Βάσεις Δεδομένων

Αναφορά phase A(*)

Εργαστηριακή Ομάδα *52*

Γιουμερτάκης Απόστολος, 2017030142
Κατσούπης Ευάγγελος, 2017030077

3 Ανάκτηση δεδομένων και υπολογισμοί

3.8) **return_thesis_same_lab_38**() : Ολοκληρώθηκε *πλήρως*

Στο πρώτο subquery (a), γίνεται ένα INNER JOIN μεταξύ του Committee και Professor, ώστε να γίνει αντιστοίχιση των διπλωματικών με το εργαστήριο που εργάζεται κάθε μέλος της επιτροπής. Χρησιμοποιώντας το "DISTINCT on (c.student_amka, c.diploma_num, labjoins)" εμφανίζονται πλειάδες διπλωματικών (student_amka, diploma_num) με διαφορετικά εργαστήρια καθηγητών (labjoins). Οπότε αν σε μια διπλωματική όλα τα μέλη της επιτροπής εργάζονται στο ίδιο εργαστήριο, τότε θα εμφανιστεί μονο **μια πλειάδα** που θα περιέχει αυτήν την διπλωματική, αλλίως θα εμφανιστούν περισσότερες.
Στο δεύτερο subquery (b), γίνεται μια ομαδοποίηση των παραπάων πλειάδων με βάση την διπλωματική (student_amka - diploma_num) και μετράμε τον αριθμό των εμφανίσεων των διπλωματικών του 1ου subquery (a). Χρησιμοποιώντας το HAVING, επιβάλουμε τον αριθμό των εμφανίσεων να είναι ίσος του 1 (count(*)=1), το οποίο όπως προαναφέρθηκε, θα επιλέξει διπλωματικές με όλα τα μέλη της επιτροπής που εργάζονται στο ίδιο εργαστήριο.
Τέλος, με INNER JOIN μεταξύ του Diploma και του 2ου subquery (b), πάνω στα κοινά (student_amka, diploma_num), θα γίνει αντιστοίχιση των διπλωματικών με τους τίτλους τους.

3.9) **find_all_main_of_dependent_39**(**depend$_c$ourse**) : Ολοκληρώθηκε *πλήρως*

"depend_tree" sub-query:
Αρχικά βρίσκουμε τα μαθήματα που είναι άμεσα προαπαιτούμενα ή συνιστώμενα, για τον κωδικό του μαθήματος που δίνεται ως όρισμα στην συνάρτηση. Στην συνέχεια, βρίσκουμε **αναδρομικά** για κάθε προαπαιτούμενο ή συνιστώμενο μάθημα του αρχικού μαθήματος, τα προαπαιτούμενα ή συνιστώμενα μαθήματά του, με INNER JOIN του "Course_depends" με το "depend_tree" sub-query, πάνω στο cd.dependent = dt.main.
Τέλος, επειδή θέλουμε και τους τίτλους του κάθε μαθηματος, κάνουμε JOIN του "depend_tree" sub-query με το "Course" (όπου υπάρχει η πληροφορία του τίτλου), πάνω c.course_code = dt.main. Χρησιμοποιώντας το keyword 'Distinct', αφαιρούνται τα διπλότυπα που υπάρχουν για τα άμεσα ή έμμεσα, προαπαιτούμενα ή συνιστώμενα μαθήματα.

6 Views

6.1) **view_committee_undergrad_61**(**amka**, **committee**) : Ολοκληρώθηκε *πλήρως*

Στο subquery (a), κάνουμε INNER JOIN μεταξύ του Diploma και του Committee, πάνω στα κοινά (student_amka, diploma_num), ώστε να πάρουμε τις διπλωματικές φοιτητών που **δεν έχουν ακόμη αποφοιτήσει** (πληροφορία που περιέχεται στο "Diploma"), καθώς και τα μέλη της επιτροπής (πληροφορία που περιέχεται στο "Committee"). Ξανακάνοντας INNER JOIN στο "Person", πάνω στο AMKA των καθηγητών που είναι στην επιτροπή (prof_amka = amka), παίρνουμε και τα στοιχεία των καθηγητών, όνομα και επίθετο, τα οποία ενώνονται σε ένα πεδίο με την συναρτηση concat(..), ως ονοματεπώνυμο. Η συνθήκη στο "WHERE", όπως προαναφέρθηκε, φιλτράρει μόνο τους φοιτητές που **δεν** έχουν αποφοιτήσει ακόμα. Κάνοντας ταξινόμηση κατά φθίνουσα σειρά (ORDER BY DESC) στο attribute "supervisor", παίρνουμε πλειάδες για κάθε διπλωματική, όπου έχουμε πρώτα το ονοματεπώνυμο που αντιστοιχεί στον επιβλέποντα και μετά στα υπόλοιπα μέλη της επιτροπής (TRUE 1st - FALSE 2nd).
Στην συνέχεια, κάνοντας ομαδοποίηση σύμφωνα με την διπλωματική (student_amka, diploma_num), μπορούμε να χρησιμοποιήσουμε την aggregate function "string_agg(column, delimiter)", ώστε να συγκεντρώσουμε τις σειρές με τα ονοματεπώνυμα καθηγητών που είναι μέλη της επιτροπής, σε μία μόνο στήλη, το ένα δίπλα στο άλλο, με πρώτο αυτό αντιστοιχεί στον επιβλέποντα.

Οπότε επιστρέφονται δυο (2) πεδία, το AMKA του φοιτητή και το/α ονοματεπώνυμο/α των μελών της επιτροπής, ως ένα Virtual View.

6.2) **view_possible_graduates_62(entry_year, amount)** : Ολοκληρώθηκε *πλήρως*

"sc_rules" sub-query:
Κάνοντας INNER JOIN του "Student" με το "SchoolRules" πάνω στο έτος εισαγωγής των φοιτητων και το έτος που ισχύουν οι κανονισμοί της σχολής, παίρνουμε τον ελάχιστο αριθμό μαθημάτων και διδακτικών μονάδων που χρειάζεται ένας φοιτητής για να αποφοιτήσει. Θεωρούμε ότι το έτος εισαγωγή του φοιτητή πρέπει να είναι το ίδιο με το έτος των κανονισμών της σχολής.

(a) sub-query:
Κάνοντας INNER JOIN του "Register" με το "Course" πάνω στο κοινό "course_code", παίρνουμε τα μαθήματα που είναι περασμένα και κάνοντας ομαδοποίηση κατά amka, obligatory, παίρνουμε 2 πλείδαες για κάθε AMKA φοιτητή, όπου η πρώτη είναι το σύνολο περασμένων υποχρεωτικών μαθημάτων και διδακτικών τους μονάδων και η δεύτερη, το σύνολο των κατ επιλογή υποχρεωτικών μαθημάτων και διδακτικών τους μονάδων. Η σειρά των πλειάδων καθορίζεται από το ORDER BY DESC (TRUE 1st - FALSE 2nd).

(b) sub-query:
Με INNER JOIN μεταξύ του "sc_rules" subquery και του subquery (a), πάνω στο κοινό πεδίο amka, πλέον αφού γνωρίζουμε τον ελάχιστο αριθμό μαθημάτων και διαδακτικών μονάδων για να μπορέσει ένας φοιτητής να αποφοιτήσει, μπορούμε να τον συγκρίνουμε με το σύνολο μαθηματών που έχει περάσει και των διδακτικών μονάδων που έχει συγκεντρώσει. Η σύγκριση αυτή γίνεται με το "HAVING", αλλά για να γίνει πρέπει να προσθέσουμε το σύνολο των υποχρεωτικών και κατ επιλογή υποχρεωτικών μαθημάτων καθώς και τις διδακτικές τους μονάδες, για κάθε φοιτητή (όπως προέκυψαν από το (a) sub-query). Αυτό γίνεται πολύ εύκολα με το GROUP by clause και την συνάρτηση sum(). Ο λόγος που δεν έγινε προηγουμένως η πρόσθεση να αναφερθεί αργότερα. Χρησιμοποιώντας το "WHERE NOT EXISTS" , βρίσκουμε τους φοιτητές που δεν έχουν ακόμη εκπονήσει διπλωματική εργασία, δηλαδή δεν υπάρχουν στον πίνακα "Diploma".

(c) sub-query:
Έχοντας πλέον πληροφορία για το ποιοί φοιτητές πληρούν τους κανονισμούς της σχολής για αποφοίτηση, βάσει το έτος εισαγωγής-εγγραφής τους, κάνουμε ομαδοποίηση κατά το έτος αυτό και μετράμε τον αριθμό των φοιτητών για αυτό το έτος με την συνάρτηση count(). Καθώς έχουμε κρατήσει πληροφορία για το πόσα υποχρεωτικά μαθήματα έχει περάσει κάθε φοιτητής (μέσω της συνάρτησης string_agg() as tot_agg, που δημιουργεί πεδίο της μορφής [υποχρ. μάθημα, μη υποχρ. μάθημα]), μπορούμε να βάλουμε έναν ακόμα περιορισμό, τα μαθήματα αυτά να ισούνται με τα συνολικά υποχρεωτικά μαθήματα που προσφέρει η σχολή (δηλαδή όσα υπάρχουν στον πίνακα "Course").

Τέλος, με RIGHT JOIN μεταξύ του (c) subquery και του "SchoolRules", πάνω στο κοινό πεδίο year, παίρνουμε το *πλήθος των φοιτητών ανά έτος εγγραφής για τα τελευταία 10 έτη οι οποίοι ικανοποιούν τις προϋποθέσεις αποφοίτησης και δεν έχουν ακόμη εκπονήσει διπλωματική εργασία*. Γίνεται RIGHT JOIN καθώς θέλουμε τα τελευταία 10 χρόνια και για κάποια από αυτά δεν έχουμε υποψήφιους φοιτητές. Για αυτά τα χρόνια, θα εμφανίζεται η τιμή 'NULL' στο πεδίο του πλήθους, οπότε με την συνάρτηση coalesce(), μετατρέπεται σε ΄0΄. Κάνουμε ταξινόμηση κατά αύξουσα σειρά για λόγους αισθητικής.
Οπότε επιστρέφονται δυο (2) πεδία, το έτος εγγραφής και το πλήθος των φοιτητών, ως ένα Virtual View.