEY([DepartementCode])

REFERENCES [dbo].[Departement] ([DepartementCode])

FOREIGN KEY([PublisherID]) REFERENCES [dbo].[Publisher] ([PublisherID])

### Tábla neve és Mezőleírás

TABLE [dbo].[BookAuthor](

[BookID] [int] NOT NULL,

[AuthorID] [smallint] NOT NULL,

### Indexek

PRIMARY KEY CLUSTERED

([BookID] ASC,[AuthorID] ASC)

### Táblakapcsolatok

ALTER TABLE [dbo].[BookAuthor] WITH CHECK ADD FOREIGN KEY([AuthorID])

REFERENCES [dbo].[Author] ([AuthorID])

ALTER TABLE [dbo].[BookAuthor] WITH CHECK ADD FOREIGN KEY([BookID])

REFERENCES [dbo].[Book] ([BookID])

## Tábla neve és mezőleírás

TABLE [dbo].[Departement](

[DepartementCode] [varchar](6) NOT NULL,

[DepartementName] [varchar](30) NOT NULL,

[FacultyCode] [char](3) NOT NULL, Melyik karhoz tartozik a tanszék

### Indexek

CONSTRAINT [PK\_\_Departem\_\_A95E16DC0F973C5C] PRIMARY KEY CLUSTERED ([DepartementCode] ASC)

### Táblakapcsolatok

CONSTRAINT [FK\_Departement\_Faculty] FOREIGN KEY([FacultyCode])

REFERENCES [dbo].[Faculty] ([FacultyCode])

## Tábla neve és Mezőleírás

TABLE [dbo].[DictUser](

[UserID] [int] NOT NULL,

[UserName] [varchar](50) NOT NULL,

[UserLogin] [varchar](100) NOT NULL,

[StoreRole] [bit] NOT NULL, Melyik adatbázis-szerepkörnek tagja a user

[AdminRole] [bit] NOT NULL,

[ReaderRole] [bit] NOT NULL,

### Indexek

PRIMARY KEY CLUSTERED ([UserID] ASC)

## Tábla neve és Mezőleírás

TABLE [dbo].[Faculty](

[FacultyCode] [char](3) NOT NULL,

[FacultyName] [varchar](15) NOT NULL,

### Indexek

PRIMARY KEY CLUSTERED ([FacultyCode] ASC)

## Tábla neve és Mezőleírás

TABLE [dbo].[Loan](

[MemberID] [int] NOT NULL,

[BookID] [int] NOT NULL,

[LoanDate] [date] NOT NULL,

[DueDate] [date] NOT NULL,

[StartDate] [datetime2](0) GENERATED ALWAYS AS ROW START NOT NULL,

[EndDate] [datetime2](0) GENERATED ALWAYS AS ROW END NOT NULL

### Táblaszintű megszorítások

CONSTRAINT [DF\_LoanSysStartDate] DEFAULT (sysutcdatetime()) FOR [StartDate]

CONSTRAINT [DF\_ProductSysEndDate] DEFAULT (CONVERT([datetime2](0),'99991231 23:59:59')) FOR [EndDate]

CONSTRAINT [CK\_Loan] CHECK (([DueDate]>[LoanDate]))

### Indexek

PRIMARY KEY CLUSTERED ([MemberID] ASC,[BookID] ASC)

WITH SYSTEM\_VERSIONING = ON ( HISTORY\_TABLE = [dbo].[LoanHistory] )

### Táblakapcsolatok

ALTER TABLE [dbo].[Loan] WITH CHECK ADD FOREIGN KEY([BookID]) REFERENCES [dbo].[Book] ([BookID])

ALTER TABLE [dbo].[Loan] WITH CHECK ADD CONSTRAINT [FK\_\_Loan\_\_MemberID\_\_4222D4EF] FOREIGN KEY([MemberID])

REFERENCES [dbo].[Member] ([MemberID])

## Tábla neve és Mezőleírás

TABLE [dbo].[Prenotation](

[MemberID] [int] NOT NULL,

[BookID] [int] NOT NULL,

[PrenotationDate] [date] NOT NULL,

### Indexek

PRIMARY KEY CLUSTERED ([MemberID] ASC,[BookID] ASC

### Táblakapcsolatok

FOREIGN KEY([BookID]) REFERENCES [dbo].[Book] ([BookID])

CONSTRAINT [FK\_\_Prenotati\_\_Membe\_\_440B1D61] FOREIGN KEY([MemberID])

REFERENCES [dbo].[Member] ([MemberID])

Tábla neve és Mezőleírás

TABLE [dbo].[LoanHistory](

[MemberID] [int] NOT NULL,

[BookID] [int] NOT NULL,

[LoanDate] [date] NOT NULL,

[DueDate] [date] NOT NULL,

[StartDate] [datetime2](0) NOT NULL,

[EndDate] [datetime2](0) NOT NULL

### Indexek

CLUSTERED INDEX [ix\_LoanHistory] ON [dbo].[LoanHistory] ([EndDate] ASC,[StartDate] ASC

## Tábla neve és Mezőleírás)

TABLE [dbo].[Member](

[MemberID] [int] IDENTITY(1,1) NOT NULL,

[LastName] [varchar](40) NOT NULL,

[FirstName] [varchar](40) NOT NULL,

[PostalCode] [varchar](10) NOT NULL,

[City] [varchar](40) NOT NULL,

[Address] [varchar](100) NOT NULL,

[EMailAddress] [nvarchar](50) NULL,

[BirthDate] [date] NOT NULL,

[EnrollmentDate] [date] NOT NULL,

### Táblaszintű megszorítások

CONSTRAINT [CK\_Member\_BirthDate] CHECK (([BirthDate]<=dateadd(year,(-18),getdate()))) 18 évnél idősebb

CONSTRAINT [CK\_Member\_EnrollmentDate] CHECK (([EnrollmentDate]<=getdate())) Nem a jövőben iratkozott be

### Indexek

CONSTRAINT [PK\_\_Member\_\_0CF04B380F30F4E1] PRIMARY KEY CLUSTERED ([MemberID] ASC)

ONCLUSTERED INDEX [IX\_Member\_EMail] ON [dbo].[Member] ([EMailAddress] ASC)

## Tábla neve és Mezőleírás

TABLE [dbo].[Publisher](

[PublisherID] [tinyint] IDENTITY(1,1) NOT NULL,

[PublisherName] [varchar](30) NOT NULL,

### Indexek

PRIMARY KEY CLUSTERED ([PublisherID] ASC)

## Tábla neve és Mezőleírás

TABLE [dbo].[Upkeeping](

[BookID] [int] NOT NULL,

### Indexek

PRIMARY KEY CLUSTERED ([BookID] ASC)

### Táblakapcsolatok

FOREIGN KEY([BookID]) REFERENCES [dbo].[Book] ([BookID])

Nézetek

## dbo.vReader

### Nézet leírása

A nézet a könyvek adatait tartalmazza, amik alapján az olvasó beazonosíthatja őket.

A nézetet az olvasó használhatja, select joga van rá, megkeresheti benne a kikölcsönözni kívánt könyvet. A ézetre az Adminisztrátornak is van rálátása, hogy segítséget nyújthasson a tagnak, ha esetleg elakad

## dbo.vStatistics

### Nézet leírása

A nézet egy összesített statisztika, hogy Karonként, tanszékenként hány könyv van, és abból hány van kölcsönben. A raktáros és az adminisztráor nézegetheti, meg persze a főnökség, a dbo.

## dbo.vBookStock

### Nézet leírása

A nézet a raktári könyvek adatait tartalmazza. Lehet bele írni is egy tárolt eljáráson és a trgBookStock insteadof triggeren keresztül. A raktáros beírja az új könyv szerzőjét, címét kiadás évét, kiadóját, olvasótermi voltát, kart, tanszéket, és az adatok bekerülek a megfelelő táblákba.

## dbo.vPrenotatedBooksAvailable

### Nézet leírása

A nézet azokat az előjegyzett könyveket tartalmazza, amiket visszahoztak, és azt, hogy ki jegyezte elő. Ez alapján az adminisztrátor, akinek select joga van, értesítheti az olvasót.

# Függvények

## dbo.LoanDueDate

### A függvény funkcionalitásának leírása

Egy megadott dátumhoz egy megadott számú napot ad hozzá, így a kölcsönzések rögzítésénél csak a kölcsönzés dátumát kell megadni, a visszahozás dátumát már ez a függvény számolja ki a tárolt eljáráson belül.

### Függvény paraméterei és visszaadott értéke

CREATE FUNCTION [dbo].[LoanDueDate] (@Date date,@Day tinyint) RETURNS date AS

BEGIN

DECLARE @D date

SET @D = DATEADD(day, @Day, @Date)

RETURN @D

END

# Tárolt eljárások

## dbo.InsertView

### A tárolt eljárás funkcionalitásának leírása

## A tárolt eljárás a dbo.vBookStock-ba való insertálást oldja meg. Bekéri a szükséges adatokat paraméterként, és egy INSERT VALUES-t alkalmaz a viewra. Insert helyett persze a trigger intézkedik. A raktárosnak van rá execute joga

### Tárolt eljárás paraméterei és visszaadott értéke

PROCEDURE [dbo].[Insertview]

@AuthorName varchar(30)=null,

@Title varchar(30)=null,

@PublisherName varchar(30)=null,

@PublishingYear smallint =null,

@DepartementName varchar(30)=null,

@FacultyName varchar(15)=null,

@OnlyForReadingroom bit=null

IF Ha bármelyik parameter null maradt, tehát nem adták meg akkor

return 1

dbo.InsertPrenotation

### A tárolt eljárás funkcionalitásának leírása

### Az előjegyzéseket lehet felvenni INSERT VALUES-szal a Prenotation táblába, ha paraméterként megadjuk a könyv, member azonosítóját, és a dátumot

PROCEDURE [dbo].[InsertPrenotation]

@MemberID int =null,

@BookID int =null,

@PrenotationDate date =null

AS

IF

Ha bármelyik parameter null maradt, tehát nem adták meg akkor

return 1

Ha nincs is ilyen könyv a raktárban.

return 2

Ha nincs ilyen tag

return 3

Ha nem a mai dátummal akarjuk felvenni a rekordot

return 4

Ha olvasótermi a könyv

return 5

Ha az olvasó még nem hozott vissza egy lejárt kölcsönzést

return 6

Ha a saját maga által kölcsönzött könyvre akar előjegyzést

return 7

## dbo.InsertDeleteUpdateLoan

### A tárolt eljárás funkcionalitásának leírása

A kölcsönzéseket insertálhatjuk a Loan táblába megadván a könyv, member azonosítóját, és a dátumot. A visszahatáridő automatikusan kerül kiszámításra. Kölcsönzésnél ez 3 hét, hosszabbításnál (update) plusz egy hét. Ha viszahozták a könyvet, akkor töröljük a rekordot.

PROCEDURE [dbo].[InsertDeleteUpdateLoan]

@Action tinyint=1,

@MemberID int =null,

@BookID int =null,

@LoanDate date =null

AS

IF @Action =1 kölcsönzés

BEGIN

IF

Ha bármelyik parameter null maradt, tehát nem adták meg akkor

return 1

Ha már kölcsönben van a könyv

return 2

Ha már 3 könyvet kölcsönöz egyszerre a tag

return 3

Ha karbantartáson van a könyv

return 4

Ha olvasótermi a könyv

return 5

Ha az olvasó még nem hozott vissza egy lejárt kölcsönzést

return 6

Ha valaki más előjegyezte a könyvet

return 7

Ha nem a mai dátummal akarjuk felvenni a rekordot

return 8

Ha nincs is ilyen könyv a raktárban.

return 9

Ha nincs ilyen tag

return 10

ELSE IF

Ha már kétszer egymás után kivette a könyvet (tehát lejárt a kölcsönzése, visszahozta, és még aznap újra kikölcsönözte) vagy már egyszer hosszabbította

return 11

@Action =2 visszahozták a könyvet, törlés

Ha @MemberID vagy @BookID null

return 1

@Action =3 Hosszabbítás (update)

BEGIN

IF

Ha @MemberID vagy @BookID null

return 1

Ha az olvasó még nem hozott vissza egy lejárt kölcsönzést

return 6

Ha már kétszer egymás után kivette a könyvet (tehát lejárt a kölcsönzése, visszahozta, és még aznap újra kikölcsönözte) vagy már egyszer hosszabbította

return 11

## dbo.DataReading

### A tárolt eljárás funkcionalitásának leírása

Javasolt volt, hogy legyen egy tárolt eljárás, ami külső fájlból olvassa be az adatokat. Gondoltam, akkor legyen a BULK INSERT, és paraméterben megadom a fájl elérési útvonalát, meg a mező- és sorszeparátorokat, de nem... A BULK INSERT csak konstanst fogad el, nem tudom paraméterezni. Idő hiányában ez a sp félbemaradt.

PROCEDURE [dbo].[DataReading]

@FilePath varchar(50)=null,

@FT varchar(5)=null,

@RT varchar(5)=null

Ha bármelyik parameter null maradt, tehát nem adták meg akkor

return 1

# Jogosultsági rendszer

A *UniLibrary* adatbázist a tervek szerint az alábbi kliens rendszerek érik el:*.*

* A raktáros, aki tisztában van a könyvállománnyal, és beszerzi, és regisztrálja az új könyveket
* Az olvasó, aki kereshet magának egy könyvet az adatbázisban.
* Az adminisztrátor, aki a tag által kiválasztott könyvet kikölcsönzi, vagy előjegyzi, vagy meghosszabbítja a kölcsönzést, vagy visszaveszi a visszahozott könyvet.
* Az olvasótermi felügyelő, aki az olvasótermi könyvhasználatot regisztrálja
* Az állapotmegőrzési felelős, aki elviszi a Kötészetre a rongyosra olvasott könyveket

## Login objektumok

A tervezett jogosultsági rendszer alapján az alábbi Login objektumokra és szerepkörökre van szükség:

* LAdmin SQL login
* LMember SQL
* LStore SQL login

## User objektumok

Az imént felsorolt Login objektumokhoz azonos nevű User objektumokként az alábbiakat kapcsoljuk hozzá a megjelölt szerepekörökkel:

* LAdmin user LibAdmin adatbázis szerepkörrel
* LMember user LibMember adatbázis szerepkörrel
* LStore user LibStore adatbázis szerepkörrel

## Database role és Application role objektumok

## LibAdmin: execute permission a dbo.InsertDeleteUpdateLoan, dbo.InsertPrenotation tárolt eljárásokhoz, execute és control permission a dbo.LoanDueDate függvényhez, select jog a vPrenotatedBookAvailable, vReader, vStatiszics nézetekre

LibMember: select permission a vReader nézetre

LibStore: execute permission az Insertview tárolt eljárásra, insert, select jog a vBookStock nézetre, select jog a vStatistics nézetre

RRApp: application role, az ApplicationInReadingRoom táblát kezeli insert, select joggal

UpkeepingApp: application role, az Upkeeping táblát kezeli insert,delete, select joggal

Így minden olyan táblának megvan a gazdája, amivel napi szinten foglalkozni kell. (Nyilván a Karokat és Tanszékeket elég emberöltőnként egyszer felvenni a megfelelő táblákba)

# Telepítés

A vizsgaremekben bemutatott tesztrendszer az alábbi alapfeltöltés során állt elő:

* Author tábla: Ide nevek kellenek, ezeket INSERT SELECT utasítással az Adventureworks adatbázis Person.Person táblájából válogattam. Nem szép megoldás, de gyors.
* Book tábla: könyvek címe, kiadó kódja, tanszék kódja, kiadás éve, olvasótermi-e, ezek az adatok kellenek. Egy paraméterezett INSERT VALUES utasítást beraktam egy While ciklusba. A címek jelképesek pl.: Mathematics1, Physics2. A kiadó, és tanszék kódját beállíto ma címnek megfelelő értékre, pl. Mathematics könyvekhez matematika tanszék,és betöltök Mathematics1-től Mathematics 1000-ig 1000 sort.
* Member tábla: sok személyes adat pl. Név, cím ,irányítószám, e.Mail-cím kell ide. Ezeket az AdventureWorks összejoinolt tábláiból INSERT SELECT utasítással töltöttem be.
* BookAthor tábla: Egy paraméterezett INSERT VALUES utasítást beraktam egy While ciklusba, ez egész számokat ír a BookID és AuthorID mezőkbe
* A kisebb táblák, mint a Faculty, Departement, Publisher,DictUser, ApplicationInReadingRoom, Upkeeping feltöltése INSERT VALUES utasításokkal történt, történik. Az első négy adatai már a könyvtár létrejöttekor adottak voltak, csak egyszer kellett beírni azt a néhány sort. Az utolsó két táblába a megfelelő application role-ok insertálhatnak szükség esetén.
* A Prenotation táblát az InsertPrenotation tárolt eljárás tölti fel a könyvtár működése közben. A Loan táblát az InsertDeleteUpdateLoan tárolt eljárás tölti fel a könyvtár működése közben. A Loan tábla egy temporal table, ezért kapcsolódik hozzá egy LoanHistory tábla, ez system versioned, automatikusan kerülnek bele a Loan-beli változások. Ha például törlünk egy kölcsönzést, akkor az belekerül a History táblába a törlés idejének megfelelő EndTime értékkel.

# Mentési stratégia

Az adatbázist Full recovery modell alatt kell kezelni. A Back Up Database – Unilibrary job minden este csinál egy Full Backupot. Ezen kívül van még két job. A DeletePrenotation kitörli azokat a rekordokat a Prenotation táblából, amik olyan könyvre vonatkoznak, amit már 3 napja visszahoztak. Ennél tovább nem tartjuk fenn az előjegyzőnek a könyvet.A ReadingRoom minden reggel truncatelii az ApplicationInReadingRoom táblát. Ha tegnap este minden könyv visszakerült a polcra, akkor nem érdekes, ki használta (ha mégsem, akkor az illetőt már keresi a rendőrség).