ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

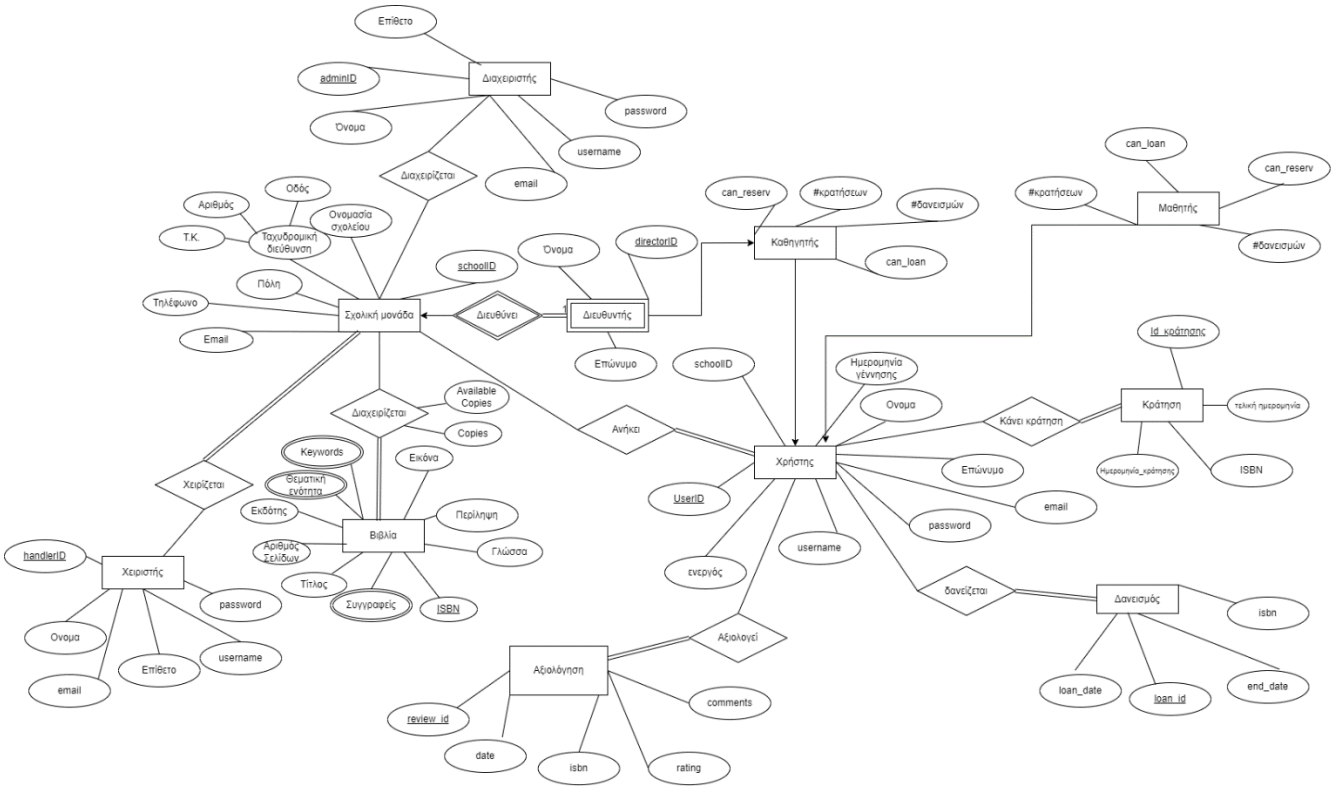
Γεώργιος Χατζηχριστοδούλου

Α.Μ.: 03120125

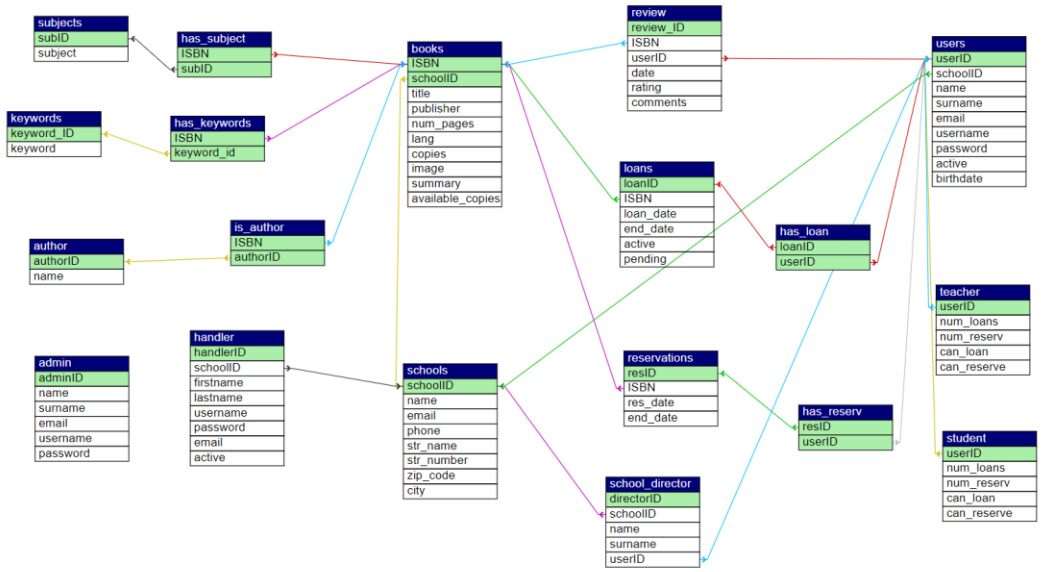
ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

Αναφορά για την Εξαμηνιαία Εργασία

ER Διάγραμμα



Σχεσιακό Διάγραμμα



Με πράσινο χρώμα στο σχεσιακό διάγραμμα φαίνονται τα primary keys.

Κατά τον θεωρητικό σχεδιασμό της βάσης δεδομένων για το σύστημα διαχείρισης σχολικών βιβλιοθηκών θεωρήσαμε τις ακόλουθες οντότητες με τις αντίστοιχες σχέσεις.

Η οντότητα «σχολική μονάδα» που περιλαμβάνει τα στοιχεία των σχολείων, όπως το όνομα, η διεύθυνση και τα στοιχεία επικοινωνίας του. Θεωρήσαμε την οντότητα «βιβλία» και την σχέση «διαχειρίζεται», όπου για κάθε βιβλίο καταχωρούμε τα στοιχεία του, ενώ στη σχέση «διαχειρίζεται» αποθηκεύουμε τα συνολικά καθώς και τα διαθέσιμα κάθε στιγμή αντίτυπα του βιβλίου κάθε σχολείου. Κάθε βιβλίο πρέπει να διαχειρίζεται οπωσδήποτε από ένα τουλάχιστον σχολείο, ενώ μπορεί να έχει πολλές θεματικές κατηγορίες, λέξεις κλειδιά και συγγραφείς. Έτσι στο σχεσιακό διάγραμμα υπάρχουν οι πίνακες “βιβλία”, “θεματική κατηγορία”, “λέξη κλειδί”, “συγγραφέας” και οι «έχει_κατηγορία», «έχει_λέξη_κλειδί», «έχει_συγγράψει» για να διαχειριστούμε τις περισσότερες από μία τιμές αυτών των χαρακτηριστικών. Στους τρεις τελευταίους πίνακες αποθηκεύουμε μόνο το ISBN του βιβλίου και το subID, resID ή autID αντίστοιχα, ενώ στους υπόλοιπους πίνακες τα υπόλοιπα χαρακτηριστικά τους, όπως το όνομα του συγγραφέα, ενώ εκεί γίνεται και η δήλωση των subID, resID και autID ως primary keys. Στη συνέχεια αποθηκεύουμε χειριστές και διαχειριστές. Οι διαχειριστές είναι υπεύθυνοι για τη συνολική λειτουργία του δικτύου και έτσι δεν συνδέονται με κάποιο άλλο πίνακα. Οι χειριστές είναι υπεύθυνοι για ένα σχολείο και έτσι αποθηκεύουμε για αυτούς στον πίνακα το schoolID ως foreign key που αναφέρεται στην οντότητα “σχολική μονάδα”. Επίσης ορίζουμε την οντότητα “Χρήστης” και τις οντότητες “Μαθητής” και “Καθηγητής” που “κληρονομούν” τα χαρακτηριστικά του “Χρήστη”. Για κάθε μαθητή και καθηγητή αποθηκεύουμε τον αριθμό των δανεισμών και κρατήσεων του, ενώ υπάρχουν και τα σχετικά constraints για να μην υπερβαίνουν την τιμή που ορίσαμε. Ο χρήστης αναγνωρίζεται μοναδικά από το userID του, ενώ στο σχεσιακό μοντέλο, στον πίνακα του αποθηκεύουμε σαν foreign key το σχολείο του μέσω του schoolID. Ο διευθυντής θεωρήθηκε ως weak entity του σχολείου, αφού δεν μπορεί να υπάρξει διευθυντής χωρίς σχολείο. Ο διευθυντής αναγνωρίζεται μοναδικά

από το directorID του, ενώ και αυτός είναι ένας καθηγητής. Υπάρχει η οντότητα «Αξιολόγηση», την οποία μπορεί να κάνει κάθε χρήστης. Τέλος υπάρχουν οι οντότητες «Δανεισμός» με primary key το loanID και «Κράτηση» με primary key resID. Κάθε χρήστης μπορεί να έχει γενικά παραπάνω από μία κράτηση και δανεισμό και έτσι δημιουργήθηκαν οι πίνακες «έχει_κράτηση» και «έχει_δανεισμό» στις οποίες αποθηκεύουμε τα resID ή loanID με το userID του χρήστη που του ανήκει. Στους δανεισμούς και τις κρατήσεις καταχωρούμε το βιβλίο και τις αρχικές και τελικές ημερομηνίες για να γίνεται έλεγχος για εκπρόθεσμους δανεισμούς και κρατήσεις. Να επισημάνουμε ότι το πεδίο num_loans των πινάκων teacher και student αναφέρεται μόνο στους ενεργούς δανεισμούς(active = 1). Κάθε δανεισμός, κράτηση και αξιολόγηση πρέπει να ανήκει οπωσδήποτε σε έναν χρήστη και γι' αυτό υπάρχει διπλή γραμμή.

DML και DDL

Για τη δημιουργία της βάσης χρησιμοποιήθηκε MySQL. Το μοντέλο που χρησιμοποιήθηκε είναι αυτό που περιγράφεται από το σχεσιακό διάγραμμα, κάθε πίνακας του σχεσιακού διαγράμματος υπάρχει και στη βάση δεδομένων.

Έχουν οριστεί οι παρακάτω περιορισμοί:

Περιορισμοί foreign key:

TABLE_NAME	COLUMN_NAME	CONSTRAINT_NAME	REFERENCED_TABLE_NAME	REFERENCED_COLUMN_NAME
books	schoolID	books_ibfk_1	schools	schoolID
handler	schoolID	handler_ibfk_1	schools	schoolID
has_keywords	ISBN	has_keywords_ibfk_1	books	ISBN
has_keywords	keyword_id	has_keywords_ibfk_2	keywords	keyword_ID
has_loan	loanID	has_loan_ibfk_1	loans	loanID
has_loan	userID	has_loan_ibfk_2	users	userID
has_reserv	resID	has_reserv_ibfk_1	reservations	resID
has_reserv	userID	has_reserv_ibfk_2	users	userID
has_subject	ISBN	has_subject_ibfk_1	books	ISBN
has_subject	subID	has_subject_ibfk_2	subjects	subID
is_author	authorID	is_author_ibfk_1	author	authorID
is_author	ISBN	is_author_ibfk_2	books	ISBN
loans	ISBN	loans_ibfk_1	books	ISBN
reservations	ISBN	reservations_ibfk_1	books	ISBN
review	ISBN	review_ibfk_1	books	ISBN
review	userID	review_ibfk_2	users	userID
school_director	schoolID	school_director_ibfk_1	schools	schoolID
school_director	userID	school_director_ibfk_2	users	userID
student	userID	student_ibfk_1	users	userID
teacher	userID	teacher_ibfk_1	users	userID
users	schoolID	users_ibfk_1	schools	schoolID

Οι λόγοι που επιλέχθηκαν αυτοί οι περιορισμοί εξηγούνται παραπάνω.

Επιπλέον περιορισμοί:

Θεωρούμε ότι καμία τιμή της βάσης δεν μπορεί να είναι NULL. Επίσης, θεωρούμε ότι πέρα από τα primary keys, οι στήλες που αφορούν username και email είναι UNIQUE όπως συμβαίνει άλλωστε και στην πραγματικότητα. Επίσης θεωρήσαμε ότι ο αριθμός των δανεισμών(num_loans) και των κρατήσεων(num_reserv) για ένα μαθητή πρέπει να μην υπερβαίνει την τιμή 2, ενώ για ένα καθηγητή την τιμή 1.

Ευρετήρια

Έχουμε ορίσει τα ακόλουθα ευρετήρια:

```
CREATE INDEX subs ON subjects (subject);

CREATE INDEX unames ON users (username);

CREATE INDEX surs ON users (surname);

CREATE INDEX auths ON author (name);

CREATE INDEX ldate ON loans (loan_date);
```

Ο βασικός λόγος που ορίζονται ευρετήρια είναι για να γίνεται γρηγορότερα η αναζήτηση και η εύρεση κάποιας τιμής που αναζητείται συχνά. Οι λόγοι που ορίστηκαν τα συγκεκριμένα είναι οι ακόλουθοι:

Subs: Ορίστηκε επειδή γίνεται πολλές φορές αναζήτηση βιβλίου, αξιολογήσεων και άλλων ερωτημάτων βάσει της κατηγορίας που ανήκουν

Unames: Χρησιμοποιούνται μόνο για τους users για να γίνεται πιο εύκολα η εξακρίβωση των στοιχείων των users. Για τις υπόλοιπες οντότητες που έχουν username και password δεν ορίστηκαν ευρετήρια αφού γενικά είναι πολύ λίγοι σε πλήθος(στην υλοποίηση μας υπάρχουν μόνο 2 admins) και είναι περιττό να γίνει ευρετήριο για πολύ λίγες τιμές.

Surs: Ορίστηκε καθώς γίνεται συχνά αναζήτηση χρήστη, δανεισμών του και άλλων χαρακτηριστικών του βάσει του επωνύμου του.

Auths: Ορίστηκε λόγω της αναζήτησης βιβλίων αλλά και άλλων ερωτημάτων με βάση το όνομα του συγγραφέα

Ldate : Ορίστηκε για να γίνεται εύκολα η εύρεση καθυστερημένων δανεισμών ή δανεισμών μιας συγκεκριμένης χρονικής περιόδου.

Άλλες προσθήκες

Πέρα από την εισαγωγή των πινάκων και των περιορισμών στη βάση, έχει οριστεί και ένας trigger ώστε κάθε φορά που γίνεται μία κράτηση, δηλαδή INSERT στον πίνακα “reservations” , να ελέγχεται αν έγινε από μαθητή ή καθηγητή και να αυξάνεται κατά ένα το πεδίο του αριθμού των κρατήσεων στον αντίστοιχο πίνακα, student ή teacher.

```
CREATE TRIGGER `increase_num_reserv_trigger` AFTER INSERT ON `has_reserv`
 FOR EACH ROW BEGIN
    DECLARE user_type VARCHAR(10);
    SET user_type = (
        SELECT CASE
            WHEN EXISTS(SELECT * FROM student WHERE userID = NEW.userID)
THEN 'student'
            WHEN EXISTS(SELECT * FROM teacher WHERE userID = NEW.userID)
THEN 'teacher'
        END
    );
```

```
    IF user_type = 'student' THEN
        UPDATE student SET num_reserv = num_reserv + 1 WHERE userID =
NEW.userID;
    ELSEIF user_type = 'teacher' THEN
        UPDATE teacher SET num_reserv = num_reserv + 1 WHERE userID =
NEW.userID;
    END IF;
END
```

Έχουν οριστεί δύο stored procedures για τους δανεισμούς και τις κρατήσεις. Το ένα ελέγχει αν ο χρήστης έχει κάνει ήδη κράτηση στο βιβλίο που κάνει αίτηση για καινούρια κράτηση(χρησιμεύει μόνο για τους μαθητές, αφού οι καθηγητές έτσι κι αλλιώς δεν μπορούν να έχουν 2 κρατήσεις). Αν υπάρχει τότε εμφανίζεται μήνυμα λάθους και διαγράφεται η κράτηση.

```
CREATE PROCEDURE `insert_has_reserv`(IN p_resID INT, IN p_userID INT)
BEGIN
    DECLARE existing_reservations INT;

    SELECT COUNT(*) INTO existing_reservations
    FROM reservations r
    JOIN has_reserv hs ON r.resID = hs.resID
    WHERE hs.userID = p_userID
      AND r.ISBN = (
        SELECT ISBN FROM reservations WHERE resID = p_resID
      );

    IF existing_reservations > 0 THEN
        DELETE FROM reservations WHERE resID = p_resID;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Duplicate
reservation not allowed for the same book and user. The previous
reservation has been deleted.';
    END IF;
END
```

Το δεύτερο αφορά τους δανεισμούς και συγκεκριμένα ελέγχει αν υπάρχουν καθυστερημένοι δανεισμοί από τον χρήστη, ή ενεργός δανεισμός του ίδιου βιβλίου. Όμοια με πριν, σε αυτή την περίπτωση εμφανίζεται μήνυμα λάθους και διαγράφεται ο δανεισμός με το procedure `delete_loan` που απλά κάνει delete το tuple με το loanID του ορίσματος.

```
CREATE PROCEDURE `insert_has_loan`(IN p_loanID INT, IN p_userID INT)
BEGIN
    DECLARE existing_delayed_loans INT;
    DECLARE existing_same_book_loan INT;

    -- Check for delayed loans
    SELECT COUNT(*) INTO existing_delayed_loans
    FROM loans l
    JOIN has_loan hl ON hl.loanID = l.loanID
    WHERE hl.userID = p_userID AND l.active = 1 AND
DATEDIFF(CURRENT_DATE, l.loan_date) > 7;

    IF existing_delayed_loans > 0 THEN
        CALL delete_loan(p_loanID);
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Delayed loan
detected. The loan has been deleted.';
    ELSE
        -- Check for loan of the same book
```

```
        SELECT COUNT(*) INTO existing_same_book_loan
        FROM loans l
            join has_loan hs on hs.loanID = l.loanID
        WHERE hs.userID = p_userID AND ISBN = (SELECT ISBN FROM loans
WHERE loanID = p_loanID) AND l.active = 1;

        IF existing_same_book_loan > 0 THEN
            CALL delete_loan(p_loanID);
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'You already
have this book. The loan has been deleted.';
        ELSE
            INSERT INTO has_loan (loanID, userID) VALUES (p_loanID,
p_userID);
        END IF;
    END IF;
END //


CREATE PROCEDURE `delete_loan`(IN p_loanID INT)
BEGIN
    DELETE FROM loans WHERE loanID = p_loanID;
END //
```

Τέλος έχει οριστεί ένα event, όπου με περίοδο μίας ημέρας ελέγχει για κρατήσεις με ημερομηνία κράτησης(res_date) που έχουν περάσει τις 7 ημέρες από την σημερινή ημέρα. Αυτές οι κρατήσεις (reservations) διαγράφονται, το αντίστοιχο tuple στον πίνακα has_reserv και μειώνεται κατά ένα η τιμή του num_reserv για τον μαθητή ή τον καθηγητή.

```
CREATE EVENT delete_expired_reservations_event
ON SCHEDULE EVERY 1 DAY
STARTS CURRENT_TIMESTAMP + INTERVAL 1 DAY
DO
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE res_id INT;
    DECLARE user_id INT;
    DECLARE user_type VARCHAR(10);

    DECLARE cur CURSOR FOR
        SELECT r.resID , hr.userID , CASE
            WHEN EXISTS(SELECT * FROM student s WHERE s.userID =
hr.userID) THEN 'student'
            WHEN EXISTS(SELECT * FROM teacher t WHERE t.userID =
hr.userID) THEN 'teacher'
        END AS user_type
        FROM reservations r
        JOIN has_reserv hr ON r.resID = hr.resID
        WHERE DATEDIFF(CURRENT_DATE, r.res_date) > 7;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN cur;

    read_loop: LOOP
        FETCH cur INTO res_id, user_id, user_type;

        IF done THEN
            LEAVE read_loop;
        END IF;
```

```
        DELETE FROM has_reserv WHERE resID = res_id;

        IF user_type = 'student' THEN
            UPDATE student SET num_reserv = num_reserv - 1 WHERE
userID = user_id;
        ELSEIF user_type = 'teacher' THEN
            UPDATE teacher SET num_reserv = num_reserv - 1 WHERE
userID = user_id;
        END IF;

        DELETE FROM reservations WHERE resID = res_id;
    END LOOP;

    CLOSE cur;
END
```

Τα δεδομένα που βρίσκονται στο αρχείο insert.sql έχουν εισαχθεί με τέτοιο τρόπο, ώστε αφενός να τηρούν τα ελάχιστα όρια σχολείων, βιβλίων, χρηστών, δανεισμών και κρατήσεων, και αφετέρου για όλα τα ερωτήματα που ζητούνται να υπάρχουν αποτελέσματα για συγκεκριμένα κριτήρια.

Για παράδειγμα στο ερώτημα 3.1.5 υπάρχουν 22 δανεισμοί για τους χειριστές των σχολείων με schoolID = 2, 3 για το έτος 2021.

Θεωρούμε για λόγους ασφάλειας ότι ένας μαθητής δεν μπορεί να τροποποιήσει κανένα στοιχείο του. Για αλλαγή του password του απευθύνεται στον υπεύθυνο χειριστή της σχολικής του μονάδας. Επίσης, ο εκπαιδευτικός μπορεί να απευθυνθεί στον χειριστή της σχολικής βιβλιοθήκης του σχολείου του για να τροποποιήσει και εκείνος τα στοιχεία του, σε περίπτωση για παράδειγμα όπου έχει ξεχάσει το password του.

Όσον αφορά τους δανεισμούς και τις κρατήσεις, στην εφαρμογή θεωρήσαμε ότι ένας χρήστης μπορεί να κάνει μόνο κράτηση, οπότε όταν θέλει να δανειστεί ένα βιβλίο, η αίτηση καταχωρείται ως κράτηση και ο χειριστής μπορεί να την αποδεχτεί και να την μετατρέψει αυτόματα σε δανεισμό. Οι παλαιές κρατήσεις διαγράφονται από την βάση δεδομένων, ενώ οι παλαιοί δανεισμοί παραμένουν με το attribute: active = 0. Συνεπώς ισχύον δανεισμός έχει active = 1. Οι παλαιοί δανεισμοί παραμένουν στη βάση για στατιστικούς λόγους.

GIT REPOSITORY ΤΗΣ ΕΦΑΡΜΟΓΗΣ

To git repository της εφαρμογής είναι :
https://github.com/gioxatz/library_management_system.git

To ER diagram : er.png

Relational schema : relational.png

DDL : tables.sql

DML : insert.sql

Requirements: requirements.txt

Queries σε python για τα ερωτήματα : queries.py

Αρχείο python με τον κώδικα της εφαρμογής, τη σύνδεση με την εφαρμογή και τα queries : lib1.py

Στον φάκελο docs:

Οδηγίες εγκατάστασης της εφαρμογής : installation_guide.pdf

User Manual : user_manual.pdf

Αναφορά : db_report.pdf

Στον φάκελο templates:

Τα templates html που χρησιμοποιήθηκαν για την εφαρμογή