# Βάσεις Δεδομένων
## Εξαμηνιαία Εργασία

**Ομάδα: 47**
**Συνεργάτες:** Βεκράκης Εμμανουήλ - 03116068
Βουλγαρίδης Ιωάννης - 03116151
Γιαννιός Γεώργιος-Ταξιάρχης - 03116156

---

## Έντυπη αναφορά

### Σχόλια πλατφόρμας εκπόνησης της εργασίας

Επιλέξαμε ως ομάδα να εκπονήσουμε την βάση σε MySQL, ενώ τόσο για το SQL script όσο και για τον κώδικα για το User Interface χρησιμοποιήθηκε το περιβάλλον ανάπτυξης NetBeans. Η επιλογή αυτή ήταν προφανής καθώς αποφασίσαμε να υλοποιήσουμε το γραφικό περιβάλλον σε γλώσσα Java, συνεργαζόμενη με το JDBC (Java DataBase Connectivity) API ώστε να κάνουμε τα κατάλληλα queries στην βάση μας, για την οποία το περιβάλλον NetBeans παρέχει αρκετά εργαλεία που μας βοήθησαν στην σχεδίαση του. Αυτό μας βοήθησε μεν στον ευκολότερο σχεδιασμό του γραφικού περιβάλλοντος αλλά προϋπέθετε υπάρχουσες γνώσεις στον προγραμματισμό σε Java και κατανόηση του αντικειμενοστραφούς τρόπου προγραμματισμού, δεξιότητες τις οποίες κατείχαν τα μέλη της ομάδας.

Για την εκτέλεση της εφαρμογής μας σε περιβάλλον Linux εκτελείται η εντολή java -jar Library/dist/Library.jar .

Από τις πρώτες δυσκολίες που αντιμετωπίσαμε ήταν ο συγχρονισμός των διαφορετικών κομματιών της εργασίας που είχε αναλάβει το κάθε μέλος. Για την επίλυση του προβλήματος αυτού, δημιουργήσαμε ένα private repository στο github με collaborators τα μέλη της ομάδας ώστε να μπορεί ο καθένας να βλέπει την πρόοδο των άλλων και να συγχρονίζει την έκδοση του κώδικα του. Το επόμενο πρόβλημα που αντιμετωπίσαμε ήταν ο καθορισμός των constraints κατά τη δημιουργία της βάσης ώστε να εξυπηρετούν την εύρυθμη λειτουργία της βιβλιοθήκης, η επίλυση του οποίου παρουσιάζεται στην επόμενη παράγραφο.

**Σχεδιασμός της Βάσης**

Το σχεσιακό μοντέλο που επιλέξαμε να υλοποιήσουμε είναι αυτό της προτεινόμενης λύσης της Πρώτης Άσκησης που αναρτήθηκε στο mycourses. Συγκεκριμένα το μοντέλο είναι το εξής:

- member(memberID, MFirst, MLast, Street, number, postalCode, Mbirthdate)
- Book(ISBN, title,pubYear,numpages, pubName) pubName FK to publisher
- author(authID, AFirst, ALast, Abirthdate)
- category(categoryName, supercategoryName) supercategoryName FK to category
- copies(ISBN, copyNr, shelf) ISBN FK to Book
- publisher(pubName, estYear, street, number, postalCode)
- employee(empID, EFirst, ELast, salary)
- permanent_employee(empID, HiringDate) empID FK to employee
- temporary_employee(empID, ContractNr) empID FK to employee
- borrows(memberID, ISBN, copyNr, date_of_borrowing, date_of_return) memberID FK to member, ISBN FK to Books, (ISBN, copyNr) FK to copies
- belongs_to(ISBN, categoryName) ISBN FK to Book, categoryName FK to category
- reminder(empID, memberID, ISBN, copyNr, date_of_borrowing, date_of_reminder) empID FK to employee, memberID FK to member, ISBN FK to Book, (memberID, ISBN, copyNr, date_of_borrowing) FK to borrows, (ISBN, copyNr) FK to copies
- written_by(ISBN, authID) ISBN FK to Book, authID FK to author

Οι **περιορισμοί** (constraints) που ορίσαμε στην βάση μας, που είναι και αυτοί που εξασφαλίζουν την αναφορική ακεραιότητα της κατά τις αλλαγές, είναι αρχικά ένα constraint για το primary key του κάθε πίνακα (τα primary keys είναι υπογραμμισμένα στο σχεσιακό μοντέλο) και ένα constraint για τα foreign keys του κάθε πίνακα, τα οποία επίσης φαίνονται στο παραπάνω σχεσιακό μοντέλο. Για τα foreign keys, πλην κάποιων εξαιρέσεων, θεωρήσαμε ότι όταν γίνεται update ή delete το κλειδί αυτό από τον πίνακα στον οποίον αναφέρεται, θέλουμε να ενημερώνεται το αντίστοιχο πεδίο στον πίνακα που περιέχει το foreign key (cascade). Οι εξαιρέσεις που προαναφέρθηκαν είναι:

- Στον εκδοτικό οίκο ενός βιβλίου, όπου αν διαγραφεί ο εκδοτικός οίκος (αν κλείσει πχ) δεν θέλουμε να σβηστούν τα αντίστοιχα βιβλία που αυτός έχει εκδώσει, αλλά βάζουμε NULL στο πεδίο pubName (SET NULL).
- Στον πίνακα του δανεισμού, αν τυχόν πάει να διαγραφεί είτε ένα μέλος που έχει δανειστεί βιβλία, είτε ένα βιβλίο δανεισμένο από ένα μέλος, είτε ένα αντίγραφο δανεισμένο από ένα μέλος, η βάση να μην επιτρέπει τη διαγραφή αυτή (NO ACTION) καθώς τότε θα χανόταν για πάντα η πράξη του δανεισμού και έτσι το μέλος θεωρητικά θα μπορούσε να κρατήσει το βιβλίο για πάντα, σε περίπτωση που αυτό δεν έχει επιστραφεί.

Τα ευρετήρια που δημιουργήσαμε είναι ένα για κάθε πίνακα, το οποίο δείχνει στο primary key του εκάστοτε πίνακα. Επιλέξαμε τα indices αυτά, για να έχουμε γρήγορη πρόσβαση (ανάγνωση, εγγραφή) στους πίνακες με βάση το εκάστοτε primary key.

## Κώδικας δημιουργίας Βάσης

Για τη δημιουργία τόσο της βάσης που περιγράφηκε παραπάνω, των όψεων και των triggers που ζητούνται, καθώς και για τις εισαγωγές δεδομένων χρησιμοποιήσαμε ένα script sql αρχείο. Το αρχείο αυτό, όταν το εκτελούμε διαγράφει την βάση Library αν υπάρχει, την δημιουργεί από την αρχή με όλους του περιορισμούς, κατασκευάζει έναν χρήση libuser με κωδικό libuser (στοιχεία τα οποία εισάγει από μόνη της η εφαρμογή για τη σύνδεση με την βάση) και εισάγουμε κάποια δεδομένα για τον έλεγχο της λειτουργίας της. Το αρχείο αυτό (ονόματι Library.sql) βρίσκεται και στον φάκελο Library/src/my/libraryui που περιέχει όλους τους κώδικες.

Ο κώδικας δημιουργίας της βάσης περιέχεται και στο παράρτημα στο τέλος της αναφοράς.

Τα ερωτήματα που επιλέξαμε να υλοποιήσουμε είναι:

**JOIN-1 Query:** Δείχνει τα βιβλία που έχουν εκδοθεί από τον εκδοτικό οίκο που επιλέγει ο χρήστης.

```
select title from Book as b inner join Publisher as p on b.pubName=p.pubName where b.pubName = NAME;
```

**JOIN-2 Query:** Δείχνει το ράφι και τον αριθμό αντιτύπων του βιβλίου που έχει επιλέξει ο χρήστης.

```
select copyNr,shelf from Copies as c inner join Book as b on c.ISBN=b.ISBN where b.title = TITLE;
```

**AGGREGATE Query:** Δείχνει τα συνολικά έξοδα της βιβλιοθήκης σε μισθούς υπαλλήλων.

```
select SUM(salary) as "Total Salary Costs" from Employee;
```

**GROUP BY Query:** Δείχνει τον αριθμό των βιβλίων που έχει συγγράψει ο κάθε συγγραφέας.

```
select a.AFirst as "First Name", a.ALast as "Last Name", COUNT(*) as "Books Written" from Written_by as w inner join Author as a on a.authID=w.authID group by w.authID;
```

**ORDER BY Query:** Δείχνει το ονοματεπώνυμο των μόνιμων υπαλλήλων από αυτόν που προσλήφθηκε πιο παλιά ως τον πιο “καινούριο”.

```
select EFirst as "First Name", Elast as "Last Name", HiringDate as "Hiring Date" from Employee, Permanent_Employee order by HiringDate;
```

**HAVING Query:** Δείχνει το ονοματεπώνυμο εκείνων των μελών που έχουν δανειστεί ακριβώς πέντε βιβλία.

```
select m.memberID as \"Member ID\", Mfirst as \"First Name\", MLast as \"Last Name\" from Member as m, Borrows as b where m.memberID = b.memberID group by b.memberID having count(b.memberID)=5;
```

**NESTED Query:** Δείχνει τους τίτλους των βιβλίων που εκδόθηκαν από εκδοτικούς οίκους με έτος ίδρυσης μετά από το έτος που επέλεξε ο χρήστης.

```
select title from Book where pubName in (select pubName from Publisher where estYear > YEAR);
```

Η πρώτη όψη που δημιουργήσαμε είναι η BookPosition η οποία περιέχει τον τίτλο των βιβλίων από τον πίνακα Book και το αντίστοιχο ράφι και αριθμό αντιτύπου του πίνακα Copies. Η όψη αυτή είναι ενημερώσιμη καθώς δεν διαθέτει κανένα από τα χαρακτηριστικά που κάνουν μία όψη να είναι μη ενημερώσιμη. Η δεύτερη όψη που δημιουργήσαμε είναι η BorrowedCount η οποία περιέχει το ονοματεπώνυμο όλων των μελών από τον πίνακα Member και το αντίστοιχο πλήθος βιβλίων που αυτά έχουν δανειστεί από τον πίνακα Borrows. Η όψη αυτή είναι μη-ενημερώσιμη, καθώς περιέχει την συνάρτηση COUNT στα πεδία που επιστρέφει. Στην περίπτωση αυτή η MySQL της απαγορεύει να ενημερωθεί.

Τα triggers που έχουμε υλοποιήσει και φαίνονται στον κώδικα sql είναι:

- Όταν πάει να δανειστεί ένα βιβλίο κάποιο μέλος το οποίο δεν δύναται να προβεί σε δανεισμό (είτε επειδή έχει ήδη πέντε δανεισμένα βιβλία είτε λόγω καθυστέρησης στην επιστροφή ενός βιβλίου) να αποτρέπει την βάση από την είσοδο της εν λόγω εγγραφής, ώστε να διατηρηθεί η εύρυθμη λειτουργία της βιβλιοθήκης.
- Όταν εισάγουμε μία εγγραφή βιβλίου-κατηγορίας στον πίνακα Belongs_to και η κατηγορία αυτή δεν υπάρχει στον πίνακα Category, τότε το trigger εισάγει την κατηγορία στον πίνακα αυτόν (χωρίς κάποια υπερκατηγορία), έτσι ώστε να διαφυλαχθεί η αναφορική ακεραιότητα της βάσης και να μην χαθεί η εγγραφή που πάμε να εισάγουμε.

Επίσης δημιουργήσαμε επιπλέον triggers για τον έλεγχο των δεδομένων σε ορισμένους πίνακες. Για τα βιβλία ελέγχουμε πριν από κάθε εγγραφή το ISBN (13 χαρακτήρες αποτελούμενοι από ψηφία και παύλες), το έτος έκδοσης (μετά το έτος ίδρυσης του εκδοτικού οίκου και μέχρι και το τρέχον έτος) και οι σελίδες (θετικός αριθμός) να είναι έγκυρα και επίσης να μην μπορεί να γίνει υπενθύμιση με ημερομηνία υπενθύμισης πριν την ημερομηνία δανεισμού. Επίσης ελέγχουμε ώστε τα πεδία copyNr και shelf του πίνακα Copies να είναι θετικοί αριθμοί.

Κάποιοι από τους παραπάνω ελέγχους τιμών μπορούσαν να έχουν γίνει με την εντολή CHECK κατά τη δημιουργία του πίνακα, όμως διαπιστώσαμε ότι η εντολή αυτή ενσωματώθηκε σε πιο πρόσφατες εκδόσεις τις MySql οπότε υπήρχε πρόβλημα ασυμβατότητας από μηχάνημα σε μηχάνημα.

Μέσω της εφαρμογής ελέγχουμε για κάθε πεδίο ότι δέχεται τον κατάλληλο τύπο εισόδου, καθώς και για καθένα από τα παραπάνω triggers και ενημερώνουμε τον χρήστη με κατάλληλο μήνυμα σε περίπτωση που μια ενημέρωση αποτύχει.

# Παράρτημα – Κώδικας Δημιουργίας της Βάσης

```
/*********************************************************************************
  Library Database
  Script: Library.sql
  Description: Creates the Library database.
  DB Server: MySql
  Update: 19-05-2019
  Author: Vekrakis, Giannios, Voulgaridis
*********************************************************************************/

/*********************************************************************************
  Drop database if it exists
*********************************************************************************/
DROP DATABASE IF EXISTS Library;

/*********************************************************************************
  Create database
*********************************************************************************/
CREATE DATABASE Library;

/*********************************************************************************
  Open database
*********************************************************************************/
USE Library;

/*********************************************************************************

  Create user libuser
*********************************************************************************/
CREATE USER IF NOT EXISTS 'libuser'@'localhost' IDENTIFIED BY 'libuser';
GRANT ALL PRIVILEGES ON Library.* TO 'libuser'@'localhost';
FLUSH PRIVILEGES;

/*********************************************************************************
  Create tables
*********************************************************************************/
CREATE TABLE Member
(
   memberID INT NOT NULL AUTO_INCREMENT,
   MFirst NVARCHAR(40) NOT NULL,
   MLast NVARCHAR(40) NOT NULL,
   Street NVARCHAR(80),
   Snumber NVARCHAR(10),
   PostalCode NVARCHAR(10),
   Mbirthdate DATE,
```

```
    CONSTRAINT PK_memberID PRIMARY KEY (memberID)
);


CREATE TABLE Book
(
    ISBN VARCHAR(15) NOT NULL,
    title NVARCHAR(120) NOT NULL,
    pubYear INT,
    numPages INT,
    pubName NVARCHAR(80),

    CONSTRAINT PK_ISBN PRIMARY KEY (ISBN)
);


CREATE TABLE Author
(
    authID INT NOT NULL AUTO_INCREMENT,
    AFirst NVARCHAR(40) NOT NULL,
    ALast NVARCHAR(40) NOT NULL,
    Abirthdate DATE,

    CONSTRAINT PK_authID PRIMARY KEY (authID)
);


CREATE TABLE Category
(
    categoryName NVARCHAR(80) NOT NULL,
    supercategoryName NVARCHAR(80),

    CONSTRAINT PK_categoryName PRIMARY KEY (categoryName)
);


CREATE TABLE Copies
(
    ISBN VARCHAR(15) NOT NULL,
    copyNr INT,
    shelf INT,

    CONSTRAINT PK_ISBN_copyNr PRIMARY KEY (ISBN, copyNr)
);


CREATE TABLE Publisher
(
    pubName NVARCHAR(80) NOT NULL,
    estYear INT,
    street NVARCHAR(80) NOT NULL,
    snumber NVARCHAR(10),
```

```
    postalCode NVARCHAR(10),

    CONSTRAINT PK_pubName PRIMARY KEY (pubName)
);


CREATE TABLE Employee
(
    empID INT NOT NULL AUTO_INCREMENT,
    EFirst NVARCHAR(40) NOT NULL,
    ELast NVARCHAR(40) NOT NULL,
    salary FLOAT,

    CONSTRAINT PK_empID PRIMARY KEY (empID)
);


CREATE TABLE Permanent_Employee
(
    empID INT NOT NULL,
    HiringDate DATE,

    CONSTRAINT PK_empID PRIMARY KEY (empID)
);


CREATE TABLE Temporary_Employee
(
    empID INT NOT NULL,
    ContractNr INT,

    CONSTRAINT PK_empID PRIMARY KEY (empID)
);


CREATE TABLE Borrows
(
    memberID INT NOT NULL,
    ISBN VARCHAR(15) NOT NULL,
    copyNr INT,
    date_of_borrowing DATE,
    date_of_return DATE,

    CONSTRAINT PK_micd PRIMARY KEY (memberID,ISBN,copyNr,date_of_borrowing)
);

CREATE TABLE Belongs_to
(
    ISBN VARCHAR(15) NOT NULL,
    categoryName NVARCHAR(80) NOT NULL,

    CONSTRAINT PK_ic PRIMARY KEY (ISBN,categoryName)
);
```

```
CREATE TABLE Reminder
(
   empID INT,
   memberID INT NOT NULL,
   ISBN VARCHAR(15) NOT NULL,
   copyNr INT,
   date_of_borrowing DATE,
   date_of_reminder DATE,

CONSTRAINT PK_emicdd PRIMARY KEY
(empID,memberID,ISBN,copyNr,date_of_borrowing,date_of_reminder)
);


CREATE TABLE Written_by
(
   ISBN VARCHAR(15) NOT NULL,
   authID INT NOT NULL,

   CONSTRAINT PK_ia PRIMARY KEY (ISBN,authID)
);


/*****************************************************************************
  Create Unique Indexes
*****************************************************************************/
CREATE INDEX INX_memberID ON Member (memberID);
CREATE INDEX INX_isbn ON Book (ISBN);
CREATE INDEX INX_authID ON Author (authID);
CREATE INDEX INX_ISBN_copyNr ON Copies (ISBN, copyNr);
CREATE INDEX INX_pubName ON Publisher (pubName);
CREATE INDEX INX_empID ON Employee (empID);
CREATE INDEX INX_empID ON Permanent_Employee (empID);
CREATE INDEX INX_empID ON Temporary_Employee (empID);
CREATE INDEX INX_micd ON Borrows (memberID,ISBN,copyNr,date_of_borrowing);
CREATE INDEX INX_emicdd ON Reminder
(empID,memberID,ISBN,copyNr,date_of_borrowing,date_of_reminder);
CREATE INDEX INX_ia ON Written_by (ISBN,authID);
CREATE INDEX INX_ic ON Belongs_to (ISBN,categoryName);


/*****************************************************************************
  Create Foreign Keys
*****************************************************************************/

/* 1. Create Foreign key: FK_BOOK_pubName in Book table to Publisher table */
ALTER TABLE Book ADD CONSTRAINT FK_BOOK_pubName
   FOREIGN KEY (pubName) REFERENCES Publisher (pubName)
   ON DELETE SET NULL
   ON UPDATE CASCADE;
```

```
/* Create Foreign key: FK_CATEGORY_cat in Category table to Category table*/
ALTER TABLE Category ADD CONSTRAINT FK_CATEGORY_cat
   FOREIGN KEY (supercategoryName) REFERENCES Category (categoryName)
   ON DELETE CASCADE
   ON UPDATE CASCADE;


/* Create Foreign key: FK_COPIES_isbn in Copies table to Book table */
ALTER TABLE Copies ADD CONSTRAINT FK_COPIES_isbn
   FOREIGN KEY (ISBN) REFERENCES Book (ISBN)
   ON DELETE CASCADE
   ON UPDATE CASCADE;


/* Create Foreign key: FK_PE_empID in Permanent_employee table to Employee table*/
ALTER TABLE Permanent_Employee ADD CONSTRAINT FK_PE_empID
   FOREIGN KEY (empID) REFERENCES Employee (empID)
   ON DELETE CASCADE
   ON UPDATE CASCADE;


/* Create Foreign key: FK_TE_empID in Temorary_employee table to Employee table*/
ALTER TABLE Temporary_Employee ADD CONSTRAINT FK_TE_empID
   FOREIGN KEY (empID) REFERENCES Employee (empID)
   ON DELETE CASCADE
   ON UPDATE CASCADE;


/* Create Foreign key: FK_BORROWS_memberID in Borrows_to table to Member table */
ALTER TABLE Borrows ADD CONSTRAINT FK_BORROWS_memberID
   FOREIGN KEY (memberID) REFERENCES Member (memberID)
   ON DELETE NO ACTION
   ON UPDATE CASCADE;


/* Create Foreign key: FK_BORROWS_isbn in Borrows table to Book table*/
ALTER TABLE Borrows ADD CONSTRAINT FK_BORROWS_isbn
   FOREIGN KEY (ISBN) REFERENCES Book (ISBN)
   ON DELETE NO ACTION
   ON UPDATE CASCADE;


/* Create Foreign key: FK_BORROWS_isbn in Borrows table to Copies table*/
ALTER TABLE Borrows ADD CONSTRAINT FK_BORROWS_isbn_copyNr
   FOREIGN KEY (ISBN,copyNr) REFERENCES Copies (ISBN,copyNr)
   ON DELETE NO ACTION
   ON UPDATE CASCADE;


/* Create Foreign key: FK_BELONGS_isbn in Belongs_to table to Book table*/
ALTER TABLE Belongs_to ADD CONSTRAINT FK_BELONGS_isbn
   FOREIGN KEY (ISBN) REFERENCES Book (ISBN)
   ON DELETE CASCADE
```

```
    ON UPDATE CASCADE;


/* Create Foreign key: FK_BELONGS_categoryName in Belongs_to table to Category table */
ALTER TABLE Belongs_to ADD CONSTRAINT FK_BELONGS_categoryName
    FOREIGN KEY (categoryName) REFERENCES Category (categoryName)
    ON DELETE CASCADE
    ON UPDATE CASCADE;


/* Create Foreign key: FK_REMINDER_empID in Reminder table to Employee table */
ALTER TABLE Reminder ADD CONSTRAINT FK_REMINDER_empID
    FOREIGN KEY (empID) REFERENCES Employee (empID)
    ON DELETE CASCADE
    ON UPDATE CASCADE;


/* Create Foreign key: FK_REMINDER_memberID in Reminder table to Member table */
ALTER TABLE Reminder ADD CONSTRAINT FK_REMINDER_memberID
    FOREIGN KEY (memberID) REFERENCES Member (memberID)
    ON DELETE CASCADE
    ON UPDATE CASCADE;


/* Create Foreign key: FK_REMINDER_isbn in Reminder table to Book table */
ALTER TABLE Reminder ADD CONSTRAINT FK_REMINDER_isbn
    FOREIGN KEY (ISBN) REFERENCES Book (ISBN)
    ON DELETE CASCADE
    ON UPDATE CASCADE;


/* Create Foreign key: FK_REMINDER_micd in Reminder table to Borrows table */
ALTER TABLE Reminder ADD CONSTRAINT FK_REMINDER_micd
    FOREIGN KEY (memberID,ISBN,copyNr,date_of_borrowing) REFERENCES Borrows
(memberID,ISBN,copyNr,date_of_borrowing)
    ON DELETE CASCADE
    ON UPDATE CASCADE;


/* Create Foreign key: FK_REMINDER_ic in Reminder table to Copies table */
ALTER TABLE Reminder ADD CONSTRAINT FK_REMINDER_ic
    FOREIGN KEY (ISBN,copyNr) REFERENCES Copies (ISBN,copyNr)
    ON DELETE CASCADE
    ON UPDATE CASCADE;


/* Create Foreign key: FK_WRRITENBY_isbn in Written_by table to Book table */
ALTER TABLE Written_by ADD CONSTRAINT FK_WRITTENBY_isbn
    FOREIGN KEY (ISBN) REFERENCES Book (ISBN)
    ON DELETE CASCADE
    ON UPDATE CASCADE;
```

```
/* Create Foreign key: FK_WRRITENBY_authid in Written_by table to Author table */
ALTER TABLE Written_by ADD CONSTRAINT FK_WRITTENBY_authid
   FOREIGN KEY (authID) REFERENCES Author (authID)
   ON DELETE CASCADE
   ON UPDATE CASCADE;


/*****************************************************************************
  Create trigger that permits members to borrow books if they are not eligible
*****************************************************************************/
DELIMITER |
CREATE TRIGGER TR_BORROWS BEFORE INSERT ON Borrows
FOR EACH ROW
BEGIN
   DECLARE total INT;
   DECLARE dayDiff INT;

   SET total := (SELECT COUNT(*) FROM Borrows WHERE NEW.memberID=memberID AND
date_of_return IS NULL);
   SET dayDiff := (SELECT DATEDIFF(DATE(NOW()), date_of_borrowing) FROM Borrows WHERE
NEW.memberID=memberID AND date_of_return IS NULL ORDER BY date_of_borrowing LIMIT 1);

   IF ((total >= 5) OR (dayDiff > 30))
   THEN
      SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = "Error! This user is not eligible to borrow a
book!";
   END IF;
END|

DELIMITER ;


/*****************************************************************************
  Create trigger that inserts category if it doesn't exist
*****************************************************************************/
DELIMITER |
CREATE TRIGGER TR_CATEGORY BEFORE INSERT ON Belongs_to
FOR EACH ROW
BEGIN
   DECLARE exist INT;
   SET exist := (SELECT COUNT(*) FROM Category WHERE categoryName = NEW.categoryName);

   IF (exist = 0)
   THEN
      INSERT INTO Category VALUES (NEW.categoryName, NULL);
   END IF;
END|

DELIMITER ;
```

```
/*****************************************************************************
  Create trigger ON INSERT in Book that checks if ISBN, pubYear, numPages are valid
*****************************************************************************/
DELIMITER |

CREATE TRIGGER TR_ISBN_ON_INSERT BEFORE INSERT ON Book
FOR EACH ROW
BEGIN
   DECLARE curYear INT;
   DECLARE numPages INT;
   DECLARE pubEstYear INT;


   /* Check if ISBN string lenght is 13 characters */
   IF (CHAR_LENGTH(new.ISBN) <> 13)
   THEN
      SIGNAL SQLSTATE "03001" SET MESSAGE_TEXT = "Error in ISBN length.";
   END IF;

   /* Check if ISBN contains only digits and "-" */
   IF NOT (SELECT new.ISBN REGEXP '^[-]|[0-9]$')
   THEN
      SIGNAL SQLSTATE "03002" SET MESSAGE_TEXT = "Error in ISBN Format. The correct format is
###-###-###-#";
   END IF;

   /* Check if year is valid */
   SET curYear := (SELECT YEAR(CURDATE()));
   SET pubEstYear := (SELECT p.estYear FROM Publisher as p WHERE new.pubName = p.pubName);
   IF ((new.pubYear < pubEstYear) OR (new.pubYear > curYear))
   THEN
      SIGNAL SQLSTATE "03003" SET MESSAGE_TEXT = "Error! Year must be greater than publisher's
establishment year and less than 2019.";
   END IF;

   /* Check if pages are valid */
   IF (new.numPages <= 0)
   THEN
      SIGNAL SQLSTATE "03004" SET MESSAGE_TEXT = "Error in pages. Must be a postitive number.";
   END IF;
END|

DELIMITER ;


/*****************************************************************************
  Create trigger ON UPDATE in Book that examine if ISBN, pubYear and numPages are correct
*****************************************************************************/
DELIMITER |

CREATE TRIGGER TR_ISBN_ON_UPDATE BEFORE UPDATE ON Book
FOR EACH ROW
BEGIN
```

```
   DECLARE curYear INT;
   DECLARE numPages INT;
   DECLARE pubEstYear INT;

   /* Check if ISBN string lenght is 13 characters */
   IF (CHAR_LENGTH(new.ISBN) <> 13)
   THEN
      SIGNAL SQLSTATE "03001" SET MESSAGE_TEXT = "Error in ISBN length";
   END IF;

   /* Check if ISBN contains only digits and "-" */
   IF NOT (SELECT new.ISBN REGEXP '^[-]|[0-9]$')
   THEN
      SIGNAL SQLSTATE "03002" SET MESSAGE_TEXT = "Error in ISBN Format. The correct format is
###-###-###-#";
   END IF;

   /* Check if year is valid */
   SET curYear := (SELECT YEAR(CURDATE()));
   SET pubEstYear := (SELECT p.estYear FROM Publisher as p WHERE new.pubName = p.pubName);
   IF ((new.pubYear < pubEstYear) OR (new.pubYear > curYear))
   THEN
      SIGNAL SQLSTATE "03003" SET MESSAGE_TEXT = "Error! Year must be greater than publisher's
establishment year and less than 2019.";
   END IF;

   /* Check if pages are valid */
   IF (new.numPages <= 0)
   THEN
      SIGNAL SQLSTATE "03004" SET MESSAGE_TEXT = "Error! Pages must be a postitive number.";
   END IF;
END|

DELIMITER ;


/*****************************************************************************
  Create trigger that checks copyNr and Shelf in Copies before insert
*****************************************************************************/
DELIMITER |
CREATE TRIGGER TR_COPIES_INSERT BEFORE INSERT ON Copies
FOR EACH ROW
BEGIN
   IF (new.copyNr <= 0 OR new.shelf <= 0)
   THEN
      SIGNAL SQLSTATE "03005" SET MESSAGE_TEXT = "Error! Copy Number and Shelf must be postitive
numbers.";
   END IF;
END|

DELIMITER ;
```

```
/*******************************************************************************
  Create trigger that checks copyNr and Shelf in Copies before update
*******************************************************************************/
DELIMITER |
CREATE TRIGGER TR_COPIES_UPDATE BEFORE UPDATE ON Copies
FOR EACH ROW
BEGIN
   IF (new.copyNr <= 0 OR new.shelf <= 0)
   THEN
      SIGNAL SQLSTATE "03005" SET MESSAGE_TEXT = "Error! Copy Number and Shelf must be postitive
numbers.";
   END IF;
END|

DELIMITER ;

/*******************************************************************************
  Create trigger that checks if date of reminder is later than date of borrowing before insert
*******************************************************************************/
DELIMITER |
CREATE TRIGGER TR_REMINDER_DATE_INSERT BEFORE INSERT ON Reminder
FOR EACH ROW
BEGIN
   IF (new.date_of_borrowing > new.date_of_reminder)
   THEN
      SIGNAL SQLSTATE "03006" SET MESSAGE_TEXT = "Error! Reminder date must be later than
borrowing date.";
   END IF;
END|

DELIMITER ;


/*******************************************************************************
  Create trigger that checks if date of reminder is later than date of borrowing before update
*******************************************************************************/
DELIMITER |
CREATE TRIGGER TR_REMINDER_DATE_UPDATE BEFORE UPDATE ON Reminder
FOR EACH ROW
BEGIN
   IF (new.date_of_borrowing > new.date_of_reminder)
   THEN
      SIGNAL SQLSTATE "03006" SET MESSAGE_TEXT = "Error! Reminder date must be later than
borrowing date.";
   END IF;
END|

DELIMITER ;


/*******************************************************************************
  Insert data into Publisher table
*******************************************************************************/
```

```
INSERT INTO Publisher VALUES("ΓΚΙΟΥΡΔΑΣ",1932,"Βαλτετσίου","90",16885);
INSERT INTO Publisher VALUES("ΚΕΔΡΟΣ",1984,"Βάρναλη Κώστα","171",16872);
INSERT INTO Publisher VALUES("ΚΛΕΙΔΑΡΙΘΜΟΣ",1977,"Βενιζέλου Ελευθέριου","93",16514);
INSERT INTO Publisher VALUES("ΒΙΒΛΙΟΝΕΤ",1996,"Βεργίνας","148",16611);
INSERT INTO Publisher VALUES("ΜΕΤΑΙΧΜΙΟ",1980,"Βορρά","57",16949);
INSERT INTO Publisher VALUES("ΕΛΕΥΘΕΡΟΥΔΑΚΗΣ",1949,"Γαρδένιας","184",16848);
INSERT INTO Publisher VALUES("ΠΑΠΑΣΩΤΗΡΙΟΥ",1978,"Γενναδίου","172",16509);
INSERT INTO Publisher VALUES("ΠΡΩΤΟΠΟΡΙΑ",1995,"Δελφών","152",16824);

/*****************************************************************************
  Insert data into Book table
*****************************************************************************/
INSERT INTO Book VALUES("960–538–174–1","Adobe Photoshop CS3",2002,622,"ΓΚΙΟΥΡΔΑΣ");
INSERT INTO Book VALUES("960–538–174–2","Audacity 1.3.13",2001,926,"ΚΕΔΡΟΣ");
INSERT INTO Book VALUES("960–538–174–3","AutoCAD 2004",1997,276,"ΚΛΕΙΔΑΡΙΘΜΟΣ");
INSERT INTO Book VALUES("960–538–174–4","Facebook",1997,963,"ΚΛΕΙΔΑΡΙΘΜΟΣ");
INSERT INTO Book VALUES("960–538–174–5","HTML5+JavaScript Δημιουργώντας
παιχνίδια",1999,664,"ΜΕΤΑΙΧΜΙΟ");
INSERT INTO Book VALUES("960–538–175–1","JAVA Getting started",1991,375,"ΚΕΔΡΟΣ");
INSERT INTO Book VALUES("960–538–175–2","Μικροϋπολογιστές",2010,279,"ΠΑΠΑΣΩΤΗΡΙΟΥ");
INSERT INTO Book VALUES("960–538–175–3","C++ Getting started",2001,904,"ΚΕΔΡΟΣ");
INSERT INTO Book VALUES("960–538–175–4","LabVIEW",2015,240,"ΓΚΙΟΥΡΔΑΣ");
INSERT INTO Book VALUES("960–538–175–5","ASSEMBLY ARM-MIPS",2013,248,"ΠΡΩΤΟΠΟΡΙΑ");
INSERT INTO Book VALUES("960–538–175–6","LaTeX για αρχάριους",2014,326,"ΜΕΤΑΙΧΜΙΟ");
INSERT INTO Book VALUES("960–538–175–7","UNIX - οκτώ μαθήματα",2005,891,"ΚΛΕΙΔΑΡΙΘΜΟΣ");
INSERT INTO Book VALUES("960–538–175–8","Αλγόριθμοι και πολυπλοκότητα",2011,168,"ΓΚΙΟΥΡΔΑΣ");


/*****************************************************************************
  Insert data into Author table
*****************************************************************************/
INSERT INTO Author VALUES(1,"Κώστας","Σιδηρόπουλος","1962-05-10");
INSERT INTO Author VALUES(2,"Ειρήνη","Μακρή","1955-11-23");
INSERT INTO Author VALUES(3,"Ελένη","Δρόσου","1968-11-02");
INSERT INTO Author VALUES(4,"Μιχάλης","Καππής","1986-02-08");
INSERT INTO Author VALUES(5,"Αναστάσης","Χατζής","1955-03-29");
INSERT INTO Author VALUES(6,"Δήμητρα","Τζιώρη","1965-03-29");


/*****************************************************************************
  Insert data into Employee table
*****************************************************************************/
INSERT INTO Employee VALUES (1,"Θεοδωσία","Καλλέργη",621);
INSERT INTO Employee VALUES (2,"Μαρία","Καλπούζου",690);
INSERT INTO Employee VALUES (3,"Άγγελος","Κοντός",750);
INSERT INTO Employee VALUES (4,"Γεώργιος","Κιατίπης",750);
INSERT INTO Employee VALUES (5,"Κωνσταντίνος","Κυριακός",1100);
INSERT INTO Employee VALUES (6,"Κυριάκος","Αντωνίου",680);
INSERT INTO Employee VALUES (7,"Ευτυχία","Τσίτου",912);
```

```
/*****************************************************************************
  Insert data into Member table
*****************************************************************************/
INSERT INTO Member VALUES (1,"Μαρία","Αλεξίου","Δοϊράνης","12","16562","1973-02-22");
INSERT INTO Member VALUES (2,"Ελένη","Δήμου","Τρικάλων","11","15771","1999-07-23");
INSERT INTO Member VALUES (3,"Δήμητρα","Βλάχου","Ρωμυλίας","128","13554","2002-09-12");
INSERT INTO Member VALUES (4,"Γεώργιος","Βίτσας","Γεννηματά","209","16561","2000-06-05");
INSERT INTO Member VALUES (5,"Νίκος","Κλεφτάρας","Σμύρνης","44","18892","1998-10-12");
INSERT INTO Member VALUES (6,"Αναστάσης","Μαντούδης","Ελ. Ανθρώπου","57","15489","1999-04-27");
INSERT INTO Member VALUES (7,"Κατερίνα","Κουτσιλέου","Αττικής","68","14325","1996-08-05");
INSERT INTO Member VALUES (8,"Νεφέλη","Δρογγίτη","Βυζαντίου","9","16567","1997-09-08");
INSERT INTO Member VALUES (9,"Μαρία","Βαλαβάνη","Βυζαντίου","45","16567","1993-07-27");
INSERT INTO Member VALUES (10,"Παναγιώτης","Δημάκας","Κρήτης","43","15987","1988-01-01");


/*****************************************************************************
  Insert data into Permanent_Employee table
*****************************************************************************/
INSERT INTO Permanent_Employee VALUES (1,"2005-10-23");
INSERT INTO Permanent_Employee VALUES (3,"2000-12-10");
INSERT INTO Permanent_Employee VALUES (5,"1999-01-23");
INSERT INTO Permanent_Employee VALUES (7,"2010-06-28");


/*****************************************************************************
  Insert data into Temporaty_Employee table
*****************************************************************************/
INSERT INTO Temporary_Employee VALUES (2,1209);
INSERT INTO Temporary_Employee VALUES (4,1210);
INSERT INTO Temporary_Employee VALUES (6,1211);


/*****************************************************************************
  Insert data into Written_by table
*****************************************************************************/
INSERT INTO Written_by VALUES ("960–538–175–7",1);
INSERT INTO Written_by VALUES ("960–538–175–7",2);
INSERT INTO Written_by VALUES ("960–538–175–7",3);
INSERT INTO Written_by VALUES ("960–538–175–8",1);
INSERT INTO Written_by VALUES ("960–538–174–3",6);
INSERT INTO Written_by VALUES ("960–538–174–3",1);
INSERT INTO Written_by VALUES ("960–538–174–5",1);
INSERT INTO Written_by VALUES ("960–538–174–5",3);
INSERT INTO Written_by VALUES ("960–538–175–2",2);
INSERT INTO Written_by VALUES ("960–538–175–2",5);
INSERT INTO Written_by VALUES ("960–538–175–2",4);
INSERT INTO Written_by VALUES ("960–538–175–6",2);
INSERT INTO Written_by VALUES ("960–538–175–1",3);
INSERT INTO Written_by VALUES ("960–538–174–2",3);
INSERT INTO Written_by VALUES ("960–538–175–3",5);
INSERT INTO Written_by VALUES ("960–538–175–4",5);
INSERT INTO Written_by VALUES ("960–538–175–5",6);
INSERT INTO Written_by VALUES ("960–538–174–1",6);
```

```
INSERT INTO Written_by VALUES ("960–538–174–4",6);


/*****************************************************************************
  Insert data into Category table
*****************************************************************************/
INSERT INTO Category VALUES ("ΕΠΙΣΤΗΜΟΝΙΚΑ",NULL);
INSERT INTO Category VALUES ("ΠΛΗΡΟΦΟΡΙΚΗ","ΕΠΙΣΤΗΜΟΝΙΚΑ");
INSERT INTO Category VALUES ("ΠΑΝΕΠΙΣΤΗΜΙΑΚΑ",NULL);
INSERT INTO Category VALUES ("ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ","ΠΑΝΕΠΙΣΤΗΜΙΑΚΑ");
INSERT INTO Category VALUES ("ΠΟΛΥΤΕΧΝΙΚΑ",NULL);
INSERT INTO Category VALUES ("ΛΕΙΤΟΥΡΓΙΚΑ","ΠΟΛΥΤΕΧΝΙΚΑ");
INSERT INTO Category VALUES ("ΕΦΑΡΜΟΓΕΣ","ΕΠΙΣΤΗΜΟΝΙΚΑ");
INSERT INTO Category VALUES ("ΔΙΑΔΙΚΤΥΟ","ΠΑΝΕΠΙΣΤΗΜΙΑΚΑ");


/*****************************************************************************
  Insert data into Belongs_to table
*****************************************************************************/
INSERT INTO Belongs_to VALUES("960–538–174–1","ΕΦΑΡΜΟΓΕΣ");
INSERT INTO Belongs_to VALUES("960–538–174–2","ΕΦΑΡΜΟΓΕΣ");
INSERT INTO Belongs_to VALUES("960–538–174–3","ΕΦΑΡΜΟΓΕΣ");
INSERT INTO Belongs_to VALUES("960–538–174–4","ΔΙΑΔΙΚΤΥΟ");
INSERT INTO Belongs_to VALUES("960–538–174–5","ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ");
INSERT INTO Belongs_to VALUES("960–538–175–1","ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ");
INSERT INTO Belongs_to VALUES("960–538–175–2","ΠΟΛΥΤΕΧΝΙΚΑ");
INSERT INTO Belongs_to VALUES("960–538–175–3","ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ");
INSERT INTO Belongs_to VALUES("960–538–175–4","ΠΑΝΕΠΙΣΤΗΜΙΑΚΑ");
INSERT INTO Belongs_to VALUES("960–538–175–5","ΠΟΛΥΤΕΧΝΙΚΑ");
INSERT INTO Belongs_to VALUES("960–538–175–6","ΕΦΑΡΜΟΓΕΣ");
INSERT INTO Belongs_to VALUES("960–538–175–7","ΛΕΙΤΟΥΡΓΙΚΑ");
INSERT INTO Belongs_to VALUES("960–538–175–8", "ΠΟΛΥΤΕΧΝΙΚΑ");


/*****************************************************************************
  Insert data into Copies table
*****************************************************************************/
INSERT INTO Copies VALUES ("960–538–175–7",1,1);
INSERT INTO Copies VALUES ("960–538–175–7",2,1);
INSERT INTO Copies VALUES ("960–538–175–7",3,1);
INSERT INTO Copies VALUES ("960–538–175–8",1,2);
INSERT INTO Copies VALUES ("960–538–174–3",1,4);
INSERT INTO Copies VALUES ("960–538–174–3",2,4);
INSERT INTO Copies VALUES ("960–538–174–5",1,4);
INSERT INTO Copies VALUES ("960–538–175–2",1,3);
INSERT INTO Copies VALUES ("960–538–175–2",2,3);
INSERT INTO Copies VALUES ("960–538–175–2",3,3);
INSERT INTO Copies VALUES ("960–538–175–6",1,1);
INSERT INTO Copies VALUES ("960–538–175–1",1,1);
INSERT INTO Copies VALUES ("960–538–174–2",1,4);
INSERT INTO Copies VALUES ("960–538–175–3",1,4);
INSERT INTO Copies VALUES ("960–538–175–4",1,4);
INSERT INTO Copies VALUES ("960–538–175–5",1,3);
```

```
INSERT INTO Copies VALUES ("960–538–175–5",2,3);
INSERT INTO Copies VALUES ("960–538–175–5",3,3);
INSERT INTO Copies VALUES ("960–538–174–1",1,2);
INSERT INTO Copies VALUES ("960–538–174–4",1,2);
INSERT INTO Copies VALUES ("960–538–174–4",2,2);
INSERT INTO Copies VALUES ("960–538–174–4",3,2);
INSERT INTO Copies VALUES ("960–538–174–4",4,2);


/*****************************************************************************
  Insert data into Borrows table
*****************************************************************************/
INSERT INTO Borrows VALUES (1,"960–538–175–7",1,"2019-05-05",NULL);
INSERT INTO Borrows VALUES (2,"960–538–174–1", 1,"2019-05-06",NULL);
INSERT INTO Borrows VALUES (1,"960–538–175–1",1,"2019-05-01",NULL);
INSERT INTO Borrows VALUES (3,"960–538–175–2",1,"2019-05-10",NULL);
INSERT INTO Borrows VALUES (4,"960–538–174–3", 1,"2019-05-12",NULL);
INSERT INTO Borrows VALUES (5,"960–538–175–4",1,"2019-05-13",NULL);
INSERT INTO Borrows VALUES (6,"960–538–175–7",2,"2019-05-14",NULL);
INSERT INTO Borrows VALUES (7,"960–538–174–3", 2,"2019-05-15",NULL);
INSERT INTO Borrows VALUES (7,"960–538–174–4", 1,"2019-05-15",NULL);
INSERT INTO Borrows VALUES (1,"960–538–175–2",3,"2019-05-20",NULL);
INSERT INTO Borrows VALUES (1,"960–538–175–5",3,"2019-05-20",NULL);
INSERT INTO Borrows VALUES (1,"960–538–175–7",3,"2019-05-20",NULL);


/*****************************************************************************
  Insert data into Reminder table
*****************************************************************************/
INSERT INTO Reminder VALUES (1,2,"960–538–174–1",1,"2019-05-06","2019-05-15");
INSERT INTO Reminder VALUES (4,3,"960–538–175–2",1,"2019-05-10","2019-05-15");


/*****************************************************************************
  Create view that shows how many books each member has borrowed
  (This view is non-updateable as it contains the COUNT function)
*****************************************************************************/
CREATE VIEW BorrowedCount AS
(SELECT MFirst AS "FirstName", MLast AS "LastName", COUNT(*) AS "BooksBorrowed" FROM Borrows
AS b, Member AS m
 WHERE m.memberID=b.memberID GROUP BY m.memberID ORDER BY COUNT(*) DESC);

/*****************************************************************************
  Create view that shows book titles, their shelf and copy number
  (This view is updateable as it doesn't contain any element that denies it)
*****************************************************************************/
CREATE VIEW BookPosition AS SELECT title, shelf, copyNr FROM Book AS b, Copies AS c WHERE
b.ISBN=c.ISBN;
```