ΕΞΑΜΗΝΙΑΙΑ ΕΡΓΑΣΙΑ
ΣΤΙΣ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ
www.cslab.ece.ntua.gr

Ομάδα 2

Μέλη:
Ανδρέας Γεωργής 03115194
Παναγιώτης Παπαντωνάκης 03115012
Δημήτρης Στυλιαράς 03115027

Περιγραφή Εργασίας

Η παρούσα εξαμηνιαία εργασία αφορά στην ανάπτυξη της βάσης δεδομένων της βιβλιοθήκης του Πολυτεχνείου καθώς και της αντίστοιχης εφαρμογής που θα την υποστηρίζει και θα είναι φιλική προς το χρήστη. Με στόχο ο χρήστης να μπορεί να δανειστεί κάποιο βιβλίο από τη βιβλιοθήκη, καθώς και τη δυνατότητα να δει τα βιβλία που έχει η βιβλιοθήκη και να διαλέξει το βιβλίο της αρεσκείας του.

Αρχιτεκτονική Εφαρμογής

Για την ανάπτυξη της βάσης χρησιμοποιήσαμε τη γλώσσα MySql σε συνδυασμό με το MySQL Workbench ενώ για του γραφικού περιβάλλοντος της εφαρμογής χρησιμοποιήσαμε ReactJS, CSS και HTML. Η επικοινωνία μεταξύ του front-end και της βάσης έγινε με τα πακέτα express και mysql του NodeJS. Η επιλογή των παραπάνω εργαλείων έγινε για τους εξής λόγους:

- Προτιμήσαμε την MySql έναντι των άλλων επιλογών, γιατί, παρόλο που έχει λίγο πιο περίπλοκο συντακτικό, είναι ένα ολοκληρωμένο RDBMS, με engine που υποστηρίζει foreign key constraints (που ήταν πολύ σημαντικά στην εφαρμογή μας), με μια μεγάλη κοινότητα που το υποστηρίζει και περιλαμβάνει πλήθος συναρτήσεων (όπως η regex_like για έλεγχο με κανονικές εκφράσεις) που μας φάνηκαν χρήσιμες. Επίσης, παρέχει το MySql Workbench που είναι ένα γραφικό περιβάλλον για την ανάπτυξη της βάσης και την εκτέλεση ερωτημάτων, το οποίο μας βοήθησε πολύ καθόλη τη διάρκεια του project.
- Η Javascript επιλέχθηκε καθώς είναι μια εύκολη scripting γλώσσα με την οποία μπορεί να παραχθεί, ενιαία, κώδικας τόσο front-end (χρησιμοποιώντας το ReactJS) όσο και back-end (με το NodeJS) και η οποία χρησιμοποιείται σε μεγάλο βαθμό από εταιρείες σε όλο τον κόσμο. Επιπλέον, διαθέτει πάρα πολλά πακέτα και βιβλιοθήκες που διεκπεραιώνουν πολύ βασικές διαδικασίες (σύνδεση με τη βάση, αποστολή ερωτημάτων κ.ά.) μέσα σε λίγες γραμμές κώδικα, σε αντίθεση με άλλες γλώσσες, όπως η Java. Σαν scripting interpreted γλώσσα, προφανώς αρκετές διαδικασίες, όπως ο καθορισμός τύπων, γίνονται δυναμικά, υστερώντας έτσι σε ταχύτητα σε σχέση με άλλες γλώσσες.
- Η CSS χρησιμοποιήθηκε για οργάνωση και βελτίωση της εμφάνισης των στοιχείων της html.

Σε ένα top-level επίπεδο ανάλυσης της αρχιτεκτονικής, η εφαρμογή δουλεύει με τον εξής τρόπο. Ο χρήστης, που λόγω περιορισμού των προϋποθέσεων της εργασίας, έχει ταυτόχρονα δικαιώματα διαχειριστή και απλού χρήστη, περιηγείται στο γραφικό περιβάλλον. Σε panels τα οποία απαιτούν επικοινωνία με τη βάση (εκτέλεση ερωτημάτων, εισαγωγών, διαγραφών ή αλλαγών) αποστέλλεται ασύγχρονο HttpRequest σε συγκεκριμένο domain που λειτουργεί ο back-end server. Εκεί, το

αίτημα αναλύεται και ανάλογα με τον τύπο του γίνεται και το αντίστοιχο ασύγχρονο ερώτημα στη βάση. Όταν η βάση απαντήσει, τα δεδομένα ακολουθούν αντίστροφη πορεία, καταλήγοντας ως απάντηση στο front-end, για να τα δει ο χρήστης. Σε περίπτωση σφάλματος, ο back-end server κάνει τη σωστή διαχείριση ώστε ο χρήστης να δει ένα επεξηγηματικό μήνυμα.

Λειτουργίες Εφαρμογής

Αναλυτικά η εφαρμογή μας, πληρώντας τις προϋποθέσεις που περιγράφονται στην εκφώνηση, παρέχει τις εξής λειτουργίες:

- Ο χρήστης μπορεί να δει τους πίνακες book, author και member της βάσης και την ενημερώσιμη όψη. Για καθένα από αυτούς τους πίνακες τού δίνεται η δυνατότητα **αναζήτησης** με βάση τα πεδία τους, περιορίζοντας τα αποτελέσματα, και δυνατότητες **εισαγωγής**, **διαγραφής** ή και **αλλαγής** κάποιας εγγραφής. Η αναζήτηση εκτελεί το ανάλογο select ερώτημα στη βάση, στο σώμα του οποίου μπορεί να γίνεται συνένωση (left join), ομαδοποίηση (group by) και ομαδοποίηση με περιορισμό (group by - having).
- Δίνεται η δυνατότητα ταξινόμησης (order by) είτε σε φθίνουσα είτε σε αύξουσα σειρά για κάθε πεδίο εκτός των αποτελεσμάτων των aggregate functions (π.χ. ο αριθμός των βιβλίων που έχει γράψει ένας συγγραφέας).
- Οι χρήστες μπορούν να αυξομοιώσουν τον αριθμό των αντιγράφων των βιβλίων.
- Ο χρήστης μπορεί να επιλέξει και να εκτελέσει επτά queries και να δει τα αποτελέσματα της μη ενημερώσιμης όψης.

- Επίσης, αν ένα μέλος έχει δανειστεί 5 βιβλία και δε τα έχει επιστρέψει, τότε εάν πάει να δανειστεί καινούριο βιβλίο η εφαρμογή θα τον αποτρέψει από τη ενέργεια αυτή.

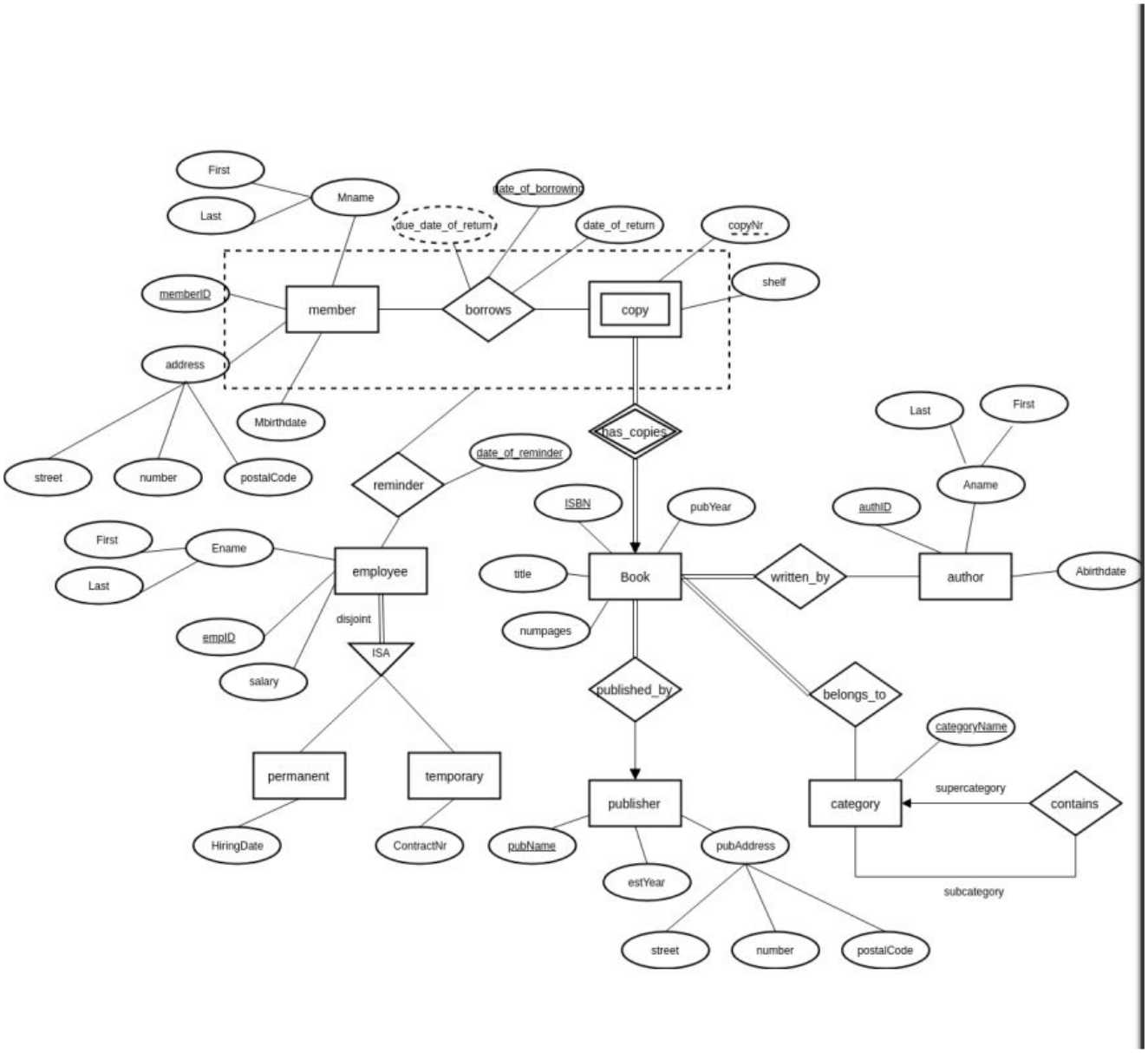
Σχόλια

Κατά την κατασκευή της εφαρμογής, εκτός από τις πρώτες μέρες που εξοικειωνόμασταν με τα διάφορα εργαλεία και στις οποίες είχαμε κάποια προβλήματα (στήσιμο back-end server, αποστολή αιτημάτων από το front-end στο back-end, χρήση σωστού engine βάσης που να υποστηρίζει foreign key constraints), δεν αντιμετωπίσαμε πολλά προβλήματα. Το σημαντικότερο από αυτά ήταν η διαχείριση των σφαλμάτων της βάσης, καθώς αυτά είναι πάρα πολλά και δεν έχουν μια κανονικοποιημένη και φιλική προς το χρήστη μορφή. Η λύση δίνεται μερικώς στο back-end, όπου γίνεται η αντιστοίχιση του κωδικού του σφάλματος με φιλικά, αλλά όχι εντελώς κατατοπιστικά προς το χρήστη μηνύματα. Για παράδειγμα, αν υπάρξει σφάλμα τύπου (ο χρήστης εισήγαγε γράμματα σε αριθμητικό πεδίο), δίνεται το

κατάλληλο μήνυμα, αλλά δεν προσδιορίζει ποιο πεδίο της φόρμας χρειάζεται διόρθωση.

Σχεσιακό Διάγραμμα Βάσης Δεδομένων

Για την ανάπτυξη της βάσης χρησιμοποιήθηκε το σχεσιακό διάγραμμα που δόθηκε ως λύση.



Το Σχεσιακό μοντέλο είναι:

- **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**(ID, 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)SBN 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

Σχεδιασμός Βάσης Δεδομένων - DDL

Οι πίνακες της Βάσης που δημιουργήσαμε καθώς και οι περιορισμοί τους είναι οι κάτωθι. Οι check και on update περιορισμοί εξηγούνται παρακάτω:

Δημιουργία πίνακα Χρήστη:

```
create table if not exists Member(
    ID int  auto_increment,
    mFirst text,
    mLast text,
    street text,
    streetNumber smallint,
    postalCode char(5),
    mBirthdate date,
    primary key (ID),
    check (regexp_like(postalcode, '[0-9]+$') and length(postalcode)
= 5)
    );
```

Δημιουργία πίνακα Εκδότη:

```
create table if not exists publisher(
   pubName varchar(60),
    estYear int,
    street text,
    streetNumber smallint,
    postalCode char(5),
    primary key (pubName),
    check (regexp_like(postalcode, '^[0-9]+$') and
length(postalcode) = 5),
    check (regexp_like(estYear, '^[0-9]+$'))
    );
```

Δημιουργία πίνακα Βιβλίων

```
create table if not exists Book(
    ISBN char(13),
    title text,
    pubYear int,
    numpages int,
    pubName varchar(60) not null,
    primary key (ISBN),
    foreign key(pubName) references publisher(pubName) on delete no
action on
    update cascade,
    check (regexp_like(isbn, '^[0-9]+$') and length(isbn) = 13),
    check (regexp_like(pubYear, '^[0-9]+$')),
    check (regexp_like(numpages, '^[0-9]+$'))
);
```

Ο περιορισμός ακεραιότητας υπαγορεύει να μην είναι δυνατή η διαγραφή ενός εκδότη, όσο υπάρχουν βιβλία του στη βιβλιοθήκη.

Δημιουργία πίνακα Συγγραφέων

```
create table if not exists author(
    ID int auto_increment,
    aFirst varchar(60),
    aLast varchar(60),
    aBirthdate date,
    primary key (ID)
);
```

Δημιουργία πίνακα Κατηγορίας

```
create table if not exists category(
```

```
    categoryName varchar(60),
    supercategoryName varchar(60) default null,
    primary key (categoryName),
    foreign key(supercategoryName) references category(categoryName)
on delete set null on update cascade
    );
```

Κατά τη διαγραφή μιας υπερκατηγορίας τίθεται η τιμή null στο αντίστοιχο πεδίο.

Δημιουργία πίνακα Αντίτυπων

```
create table if not exists copies(
    ISBN char(13) not null,
    copyNr int default 1,
    shelf  varchar(20) default 'A800',
    primary key(ISBN,copyNr),
    foreign key(ISBN) references Book(ISBN) on delete cascade on
update cascade
);
```

Κατά τη διαγραφή του βιβλίου, διαγράφονται και όλα τα αντίτυπά του

Δημιουργία πίνακα Εργαζομένων

```
create table if not exists employee(
     ID int auto_increment,
    eFirst varchar(60),
    eLast varchar(60),
    salary int,
    primary key (ID)
);
```

Δημιουργία πίνακα Μόνιμων Εργαζόμενων

```
create table if not exists permanent_employee(
    empID int auto_increment,
    hiringDate date,
    primary key (empID),
    foreign key(empID) references employee(ID) on delete cascade on
update cascade
);
```

Κατά τη διαγραφή της υπερκλάσης «εργαζόμενος», διαγράφεται και η υποκλάση.

Δημιουργία πίνακα Προσωρινών Εργαζομένων

```
create table if not exists temporary_employee(
    empID int auto_increment,
    contractNr varchar(30),
    primary key (empID),
    foreign key(empID) references employee(ID) on delete cascade on
update cascade
    );
```

Κατά τη διαγραφή της υπερκλάσης «εργαζόμενος», διαγράφεται και η υποκλάση.

Δημιουργία πίνακα Δανεισμού

```
create table if not exists borrows(
    ID int not null,
    ISBN char(13) not null,
    copyNr int not null,
    date_of_borrowing date not null,
    date_of_return date null,
    primary key(ID, ISBN, copyNr, date_of_borrowing),
    foreign key(ID) references Member(ID) on delete cascade on
update cascade,
    foreign key(ISBN) references Book(ISBN) on delete cascade on
update cascade,
    foreign key(ISBN,copyNr) references copies(ISBN,copyNr) on
delete cascade on update cascade,
    check (date_of_return > date_of_borrowing)
);
```

Κατά τη διαγραφή του μέλους, του βιβλίου ή του αντιγράφου του βιβλίου, διαγράφονται οι αντίστοιχες εγγραφές

Δημιουργία πίνακα που δείχνει την κατηγορία στην οποία ανήκει το βιβλίο

```
create table if not exists belongs_to(
    isbn char(13) not null,
    categoryName varchar(60) not null,
    primary key(ISBN,categoryName),
    foreign key(ISBN) references Book(ISBN) on delete cascade on
update cascade,
    foreign key(categoryName) references category(categoryName) on
delete cascade on update cascade
);
```

Κατά τη διαγραφή του βιβλίου, διαγραφή των αντίστοιχων εγγραφών, ενώ απαγορεύεται η διαγραφή της κατηγορίας στην οποία ανήκουν βιβλία της βιβλιοθήκης.

Δημιουργία πίνακα Υπενθύμισης

```
create table if not exists reminder(
      empID int not null,
    memID int not null,
    ISBN char(13) not null,
    copyNr int not null,
    date_of_borrowing date not null,
    date_of_reminder date not null,
    primary
key(empID,memID,ISBN,copyNr,date_of_borrowing,date_of_reminder),
    foreign key(empID) references employee(ID) on delete cascade on
update cascade,
    foreign key(memID) references Member(ID) on delete cascade on
update cascade,
    foreign key(ISBN) references Book(ISBN) on delete cascade on
update cascade,
    foreign key(memID, ISBN, copyNr,date_of_borrowing) references
borrows(
    ID, ISBN, copyNr, date_of_borrowing) on delete cascade on update
cascade
);
```

Κατά τη διαγραφή του υπαλλήλου, του βιβλίου, του μέλους ή της εγγραφής δανεισμού, διαγράφονται οι αντίστοιχες εγγραφές.

Δημιουργία πίνακα Συγγραφέας του

```
create table if not exists written_by(
      ISBN char(13) not null,
    ID int not null,
    primary key(ISBN, ID),
    foreign key(ISBN) references Book(ISBN) on delete cascade on
update cascade,
    foreign key(ID) references author(ID) on delete no action on
update cascade
);
```

Κατά τη διαγραφή του βιβλίου, διαγραφή της εγγραφής, ενώ απαγορεύεται η διαγραφή ενός συγγραφέα που έχει βιβλία στη βάση.

Περιορισμοί και triggers Βάσης

Στην βάση εισήχθησαν περιορισμοί στηλών (check constraint), πεδίου τιμών (domain constraints), αναφορικής ακεραιότητας (foreign key constraints), ακεραιότητας οντότητας (primary key) και κάποιοι άλλοι ορισμένοι από εμάς (6 triggers).
Τα triggers έχουν ως εξής:

1) Όταν εισάγουμε κάποιο βιβλίο σε μια κατηγορία (στον πίνακα belongs_to) και αυτή δεν υπάρχει πριν από την εισαγωγή τότε την εισάγουμε. Το trigger αυτό επιλέχθηκε για να επιτελεί μια πολύ απλή λειτουργία που θα κάνει πιο εύκολη την εισαγωγή κάποιων στοιχείων.

```
create trigger create_category_on_book_insert
before insert on belongs_to
for each row
begin
if not exists (select * from category
     where (categoryName = new.categoryName))
then
insert into category(categoryName)
values (new.categoryName)
on duplicate key update categoryName= new.categoryName;
end if;
end;
```

2) Αν ο χρήστης προσπαθήσει να δανειστεί παραπάνω των 5 βιβλίων τότε η βάση απορρίπτει τη νέα εισαγωγή. Το trigger αυτό υπαγορεύεται από την εκφώνηση.

```
create trigger borrow_5_books
before insert on borrows
for each row
begin
if exists (select distinct borrows.id
           from borrows
           where borrows.date_of_return is null and borrows.id = new.id
           having count(*) = 5)
then
signal sqlstate '45000'
SET MESSAGE_TEXT = 'Cannot borrow more than 5 books!';
end if;
end; $$
```

3) Αν ο χρήστης προσπαθήσει να ενοικιάσει βιβλίο, ενώ έχει στην κατοχή του βιβλίο που έχει λήξει η ημερομηνία δανεισμού του, τότε αποτρέπεται από την ενέργεια αυτή. Το trigger αυτό υπαγορεύεται από την εκφώνηση.

```
create trigger cannot_borrow
before insert on borrows
for each row
begin
if exists (select distinct borrows.id
                      from borrows
           where borrows.id=new.id and borrows.date_of_return is null and
date_add(borrows.date_of_borrowing, interval 30 day) < curdate()
           )
then
signal sqlstate '45000'
SET MESSAGE_TEXT = 'Expired book found!';
end if;
end;
```

4) Πριν την εισαγωγή νέου βιβλίου στη βιβλιοθήκη, εισάγεται και ένα αντίγραφό του. Δεν έχει νόημα να εισαχθεί νέο βιβλίο στη βιβλιοθήκη χωρίς να έχουμε αντίτυπό του.

```
create trigger copy_on_first_insert
after insert on book
for each row
begin
if (select count(*)
    from copies
    where copies.isbn = new.isbn) = 0
then
insert into copies(isbn, copyNr, shelf)
values (new.isbn);
end if;
end;
```

5) Δημιουργία αυτόματης αύξησης του πρωταρχικού κλειδιού του πίνακα των αντιτύπων κατά 1, όταν εισάγουμε κάποιο αντίτυπο. Η λειτουργία αυτή εισήχθη για ευκολία στην εισαγωγή των αντιτύπων.

```
create trigger auto_increment_copies_number
before insert on copies
for each row
begin
if exists (select distinct isbn from copies where copies.isbn = new.isbn)
then
set new.copyNr = (select max(copyNr)
from copies
where copies.isbn = new.isbn) + 1;
end if;
end;
```

6) Απαγόρευση δανεισμού βιβλίου που έχει δανειστεί κάποιος άλλος και δεν το έχει επιστρέψει.

```
create trigger cannot_borrow_borrowed_book
before insert on borrows
for each row
begin
if exists (select * from copies inner join borrows on copies.isbn = borrows.isbn and
copies.copyNr = borrows.copyNr where copies.isbn = new.isbn and copies.copyNr =
new.copyNr and borrows.date_of_return is null)
then
signal sqlstate '45000'
SET MESSAGE_TEXT = 'Cannot borrow borrowed book!';
end if;
end;
```

Για τους υπόλοιπους περιορισμούς έχουμε:

- Τα primary_keys τα ορίζουμε (είτε εμείς είτε αυτόματα η βάση) ως unique και not_null.
- Στις σχέσεις τα foreign_keys τα ορίζουμε ως not_null για να υπάρχει συνέπεια στη βάση και να συμφωνούν με τα primary_keys των σχέσεων στις οποίες ανήκουν. Επίσης, σε όλους τους περιορισμούς αναφορικής ακεραιότητας κατά την αλλαγή στοιχείων, αλλάζουν και όλες οι επηρεαζόμενες εγγραφές.
- Δεν ορίσαμε ως not_null χαρακτηριστικά τα οποία θεωρούνται ως δευτερεύοντα για τον προσδιορισμό των σχέσεων (π.χ. superCategory στον πίνακα category).
- Διάφορα check constraints που εγγυώνται τη σωστή μορφή των τιμών. Συγκεκριμένα, οι ταχυδρομικοί κώδικες πρέπει να έχουν 5 αριθμούς, το ISBN πρέπει να αποτελείται από 13 ψηφία, η ημερομηνία επιστροφής ενός βιβλίου πρέπει να είναι πιο μετά από την ημερομηνία δανεισμού του.
- Τέλος δεν υλοποιήσαμε κάποιο ευρετήριο στη βάση μας και επομένως τα υπάρχοντα ευρετήρια είναι αυτά που χρησιμοποιεί ο DBMS του MySQL Workbench (κυρίως B+ trees indexes ως προς το primary key, Hash indexes).

Αυτό το αποφασίσαμε επειδή δεν θέλαμε να προσθέσουμε πολυπλοκότητα στη βάση μας, τα select ερωτήματα είναι ήδη αρκετά γρήγορα λόγω του μικρού αριθμού των εγγραφών και δεν θέλαμε καθυστερήσεις σε εισαγωγές, διαγραφές, αλλαγές στοιχείων.

Queries

Έχουμε τα query που προαναφέρθηκαν, με τα οποία μπορούμε να αναζητούμε με βάση τα πεδία τους, τους αντίστοιχους πίνακες συγγραφέων, βιβλίων και χρηστών. Πέρα από αυτά, υλοποιήσαμε επιπλέον 7 queries που είναι τα εξής:

1 Query: select count(*) from member;

Απλός υπολογισμός των αριθμών των μελών (aggregate)

2 Query: select copies.isbn, count(*) from copies group by copies.isbn;

Απλός υπολογισμός των αντιτύπων κάθε βιβλίου (group by)

3 Query: select book.isbn, book.title, book.numpages from book order by book.numpages desc;

Εύρεση των μεγαλύτερων βιβλίων με ταξινόμησή τους με βάση φθίνον αριθμό σελίδων (order by)

4 Query: select member.mFirst, member.mLast, count(*) from member inner join borrows on borrows.id = member.id where borrows.date_of_return is null group by member.id having count(*) >= 3;

Εύρεση των ατόμων που χρωστούν παραπάνω από 3 βιβλία και είναι πιθανό να έχουν φτάσει το μέγιστο όριο των 5 βιβλίων, καθώς και του τελικού αριθμού των βιβλίων που χρωστάνε (group by - having)

5 Query: select distinct borrows.isbn from (select borrows.isbn as isbn, m1.ID as ID from member as m1 inner join borrows on m1.ID = borrows.ID) as borrowed_books_by inner join borrows on borrows.isbn = borrowed_books_by.isbn where borrows.ID != borrowed_books_by.ID;

Εύρεση των isbn των βιβλίων που τα έχουν δανειστεί δύο διαφορετικά άτομα, πιθανώς για να βρούμε τα πιο συνηθισμένα και τα πιο εξειδικευμένα βιβλία (subquery)

6 Query: select book.isbn, book.title, belongs_to.categoryName from book left join belongs_to on book.isbn = belongs_to.isbn;

Εύρεση των κατηγοριών των βιβλίων (join)

7 Query: select book.isbn, book.title, author.aFirst, author.aLast from book left join written_by on book.isbn = written_by.isbn inner join author on written_by.ID = author.ID;

Εύρεση των συγγραφέων των βιβλίων (join)

View

Τα views που υλοποιήσαμε είναι τα εξής:

Μη ενημερώσιμη όψη που δείχνει τα άτομα που χρωστάνε βιβλία κι επομένως θα πρέπει να τους σταλεί ειδοποίηση.

```
create view to_remind as
 select member.id as 'ID', member.mFirst as 'First Name', member.mLast as 'Last
 Name', count(*) as 'Number of books'

 from borrows inner join member


  on member.id = borrows.id


  where borrows.date_of_return is null and date_add(borrows.date_of_borrowing,
interval 30 day) < curdate()

  group by member.id;
```

Πλήρως ενημερώσιμη όψη που λειτουργεί σαν πίνακας δείχνει τα ονόματα των εργαζομένων, χωρίς το πεδίο του μισθού. Θα μπορούσε να χρησιμοποιηθεί από τον διαχειριστή για να δει το προσωπικό του και να εισάγει νέα άτομα

```
create view employee_no_salary as

select eFirst as 'First Name' , eLast as 'Last Name'

from employee
```