ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

2023-2024

(ΠΠΣ-183) - Διαχείριση Δεδομένων για Σχεσιακές και μη Σχεσιακές Βάσεις Δεδομένων

Απαλλακτική εργασία
Καθηγητής: Χρήστος Δουλκερίδης

ΦΙΛΙΠΠΟΣ ΡΙΧΤΕΡ
ΑΜ: ΜΕ2350

Κατανοώντας ότι η εργασία σχεδιάστηκε ως ομαδική δύο ατόμων, επέλεξα να προχωρήσω αυτόνομα λόγω των ιδιαίτερων χρονικών δεσμεύσεων που επιβάλλουν τα στρατιωτικά μου καθήκοντα. Ήθελα να αποφύγω να επιβαρύνω έναν συμφοιτητή με το χρονοδιάγραμμά μου, επιλέγοντας μια πιο ευέλικτη διαχείριση του χρόνου μου. Ελπίζω αυτή η αυτόνομη προσέγγιση να μην αποτελέσει ζήτημα για τον εξεταστή.

TABLE OF CONTENTS

1. Περιγραφή & Παραδοχές Βάσης Δεδομένων

1.1. Περιγραφή

Ο σκοπός της εργασίας είναι η μοντελοποίηση και υλοποίηση μιας βάσης δεδομένων από την άποψη ενός πανεπιστημίου που έχει ως απώτερο σκοπό την καταγραφή πληροφοριών για τους αποφοίτους του, στην προκειμένη περίπτωση του πανεπιστημίου Πειραιά. Η χρήση του θα γίνεται από διαχειριστές και αλλους εξουσιοδοτημένους χρήστες με επίκεντρο την καταγραφη δεδομένων των αποφοίτων, τις ιστορικές εγγραφές τους σε προγράμματα σπουδών από διάφορα πανεπιστήμια, την εργασιακή εμπειρία αυτών και άλλες πληροφορίες που ενδιαφέρουν το επιχειρηματικό πεδίο. Οντότητες που δεν συμμετέχουν άμεσα στον σκοπό καταγραφής δεδομένων για τους φοιτητές δεν έχουν συμπεριληφθεί, πχ διδάσκοντες, αίθουσες διδασκαλίας κλπ.

Η εισαγωγή για δεδομένα των αποφοίτων όπως ιστορικές εγγραφές σε προγράμματα σπουδών, παλαιότερες εργασιακές εμπειρίες και άλλα, μπορεί να γίνεται είτε από τους διαχειριστές απευθείας ή μέσω ανταλλαγής πληροφοριών μεταξύ συστημάτων πανεπιστημίων τα οποία θα μοιράζονται πληροφορίες για τους αποφοίτους τους μέσω ενός API (Στο μέλλον). Ο τρόπος και η αποτελεσματικότητα για την εισαγωγή δεδομένων στην βάση είναι εκτός του σκοπού της εν λόγω εργασίας και χρήζει περαιτέρω ανάλυσης.

1.2. Παραδοχές Βάσης Δεδομένων / Λειτουργίας

Με βάση την δοθείσα περιγραφή, έχουν γίνει οι παρακάτω θεωρήσεις για το σύστημα και την λειτουργία του. Παρατίθενται περαιτέρω αναλυτικές παραδοχές σε άλλο κεφάλαιο.

1.Τα δεδομένα αφορούν απόφοιτους, τωρινούς και μελλοντικούς φοιτητες. Η βάση έχει σχεδιαστεί για να επιτρέπει ιστορικές εγγραφές φοιτητών και την εισαγωγή στοιχείων αποφοίτησης για αποφοίτους αλλά και στοιχεία για τωρινούς η μελλοντικούς μαθητές που είναι εγγεγραμμένοι σε τρέχων ή μελλοντικά προγράμματα σπουδών.

2. Όλοι οι μαθητές (**Students**) έχουν ένα μοναδικό κωδικό που χρησιμεύει για όλες τις συναλλαγές τους με το πανεπιστημιακό σύστημα (**student_id)**. Αναλυτικές παραδοχές για τους μαθητές:

- Ένας φοιτητής μπορεί να εγγραφεί σε πολλά τμήματα προγράμματα σπουδών.
- Για λόγους ακεραιότητας των δεδομένων, για την εγγραφή ενός φοιτητή σε μεταπτυχιακό ή διδακτορικό πρόγραμμα σπουδών θα πρέπει να υπάρχει τουλάχιστον μια εισαγωγή προπτυχιακού τίτλου σπουδών από οποιοδήποτε πανεπιστήμιο. Αυτό έχει υλοποιηθεί μέσω ενός trigger στην βάση δεδομένων.

- Έχουμε την δυνατότητα να καταγράφουμε πληροφορίες για προγράμματα σπουδών του αποφοίτου ακόμα και ύστερα από την αποφοίτηση του από ένα πρόγραμμα από το πανεπιστήμιο Πειραιά.
- Έχουμε την δυνατότητα να καταγράφουμε εργασιακές εμπειρίες αποφοίτων ακόμα και για αυτές που είναι χρονικά πριν από την εγγραφή τους με ένα πρόγραμμα σπουδών στο πανεπιστήμιο Πειραιά.
- Εφόσον έχουμε ως επίκεντρο αποφοίτους του πανεπιστημίου Πειραιώς, τότε θεωρούμε πως μια εγγραφή σε πρόγραμμα σπουδών για κάθε φοιτητή θα έπρεπε να σχετίζεται με το πανεπιστήμιο Πειραιά.
- Η ημερομηνία εγγραφής ενός φοιτητή σε ένα πρόγραμμα σπουδών θα έπρεπε να ελέγχεται να μην είναι επιτρεπτή εφόσον το τμήμα προγράμματος σπουδών στο οποίο εγγράφεται έχει ξεπεραστεί κατά ένα χρονικό διάστημα από την ημερομηνία έναρξης αυτού. Αυτό **ΔΕΝ** έχει υλοποιηθεί καθώς σύστηνε μεγάλη πολυπλοκότητα στην εισαγωγή δεδομένων με python. Αναγνωρίζουμε την ανάγκη για μελλοντική ανάπτυξη τέτοιας λειτουργίας για την ακεραιότητα των δεδομένων στο σύστημα.
- Οι εγγραφές φοιτητών δεν θα έπρεπε να ξεπερνούν τον αριθμό του πεδίου max_capacity στον πίνακα Program Term. Αυτό **ΔΕΝ** έχει υλοποιηθεί καθώς σύστηνε μεγάλη πολυπλοκότητα στην εισαγωγή δεδομένων. Αναγνωρίζουμε την ανάγκη για μελλοντική ανάπτυξη τέτοιας λειτουργίας για την ακεραιότητα των δεδομένων στο σύστημα.

3. Εφόσον καταγράφουμε πληροφορίες για προγράμματα σπουδών από άλλα πανεπιστήμια, έχουμε συμπεριλάβει την οντότητα πανεπιστημίου (University). Υποθέτουμε πως τα πανεπιστήμια αυτά έχουν τα ίδια πεδία πληροφοριών μεταξύ τους και έναν κωδικό που τα αναγνωρίζει ξεχωριστά (**university_Id**). Όλα είναι δημόσια, δωρεάν και βρίσκονται στην Ελλάδα με την ίδια αναγνώριση καθώς αλλιώς θα έπρεπε να συμπεριλάβουμε οντότητες όπως αναγνώριση πτυχίων, μετατροπή βαθμών, πληρωμές κλπ. συστήνοντας μεγάλη πολυπλοκότητα στην βάση δεδομένων.

4. Το κάθε πανεπιστήμιο αποτελείται από πολλά τμήματα (**Faculty**), ενώ το κάθε faculty ανήκει σε ένα πανεπιστήμιο. Στον συγκεκριμένο σχεδιασμό **ΔΕΝ** έχει γίνει η παραδοχή για σχολές στις οποίες μπορεί να ανήκουν τα τμήματα και υποθέτουμε πως τα πανεπιστήμια έχουν απλά τμήματα χωρίς σχολές.

5. Τα προγράμματα σπουδών (**Program**) διοργανώνονται από το τμήμα του πανεπιστημίου (**Faculty**) και όχι απευθείας από το πανεπιστήμιο (**University**).

6. Το κάθε πρόγραμμα σπουδών (**Program**) θα έχει μία ή παραπάνω τάξεις (**Program Term**) που μπορεί να τρέχει παράλληλα (σε περίπτωση πολλών ενδιαφερομένων) ή σε

διαφορετικές περιόδους έναρξης (πχ. τάξη καλοκαιρινού εξαμήνου 2024, χειμερινού 2024 κλπ). Σημειώνεται πως οι φοιτητές εγγράφονται σε ένα τμήμα προγράμματος σπουδών **Program term** και όχι στο πρόγραμμα σπουδών **Program** απευθείας.

7. Συστήνεται η οντότητα μάθημα (**Module**) καθώς σχετίζεται άμεσα με τους φοιτητές. Θεωρούμε πως το κάθε μάθημα (**Module**) μπορεί να σχετίζεται μόνο με ένα τμήμα προγράμματος σπουδών (**Program Term**) καθώς μπορεί τα μαθήματα να είναι παρόμοια μεταξύ διαφορετικών προγραμμάτων σπουδών αλλά ποτέ ίδια. Επίσης, εφόσον τα μαθήματα μπορεί να αλλάξουν ανά cohort (Τμήμα Προγράμματος Σπουδών), έχουν συνδεθεί με το τμήμα (**Program Term**) παρά με το πρόγραμμα σπουδών (**Program**). Επίσης, **ΔΕΝ** ενδιαφέρει η αναλυτική καταγραφή επίδοσης φοιτητών ανά μάθημα στο σύστημα, μόνο η συμμετοχή φοιτητών σε αυτά.

8. Μία εργασιακή εμπειρία (**Work Experience**) σχετίζεται με έναν μαθητή (**Student**) και με μία εταιρία (**Company**).

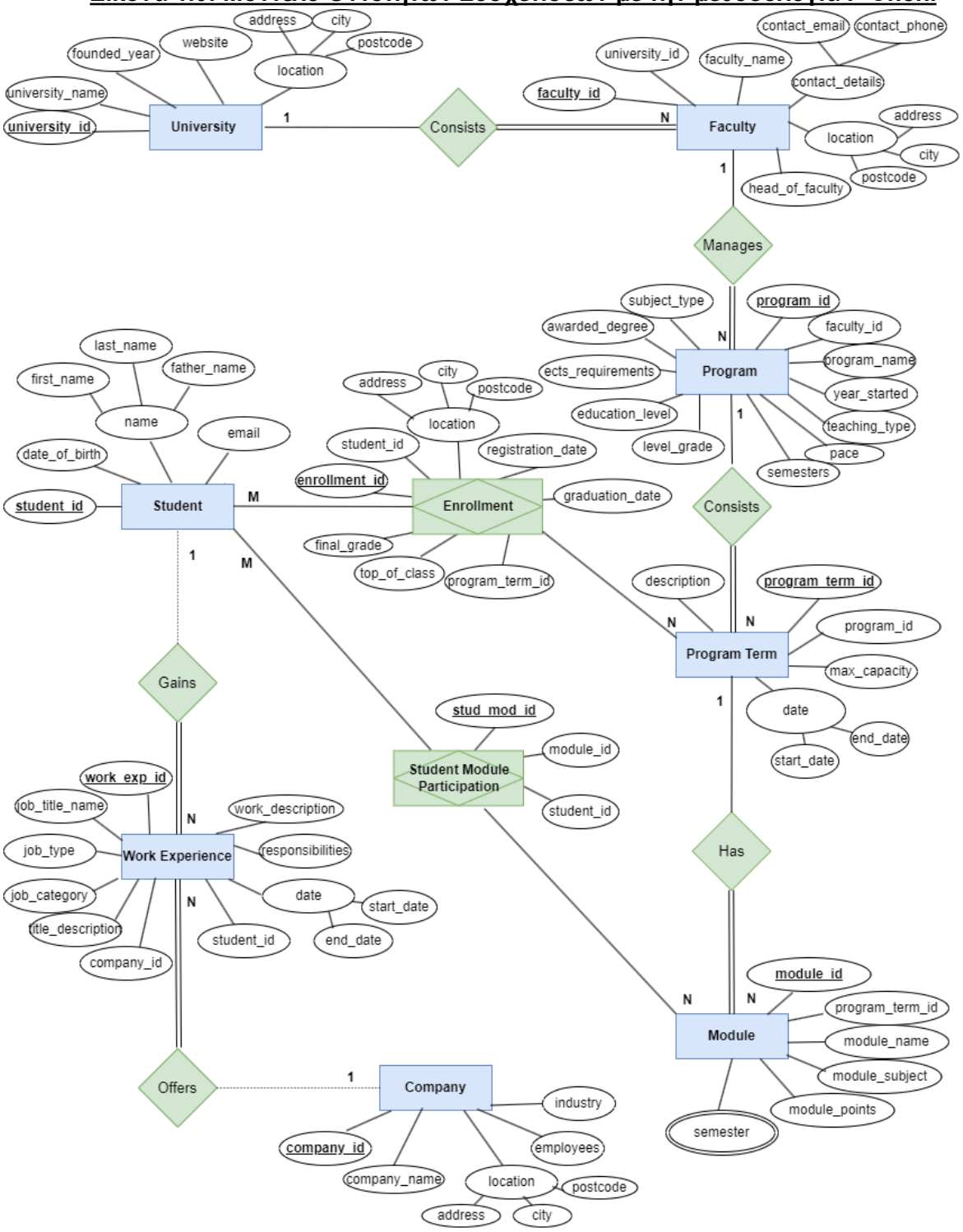
9. Άλλες παραδοχές:

- Η διαγραφή των δεδομένων δεν συνιστάται στην βάση εφόσον μας ενδιαφέρουν ιστορικά δεδομένα φοιτητών / αποφοίτων. Η εντολή ON DELETE CASCADE **ΔΕΝ** έχει συμπεριληφθεί για την ακεραιότητα των δεδομένων.
- Για κάθε εισαγωγή σε οποιονδήποτε πίνακα, το πεδίο id πρέπει να οριστεί και **ΔΕΝ** είναι αυτό αυξανόμενο.
- Έχει γίνει μεγάλη προσπάθεια εισαγωγής δεδομένων μέσω της python τα οποία είναι ρεαλιστικά, όπως αντιστοίχιση τμημάτων με πανεπιστημίων, προγραμμάτων σπουδών με τμήματα κλπ. Σε ορισμένες περιπτώσεις, όπως τα μαθήματα, τα ονόματα εταιρειών και οι τίτλοι εργασιακής εμπειρίας έχουν εισαχθεί μέσω python και της βιβλιοθήκης faker που παράγει ψεύτικα δεδομένα τα οποία **ΔΕΝ** έχουν αντιστοίχιση στην πραγματικότητα. Το ίδιο ισχύει για σχέσεις μεταξύ τους, πχ εγγραφές μαθήματος intro to computer science το οποίο μπορεί να έχει γραφτεί σε ένα πρόγραμμα σπουδών που δεν έχει να κάνει με πληροφορική. Η μία εργασιακή εμπειρία σε μία εταιρία πληροφορικής να έχει τίτλο κηπουρός κλπ.

2. Μοντέλο οντοτήτων-συσχετίσεων

2.1 Σχήμα Μοντέλου Οντοτήτων Συσχετίσεων

Εικόνα 1.0: Μοντέλο Οντοτήτων Συσχετίσεων με την μεθοδολογία P Chen.



2.2 Περιγραφή εννοιολογικού μοντέλου:

Περιγραφή:
Η μοντελοποίηση δεδομένων, αναφέρεται στις ενδιαφέρουσες υπό μελέτη, οντότητες και συσχετίσεις μεταξύ τους, καθώς και στα ενδιαφέροντα χαρακτηριστικά στο πλαίσιο ενός επιχειρηματικού πεδίου (Βασιλακόπουλος, Γ., 2023, σ.33). Επιπλέον, η απόφαση για το επίπεδο λεπτομέρειας μπορεί να έχει σημαντικό αντίκτυπο στην ανάπτυξη και συντήρηση του συστήματος καθότι μπορεί να επιφέρει σημαντική πρόσθετη οικονομική επιβάρυνση από αυτή που πραγματικά απαιτείται (Βασιλακόπουλος, Γ., 2023, σ.46). Επομένως, συμπεριλαμβάνουμε στο μοντέλο ΟΣ μόνο τις οντότητες που συμμετέχουν στην καταγραφή δεδομένων των φοιτητών, των προγραμμάτων σπουδών στα οποία έχουν αποφοιτήσει, τις εργασιακές εμπειρίες, και τις συμμετοχές τους σε μαθήματα.

Το μοντέλο ΟΣ του σχήματος 1.0 αποτυπώνει, τις οντότητες που ενδιαφέρουν το επιχειρηματικό πεδίο (πανεπιστημιακό σύστημα διαχείρισης φοιτητών / αποφοίτων), τις σχέσεις μεταξύ τους και τον τύπο σχέσεων και τα χαρακτηριστικά οντοτήτων τα οποία είναι κρίσιμα για την ποιοτική καταγραφή δεδομένων των αποφοίτων.

Παραδοχές Σχέσεων μοντέλου ΟΣ:

- Ένα πανεπιστήμιο (**University**) αποτελείται από πολλά τμήματα (**Faculties**) (δεν έχει γίνει αναφορά για σχολές, υποθέτουμε πως υπάρχουν μόνο τμήματα) και ένα τμήμα από ένα πανεπιστήμιο (1:N).
- Ένα τμήμα (**Faculty**) διοργανώνει πολλά προγράμματα σπουδών (**Program**) και ένα πρόγραμμα σπουδών διοργανώνεται από ένα τμήμα πανεπιστημίου (1:N) με ισχυρή συμμετοχή.
- Ένα πρόγραμμα σπουδών (**Program**) αποτελείται από ένα ή πολλά τμήματα του προγράμματος (**Program Term**) αλλά ένα τμήμα είναι μέρος ενός και μόνο προγράμματος σπουδών (1:N) με ισχυρή συμμετοχή.
- Ένα τμήμα προγράμματος σπουδών (**Program Term**) έχει πολλούς μαθητές (**Student**) και οι μαθητές μπορούν να γραφτούν σε πολλά τμήματα σπουδών (M:N), εξού η οντότητα συσχέτισης εγγραφής **Enrollment**.
- Ένα πρόγραμμα σπουδών (**Program Term**) έχει πολλά μαθήματα (**Module**) και ένα module ένα πρόγραμμα σπουδών (1:N) με ισχυρή συμμετοχή καθώς δεν νοείται module χωρίς το τμήμα του προγράμματος σπουδών.
- Ένας φοιτητής (**Student**) μπορεί να συμμετέχει σε πολλά μαθήματα (**Module**) και ένα μαθήματα μπορεί να έχει πολλούς φοιτητές (M:N), εξού η οντότητα συσχέτισης **Student Module Participation**.
- Ένας φοιτητής (**Student**) **μπορεί** να έχει μία ή πολλές εργασιακές εμπειρίες (**Work Experience**) αλλά μία εργασιακή εμπειρία μόνο ένα φοιτητή (0:N) με ισχυρή συμμετοχή καθώς δεν νοείται εργασιακή εμπειρία χωρίς τον εργαζόμενο φοιτητή.
- Μία εταιρεία (**Company**) **μπορεί** να προσφέρει μία ή παραπάνω εργασιακή εμπειρία (**Work Experience**) και μία εργασιακή εμπειρία μπορεί να προσφερθεί

από μία εταιρεία (0:N), με ισχυρή συμμετοχή καθώς δεν νοείται εργασιακή εμπειρία χωρίς την εταιρία στην οποία γίνεται.

Παραδοχές μοντέλου ΟΣ:

- Στο μοντέλο ΟΣ οι οντότητες Program Term και Modules θα μπορούσαν να θεωρηθούν ως ασθενής οντότητες καθώς εξαρτώνται από τις γονικές του οντότητες. Όμως εφόσον αυτές οι οντότητες έχουν μοναδικούς προσδιοριστές που τις καθορίζουν ξεχωριστά (primary key) θα τις συμπεριλάβουμε σαν κανονικές οντότητες που έχουν ολική (ισχυρή) συμμετοχή στις γονικές τους οντότητες.
- Οι σχέσεις M:N έχουν παραμείνει στο μοντέλο ΟΣ όπως αυτές περιγράφονται στον πραγματικό κόσμο, στην θέση αυτών των σχέσεων έχουμε ορίσει τις σχέσεις ως associative entities στο μοντέλο ΟΣ που διασπώνται στο σχεσιακό.

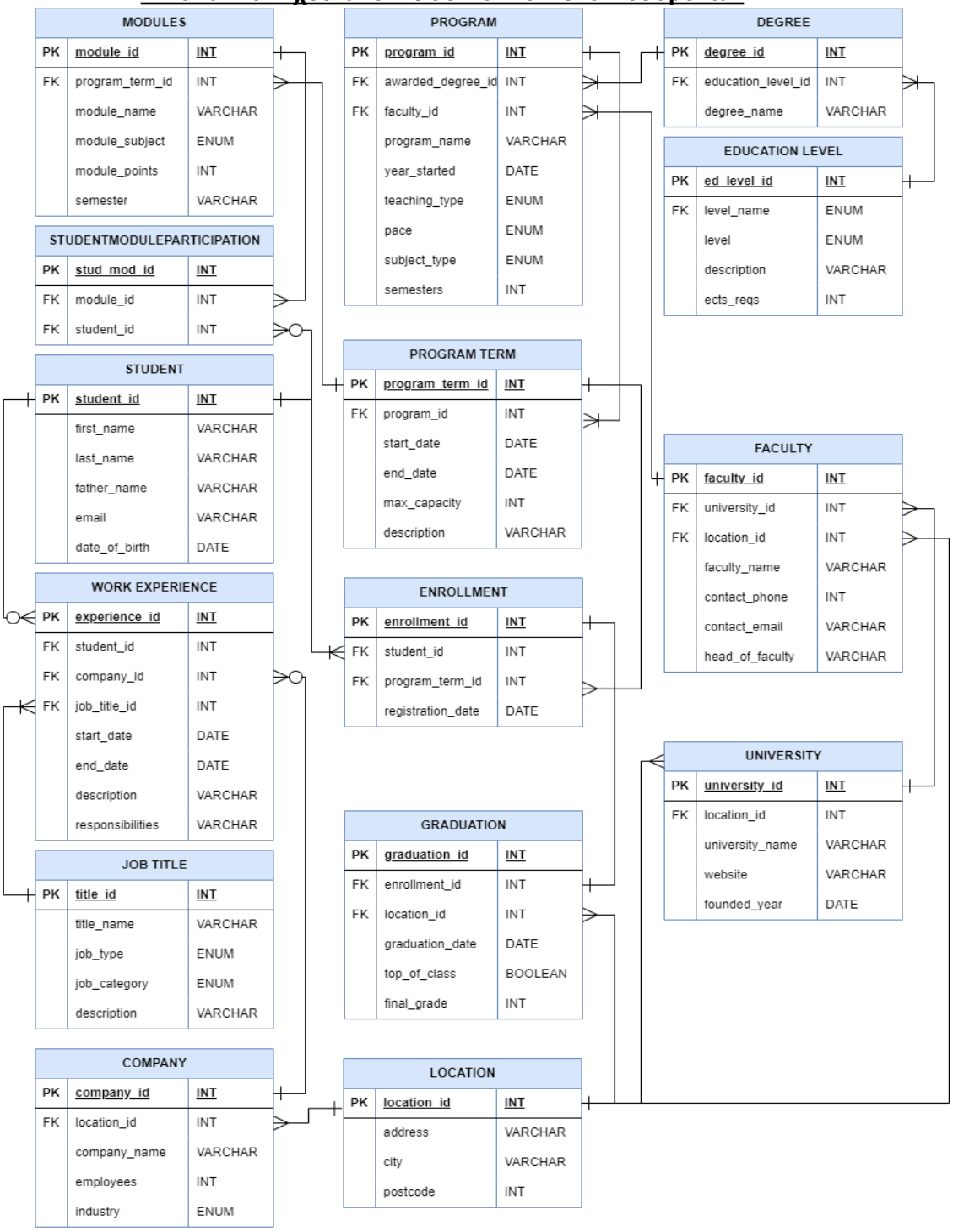
Πίνακας 1.0: Χαρακτηριστικά οντοτήτων που παρουσιάζονται στο μοντελο ΟΣ

Student	(student_id, name, email, date_of_birth)
University	(university_id, university_name, founded_year, website, location)
Faculty	(faculty_id, university_id, faculty_name, contact_details, location, head_of_faculty)
Program	(program_id, faculty_id, program_name, year_started, teaching_type, semesters, pace, subject_type, ects_requirements, awarded_degree, level_grade, education_level)
Program Term	(program_term_id, program_id, date, max_capacity, description)
Modules	(module_id, program_term_id, module_name, module_subject, module_points, semester)
Work Experience	(work_exp_id, company_id, student_id, job_title_name, job_type, work_description, job_category, title_description, date, responsibilities)
Company	(company_id, company_name, location, employees, industry)
Enrollment	(enrollment_id, student_id, program_term_id, location, registration_date, graduation_date, final_grade, top_of_class)
StudentModule Participation	(stud_mod_id, module_id, student_id)

3. Σχεσιακό Μοντέλο

3.1 Σχεσιακό Μοντέλο

Εικόνα 2.0: Σχεσιακό - Φυσικό Μοντέλο Δεδομένων



3.2 Περιγραφή σχεσιακού μοντέλου:

Περιγραφή

Τα διαγράμματα σχήματος βάσης δεδομένων, επίσης γνωστά ως σχεσιακά μοντέλα, είναι οπτικές αναπαραστάσεις της λογικής δομής μιας βάσης δεδομένων. Χτίζουν πάνω στο μοντέλο ΟΣ αντικατοπτρίζοντας το διάγραμμα του πραγματικού κόσμου σε ένα διάγραμμα με έμφαση τα δεδομένα, και των διαφόρων σχέσεων μεταξύ τους έτοιμα προς υλοποίηση σε μία βάση δεδομένων. Βοηθούν στην κατανόηση και την επικοινωνία της δομής των δεδομένων μεταξύ χρηστών, συμβάλλει στη βελτίωση της ποιότητας των σχεδιαστικών αποφάσεων και επιτρέπει την εύκολη τροποποίηση της βάσης δεδομένων όταν απαιτείται. Για το συγκεκριμένο διάγραμμα έχουμε χρησιμοποιήσει την μεθοδολογία crow's foot notation για να απεικονίσουμε τις σχέσεις μεταξύ πινάκων, την πολλαπλότητα τους και αν έχουν αναγκαστική η προαιρετική συμμετοχή. Τα σύμβολα με τους κύκλους στα άκρα συμβολίζουν προαιρετική συμμετοχή.

Το μοντέλο ΟΣ που προκύπτει στο σχήμα 1.0 έχει μεταφραστεί σε σχεσιακό στο σχήμα 2.0 μέσω διάσπασης πινάκων σε μικρότερους και πιο διαχειρίσιμους για λόγους κανονικοποίησης, που συνδέονται μέσω ξένων κλειδιών (foreign keys). Οι καινούργιοι πίνακες **Student Module Participation**, **Enrollment, Location, Job Title και Graduation** είναι παράδειγμα του πώς το μοντέλο ΟΣ εξελίσσεται για να διατηρηθούν οι κανονικές μορφές, η Πρώτη Κανονική Μορφή (1NF), η Δεύτερη Κανονική Μορφή (2NF), και η Τρίτη Κανονική Μορφή (3NF) βοηθωντας στην αποφυγή προβλημάτων όπως διπλότυπης αποθήκευσης δεδομένων, και ασυνέπειες αυτών (data inconsistencies).

Παραδοχές Σχεσιακού μοντέλου:

- Αρκετές οντότητες εμφανίζουν χαρακτηριστικά τοποθεσίας οπότε για την αποφυγή επανάληψης, έχει δημιουργηθεί ο πίνακας (Location) που συνδέεται με τις κατάλληλες οντότητες μέσω ξένου κλειδιού.
- Με βάση τους κανόνες κανονικοποίησης, και την παραδοχή πως καταγράφονται απόφοιτοι μεταπτυχιακών και διδακτορικών πέρα από των προπτυχιακών σπουδών. Έχουμε συμπεριλάβει έναν πίνακα για τα επίπεδα σπουδών (**EducationLevel**) και έναν για τους τίτλους σπουδών (**Degrees**). Με αυτόν τον τρόπο επιτυγχάνουμε την αποφυγή επανάληψης δεδομένων και διατήρηση του προτύπου της 3ης μορφής κανονικοποίησης (3NF). Ένα education level έχει πολλά degree, ενώ ένα degree μόνο ένα education level (1:N).
- Καταγράφουμε την συμμετοχή μαθητών σε μαθήματα μέσω του ενδιάμεσου πίνακα **StudentModuleParticipation** εφόσων ένας μαθητής μπορεί να εγγραφεί σε πολλά μαθήματα και ένα μάθημα έχει πολλούς φοιτητές.
- Ξανά, για λόγους κανονικοποίησης, ο τίτλος μιας εργασιακής εμπειρίας (**Job Title**) έχει δημιουργηθεί ως καινούργιος πίνακας καθώς έχει πεδία που αφορούν

αυτή την οντότητα και μόνο. Ένα Job Title συνδέεται με πολλά work experiences και ένα work experience έχει ένα Job Title, μια σχέση 1:N.

- Τα στοιχεία αποφοίτησης (**Graduation**) ενος φοιτητή έχουν προστεθεί σαν καινούργιος πίνακας για λόγους κανονικοποίησης με μία σχέση 1:1 με την κάθε εγγραφή (Enrollment) ενός φοιτητή.
- Τα εξάμηνα (semester) κάθε μαθήματος θα μπορούσε να είναι ξεχωριστός πίνακας για λόγους κανονικοποίησης, αλλά εφόσον δεν αποτελεί πρωταρχικό στόχο η καταγραφή των δεδομένων αυτών, έχει θεωρηθεί πρόσθετη πολυπλοκότητα και δεν έχει συμπεριληφθεί.

Επομένως το μοντέλο ΟΣ στο σχεσιακό μοντέλο έχει εξελιχθεί στους παρακάτω πίνακες:

Πίνακας 2.0: Πίνακες και χαρακτηριστικά αυτών στο σχεσιακό μοντέλο

Student	(student_id, first_name, last_name, father_name, email, date_of_birth)
Enrollment	(enrollment_id, student_id, program_term_id, registration_date)
University	(university_id, location_id, university_name, website, founded_year)
Faculty	(faculty_id, university_id, location_id, faculty_name, contact_phone, contact_email, head_of_faculty)
Program	(program_id, awarded_degree_id, faculty_id, program_name, year_started, teaching_type, pace, subject_type, semesters)
Program_Term	(program_term_id, program_id, start_date, end_date, max_capacity, description)
Graduation	(graduation_id, enrollment_id, location_id, graduation_date, top_of_class, final_grade)
Modules	(module_id, program_term_id, module_name, module_subject, module_points, semester)
StudentModuleResults	(stud_mod_id, module_id, student_id)
Degree	(degree_id, education_level_id, degree_name)
EducationLevel	(ed_level_id, level_name, level, description, ects_reqs)
WorkExperience	(experience_id, student_id, company_id, job_title_id, start_date, end_date, description, responsibilities)
Company	(company_id, location_id, company_name, employees, industry)
JobTitle	(title_id, title_name, job_type, job_category, description)
Location	(location_id, address, city, postcode)

4. Ενδεικτικές Εντολές Insert, Update, Delete

Πίνακας Location	
INSERT	`INSERT INTO Location (location_id, address, city, postcode)` `VALUES (location_id_value, 'address_value', 'city_value', postcode_value);`
DELETE	`DELETE FROM Location WHERE location_id = location_id_value;`
UPDATE	`UPDATE Location SET address = 'new_address_value' WHERE location_id = location_id_value;`
Πίνακας University	
INSERT	`INSERT INTO University (university_id, university_name, founded_year, website, location_id)` `VALUES (university_id_value, 'university_name_value', founded_year_value, 'website_value', location_id_value);`
DELETE	`DELETE FROM University WHERE university_id = university_id_value;`
UPDATE	`UPDATE University SET university_name = 'new_university_name_value' WHERE university_id = university_id_value;`
Πίνακας Faculty	
INSERT	`INSERT INTO Faculty (faculty_id, university_id, faculty_name, contact_phone, contact_email, location_id, head_of_faculty)` `VALUES (faculty_id_value, university_id_value, 'faculty_name_value', contact_phone_value, 'contact_email_value', location_id_value, 'head_of_faculty_value');`
DELETE	`DELETE FROM Faculty WHERE faculty_id = faculty_id_value;`
UPDATE	`UPDATE Faculty SET faculty_name = 'new_faculty_name_value' WHERE faculty_id = faculty_id_value;`
Πίνακας Education Level	
INSERT	`INSERT INTO EducationLevel (level_id, level_name, level, description, ects_requirements)` `VALUES (level_id_value, 'level_name_value', 'level_value', 'description_value', ects_requirements_value);`
DELETE	`DELETE FROM EducationLevel WHERE level_id = level_id_value;`

UPDATE	`UPDATE EducationLevel SET level_name = 'new_level_name_value' WHERE level_id = level_id_value;`
Πίνακας Degree	
INSERT	`INSERT INTO Degree (degree_id, education_level_id, degree_name) VALUES (degree_id_value, education_level_id_value, 'degree_name_value');`
DELETE	`DELETE FROM Degree WHERE degree_id = degree_id_value;`
UPDATE	`UPDATE Degree SET degree_name = 'new_degree_name_value' WHERE degree_id = degree_id_value;`
Πίνακας Program	
INSERT	`INSERT INTO Program (program_id, awarded_degree, faculty_id, program_name, year_started, teaching_type, pace, subject_type, semesters) VALUES (program_id_value, awarded_degree_value, faculty_id_value, 'program_name_value', year_started_value, 'teaching_type_value', 'pace_value', 'subject_type_value', semesters_value);`
DELETE	`DELETE FROM Program WHERE program_id = program_id_value;`
UPDATE	`UPDATE Program SET program_name = 'new_program_name_value' WHERE program_id = program_id_value;`
Πίνακας Program_Term	
INSERT	`INSERT INTO Program_Term (program_term_id, program_id, start_date, end_date, max_capacity, description) VALUES (program_term_id_value, program_id_value, 'start_date_value', 'end_date_value', max_capacity_value, 'description_value');`
DELETE	`DELETE FROM Program_Term WHERE program_term_id = program_term_id_value;`
UPDATE	`UPDATE Program_Term SET description = 'new_description_value' WHERE program_term_id = program_term_id_value;`
Πίνακας Modules	
INSERT	`INSERT INTO Modules (module_id, program_term_id, module_name, module_subject, module_points, semester) VALUES (module_id_value, program_term_id_value, module_name_value, module_subject_value, module_points_value, semester_value);`

DELETE	`DELETE FROM Modules WHERE module_id = module_id_value;`
UPDATE	`UPDATE Modules SET module_name = new_module_name_value WHERE module_id = module_id_value;`
Πίνακας Student	
INSERT	`INSERT INTO Student (student_id, first_name, last_name, father_name, email, date_of_birth) VALUES (student_id_value, first_name_value, last_name_value, father_name_value, email_value, date_of_birth_value);`
DELETE	`DELETE FROM Student WHERE student_id = student_id_value;`
UPDATE	`UPDATE Student SET first_name = new_first_name_value WHERE student_id = student_id_value;`
Πίνακας Student Module Participation	
INSERT	`INSERT INTO StudentModuleParticipation (stud_mod_id, module_id, student_id) VALUES (stud_mod_id_value, module_id_value, student_id_value);`
DELETE	`DELETE FROM StudentModuleParticipation WHERE stud_mod_id = stud_mod_id_value;`
UPDATE	`UPDATE StudentModuleParticipation SET module_id = new_module_id_value WHERE stud_mod_id = stud_mod_id_value;`
Πίνακας Enrollment	
INSERT	`INSERT INTO Enrollment (enrollment_id, student_id, program_term_id, registration_date) VALUES (enrollment_id_value, student_id_value, program_term_id_value, registration_date_value);`
DELETE	`DELETE FROM Enrollment WHERE enrollment_id = enrollment_id_value;`
UPDATE	`UPDATE Enrollment SET registration_date = new_registration_date_value WHERE enrollment_id = enrollment_id_value;`
Πίνακας Graduation	
INSERT	`INSERT INTO Graduation (graduation_id, enrollment_id, final_grade, graduation_date, top_of_class, location_id) VALUES (graduation_id_value, enrollment_id_value, final_grade_value, graduation_date_value, top_of_class_value, location_id_value);`

DELETE	`DELETE FROM Graduation WHERE graduation_id = graduation_id_value;`
UPDATE	`UPDATE Graduation SET final_grade = new_final_grade_value WHERE graduation_id = graduation_id_value;`
Πίνακας Company	
INSERT	`INSERT INTO Company (company_id, company_name, location_id, employees, industry)` `VALUES (company_id_value, company_name_value, location_id_value, employees_value, industry_value);`
DELETE	`DELETE FROM Company WHERE company_id = company_id_value;`
UPDATE	`UPDATE Company SET company_name = new_company_name_value WHERE company_id = company_id_value;`
Πίνακας Job Title	
INSERT	`INSERT INTO JobTitle (title_id, title_name, job_type, job_category, description)` `VALUES (title_id_value, title_name_value, job_type_value, job_category_value, description_value);`
DELETE	`DELETE FROM JobTitle WHERE title_id = title_id_value;`
UPDATE	`UPDATE JobTitle SET title_name = new_title_name_value WHERE title_id = title_id_value;`
Πίνακας Work Experience	
INSERT	`INSERT INTO WorkExperience (experience_id, student_id, company_id, job_title_id, start_date, end_date, description, responsibilities)` `VALUES (experience_id_value, student_id_value, company_id_value, job_title_id_value, start_date_value, end_date_value, description_value, responsibilities_value);`
DELETE	`DELETE FROM WorkExperience WHERE experience_id = experience_id_value;`
UPDATE	`UPDATE WorkExperience SET description = new_description_value WHERE experience_id = experience_id_value;`

5. Ενδεικτικές Εντολές SQL

Παρακάτω παρατίθενται μερικές εντολές SQL (MySQL) για την ανάδειξη των δυνατοτήτων των πινάκων και των σχέσεων μεταξύ τους.

1. Πλήθος εγγραφών ανά πρόγραμμα σπουδών για το πανεπιστήμιο Πειραιά

Ανάκτηση πλήθους εγγραφών φοιτητών για κάθε πρόγραμμα που προσφέρεται από το Πανεπιστήμιο του Πειραιά. Παρέχει μια σαφή εικόνα της δημοτικότητας και της προσέλκυσης που έχει κάθε πρόγραμμα σπουδών μέσα στο πανεπιστήμιο.

```
SELECT p.program_id, p.program_name, COUNT(e.student_id) AS `Σύνολο Εγγραφών`
FROM Program p
JOIN Program_Term pt ON p.program_id = pt.program_id
JOIN Enrollment e ON pt.program_term_id = e.program_term_id
JOIN Faculty f ON p.faculty_id = f.faculty_id
JOIN University u ON f.university_id = u.university_id
WHERE u.university_name = 'University of Piraeus'
GROUP BY p.program_id, p.program_name;
```

Αποτέλεσμα

program_id	program_name	Σύνολο Εγγραφών
1	Masters in Advanced Information Systems	3600
2	Masters in Business Analytics	3585
6	Associate Degree in Business and Finance	1755
7	Masters in Technology Management	3264
13	Masters in Financial Management	290

2. Πλήθος και ποσοστό παροχής εργασιακών εμπειριών ανά εταιρεία

Ανάκτηση πλήθους εργασιακών εμπειριών ανά εταιρεία, και ποσοστό σε σχέση με το σύνολο. Βοηθάει στην αξιολόγηση εταιρειών που παρέχουν εργασιακή εμπειρία.

```
SELECT c.company_name, COUNT(we.experience_id) AS `Αριθμός Προσλήψεων`,
ROUND((COUNT(we.experience_id) * 100.0 / total.total_count), 2) AS `Ποσοστό σε
σχέση με τις προσλήψεις`
FROM Company c
JOIN WorkExperience we ON c.company_id = we.company_id
CROSS JOIN (SELECT COUNT(*) as total_count FROM WorkExperience) total
GROUP BY c.company_name, total.total_count
ORDER BY `Αριθμός Προσλήψεων` DESC;
```

Αποτέλεσμα

company_name	Αριθμός Προσλήψεων	Ποσοστό σε σχέση με τις προσλήψεις
Κρεμμύδας, Μπαντής και Γάτος	54	0.54
Βουτσελας Ο.Ε.	49	0.49
Δημοπούλου, Παυλίδου και Μακρή	49	0.49
Φωτόπουλος-Πάγκαλος	48	0.48
Αποστολόπουλος Α.Β.Ε.Ε.	45	0.45
Βαρακλής-Τσίμη	45	0.45

3. Ποσοστό αριστούχων μαθητών ανά πανεπιστήμιο.

Ανάκτηση ποσοστού αποφοίτων που αποφοίτησαν στην κορυφή της τάξης για κάθε πανεπιστήμιο. Αυτή η πληροφορία μπορεί να είναι χρήσιμη για τα πανεπιστήμια για να κατανοήσουν την απόδοση των φοιτητών τους και την ακαδημαϊκή αρτιότητα του προγράμματος σπουδών τους.

```
SELECT u.university_name, COUNT(CASE WHEN g.top_of_class THEN 1 ELSE NULL END) *
100.0 / COUNT(*) AS `% Αριστούχοι`
FROM Graduation g
JOIN Enrollment e ON g.enrollment_id = e.enrollment_id
JOIN Program_Term pt ON e.program_term_id = pt.program_term_id
JOIN Program p ON pt.program_id = p.program_id
JOIN Faculty f ON p.faculty_id = f.faculty_id
JOIN University u ON f.university_id = u.university_id
GROUP BY u.university_name
ORDER BY `% Αριστούχοι`DESC;
```

Αποτέλεσμα

university_name	% Αριστούχοι
National and Kapodistrian University of Athens	50.02331
University of Piraeus	49.79952
National Technical University of Athens	49.68734
Hellenic Open University	48.64865

4. Πλήθος μαθημάτων ανά πρόγραμμα σπουδών ανα πανεπιστήμιο.

Ανάκτηση συνολικού αριθμού μαθημάτων ανά πρόγραμμα σπουδών ανά πανεπιστήμιο. Χρήσιμο για την αξιολόγηση και σύγκριση φόρτου μαθημάτων μεταξύ πανεπιστημίων.

```
SELECT u.university_name AS Πανεπιστήμιο, p.program_name AS Πρόγραμμα_Σπουδών,
    COUNT(m.module_id) AS Συνολικός_Αριθμός_Μαθημάτων
FROM University u
JOIN Faculty f ON u.university_id = f.university_id
JOIN Program p ON f.faculty_id = p.faculty_id
JOIN Program_Term pt ON p.program_id = pt.program_id
JOIN Modules m ON pt.program_term_id = m.program_term_id
GROUP BY u.university_name, f.faculty_name, p.program_name
ORDER BY u.university_name, f.faculty_name, p.program_name;
```

Αποτέλεσμα

Πανεπιστήμιο	Πρόγραμμα_Σπουδών	Συνολικός_Αριθμός_Μαθημάτων
National and Kapodistrian University of Athens	Bachelor of Arts in Political Science and Internat...	4
National and Kapodistrian University of Athens	Ph.D. in Civil Engineering Structures	4
National and Kapodistrian University of Athens	Bachelor of Arts in English Literature	8
National and Kapodistrian University of Athens	Master of Law in Intellectual Property	9
National and Kapodistrian University of Athens	Masters in Marketing Analytics	3
National and Kapodistrian University of Athens	Ph.D. in Environmental Engineering Research	6
University of Piraeus	Associate Degree in Business and Finance	8
University of Piraeus	Masters in Business Analytics	1
University of Piraeus	Masters in Advanced Information Systems	8
University of Piraeus	Masters in Technology Management	3

5. Πλήθος και ποσοστό εργασιακών τίτλων.

Ανάκτηση των πιο δημοφιλών θέσεων εργασιακών εμπειριών που έχουν καταχωρηθεί στη βάση δεδομένων.

```
SELECT jt.title_name AS `Θέση_Εργασίας`, COUNT(we.job_title_id) AS `Πλήθος`,
ROUND((COUNT(we.job_title_id) * 100.0 / (SELECT COUNT(*) FROM WorkExperience)), 2) AS
`% Συνόλου`
FROM WorkExperience we
JOIN JobTitle jt ON we.job_title_id = jt.title_id
GROUP BY jt.title_name
ORDER BY `Πλήθος` DESC;
```

Αποτέλεσμα

	Θέση_Εργασίας	Πλήθος	% Συνόλου
▶	Ειδικός Ιατρικών και Βιολογικών Εργαστηρίων Α...	87	0.87
	Στέλεχος Κοστολόγησης στη Βιομηχανία - Βιοτε...	85	0.85
	Επαγγελματίας Αθλητής	81	0.81
	Ανθοκόμος	75	0.75
	Βρεφοκόμος	69	0.69
	Ελαιοχρωματιστής	67	0.67
	Πλέκτης	65	0.65

6. 10 πόλεις με τις περισσότερες εταιρείες

Ανάκτηση των 10 πόλεων που εδρεύουν οι περισσότερες εταιρείες.

```
SELECT l.city AS Πόλη, COUNT(DISTINCT c.company_id) AS Πλήθος_Εταιρειών
FROM Company c
JOIN Location l ON c.location_id = l.location_id
GROUP BY l.city
ORDER BY Πλήθος_Εταιρειών DESC
LIMIT 10;
```

Αποτέλεσμα

	Πόλη	Πλήθος_Εταιρειών
▶	Τρίκαλα	19
	Καρδίτσα	17
	Γρεβενά	17
	Ναύπλιο	15
	Κοζάνη	14
	Σάμος	13

5. Stored procedures (Αποθηκευμένες Διαδικασίες)

Τα stored procedures είναι ομάδες εντολών SQL που αποθηκεύονται στη βάση δεδομένων και εκτελούνται ως μια εντολή. Ενέργειες που απαιτούν πολλαπλές ερωτήσεις και υπολογισμούς μπορούν να συμπυκνωθούν σε ένα μόνο κάλεσμα, εξοικονομώντας πόρους και βελτιώνοντας την απόδοση. Αυτό καθιστά την χρήση τους στην συγκεκριμένη βάση δεδομένων αρκετά χρήσιμη καθώς οι χρήστες της βάσης δεδομένων και οι διαχειριστές μπορούν με εύκολο τρόπο να καλούν τυποποιημένες εργασίες μειώνοντας τα λάθη ανάκτησης δεδομένων.

1. **Εύρεση φοιτητών που δεν έχουν εργασιακή εμπειρία για το επιλεγμένο διάστημα σε ένα επιλεγμένο τμήμα.**

Εύρεση φοιτητών συσχετιζόμενοι με ένα δοθέν τμήμα που δεν έχουν εργασιακή εμπειρία μέσα σε ένα δοθέν χρονικό διάστημα. Βοηθάει στην εύρεση και υποστήριξη μαθητών που δυσκολεύονται να βρουν εργασιακή εμπειρία άμεσα. Συνδέοντας πίνακες φοιτητών, εγγραφών, προγραμμάτων χρησιμοποιεί LEFT JOIN στον πίνακα WorkExperience με συγκεκριμένες συνθήκες για να εξαιρέσει τις εμπειρίες εργασίας εντός των καθορισμένων ημερομηνιών και χρησιμοποιεί ORDER BY για την ταξινόμηση των αποτελεσμάτων βάσει του επωνύμου και του ονόματος του φοιτητή.

```
DELIMITER $$
CREATE PROCEDURE FacultyStudentsWithoutWorkInPeriod(
    IN facultyName VARCHAR(255),
    IN startDate DATE,
    IN endDate DATE
)
BEGIN
    SELECT DISTINCT s.student_id, s.first_name, s.last_name, s.email
    FROM Student s
    JOIN Enrollment e ON s.student_id = e.student_id
    JOIN Program_Term pt ON e.program_term_id = pt.program_term_id
    JOIN Program p ON pt.program_id = p.program_id
    JOIN Faculty f ON p.faculty_id = f.faculty_id
    LEFT JOIN WorkExperience we ON s.student_id = we.student_id
        AND (
            (we.start_date BETWEEN startDate AND endDate)
            OR
            (we.end_date BETWEEN startDate AND endDate)
        )
    WHERE f.faculty_name = facultyName
```

```
        AND we.student_id IS NULL;
END$$
DELIMITER ;
-- ΠΑΡΑΔΕΙΓΜΑ
CALL FacultyStudentsWithoutWorkInPeriod('UoP Department of Digital Systems', '1900-08-01', '2500-08-01');
```

Αποτέλεσμα

	student_id	first_name	last_name	email
▶	41	Σεραφείμ	Ζαφείρης	antipatros71@example.com
	112	Οδυσσέας	Κελλάρης	melina82@example.com
	329	Πολύβια	Φιλιππάτος	pchatzopoulos@example.com
	424	Πολύδωρος	Κατσίπης	ioulianos19@example.org
	446	Ιορδάνης	Κυρίτση	hsivva@example.org
	491	Συμεωνία	Πανούσης	platon.polychronopoulos@example.net
	567	Γραμματική	Πουλίδα	afentis.zacharioudaki@example.org
	597	Χριστίνα	Κολοκάθης	sergios68@example.com
	655	Ρεγγίνα	Λιόντη	antoneas.epameinondas@example.com

2. **Εύρεση μέσου όρου βαθμού αποφοίτησης ανα πρόγραμμα σπουδών για ένα δοθέν τμήμα.**

Παρέχει στους διαχειριστές τη δυνατότητα να ανακτήσουν γρήγορα τον ΜΟ των τελικών βαθμών των φοιτητών ανά πρόγραμμα σπουδών εντός ενός δοθέν τμήματος. Συνδέει τους πίνακες Program, Faculty, Program_Term, Enrollment και Graduation για να δημιουργήσει μια σχέση μεταξύ των προγραμμάτων σπουδών και των τελικών βαθμολογιών των φοιτητών, στο δοθέν τμήμα και υπολογίζει τον μέσο όρο της τελικής βαθμολογίας για τους φοιτητές που αποφοίτησαν από το συγκεκριμένο πρόγραμμα.

```
DELIMITER $$
CREATE PROCEDURE GetAverageGradesByProgramInFaculty(IN
    inputFacultyName VARCHAR(255)
)
BEGIN
    SELECT
        p.program_name,
        AVG(g.final_grade) AS `ΜΟ Τελικού Βαθμού`
    FROM Graduation g
    JOIN Enrollment e ON g.enrollment_id = e.enrollment_id
    JOIN Program_Term pt ON e.program_term_id = pt.program_term_id
    JOIN Program p ON pt.program_id = p.program_id
    JOIN Faculty f ON p.faculty_id = f.faculty_id
```

```
    WHERE f.faculty_name = inputFacultyName
    GROUP BY p.program_name
    ORDER BY `ΜΟ Τελικού Βαθμού` DESC;
END$$
DELIMITER ;
-- ΠΑΡΑΔΕΙΓΜΑ
CALL GetAverageGradesByProgramInFaculty('UoP Department of Digital Systems');
```

Αποτέλεσμα

	program_name	ΜΟ Τελικού Βαθμού
▶	Bachelor of Science in Computer Science	81.4104
	Associate Degree in Mobile App Development	80.2533
	Masters in Advanced Information Systems	80.0631
	Masters in Information Technology	79.9611
	Associate Degree in Network Security	79.9171
	Masters in Information Systems Management	79.8653
	Masters in Technology Management	79.6422
	Ph.D. in Artificial Intelligence	79.6025

3. **Ποσοστό επιτυχίας εύρεσης εργασίας ανά το πτυχίο των αποφοίτων για ένα δοθέν επίπεδο σπουδών.**

Δυνατότητα ανάκτησης στοιχείων σχετικών με την επιτυχία των φοιτητών στην εύρεση εργασιακής εμπειρίας ανάλογα με το πτυχίο για ένα δοθέν επίπεδο σπουδών. Συνδέει τους πίνακες EducationLevel, Degree, Program, Program_Term, Enrollment, Student και WorkExperience για να αναλύσει τα ποσοστά επιτυχίας των αποφοίτων ανά είδος πτυχίου, βάσει του επιπέδου εκπαίδευσης που δίνεται ως είσοδος, φιλτράροντας τα αποτελέσματα για το συγκεκριμένο επίπεδο εκπαίδευσης και υπολογίζοντας το ποσοστό επιτυχίας ως την αναλογία των φοιτητών με εμπειρία εργασίας προς τον συνολικό αριθμό των αποφοίτων, ομαδοποιώντας και ταξινομώντας τα αποτελέσματα βάσει του ονόματος του πτυχίου και του ποσοστού επιτυχίας σε φθίνουσα σειρά.

```
DELIMITER $$
CREATE PROCEDURE GetSuccessRatesByEducationLevel(IN inputEducationLevel VARCHAR(255))
BEGIN
    SELECT
        D.degree_name,
        COUNT(DISTINCT S.student_id) AS Σύνολο_Αποφοίτων,
        COUNT(DISTINCT W.student_id) AS Απόφοιτοι_Με_Εμπειρία,
```

```
        IF(COUNT(DISTINCT S.student_id) > 0, (COUNT(DISTINCT W.student_id) /
COUNT(DISTINCT S.student_id)) * 100, 0) AS Ποσοστό_Επιτυχίας
    FROM
        EducationLevel EL
    JOIN Degree D ON EL.level_id = D.education_level_id
    JOIN Program P ON D.degree_id = P.awarded_degree
    JOIN Program_Term PT ON P.program_id = PT.program_id
    JOIN Enrollment E ON PT.program_term_id = E.program_term_id
    JOIN Student S ON E.student_id = S.student_id
    LEFT JOIN WorkExperience W ON S.student_id = W.student_id
    WHERE EL.level_name = inputEducationLevel
    GROUP BY D.degree_name
    ORDER BY Ποσοστό_Επιτυχίας DESC;
END$$
DELIMITER ;
-- ΠΑΡΑΔΕΙΓΜΑ
CALL GetSuccessRatesByEducationLevel('Bachelors');
```

Αποτέλεσμα

	degree_name	Σύνολο_Αποφοίτων	Απόφοιτοι_Με_Εμπειρία	Ποσοστό_Επιτυχίας
▶	Bachelor of Law	1053	688	65.3371
	Bachelor of Architecture	642	419	65.2648
	Bachelor of Science	1508	972	64.4562
	Bachelor of Arts	6187	3912	63.2294
	Bachelor of Engineering	898	557	62.0267

4. **Για ένα δοθέν τμήμα, επιστροφή εργασιακών τίτλων των αποφοίτων και πλήθος αυτών ανά πρόγραμμα σπουδών.**

Χρήσιμο για την ανάλυση των εργασιακών εμπειριών ανά τμήμα για την προώθηση των προγραμμάτων σπουδών με νούμερα για προσλήψεις. Πραγματοποιεί σύζευξη μεταξύ των πινάκων για να συλλέξει δεδομένα σχετικά με τις προσλήψεις αποφοίτων ανά πρόγραμμα και θέση εργασίας, φιλτράροντας τα αποτελέσματα με βάση το όνομα της δοθείσας σχολής.

```
DELIMITER $$
CREATE PROCEDURE getJobTitleHiringByProgram(IN inputFacultyName VARCHAR(255))
BEGIN
 SELECT
    p.program_name,
    jt.title_name AS job_title,
    COUNT(*) AS Προσλήψεις
```

```
 FROM Student s
 JOIN Enrollment e ON s.student_id = e.student_id
 JOIN Program_Term pt ON e.program_term_id = pt.program_term_id
 JOIN Program p ON pt.program_id = p.program_id
 JOIN Faculty f ON p.faculty_id = f.faculty_id
 JOIN WorkExperience we ON s.student_id = we.student_id
 JOIN JobTitle jt ON we.job_title_id = jt.title_id
 WHERE f.faculty_name = inputFacultyName
 GROUP BY p.program_name, jt.title_name
 ORDER BY p.program_name, COUNT(*) DESC;
END$$
DELIMITER ;
-- ΠΑΡΑΔΕΙΓΜΑ
CALL getJobTitleHiringByProgram('UoP Department of Digital Systems');
```

Αποτέλεσμα

program_name	job_title	Προσλήψεις
Associate Degree in Computer Science	Εκπαιδευτικός Ειδικής Αγωγής	3
Associate Degree in Computer Science	Τεχνικός Ελέγχου Ρύπανσης και Εγκαταστάσεω...	3
Associate Degree in Computer Science	Επιστήμων Πληροφορικής και Η/Υ	3
Associate Degree in Computer Science	Βρεφοκόμος	3
Associate Degree in Computer Science	Ανθοκόμος	3
Associate Degree in Computer Science	Τεχνολόγος Φυτικής Παραγωγής	3
Associate Degree in Computer Science	Τεχνικός Εφαρμογών Πληροφορικής, Δικτύων ...	3
Associate Degree in Computer Science	Επαγγελματίας Αθλητής	2
Associate Degree in Computer Science	Ειδικός Εμπορικών Επιχειρήσεων	2

5. **Για ένα δοθέν κωδικό φοιτητή, αναφορά για τις εγγραφές του σε προγράμματα σπουδών και εργασιακές εμπειρίες που έχει αποκτήσει.**

Επιτρέπει στους χρήστες να ανακτούν όλες τις σημαντικές πληροφορίες για έναν φοιτητή με ένα απλό κάλεσμα διαδικασίας. Εκτελεί μετά μια σειρά επιλογών από τους πίνακες Enrollment, Program_Term, Program, και Graduation για να παρουσιάσει τις εγγραφές του φοιτητή σε προγράμματα, συμπεριλαμβανομένων των ημερομηνιών έναρξης και λήξης κάθε προγράμματος, καθώς και τους τελικούς βαθμούς. Τέλος, συλλέγει και παρουσιάζει δεδομένα από τις εργασιακές εμπειρίες του φοιτητή, όπως ημερομηνίες έναρξης και λήξης, εταιρείες, θέσεις εργασίας, περιγραφές και αρμοδιότητες, μέσω συζεύξεων με τους πίνακες Company και JobTitle.

```
DELIMITER $$
CREATE PROCEDURE `GetStudentProgressReport`(IN student_id_input INT)
BEGIN
```

```
    DECLARE student_first_name VARCHAR(255);
    DECLARE student_last_name VARCHAR(255);

    SELECT first_name, last_name
    INTO student_first_name, student_last_name
    FROM Student
    WHERE student_id = student_id_input;

    SELECT CONCAT('Progress Report for Student: ', student_first_name, ' ',
student_last_name) AS 'Student Information';

    SELECT E.enrollment_id, p.program_name, pt.start_date AS 'Program Start Date',
pt.end_date AS 'Program End Date', g.final_grade
    FROM Enrollment e
    JOIN Program_Term pt ON e.program_term_id = pt.program_term_id
    JOIN Program p ON pt.program_id = p.program_id
    LEFT JOIN Graduation g ON e.enrollment_id = g.enrollment_id
    WHERE e.student_id = student_id_input;

    SELECT
        w.start_date AS 'Start Date',
        w.end_date AS 'End Date',
        c.company_name AS 'Company',
        jt.title_name AS 'Job Title',
        w.description AS 'Description',
        w.responsibilities AS 'Responsibilities'
    FROM WorkExperience w
    JOIN Company c ON w.company_id = c.company_id
    JOIN JobTitle jt ON w.job_title_id = jt.title_id
    WHERE w.student_id = student_id_input;
END$$
DELIMITER ;
-- ΠΑΡΑΔΕΙΓΜΑ
CALL GetStudentProgressReport(155);
```

Αποτέλεσμα 1

	Student Information
▶	Progress Report for Student: Ευμένιος Ρόγγα

Result 585 | Result 586 | Result 587

Αποτέλεσμα 2

	enrollment_id	program_name	Program Start Date	Program End Date	final_grade
▶	155	Masters in Human Resource Development	2021-05-26	2022-05-26	87
	10153	Bachelor of Science in Computer Science	2014-09-29	2015-09-29	78
	20244	Masters in Technology Management	2017-06-06	2018-06-06	87
	23661	Master of Law in Intellectual Property	2017-01-14	2018-01-14	63
	27888	Masters in Business Analytics	2017-01-16	2018-01-16	87
	32206	Ph.D. in Renewable Energy Engineering	2022-08-12	2023-08-12	74

Result 585 | Result 586 | Result 587

Αποτέλεσμα 3

	Start Date	End Date	Company	Job Title	Description	Responsibilities
▶	2020-10-22	2023-05-24	Δίγκας-Δουλάμη	Ωκεανογράφος	Απομόνωση γνωρίζουμ...	Χρονοδιαγράμματα ώρα μι
	2021-02-02	2024-12-08	Κτενίδης, Ρουπακάς και Τ...	Ταμίας	Γειτονιάς προβληματική...	Λεπτά μεταγλωτίσει γιαυτά

Result 585 | Result 586 | Result 587

6. **Για ένα δοθέν χρονικό διάστημα ετών και τμήμα πανεπιστημίου, επιστροφή του πλήθους εργασιακών εμπειριών ανά τομέα απασχόλησης αποφοίτων**.

Επιτρέπει την έρευνα τομέων εργασιών για αποφοίτους από ένα δοθέν τμήμα πανεπιστημίου σε ένα χρονικό εύρος. Δημιουργεί ένα εσωτερικό υποερώτημα για να εντοπίσει τους αποφοίτους συγκεκριμένης σχολής με βάση το όνομα της δοθέν σχολής, οι οποίοι αποφοίτησαν εντός του δοθέντος χρονικού διαστήματος, συνδέοντας τους πίνακες Graduation, Enrollment, Program_Term, Program, και Faculty. Στη συνέχεια, το κύριο ερώτημα συνδέει αυτούς τους αποφοίτους με τις εμπειρίες εργασίας τους (WorkExperience) και τις αντίστοιχες εταιρείες (Company), ομαδοποιώντας τα δεδομένα βάσει του κλάδου της εταιρείας (industry) και υπολογίζοντας το συνολικό αριθμό των μοναδικών αποφοίτων που έχουν εργαστεί σε κάθε κλάδο.

```
DELIMITER $$
```

```
CREATE PROCEDURE GraduateExperiencePerIndustryForGivenTimeAndFaculty(
    IN in_start_year INT,
    IN in_end_year INT,
    IN in_faculty_name VARCHAR(255)
)
BEGIN
    SELECT
        c.industry AS Sector,
        COUNT(DISTINCT we.student_id) AS NumberOfGraduates
    FROM
        WorkExperience we
    JOIN Company c ON we.company_id = c.company_id
    JOIN (
        SELECT DISTINCT e.student_id
        FROM Graduation g
        JOIN Enrollment e ON g.enrollment_id = e.enrollment_id
        JOIN Program_Term pt ON e.program_term_id = pt.program_term_id
        JOIN Program p ON pt.program_id = p.program_id
        JOIN Faculty f ON p.faculty_id = f.faculty_id
        WHERE YEAR(g.graduation_date) BETWEEN in_start_year AND in_end_year
        AND f.faculty_name = in_faculty_name
    ) AS Graduates ON we.student_id = Graduates.student_id
    GROUP BY c.industry
    ORDER BY NumberOfGraduates DESC;
END$$
DELIMITER ;
-- ΠΑΡΑΔΕΙΓΜΑ
CALL GraduateExperiencePerIndustryForGivenTimeAndFaculty(2010, 2050, 'UoP Department
of Business Administration');
```

Αποτέλεσμα

	Sector	NumberOfGraduates
▶	Engineering	692
	Software	656
	Shipping	641
	Auditing	593
	Telecommunications	528
	Banking	506
	Other	496
	Hospitality	476

7. **Απόφοιτοι που εγγράφηκαν σε παραπάνω από ένα προγράμματα σπουδών την ίδια ημερομηνία.**

Βοηθάει στην αναγνώριση πιθανών λάθος εισαγωγών στο σύστημα για τυχόν φοιτητές που εγγράφηκαν σε πολλαπλά προγράμματα σπουδών την ίδια χρονική στιγμή. Εκτελεί αυτοσυνδετικό έλεγχο στον πίνακα Enrollment για να εντοπίσει φοιτητές με πολλαπλές εγγραφές την ίδια ημερομηνία, χρησιμοποιώντας ως κριτήρια την ίδια registration_date και διαφορετικό enrollment_id μεταξύ δύο εγγραφών του ίδιου φοιτητή. Στη συνέχεια, ομαδοποιεί τα δεδομένα ανά student_id και υπολογίζει τον αριθμό των διακριτών program_term_id για κάθε φοιτητή, φιλτράροντας με τη συνθήκη να έχουν περισσότερες από μία εγγραφές, παρέχοντας έτσι μια εικόνα των φοιτητών με παράλληλες εγγραφές σε διάφορα προγράμματα σπουδών την ίδια ημερομηνία.

```
DELIMITER $$
CREATE PROCEDURE FindConcurrentEnrollments()
BEGIN
    SELECT e1.student_id, COUNT(DISTINCT e1.program_term_id) AS
concurrent_enrollments
    FROM Enrollment e1
    JOIN Enrollment e2 ON e1.student_id = e2.student_id
                       AND e1.enrollment_id <> e2.enrollment_id
                       AND e1.registration_date = e2.registration_date
    GROUP BY e1.student_id
    HAVING concurrent_enrollments > 1;
END$$
DELIMITER ;
-- ΠΑΡΑΔΕΙΓΜΑ
CALL FindConcurrentEnrollments();
```

Αποτέλεσμα

student_id	concurrent_enrollments
136	2
1098	2
1213	2
1958	2
2369	2
2646	2
3327	2

8. **Για ένα δοθέν όνομα Πανεπιστημίου, και χρονικό εύρος, αναφορά για τα προγράμματα σπουδών, τα τμήματα αυτών και τις σχετικές εγγραφές φοιτητών.**

Δίνει μία πλήρη κατανόηση για πληροφορίες που αφορούν τα πανεπιστήμια. Συγκεντρώνει και παρουσιάζει στοιχεία για τα προγραμμάτα σπουδών και τις εγγραφές σε αυτά, ανά κατηγορία, για ένα συγκεκριμένο πανεπιστήμιο και για ένα δοθέν χρονικό διάστημα. Συνδέει τους πίνακες University, Faculty, Program, Program_Term, και Enrollment για να φιλτράρει τα προγράμματα βάσει του ονόματος του πανεπιστημίου και των ημερομηνιών έναρξης και λήξης των προγραμμάτων. Μετρά τον αριθμό των τμημάτων και των εγγραφών τον φοιτητών.

```
DELIMITER $$
CREATE PROCEDURE `GenerateUniversityProgramReport`(
    IN `university_name` VARCHAR(255),
    IN `start_date` DATE,
    IN `end_date` DATE
)
BEGIN
    SELECT
        p.subject_type,
        COUNT(DISTINCT pt.program_term_id) AS number_of_program_terms,
        COUNT(e.enrollment_id) AS number_of_enrollments
    FROM
        University u
        JOIN Faculty f ON u.university_id = f.university_id
        JOIN Program p ON f.faculty_id = p.faculty_id
        JOIN Program_Term pt ON p.program_id = pt.program_id
        LEFT JOIN Enrollment e ON pt.program_term_id = e.program_term_id
    WHERE
        u.university_name = university_name
        AND pt.start_date >= start_date
        AND pt.end_date <= end_date
    GROUP BY
        p.subject_type;
    SELECT
        p.program_name,
        COUNT(DISTINCT pt.program_term_id) AS number_of_program_terms,
        COUNT(e.enrollment_id) AS registrations
    FROM
        University u
        JOIN Faculty f ON u.university_id = f.university_id
```

```
        JOIN Program p ON f.faculty_id = p.faculty_id
        JOIN Program_Term pt ON p.program_id = pt.program_id
        LEFT JOIN Enrollment e ON pt.program_term_id = e.program_term_id
    WHERE
        u.university_name = university_name
        AND pt.start_date >= start_date
        AND pt.end_date <= end_date
    GROUP BY
        p.program_name;
END$$
DELIMITER ;
-- ΠΑΡΑΔΕΙΓΜΑ
CALL GenerateUniversityProgramReport('University of Piraeus', '1997-01-01',
'2040-01-01');
```

Αποτέλεσμα 1

	subject_type	number_of_program_terms	number_of_enrollments
▶	Business & Finance	27	6377
	Technology	34	9585

Result 601 × Result 602

Αποτέλεσμα 2

	program_name	number_of_program_terms	registrations
▶	Associate Degree in Business Administration	1	170
	Associate Degree in Business and Finance	11	1755
	Associate Degree in Computer Science	1	169
	Associate Degree in Information Technology Managem...	1	160
	Associate Degree in Mobile App Development	1	150
	Associate Degree in Network Security	1	181

Result 601 Result 602 ×

6. Triggers

Τα triggers, είναι ειδικά scripts σε μια βάση δεδομένων που ενεργοποιούνται αυτόματα όταν συμβούν συγκεκριμένα γεγονότα, όπως εισαγωγές, ενημερώσεις ή διαγραφές εγγραφών. Είναι ιδιαίτερα χρήσιμα για την εγγύηση της ακεραιότητας των δεδομένων, την αυτοματοποίηση επαναλαμβανόμενων εργασιών και την εφαρμογή επιχειρησιακών κανόνων στη βάση δεδομένων χωρίς την ανάγκη παρέμβασης του χρήστη ή της εφαρμογής, εξασφαλίζοντας έτσι συνέπεια και αξιοπιστία στη διαχείριση των δεδομένων. Ξανά, αυτή η λειτουργία είναι πολύ χρήσιμη στην περίπτωση της βάσης δεδομένων του πανεπιστημίου καθώς μειώνει την ανάγκη για περαιτέρω ελέγχους και ενέργειες από τους διαχειριστές της βάσης δεδομένων, μειώνοντας ταυτόχρονα την πιθανότητα λάθους.

1. **Έλεγχος έγκυρου πεδίου email για τους φοιτητές.**

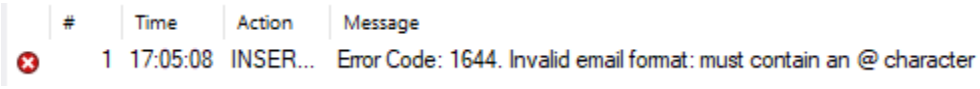
Το trigger ενεργοποιείται πριν από κάθε εισαγωγή εγγραφής στον πίνακα Student, ελέγχοντας αν το πεδίο email της νέας εγγραφής περιέχει τον χαρακτήρα @. Εάν το email δεν έχει την απαιτούμενη μορφή, το trigger διακόπτει την ενέργεια εισαγωγής και επιστρέφει μήνυμα λάθους, εξασφαλίζοντας ότι όλες οι διευθύνσεις email που εισάγονται στη βάση δεδομένων έχουν έγκυρη μορφή καθώς τα έγκυρα email των φοιτητών είναι κρίσιμα.

```
DELIMITER $$
CREATE TRIGGER BeforeStudentInsertOrUpdate
BEFORE INSERT ON Student
FOR EACH ROW
BEGIN
    IF NEW.email NOT LIKE '%@%' THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid email format: must
contain an @ character';
    END IF;
END$$
DELIMITER ;
```

Δοκιμή

```
INSERT INTO Student (student_id, first_name, last_name, email, date_of_birth)
VALUES (1, 'test', 'test', 'invalidemail.com', '2000-01-01');
```

Αποτέλεσμα



2. **Για να γίνει εισαγωγή φοιτητή σε μεταπτυχιακό πρόγραμμα θα πρέπει να υπάρχει εισαγωγή προπτυχιακού τίτλου σπουδών.**

Το trigger ελέγχει πριν από κάθε εισαγωγή στον πίνακα Enrollment αν ο φοιτητής που πρόκειται να εγγραφεί σε ένα πρόγραμμα μεταπτυχιακού ή διδακτορικού επιπέδου (Masters, Phd) έχει ήδη ολοκληρώσει ένα πρόγραμμα προπτυχιακού επιπέδου (Bachelors).

```
DELIMITER $$
CREATE TRIGGER BeforeEnrollInAdvancedProgram
BEFORE INSERT ON Enrollment
FOR EACH ROW
BEGIN
    DECLARE hasBachelors INT;
    SELECT COUNT(*) INTO hasBachelors
    FROM Enrollment e
    JOIN Program_Term pt ON e.program_term_id = pt.program_term_id
    JOIN Program p ON pt.program_id = p.program_id
    JOIN Degree d ON p.awarded_degree = d.degree_id
    JOIN EducationLevel el ON d.education_level_id = el.level_id
    WHERE e.student_id = NEW.student_id
    AND el.level_name = 'Bachelors';

    SELECT IF(el.level_name IN ('Masters', 'Phd'), TRUE, FALSE) INTO @isAdvanced
    FROM Program_Term pt
    JOIN Program p ON pt.program_id = p.program_id
    JOIN Degree d ON p.awarded_degree = d.degree_id
    JOIN EducationLevel el ON d.education_level_id = el.level_id
    WHERE pt.program_term_id = NEW.program_term_id;

    IF @isAdvanced AND hasBachelors = 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Student must be enrolled in a
Bachelor''s program before enrolling in a Master''s or PhD program.';
    END IF;
END$$
DELIMITER ;
```

Δοκιμή 1

```
INSERT INTO Student (student_id, first_name, last_name, email, date_of_birth)
VALUES (10500, 'test', 'test', 'test@example.com', '1995-03-15');
-- Ξέρουμε πως το program_term_id = 1 ισούται με ένα πρόγραμμα σπουδών που είναι μάστερ
INSERT INTO Enrollment (enrollment_id, student_id, program_term_id,
registration_date)
VALUES (50000, 10500, 1, '2050-01-01');
```

Αποτέλεσμα

#	Time	Action	Message
1	18:11:04	INSE...	1 row(s) affected
2	18:11:08	INSE...	Error Code: 1644. Student must be enrolled in a Bachelor's program before enrolling in a Master's or PhD program.

Δοκιμή 2

```
INSERT INTO Student (student_id, first_name, last_name, email, date_of_birth)
VALUES (10500, 'test', 'test', 'test@example.com', '1995-03-15');
-- Ξέρουμε πως το program_term_id = 4 ισούται με ένα πρόγραμμα σπουδών που είναι
bachelor
INSERT INTO Enrollment (enrollment_id, student_id, program_term_id,
registration_date)
VALUES (50000, 10500, 4, '2050-01-01');
-- Ξέρουμε πως το program_term_id = 1 ισούται με ένα πρόγραμμα σπουδών που είναι
μάστερ
INSERT INTO Enrollment (enrollment_id, student_id, program_term_id,
registration_date)
VALUES (50001, 10500, 1, '2050-01-01');
-- Ξέρουμε πως το program_term_id = 3 ισούται με ένα πρόγραμμα σπουδών που είναι
PhD
INSERT INTO Enrollment (enrollment_id, student_id, program_term_id,
registration_date)
VALUES (50002, 10500, 3, '2050-01-01');
```

Αποτέλεσμα

#	Time	Action	Message
1	18:26:20	INSE...	1 row(s) affected
2	18:26:23	INSE...	1 row(s) affected
3	18:26:26	INSE...	1 row(s) affected
4	18:27:22	INSE...	1 row(s) affected

3. Αύξηση αριθμού εργαζομένων εταιρείας με βάση τις εργασιακές εμπειρίες των φοιτητών.

Το trigger αυξάνει αυτόματα τον αριθμό των εργαζομένων σε μια εταιρεία κάθε φορά που προστίθεται μια νέα εγγραφή εργασιακής εμπειρίας στον πίνακα WorkExperience, εφόσων ο τύπος πρόσληψης της εργασιακής εμπειρίας είναι PERMANENT.

```
DELIMITER $$
CREATE TRIGGER IncrementEmployeeCountAfterWorkExperienceAdded
AFTER INSERT ON WorkExperience
FOR EACH ROW
BEGIN
    DECLARE jobType VARCHAR(255);
    SELECT jt.job_type INTO jobType
    FROM WorkExperience we
    JOIN JobTitle jt ON we.job_title_id = jt.title_id
    WHERE we.experience_id = NEW.experience_id;
    IF jobType = 'Permanent' THEN
        UPDATE Company
        SET employees = employees + 1
        WHERE company_id = NEW.company_id;
    END IF;
END$$
DELIMITER ;
```

Η εταιρεία με id = 123 πριν από το trigger

company_name	location_id	employees	industry
Μπελέκου, Μηλιώρης και Μπούκος	48	730	Telecommunications

Δοκιμαστική εισαγωγή του job title id = 15 που έχει job_type = permanent

```
INSERT INTO WorkExperience (experience_id, student_id, company_id, job_title_id,
start_date, end_date, description, responsibilities)
VALUES (10500, 1, 1, 15, '2022-01-01', '2022-12-31', 'Developed and maintained web
applications.', 'Responsible for back-end development');
```

Η ίδια εταιρεία με id = 1 ύστερα από την εισαγωγή της παραπάνω εντολής

	company_name	location_id	employees	industry
▶	Μπελέκου, Μηλιώρης και Μπούκος	48	731	Telecommunications

4. Χρήσιμα Triggers για μελλοντική ανάπτυξη που δεν έχουν υλοποιηθεί

1.Απαγόρευση εισαγωγής εγγραφής φοιτητή αν το όριο εγγραφών ανά τμήμα προγράμματος σπουδών έχει ξεπεραστεί. Σημαντικό! Για λόγους μαζικής εισαγωγής δεδομένων δεν υλοποιήθηκε.
2.Περιορισμός καταγραφής εργασιακών εμπειριών μόνο ύστερα από την ημερομηνία αποφοίτησης από ένα πρόγραμμα πανεπιστημίου Πειραιώς. (Σε περίπτωση που δεν ενδιαφέρουν οι προηγούμενες εργασιακές εμπειρίες φοιτητών)
3. Κάθε φοιτητής πρέπει να είναι άνω των 18 ετών.
4. Απαγόρευση εισαγωγής φοιτητή σε πρόγραμμα σπουδών εφόσον η ημερομηνία εγγραφής του είναι 60 ημερολογιακές ημέρες ύστερα από την ημερομηνία έναρξης του προγράμματος σπουδών. Σημαντικό! Για λόγους μαζικής εισαγωγής δεδομένων δεν υλοποιήθηκε.
5. Σε περίπτωση που θέλουμε η βάση να κρατάει πληροφορίες **μόνο** για αποφοίτους μπορούμε να δημιουργήσουμε ελέγχους ώστε κάθε enrollment να έχει ένα graduation και η ημερομηνία αποφοίτησης πρέπει να είναι πριν την σημερινή.
6. Απαγόρευση διπλότυπης εγγραφής σε ίδιο πρόγραμμα σπουδών εάν το student_id εμφανίζεται δύο φορές σε ένα program term. Σημαντικό! Για λόγους μαζικής εισαγωγής δεδομένων δεν υλοποιήθηκε.

7. Indexes (Ευρετήρια)

Για τη βελτιστοποίηση αναζήτησης και ταξινόμησης στη βάση δεδομένων μας, έχουμε επιλέξει την υλοποίηση ευρετηρίων με χρήση B-trees, τα οποία διαχειρίζονται αποτελεσματικά τις αναζητήσεις ισότητας, τις λειτουργίες ταξινόμησης και τα ερωτήματα εύρους (Comer, 1991). Χρήσιμο στα σενάρια αναζήτησης προγραμμάτων ή εργασιακών εμπειριών εντός συγκεκριμένων ημερομηνιών στους πίνακες Program και Work Experience. Παρόλο που η υποστήριξη των B-trees μπορεί να προσθέσει κάποια επιβάρυνση κατά τις ενημερώσεις - συμβάλλει σημαντικά στην επιτάχυνση των λειτουργιών αναζήτησης και ανάκτησης, κρίσιμο για πίνακες με συχνές αναζητήσεις όπως Student, Program, Enrollment, Faculty, κ.ά. Επιπλέον, τα B-trees υποστηρίζουν τη δημιουργία σύνθετων ευρετηρίων πάνω σε πολλαπλές στήλες, παρέχοντας πλεονεκτήματα σε πίνακες με πολλά ξένα κλειδιά όπως τα Program και Enrollment.

Στην βάση δεδομένων που έχουμε δημιουργήσει στην MySQL, το clustered ευρετήριο δημιουργείται αυτόματα για το πρωτεύον κλειδί του πίνακα. Όταν δημιουργείται ένα επιπλέον ευρετήριο με την εντολή **CREATE INDEX**, είναι αυτόματα non-clustered, B-tree, οπότε εφόσον δεν χρειαζόμαστε άλλο τύπου ευρετήριού (όπως FULLTEXT ή HASH), τότε απλά δημιουργούμε τα ευρετηρια indexes με την εντολη CREATE INDEX για την ταχύτερη αναζήτηση δεδομένων. Τα ευρετήρια μπορούν να δημιουργηθούν τόσο πριν όσο και μετά την εισαγωγή δεδομένων στην βάση. Στην προκειμένη περίπτωση δημιουργούνται εξ αρχής για να διασφαλίσει αποδοτική ανάκτηση δεδομένων από την αρχή.

Η επιλογή να προστεθούν ευρετήρια στις παρακάτω στήλες ακολουθεί τη λογική της βελτιστοποίησης της απόδοσης της βάσης δεδομένων για συχνές και κρίσιμες λειτουργίες ανάγνωσης και αναζήτησης, ενώ ταυτόχρονα διατηρεί μια ισορροπία για να αποφεύγεται η υπερβολική επιβάρυνση στις λειτουργίες εγγραφής και αλλαγής. Μικροί πίνακες όπως το Education Level, Location, Modules & Company δεν έχουν γίνει indexed καθώς δεν υπάρχει μεγάλο κόστος αναζήτησης του πίνακα με βάση το μέγεθος του και επίσης δεν χρησιμοποιείται τόσο συχνά όσο άλλα πεδία και πίνακες.

1. Πίνακας Student

Παρόλο που το ευρετήριο του πρωτεύοντα κλειδιού είναι αρκετός για πλειονότητα των ερωτημάτων που χρησιμοποιούν τον πίνακα student, θεωρούμε πως υπάρχει συχνά η ανάγκη οι διαχειριστές του συστήματος να ανακτούν το ονοματεπώνυμο ενός φοιτητή και την ηλεκτρονική του διεύθυνση, οπότε στον συγκεκριμένο πίνακα θα προσθέσουμε ευρετήρια στα παρακάτω πεδία για αυτόν τον πίνακα. Περαιτέρω, έχουμε δημιουργήσει

μοναδικό ευρετήριο στο πεδίο email γιατί εξασφαλίζει ότι δεν θα μπορούν να εισαχθούν δυο εγγραφές με το ίδιο email, διατηρώντας την ακεραιότητα των δεδομένων.

```
CREATE INDEX idx_student_last_name ON Student(last_name);
CREATE INDEX idx_student_first_name ON Student(first_name);
CREATE UNIQUE INDEX idx_student_email_unique ON Student(email);
```

2. **Πίνακας Enrollment**

Στον πίνακα enrollment θα δημιουργήσουμε ευρετήρια στα ξένα κλειδιά του πίνακα αλλά και στην ημερομηνία εγγραφής καθώς αυτό χρησιμοποιείται η ενδέχεται να χρησιμοποιηθεί σε αρκετά WHERE, ORDER BY, or GROUP BY clauses.

```
CREATE INDEX idx_enrollment_student ON Enrollment(student_id);
CREATE INDEX idx_enrollment_program_term ON Enrollment(program_term_id);
CREATE INDEX idx_enrollment_registration_date ON Enrollment(registration_date);
```

3. **Πίνακας Program Term**

Στον πίνακα Program Term θα δημιουργήσουμε ευρετήριο στο ξένο κλειδί με το πρόγραμμα σπουδών στο οποίο ανήκει για τις λειτουργίες JOIN κλπ. Επίσης θα δημιουργήσουμε ευρετήρια στην ημερομηνία έναρξης και λήξης το προγράμματος καθώς ενδέχεται να φανούν χρήσιμα για διάφορες ενέργειες επιλογής στην βάση.

```
CREATE INDEX idx_program_term_program ON Program_Term(program_id);
CREATE INDEX idx_program_term_start_date ON Program_Term(start_date);
CREATE INDEX idx_program_term_end_Date ON Program_Term(end_date);
```

4. **Πίνακας Program**

Ευρετήρια στα ξένα κλειδιά του πίνακα και στο program_name καθώς χρησιμοποιείτε συχνά.

```
CREATE INDEX idx_program_faculty ON Program(faculty_id);
CREATE INDEX idx_program_awarded_degree ON Program(awarded_degree);
CREATE INDEX idx_program_name ON Program(program_name);
```

5. Πίνακας Degree

Στον πίνακα degree θα δημιουργήσουμε ευρετήριο για το education level καθώς αυτό χρησιμοποιείται σαν ξένο κλείδι για ενώσεις με τον πίνακα Education Level.

```
CREATE INDEX idx_degree_education_level ON Degree(education_level_id);
```

6. Πίνακας Faculty

Ο πίνακας Faculty είναι ένας πίνακας που χρησιμοποιείται αρκετά συχνά στην βάση δεδομένων μας οπότε θα δημιουργήσουμε ευρετήρια για τα ξένα κλειδιά του πίνακα αλλά και για το όνομα του καθώς χρησιμοποιείται σε πολλούς τελεστές ισότητας σε διάφορα queries.

```
CREATE INDEX idx_faculty_university ON Faculty(university_id);
CREATE INDEX idx_faculty_location ON Faculty(location_id);
CREATE INDEX idx_faculty_name ON Faculty(faculty_name);
```

7. Πίνακας University

Εκτός από το ξένο κλειδί location_id, ένα ευρετήριο στο university_name μπορεί να βελτιώσει τις ερωτήσεις που φιλτράρουν βάσει ονόματος πανεπιστημίου κάτι το οποίο ενδέχεται να χρησιμοποιηθεί συχνά. Ο πίνακας είναι σχετικά μικρός οπότε αυτή η παραδοχή γίνεται για μελλοντικές περιπτώσεις που τα πανεπιστήμια θα είναι πιο πολλά.

```
CREATE INDEX idx_university_location ON University(location_id);
CREATE INDEX idx_university_name ON University(university_name);
```

8. Πίνακας Work Experience

Ο πίνακας work experience όπως και ο πίνακας enrollment είναι πίνακες που χρησιμοποιούνται πολύ συχνά για queries κυρίου ενδιαφέροντος. Επομένως, θα δημιουργήσουμε ευρετήρια στα ξένα του κλειδιά αλλά και στις ημερομηνίες καθώς αυτές χρησιμοποιούνται συχνά για να βρούμε αποφοίτους με εργασιακές εμπειρίες σε συγκεκριμένα διαστήματα.

```
CREATE INDEX idx_work_experience_student ON WorkExperience(student_id);
CREATE INDEX idx_work_experience_company ON WorkExperience(company_id);
CREATE INDEX idx_work_experience_job_title ON WorkExperience(job_title_id);
CREATE INDEX idx_work_experience_start_date ON WorkExperience(start_date);
CREATE INDEX idx_work_experience_end_date ON WorkExperience(end_date);
```

9. Πίνακας Job Title:

Καθώς χρησιμοποιούμε το πεδίο title_name σχετικά συχνά για διάφορα queries θα δημιουργήσουμε ένα ευρετήριο σε αυτό το πεδίο.

```
CREATE INDEX idx_job_title_name ON JobTitle(title_name);
```

10. Πίνακας Modules

Η συγκεκριμένη βάση δεδομένων δεν εχει πολλές εισαγωγές για τα modules που υπάρχουν σε κάθε πρόγραμμα σπουδών. Με βάση όμως την παραδοχή πως στο μέλλον ενδέχεται να προστεθούν πολλές εγγραφές μαθημάτων καθώς ένα πρόγραμμα σπουδών έχει πολλά μαθήματα, και πως ανακτούμε συχνά πληροφορίες για το όνομα του κάθε μαθήματος, θα δημιουργήσουμε τα παρακάτω ευρετήρια.

```
CREATE INDEX idx_module_program_term ON Modules(program_term_id);
CREATE INDEX idx_module_name ON Modules(module_name);
```

11. Πίνακας StudentModuleParticipation

```
CREATE INDEX idx_student_module_part_stud ON StudentModuleParticipation(student_id);
CREATE INDEX idx_student_module_part_mod ON StudentModuleParticipation(module_id);
```

12. Πίνακας graduation

Δημιουργούμε ευρετήρια στα ξένα κλειδιά του πίνακα αλλα και στον τελικό βαθμό καθώς το χρησιμοποιούμε σε διάφορα queries.

```
CREATE INDEX idx_graduation_enrollment ON Graduation(enrollment_id);
CREATE INDEX idx_graduation_final_grade ON Graduation(final_grade);
CREATE INDEX idx_graduation_final_grade ON Graduation(location_id);
```

13. Education Level, Location, Modules, Company

Για τους πίνακες αυτούς, δεν θα δημιουργηθούν εξτρά ευρετήρια καθώς το ευρετήριο που δημιουργείται αυτόματα στα πρωτεύοντα κλειδιά αρκεί, δεν συμβάλλουν σε σημαντικό όγκο query και είναι πίνακες με σχετικά μικρό όγκο δεδομένων.

8. Άλλες χρήσιμες λειτουργίες

ΡΟΛΟΙ

Για την εφαρμογή μας θεωρούμε χρήσιμη λειτουργία να υπάρχουν διαφορετικοί ρολόι για ενέργειες CRUD (Create, Read, Update, Delete) στους πίνακες της βάσης δεδομένων με βάση την αρμοδιότητα του κάθε χρήστη. Οι ρόλοι είναι χρήσιμοι για το σύστημα καθώς διασφαλίζουν την αποτελεσματική και ασφαλή διαχείριση των διαφόρων λειτουργικών τμημάτων της βάσης δεδομένων, ενισχύοντας τη διαφάνεια, την προσβασιμότητα και την ακεραιότητα των δεδομένων. Παρατήρη: Η βάση δεδομένων που χρησιμοποιούμε ονομάζεται **university_db**, αν δεν οριστεί δημιουργούνται priviledges για όλες τις βάσεις δεδομένων που υπάρχουν στον server MySQL.

1. **Administrator:** Ο ρόλος του Διαχειριστή είναι κρίσιμος για το σύστημα, καθώς παρέχει πλήρη πρόσβαση σε όλες τις λειτουργίες και τα δεδομένα της βάσης δεδομένων. Οι διαχειριστές μπορούν να προσθέτουν, να ενημερώνουν ή να διαγράφουν εγγραφές, να διαχειρίζονται λογαριασμούς χρηστών και να ρυθμίζουν τις παραμέτρους του συστήματος. Είναι υπεύθυνοι για τη συνολική συντήρηση και ασφάλεια της βάσης δεδομένων.

Δημιουργία ρόλου:

```
CREATE ROLE 'Administrator';
```

Παραχώρηση δικαιωμάτον:

```
GRANT ALL PRIVILEGES ON university_db.* TO 'Administrator';
```

2. **AcademicStaff:** Το Ακαδημαϊκό Προσωπικό αποτελείται από καθηγητές και διδάσκοντες που χρησιμοποιούν το σύστημα για να διαχειρίζονται τα μαθήματα. Έχουν τη δυνατότητα να καταχωρούν και να ενημερώνουν πληροφορίες σχετικά με τα μαθήματα, προκειμένου να παρέχουν στους φοιτητές τις απαραίτητες πληροφορίες και υποστήριξη.

```
CREATE ROLE 'AcademicStaff';
```

```
GRANT SELECT, INSERT, UPDATE ON university_db.`Modules` TO 'AcademicStaff';
```

3. **Analyst:** Οι Αναλυτές χρησιμοποιούν το σύστημα για να συλλέγουν και να αναλύουν δεδομένα, προκειμένου να κατανοήσουν καλύτερα την αποδοτικότητα των προγραμμάτων, την επιτυχία των φοιτητών και άλλες σημαντικές μετρήσεις. Τους παραχωρείται μόνο SELECT access.

```
GRANT SELECT ON university_db.`University` TO 'DataAnalyst';
GRANT SELECT ON university_db.`Faculty` TO 'DataAnalyst';
GRANT SELECT ON university_db.`Program` TO 'DataAnalyst';
GRANT SELECT ON university_db.`Modules` TO 'DataAnalyst';
GRANT SELECT ON university_db.`Student` TO 'DataAnalyst';
GRANT SELECT ON university_db.`Enrollment` TO 'DataAnalyst';
GRANT SELECT ON university_db.`Company` TO 'DataAnalyst';
GRANT SELECT ON university_db.`WorkExperience` TO 'DataAnalyst';
```

4. **CompanyPartnershipsOfficer:** Ο Υπεύθυνος Συνεργασιών με Εταιρείες διαχειρίζεται τις σχέσεις μεταξύ του εκπαιδευτικού ιδρύματος και των εταιρειών, επιδιώκοντας να δημιουργήσει ευκαιρίες για πρακτική άσκηση και εργασία για τους φοιτητές. Έχει πρόσβαση σε σχετικά δεδομένα για να διαχειρίζεται αυτές τις σχέσεις και να προωθεί τη συνεργασία.

```
GRANT SELECT, INSERT, UPDATE ON university_db.`Company` TO 'CompanyRepresentative';
GRANT SELECT, INSERT, UPDATE ON university_db.`WorkExperience` TO 'CompanyRepresentative';
```

5. **StudentOfficer (Γραμματεία):** Ο ρόλος της Γραμματείας συνδέεται με τη διαχείριση των φοιτητικών υποθέσεων, από εγγραφές φοιτητών, δεδομένα αυτών και δεδομένα σχετικά με την αποφοίτηση. Αυτός ο ρόλος είναι κρίσιμος για την ορθή λειτουργία του εκπαιδευτικού οργανισμού, καθώς εξασφαλίζει την ομαλή καταγραφή των φοιτητών μέσα στο σύστημα.

```
GRANT SELECT, INSERT, UPDATE ON university_db.`Student` TO 'StudentServicesOfficer';
GRANT SELECT, INSERT, UPDATE ON university_db.`Enrollment` TO 'StudentServicesOfficer';
GRANT SELECT, INSERT, UPDATE ON university_db.`Graduation` TO 'StudentServicesOfficer';
```

Ύστερα από κάποια δημιουργία των δικαιωμάτων, θα κάνουμε assign αυτούς τους ρόλους σε MySQL λογαριασμούς για τον κάθε χρήστη. Για παράδειγμα, για τον διαχειριστή:

```
GRANT 'Administrator' TO 'username'@'host';
FLUSH PRIVILEGES; // Για την εφαρμογή αλλαγών των δικαιωμάτων στην βάση.
```

Επαναλαμβάνουμε αυτή τη διαδικασία για όλους τους χρήστες στους οποίους θέλουμε να δώσουμε τα ανάλογα δικαιώματα στην βάση.

Η παραπάνω δομή παρέχει ένα σύστημα για role-based πρόσβαση, ανάκτηση και επεξεργασία των δεδομένων το οποίο κάνει το σύστημα πιο αποδοτικό, διαχειρίσιμο, ευελικτο και ασφαλές.

VIEWS

Στη βάση δεδομένων, οι προβολές (views) αποτελούν ένα ισχυρό εργαλείο που επιτρέπει στους χρήστες να δημιουργούν εικονικούς πίνακες βασισμένους σε ερωτήματα SQL πάνω σε έναν ή περισσότερους πραγματικούς πίνακες. Αυτές οι εικονικές απεικονίσεις δεδομένων δεν αποθηκεύουν φυσικά δεδομένα, αλλά παρέχουν ένα δυναμικό στρώμα αφαίρεσης, επιτρέποντας την προσαρμοσμένη πρόσβαση και την απλοποιημένη προβολή των δεδομένων. Αυτό είναι ιδιαίτερα χρήσιμο σε περίπλοκες βάσεις δεδομένων, όπως αυτή που περιγράφεται, όπου η διαχείριση διαφορετικών επιπέδων δεδομένων και η ερμηνεία σχεσιακών δεδομένων μπορεί να γίνει περίπλοκη. Οι προβολές διευκολύνουν την ανάκτηση, την ανάλυση και την παρουσίαση των δεδομένων, επιτρέποντας στους χρήστες να επικεντρωθούν σε συγκεκριμένα υποσύνολα δεδομένων. Επιπλέον, βοηθούν στην ενίσχυση της ασφάλειας δεδομένων περιορίζοντας την πρόσβαση σε ευαίσθητα δεδομένα, καθώς μπορούν να παρέχουν πρόσβαση μόνο στα απαραίτητα στοιχεία, χωρίς να εκθέτουν το σύνολο των υποκειμένων δεδομένων.

1. **VIEW για φοιτητές χωρίς συσχέτιση με το πανεπιστήμιο Πειραιώς.**

Όπως αναφέραμε και στις παραδοχές αρχικά, εφόσον έχουμε ως επίκεντρο αποφοίτους του πανεπιστημίου Πειραιώς, τότε θεωρούμε πως μια εγγραφή σε πρόγραμμα σπουδών για κάθε φοιτητή θα έπρεπε να σχετίζεται με το πανεπιστήμιο Πειραιά. Αυτό δεν γίνεται να υλοποιηθεί μέσω trigger εφόσον το προπτυχιακό πρόγραμμα μπορεί να είναι από άλλο πανεπιστήμιο και αυτό δεν θα μας επέτρεπε την εισαγωγή δεδομένων στην βάση. Όποτε έχουμε δημιουργήσει ένα VIEW το οποίο καταχωρεί φοιτητές οι οποίοι **ΔΕΝ** έχουν συσχέτιση με πρόγραμμα σπουδών από το πανεπιστήμιο Πειραιά. Για ανάδειξη της λειτουργίας αυτής, από τις 10,000 εισαγωγές φοιτητών ο ένας δεν έχει συσχέτιση με το πανεπιστήμιο Πειραιώς.

```
CREATE VIEW NotAffiliatedWithUniPi AS
SELECT s.student_id, s.first_name, s.last_name
FROM Student s
WHERE NOT EXISTS (
    SELECT 1
    FROM Enrollment e
    JOIN Program_Term pt ON e.program_term_id = pt.program_term_id
    JOIN Program p ON pt.program_id = p.program_id
    JOIN Faculty f ON p.faculty_id = f.faculty_id
    JOIN University u ON f.university_id = u.university_id
    WHERE s.student_id = e.student_id
    AND u.university_name = 'University of Piraeus'
```

```
);
```

```
SELECT * FROM NotAffiliatedWithUniPi;
```

2. **VIEW Για προγράμματα σπουδών ανα έτος και πλήθος τμημάτων αυτών μεταξυ 2013 και 2023**

Προσφέρει στους διαχειριστές των εκπαιδευτικών ιδρυμάτων, στους εκπαιδευτικούς, αλλά και στους φοιτητές να έχουν μια σαφή εικόνα της πορείας και της εξέλιξης των προγραμμάτων μέσα στο χρόνο.

```
CREATE VIEW ProgramTermsPerYear AS
SELECT p.program_id, p.program_name, YEAR(pt.start_date) AS year,
COUNT(pt.program_term_id) AS term_count
FROM Program p
JOIN Program_Term pt ON p.program_id = pt.program_id
WHERE YEAR(pt.start_date) BETWEEN 2013 AND 2023
GROUP BY p.program_id, p.program_name, YEAR(pt.start_date)
ORDER BY YEAR(pt.start_date) ASC, p.program_id;
```

```
SELECT * FROM ProgramTermsPerYear;
```

3. **VIEW πλήθους προγραμμάτων που προσφέρεται από κάθε τμήμα πανεπιστημίου**

Παρέχει μια συνοπτική εικόνα του αριθμού των προγραμμάτων που προσφέρει κάθε Τμήμα ενός Εκπαιδευτικού Ιδρύματος.

```
CREATE VIEW FacultyProgramsCountView AS
SELECT f.faculty_id, f.faculty_name, COUNT(p.program_id) AS program_count
FROM Faculty f
JOIN Program p ON f.faculty_id = p.faculty_id
GROUP BY f.faculty_id, f.faculty_name;
```

```
SELECT * FROM FacultyProgramsCountView;
```

4. **VIEW πλήθους εργασιακών εμπειριών ανά εταιρεία.**

Παρέχει μια αναλυτική εικόνα της συνολικής εργασιακής εμπειρίας που προσφέρει κάθε εταιρεία. Διευκολύνει την αναγνώριση των εταιρειών με την μεγαλύτερη συνεισφορά στην πρακτική εκπαίδευση και εξέλιξη των φοιτητών.

```
CREATE VIEW CompanyExperienceCountView AS
SELECT c.company_id, c.company_name, c.industry, COUNT(we.experience_id) AS
total_experiences
FROM Company c
LEFT JOIN WorkExperience we ON c.company_id = we.company_id
GROUP BY c.company_id, c.company_name, c.industry
ORDER BY total_experiences DESC;
```

```
SELECT * FROM CompanyExperienceCountView;
```

9. Βιβλιογραφία

Βασιλακόπουλος, Γ. (2023). Σχεδιασμός Βάσεων Δεδομένων, Δεύτερη Έκδοση. Αθήνα: Εκδόσεις Τσότρας.

Comer, D. (1991). The ubiquitous B-tree. Στο A. Silberschatz, M. Stonebraker και J. D. Ullman (επιμ) Readings in database systems. MIT Press.

Robinson, A. (2020). Understanding crow's foot notation. Database Guides. https://www.databaseguides.com/understanding-crows-foot-notation