



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

7ο εξάμηνο, Ροή Α, ΗΜΜΥ

Μία Επιχειρησιακή Εφαρμογή για τον Ακαδημαϊκό Κόσμο

Σταυρακάκης Χρήστος, 03106165
Τσιούρης Ιωάννης, 03106193
Πάρης Συμινελάκης, 03106185

Η/Μ Παράδοσης: 23 Φεβρουαρίου 2010

Αθήνα, 2010

Περιεχόμενα

1	Εισαγωγή	2
2	Περιβάλλον ανάπτυξης - Εργαλεία	2
3	Η αρχιτεκτονική της βάσης	4
4	Σχεδιασμός της Βάσης	7
5	Ερωτήσεις SQL	9
5.1	Τδρυμα	9
5.2	Σχολή	10
5.3	Ερευνητής	10
5.4	Χρηματοδότης	12
5.5	Project	12
5.6	Δημοσίευση	13
6	Παράρτημα	14
6.1	Makefile	14
6.2	DDL	14

1 Εισαγωγή

Στην άσκηση αυτή ζητείται ο σχεδιασμός μιας επιχειρησιακής εφαρμογής με σκοπό τη διαχείριση της βάσης δεδομένων (που σχεδιάσαμε στην άσκηση 1) για τον Ακαδημαϊκό Κόσμο. Η βάση περιλαμβάνει πληροφορίες για ερευνητές, ακαδημαϊκά ιδρύματα και συνεργασίες μεταξύ ερευνητών.

2 Περιβάλλον ανάπτυξης - Εργαλεία

Όλη η ανάπτυξη της εφαρμογής έγινε σε GNU/Linux λειτουργικό σύστημα (Debian και Ubuntu), με χρήση της PostgreSQL για τη διαχείριση της βάσης (DBMS) και της Python ως γλώσσα προγραμματισμού για την διαχείριση των queries και των αποτελεσμάτων τους και την ανάπτυξη του γραφικού περιβάλλοντος της εφαρμογής.

Επιλέξαμε να δουλέψουμε σε **GNU/Linux** λειτουργικό σύστημα όχι μόνο λόγω της πληθώρας επιλογών που προσφέρει σε επίπεδο εφαρμογών και της ευκολίας εγκατάστασης και παραμετροποίησης αυτών (που απορρέουν από την κεντρική διαχείριση πακέτων που υποστηρίζουν τα περισσότερα τέτοια λειτουργικά) αλλά κυρίως για την γενικότερη φιλοσοφία του Ελεύθερου Λογισμικού (Free Software) και Λογισμικού Ανοιχτού Κώδικα (Open Source Software) που αυτά υποστηρίζουν και προωθούν¹.



Σχήμα 1: Μερικά από τα εργαλεία που χρησιμοποιήθηκαν

Η **PostgreSQL** αποτελεί μία ανοιχτού κώδικα σχεσιακή βάση δεδομένων (Object-Relational Database Management System - ORDBMS). Υποστηρίζει τα περισσότερα κομμάτια του SQL 2003 προτύπου καθώς και μια πλειάδα από δικά της χαρακτηριστικά και επεκτάσεις (όπως ο ορισμός data types, operators και functions). Εκτός από την υποστήριξη referential integrity και προηγμένη διαχείριση transaction, η PostgreSQL διαθέτει τη δυνατότητα ορισμού triggers και κανόνων για τον έλεγχο πρόσβασης σε αντικείμενα της βάσης δεδομένων. Επιπλέον, παρέχει υποστήριξη Foreign Keys και CHECK constraints καθώς και πλήρη υποστήριξη για UTF8. Στα συν της το ότι παρέχεται χωρίς κόστος άδειας χρήσης καθώς διανέμεται υπό την ελεύθερη και ανοιχτού κώδικα άδεια του Berkeley (BSD license²). Για το administration της βάσης, μετά τη δημιουργία του σχήματος και το bulk loading

¹Χρήσιμα links:

- GNU: <http://www.gnu.org/>
- Free Software Foundation: <http://www.fsf.org/>
- Debian Manifesto: <http://www.debian.org/doc/manuals/project-history/ap-manifesto.en.html>
- Open Source Initiative: <http://www.opensource.org/>

²BSD license: http://en.wikipedia.org/wiki/BSD_license

των δεδομένων, χρησιμοποιήσαμε τον interpreter της PostgreSQL και το *phpPgAdmin* το οποίο είναι μια εφαρμογή γραμμένη σε php με web interface και παρέχει την ευκολία ενός GUI. Ένα από τα αρνητικά αυτού του RDBMS είναι η μειωμένη απόδοσή του, σε σχέση για παράδειγμα με τη MySQL, γεγονός που μπορεί να παρατηρηθεί μόνο όταν το μέγεθος της βάσης ή το traffic στο server αυξηθεί σημαντικά (κάτι που προφανώς δεν συμβαίνει στην περίπτωση μας).

Η γλώσσα προγραμματισμού αποτελεί σημαντικό κομμάτι του project καθώς ο περισσότερος κώδικας θα γράφονταν σε αυτήν. Για το λόγο αυτό θέλαμε μία "εύκολη", γρήγορη και, σίγουρα, αντικειμενοστραφή γλώσσα. Κρίναμε πως η *Python* ήταν η καλύτερη επιλογή. Η Python είναι μία γλώσσα πολύ υψηλού επιπέδου, πλήρως αντικειμενοστραφής με πολλά στοιχεία scripting που δίνει ιδιαίτερη έμφαση στην αναγνωσιμότητα κώδικα (code readability³). Προσφέρει μεγάλη ευκολία καθώς η standard library της περιλαμβάνει πολλές έτοιμες δομές και εργαλεία και, επιπλέον, λόγω του modularity που χαρακτηρίζει τη γλώσσα, υπάρχει εύκολη διασύνδεση με μεγαλύτερες βιβλιοθήκες και modules συγκεκριμένου σκοπού ("batteries included" φιλοσοφία της Python). Γι' αυτό προτιμάται μέρα με τη μέρα και περισσότερο τόσο για Web και GUI development όσο και για την ανάπτυξη γενικής χρήσεως εφαρμογών και όλα δείχνουν ότι θα κυριαρχήσει στο μέλλον στη "μάχη" των γλωσσών προγραμματισμού για το μερίδιο της αγοράς. Η Python διανέμεται υπό μία BSD-style license την Python Software Foundation License (PSFL⁴) η οποία είναι συμβατή και με την GPL.

Απαραίτητο εργαλείο αποτέλεσε το *Psycopg2* module (Python-PostgreSQL Database Adapter) το οποίο είναι ουσιαστικά ένας "προσαρμογέας"/"driver" ώστε να μπορούμε να διατυπώνουμε ερωτήματα SQL και να διαχειριζόμαστε αποτελέσματα από μία βάση μέσω της PostgreSQL που είναι ήδη εγκατεστημένη στο σύστημα. Το psycopg2 είναι πλήρως συμβατό με το Python DBAPI 2.0⁵ (για την ακρίβεια υλοποιεί αυτό το API) κι έτσι προσφέρει πολύ καλή διασύνδεση με τα builtin data types της Python. Σχεδιάστηκε ώστε να υποστηρίζει με ασφάλεια παράλληλη εκτέλεση INSERTs ή/και UPDATEs από διαφορετικές εφαρμογές (thread safe). Και αυτό το module διατίθεται υπό άδεια Ελεύθερου Λογισμικού (GPLv2⁶).

Εν κατακλείδι, ένα από τα πιο βασικά συστατικά που συνετέλεσε στην ευκολία ανάπτυξης και μετέπειτα χρήσης του project ήταν το *Qt Framework*. Το Qt είναι ένα ανεξάρτητο-λειτουργικού-συστήματος (cross-platform) πλαίσιο ανάπτυξης εφαρμογών (applications) και διεπαφών χρήστη (User Interfaces - UIs) το οποίο αναπτύχθηκε και υποστηρίζεται από την Trolltech (που πλέον ανήκει στην Nokia). Ουσιαστικά αποτελεί μία τεράστια βιβλιοθήκη ή καλύτερα μια σειρά βιβλιοθηκών που αφορούν ανάπτυξη GUI (Advanced GUI), 3D Γραφικών με OpenGL (3D Graphics with OpenGL), Πολυνηματικών Εφαρμογών (Multithreading), Εφαρμογών για Ενσωματωμένα Συστήματα (Embedded Windowing System), 2D Γραφικών (2D Graphics), Multimedia Εφαρμογών (Multimedia Framework), Διεπαφή Διεπικοινωνίας Αντικειμένων (Inter-Object Communication), Webkit Integration, Εφαρμογών δικτύου (Network Connectivity), Εφαρμογών XML, Διεπαφή για Μηχανή Σεναρίων (Scripting Engine), Διεπαφή για Βάση Δεδομένων (DB) κ.ά. Το καλό με το συγκεκριμένο framework είναι ότι υποστηρίζει Python bindings εκτός των αρχικών C++ bindings μέσω του *PyQt*. Στην εργασία αυτή χρησιμοποιήσαμε εκτενώς τα modules PyQt4.QtGui και PyQt4.QtCore για την ανάπτυξη του GUI της εφαρμογής (Widget Toolkit). Επιπλέον, η Nokia διαθέτει το Qt και υπό ελεύθερη άδεια: Qt Open Source Edition License⁷ για να εκφράσει την υποστήριξη της σε έργα Ελεύθερου Λογισμικού. Στα αρνητικά της επιλογής του Qt το γεγονός ότι αυτό αποτελεί ένα τεράστιο, πλήρες framework πολλών χρήσεων το οποίο υλοποιεί ξανά πράγματα τα οποία η Python ενσωματώνει εγγενώς στη standard library της κι επομένως είναι δύσκολο να κρατηθούν οι ισορροπίες μεταξύ χρήσης Qt και

³ Απαραίτητο στοιχείο όταν πρόκειται για project (σχετικά) μεγάλης έκτασης που γράφουν 3 άτομα ταυτόχρονα :-)

⁴ http://en.wikipedia.org/wiki/Python_Software_Foundation_License

⁵ <http://www.python.org/dev/peps/pep-0249/>

⁶ http://en.wikipedia.org/wiki/Gplv2#Version_2

⁷ <http://qt.nokia.com/products/licensing>

Python (π.χ. το Qt υλοποιεί διεπαφή για επικοινωνία με βάση δεδομένων). Αυτό το πρόβλημα αντιμετωπίστηκε συμφωνώντας εξ αρχής ότι θα χρησιμοποιηθούν τα δύο προαναφερθέντα module του PyQt και τίποτα παραπάνω.

Όλη η εργασία γράφηκε με χρήση του *VIM* editor, κυρίως λόγω της ευκολίας που αυτός παρέχει μέσω του syntax highlight (που διαθέτει by-default για πάρα πολλές γλώσσες προγραμματισμού), των διαφόρων views (π.χ. split, vertical split κλπ), της πληθώρας από keybindings (που διευκολύνουν και επιταχύνουν σημαντικά τη συγγραφή κώδικα), της αλληλεπίδρασής του με πολλά unix tools (π.χ. zsh, screen, sed) κ.ά.

3 Η αρχιτεκτονική της βάσης

Η βάση που επιλέξαμε τελικά να υλοποιήσουμε (ονόματι acad_world) είναι αυτή που περιγράφηκε στην αναφορά της Άσκησης 1 τόσο σε μοντέλο Εκτεταμένων Συσχετίσεων-Οντοτήτων (Extended E-R Model) όσο και σε Σχεσιακό Μοντέλο (Relational Model).

Οι πίνακες της βάσης μας μαζί με τα βασικά τους γνωρίσματα αναφέρονται, για επανάληψη, και παρακάτω:

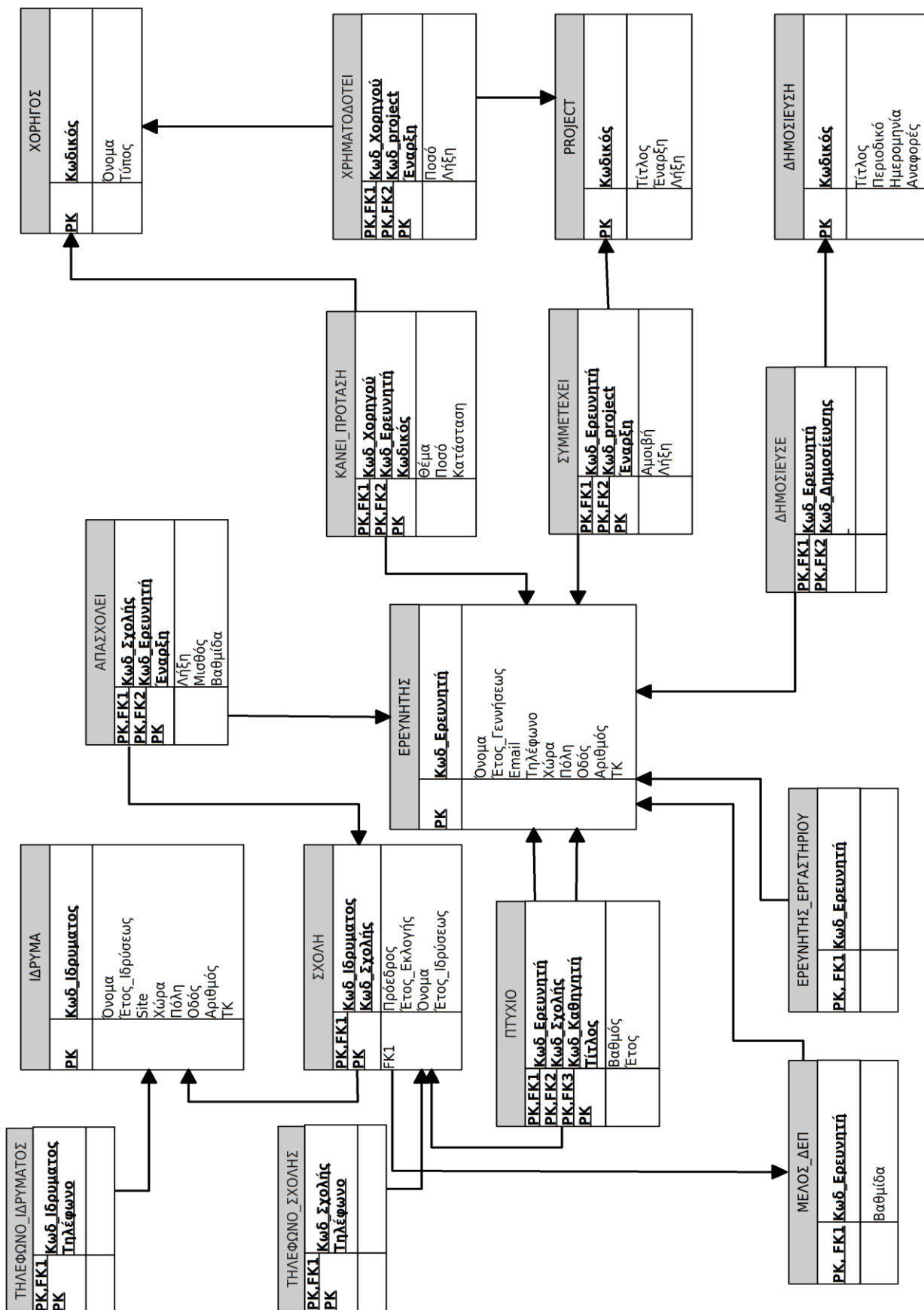
- **ΙΑΡΥΜΑ** (Κωδ_Ιδρύματος, Όνομα, Έτος_Ιδρύσεως, Site, Χώρα, Πόλη, Οδός, Αριθμός, TK)
- **ΣΧΟΛΗ** (Κωδ_Ιδρύματος, Κωδ_Σχολής, Πρόεδρος, Έτος_Εκλογής, Όνομα, Έτος_Ιδρύσεως)
- **ΕΡΕΥΝΗΤΗΣ** (Κωδ_Ερευνητή, Όνομα, Έτος_Γεννήσεως, Email, Τηλέφωνο, Χώρα, Πόλη, Οδός, Αριθμός, TK)
- **ΜΕΛΟΣ_ΔΕΠ** (Κωδ_Ερευνητή)
- **ΕΡΕΥΝΗΤΗΣ_ΕΡΓΑΣΤΗΡΙΟΥ** (Κωδ_Ερευνητή, Βαθμίδα)
- **ΠΤΥΧΙΟ** (Κωδ_Ερευνητή, Κωδ_Σχολής, Κωδ_Καθηγητή, Τίτλος, Βαθμός, Έτος_Παραλαβής)
- **ΧΟΡΗΓΟΣ** (Κωδικός, Όνομα, Τύπος)
- **PROJECT** (Κωδικός, Τίτλος, Έναρξη, Λήξη)
- **ΔΗΜΟΣΙΕΥΣΗ** (Κωδικός, Τίτλος, Περιοδικό, Ημερομηνία, Αναφορές)
- **ΑΠΑΣΧΟΛΕΙ** (Κωδ_Ιδρύματος, Κωδ_Σχολής, Έναρξη, Λήξη, Μισθός, Βαθμίδα)
- **ΣΥΜΜΕΤΕΧΕΙ** (Κωδ_Ερευνητή, Κωδ_Project, Έναρξη, Αμοιβή, Λήξη)
- **ΧΡΗΜΑΤΟΔΟΤΕΙ** (Κωδ_Χορηγού, Κωδ_Project, Έναρξη, Ποσό, Λήξη)
- **ΚΑΝΕΙ_ΠΡΟΤΑΣΗ** (Κωδ_Χορηγού, Κωδ_Ερευνητή, Κωδικός, Θέμα, Ποσό, Κατάσταση)
- **ΔΗΜΟΣΙΕΥΣΕ** (Κωδ_Ερευνητή, Κωδ_Δημοσίευσης)
- **ΤΗΛΕΦΩΝΟ_ΙΑΡΥΜΑΤΟΣ** (Κωδ_Ιδρύματος, Τηλέφωνο)
- **ΤΗΛΕΦΩΝΟ_ΣΧΟΛΗΣ** (Κωδ_Σχολής, Τηλέφωνο)

Υπενθυμίζουμε ότι το όνομα κάθε σχέσης σημειώνεται στην αρχή με κεφαλαία έντονα γράμματα και το αντίστοιχο πρωτεύον κλειδί με έντονα υπογραμμισμένα γράμματα. Αναλυτική περιγραφή και αιτιολόγηση των ιδιοτήτων κάθε σχέσης είναι διαθέσιμη στην παράγραφο ΣΧΕΣΙΑΚΟ ΜΟΝΤΕΛΟ της 1ης Άσκησης.

Παραθέτουμε και το Σχεσιακό Μοντέλο⁸ (Σχήμα 2) για να είναι πιο εύκολη η ανάγνωση της παραγράφου "Σχεδιασμός της βάσης" που ακολουθεί (Παράγραφος 4).

⁸Με κάποιες διορθώσεις καθώς αυτό της Άσκησης 1 είχε μερικά λάθη ως προς τη φορά των βελών που εκφράζουν τις εξαρτήσεις μεταξύ των πινάκων

Σχήμα 2: Relation Model



4 Σχεδιασμός της Βάσης

Στο σημείο αυτό θα παρουσιάσουμε τις επιλογές μας ως προς το σχεδιασμό της βάσης δεδομένων. Οι επιλογές αυτές αφορούν την δημιουργία πινάκων, τύπων δεδομένων, πεδίων τιμών, περιορισμών, συναρτήσεων και όψεων. Κάθε επιλογή έγινε με γνώμονα την ακεραιότητα και συνέπεια της βάσης δεδομένων καθώς και της προσφερόμενης χρηστικότητας. Η παρουσίαση των επιλογών θα γίνει αναλύοντας δύο από τους πιο αντιπροσωπευτικούς πίνακες της βάσης δεδομένων μας. Οι υπόλοιποι πίνακες ακολουθούν την ίδια προσέγγιση και οι λεπτομερείς υλοποίησης τους μπορούν να βρεθούν στο παράρτημα.

Ο πρώτος πίνακας που θα περιγραφεί αφορά στην σχέση Ερευνητής. Το σχήμα αυτού του πίνακα δημιουργείται με τις παρακάτω SQL εντολές :

```
CREATE TABLE researcher (
    rescode serial PRIMARY KEY,
    name fullname NOT NULL,
    phone VARCHAR(20) ,
    site VARCHAR(60) ,
    email VARCHAR(60) ,
    loc location NOT NULL,
    addr address
);
```

Όπως φαίνεται το πρωτεύον κλειδί αυτής της σχέσης είναι ο κωδικός ερευνητή (rescode). Για τους κωδικούς ερευνητή έχει επιλεγεί ο "ψευδο-τύπος" δεδομένων serial. Στην πραγματικότητα η χρήση του είναι ανάλογη με την δημιουργία ενός μοναδικού αναγνωριστή στήλης και θυμίζει την επιλογή auto-increment άλλων συστημάτων διαχείρισης βάσεων δεδομένων. Σημειώνουμε ότι κατά την εισαγωγή μιας πλειάδας, επιλέγεται η επιλογή default για τις στήλες τύπου serial.

Στην συνέχεια υπάρχει το όνομα του ερευνητή. Ο τύπος δεδομένων αυτής της στήλης είναι fullname. Αυτός δεν είναι ένας ενσωματωμένος τύπος της PostgreSQL αλλά ένας τύπος δεδομένων που εμείς δημιουργήσαμε. Η δημιουργία του επιτυγχάνεται με τις εξής εντολές :

```
CREATE TYPE fullname AS (
    firstn VARCHAR(20) ,
    lastn VARCHAR(20)
);
```

Αυτός ο τύπος δεδομένων αποτελείται από 2 στήλες (firstn και lastn) τύπου μεταβλητού μεγέθους συμβολοσειράς, με μέγιστο μέγεθος 20. Ακόμα για το όνομα του ερευνητή έχει χρησιμοποιηθεί ο περιορισμός NOT NULL, καθώς δεν έχει νόημα η εισαγωγή ενός ερευνητή αν δεν είναι γνωστό το όνομα του.

Ξεχωριστός τύπος δεδομένων έχει δημιουργηθεί και για τις στήλες loc και addr που αντιπροσωπεύουν την Τοποθεσία του ερευνητή (Χώρα,Πόλη) και τη διεύθυνση του αντίστοιχα. Οι τύποι αυτοί ορίζονται ως εξής:

```
CREATE TYPE location AS (
    country VARCHAR(50) ,
    city VARCHAR(50)
);
```

```
CREATE TYPE address AS (
    street VARCHAR(50) ,
```



```

num INTEGER,
zipcode VARCHAR(10)
);

```

Η σημασία τους είναι προφανής. Για την τοποθεσία (loc) έχει χρησιμοποιηθεί επίσης ο περιορισμός NOT NULL καθώς αποτελεί σημαντική πληροφορία για έναν ερευνητή.

Στην συνέχεια ακολουθεί η παρουσίαση της σχέσης απασχολεί (occupy). Η σχέση αυτή παρουσιάζει το ποιος ερευνητής εργάζεται ή έχει εργαστεί σε ποια σχολή. Ο πίνακας αυτής της σχέσης δημιουργείται με την εξής εντολή:

```

CREATE TABLE occupy (
    schcode integer REFERENCES school(schcode) ON DELETE
    CASCADE ON UPDATE
    CASCADE,
    rescode integer REFERENCES researcher(rescode) ON
    DELETE CASCADE ON UPDATE CASCADE,
    started date NOT NULL,
    ended date NOT NULL,
    salary INTEGER NOT NULL,
    rank VARCHAR(20) NOT NULL,
    PRIMARY KEY(schcode , rescode , started) ,
    CHECK ( started < ended )
);

```

Όπως φαίνεται το πρωτεύον κλειδί αυτής της σχέσης είναι ο συνδυασμός των γνωρισμάτων schcode, rescode, started που αντιστοιχούν στον κωδικό σχολής, κωδικό ερευνητή και ημερομηνία έναρξης εργασίας. Η επιλογή αυτή επιβάλλεται καθώς κάποιος ερευνητής μπορεί να έχει εργαστεί σε πολλές διαφορετικές σχολές σε διαφορετικές χρονικές περιόδους.

Τα πεδία schcode και rescode αποτελούν ξένα κλειδιά. Το schcode αναφέρεται στο schcode της σχέσης school και το rescode στο rescode της σχέσης researcher. Για να μην δημιουργείται πρόβλημα με τους περιορισμούς αναφοράς, έχει επιλέγει όταν διαγράφονται (ON DELETE) ή ενημερώνονται (ON UPDATE) οι αντίστοιχες πλειάδες στις αντίστοιχες σχέσεις να διαγράφεται ή ενημερώνεται και η σχέση απασχολεί (επιλογή CASCADE). Ο λόγος που επιλέξαμε να γίνεται CASCADE είναι γιατί δεν έχει νόημα να διατηρείται ένα ιστορικό εργασίας όταν δεν υπάρχει η αντίστοιχη πληροφορία για τον ερευνητή ή τη σχολή που εμπλέκεται.

Τα πεδία started, ended, salary και rank έχουν τον περιορισμό ακεραιότητας NOT NULL. Η επιλογή αυτή είναι υποχρεωτική (και αυτονόητη) για το started αφού αποτελεί μέρος πρωτεύουν κλειδιού. Για τα υπόλοιπα πεδία, υπαγορεύεται από τη σημασία των πεδίων. Τα πεδία αυτά περιέχουν απαραίτητες πληροφορίες για ένα ιστορικό εργασίας.

Τέλος έχουμε δημιουργήσει ένα CHECK constraint, το CHECK (started<ended). Ο έλεγχος αυτός επιβάλλεται από τη σημασιολογία καθώς δεν μπορεί η ημερομηνία έναρξης εργασίας να είναι πιο πριν από την ημερομηνία λήξης.

Ακόμα, κατά το σχεδιασμό της βάσης δεδομένων κρίθηκε απαραίτητη η δημιουργία τριών όψεων με σκοπό την διευκόλυνση της διατύπωσης των ερωτημάτων και της "δουλειάς" του optimizer.

Η πρώτη όψη μας είναι η εξής:

```

CREATE VIEW institution_schools as
SELECT ins.inscode , ins.name AS ins_name
    ,sch.schcode ,sch.name AS sch_name
FROM institution AS ins

```

```
JOIN school AS sch ON (ins.inscode=sch.inscode)
```

Η όψη αυτή αποτελείται από την συνένωση των πινάκων `institution` και `school` στο γνώρισμα `inscode` και την προβολή ορισμένων γνωρισμάτων τους. Στην ουσία χρησιμοποιείται ώστε να υπάρχει σε έναν πίνακα ο κωδικός και το όνομα των ιδρυμάτων και των σχολών που ανήκουν σε αυτά.

Η δεύτερη όψη είναι η εξής :

```
CREATE VIEW school_occupys AS
SELECT sch.schcode , sch.name , occ.rescode , occ.sallary ,
      occ.rank , occ.started , occ.ended
FROM school AS sch JOIN occupy AS occ
ON sch.schcode=occ.schcode
```

Στην όψη αυτή συνενώνονται οι πίνακες `school` και `occupy`. Στην όψη αυτή παρουσιάζεται ο κωδικός και το όνομα μιας σχολής, μαζί με πληροφορίες για όσους έχουν εργαστεί σε αυτή τη σχολή.

Και η τελευταία μας όψη είναι η εξής :

```
CREATE VIEW sponsor_funds_project AS
SELECT spon_occ.sponcode , spon_occ.name ,
      spon_occ.amount , proj.title
FROM ( SELECT sponsor.sponcode , sponsor.name ,
      funds.projcode , funds.amount
      FROM sponsor JOIN funds
      ON sponsor.sponcode=funds.sponcode ) AS spon_occ
NATURAL JOIN project AS proj;
```

Σε αυτή την όψη γίνεται έμμεση συνένωση τριών πινάκων. Στην όψη αυτή υπάρχει πληροφορία για κάθε χορηγό και για τα projects τα οποία αυτός έχει χρηματοδοτήσει.

Τέλος, δημιουργήθηκαν δύο συναρτήσεις (functions). Η χρήση αυτών των συναρτήσεων είναι η εκτύπωση των περιεχομένων των τύπων `fullname` και `location` σαν μία στήλη, αντί της μίας σχέσης μέσα σε μία στήλη. Αυτό επιτεύχθηκε με τη συνένωση των δύο συμβολοσειρών που υπάρχουν στις 2 στήλες των τύπων αυτών. Οι συναρτήσεις αυτές είναι οι εξής :

```
CREATE FUNCTION printname(fullname) RETURNS VARCHAR
AS $$ select ($1).first || ' ' || ($1).last $$ language sql;

CREATE FUNCTION printlocation(location) RETURNS VARCHAR
AS $$ select ($1).country || ' ' || ($1).city $$ language sql;
```

5 Ερωτήσεις SQL

Στο σημείο αυτό θα αναφέρουμε όλες τις SQL ερωτήσεις οι οποίες χρησιμοποιήθηκαν ώστε να δοθούν στο χρήστη του συστήματος οι απαραίτητες πληροφορίες που εμπεριέχονται στη βάση δεδομένων. Θα κατηγοριοποιήσουμε τις ερωτήσεις με βάση την οντότητα (και, κατ' επέκταση, διαφορετικό Widget του GUI) στην όποια αναφέρονται.

5.1 Ίδρυμα

- Επερώτηση 1: Για κάθε ίδρυμα να βρεθεί το όνομα του.

```
SELECT inscode , name
FROM institution
```

- Επερώτηση 2: Για το ίδρυμα με κωδικό X να βρεθεί ο κωδικός και το όνομα των σχολών του.

```
SELECT sch.name , sch.schcode
FROM school AS sch S
WHERE sch.inscode = X
```

5.2 Σχολή

- Επερώτηση 3: Για κάθε σχολή να βρεθεί ο κωδικός της, το όνομα της και το ίδρυμα στο οποίο ανήκει.

```
SELECT schcode , sch_name , ins_name
FROM institution_schools
```

- Επερώτηση 4: Για τη σχολή με κωδικό X να βρεθεί το όνομα του προέδρου της.

```
SELECT printname(e.name) AS name
FROM dep AS d, school AS s, researcher AS e
WHERE ((d.rescode = e.rescode) AND (d.rescode=s.head) AND
(s.schcode = X))
```

- Επερώτηση 5: Για τη σχολή με κωδικό X να βρεθεί το άθροισμα των μισθών όλων των ερευνητών των οποίων απασχολεί.

```
SELECT SUM(salary)
FROM school_occupys
WHERE schcode=X AND ended>current_date
```

- Επερώτηση 6: Για τη σχολή με κωδικό X να βρεθεί το όνομα της, ο αριθμός των εργαζόμενων της και ο μέσος μισθός, καθόλη τη διάρκεια της ύπαρξής της.

```
SELECT name , COUNT(*) , ROUND(AVG(salary))
FROM school_occupys
WHERE schcode=X
GROUP BY schcode , name;
```

5.3 Ερευνητής

- Επερώτηση 7: Για κάθε ερευνητή να βρεθεί ο κωδικός του, το όνομα του, το site και το e-mail του.

```
SELECT rescode , printname(name) , site , email
FROM researcher
```

- Επερώτηση 8: Για κάθε μέλος ΔΕΠ να βρεθεί ο κωδικός του, το όνομα του και η βαθμίδα του.

```
SELECT r.rescode , printname(r.name) , d.rank
FROM researcher AS r, dep AS d WHERE r.rescode = d.rescode
```

- Επερώτηση 9: Για κάθε μέλος εργαστηρίου να βρεθεί ο κωδικός του και το όνομα του.

```
SELECT r.rescode , printname(r.name)
FROM researcher AS r, lab_researcher AS l
WHERE r.rescode = l.rescode
```

- Επερώτηση 10: Για τον ερευνητή με κωδικό X να βρεθεί το όνομα του, και ο τίτλος των πτυχίων του. Επίσης για κάθε πτυχίο του να βρεθεί ο βαθμός του πτυχίου καθώς και η χρονιά και το όνομα της σχολής στο οποίο το απέκτησε.

```
SELECT printname(r.name) , d.title ,d.grade ,d.year , s.name
FROM researcher AS r, degree AS d , school AS s
WHERE (r.rescode = X) AND (r.rescode=d.rescode) AND
      (s.schcode=d.schcode)
```

- Επερώτηση 11: Για τον ερευνητή με κωδικό X να βρεθούν πληροφορίες που αφορούν ερευνητές που είχαν τον ερευνητή X σαν επιβλέπων στο πτυχίο του. Συγκεκριμένα να βρεθεί ο κωδικός και το όνομα κάθε ερευνητή, ο τίτλος, ο βαθμός και η χρονιά του πτυχίου του καθώς και η σχολή στο οποίο το απέκτησε.

```
SELECT r.rescode , printname(r.name) , d.title , d.grade ,
      d.year , s.name
FROM researcher AS r, degree AS d, school AS s
WHERE (d.profcode=X) AND (d.rescode=r.rescode) AND
      (d.schcode=s.schcode)
```

- Επερώτηση 12: Για τον ερευνητή με κωδικό X να βρεθούν πληροφορίες για το που έχει εργαστεί. Συγκεκριμένα ζητείται το όνομα κάθε σχολής στην οποία έχει δουλέψει και το ίδρυμα στο οποίο ανήκει η σχολή. Ακόμα ζητείται η βαθμίδα που είχε όταν δούλεψε, πότε ξεκίνησε, πότε τελείωσε και ο μισθός που έπαιρνε.

```
SELECT a.ins_name , b.name , b.rank , b.salary , b.started ,
      b.ended
FROM institution_schools AS a join
      (SELECT *
       FROM school_occupys
       WHERE rescode = X) AS b ON a.schcode=b.schcode
```

- Επερώτηση 13: Κατάταξε τους ερευνητές με βάση τον αριθμό των project στα οποία έχει απασχοληθεί. Για όσους ερευνητές "ισοβαθμούν" κατέταξε τους με βάση των αριθμό των δημοσιεύσεων. Για κάθε ερευνητή ζητείται ο κωδικός του, το όνομα του, ο αριθμός των project και δημοσιεύσεων.

```
SELECT r.rescode , printname(r.name) , pjb.proj_num ,
      pjb.pub_num
FROM researcher AS r ,
```

```

        (SELECT pj.rescode , pj.proj_num , pb.pub_num
FROM      (SELECT rescode ,COUNT(*) AS proj_num
FROM      works_in group by rescode) AS pj
JOIN      (SELECT rescode ,COUNT(*) AS pub_num
FROM      published GROUP BY rescode) AS pb ON
        pj.rescode=pb.rescode)
AS pjb
WHERE (pjb.rescode=r.rescode) ORDER BY pjb.proj_num
DESC, pjb.pub_num DESC

```

5.4 Χρηματοδότης

- Επερώτηση 14: Για κάθε χρηματοδότη να βρεθεί ο κωδικός του, το όνομα του και ο τύπος του.

```

SELECT sponcode , name , sptype
FROM sponsor

```

- Επερώτηση 15: Να κατατάξεις τους χρηματοδότες βάση το μέσο ποσό χρηματοδότησης.

```

SELECT sponcode , name , ROUND(AVG(amount)) AS am
FROM sponsor_funds_project
GROUP BY sponcode , name
ORDER BY am DESC

```

- Επερώτηση 16: Να κατατάξεις τους χρηματοδότες βάση του αριθμού των χρηματοδοτήσεων που έχει προσφέρει.

```

SELECT sponcode , name , COUNT(*) AS numberr
FROM sponsor_funds_project
GROUP BY sponcode , name
ORDER BY numberr DESC

```

- Επερώτηση 17: Για τον χρηματοδότη με κωδικό X να βρεθεί ο τίτλος των project που έχει χρηματοδοτήσει.

```

SELECT DISTINCT( title )
FROM sponsor_funds_project
WHERE sponcode=X;

```

5.5 Project

- Επερώτηση 18: Για κάθε project να βρεθεί ο κωδικός και ο τίτλος του.

```

SELECT projcode , title
FROM project

```

- Επερώτηση 19: Να βρεθούν τα project με παραπάνω από X εργαζόμενους. Για κάθε project να βρεθεί ο κωδικός του, ο τίτλος του και ο αριθμός των ερευνητών που απασχολούνται σε αυτό.

```

SELECT p.projcode , p.title , w.numberr
FROM project AS p ,
      (SELECT projcode , COUNT(*) AS numberr
       FROM works_in GROUP BY projcode) AS w
WHERE (w.projcode=p.projcode) AND (numberr>X)

```

- Επερώτηση 20: Για το project με κωδικό X να βρεθεί ο κωδικός και το όνομα όλων των ερευνητών που απασχολούνται σε αυτό.

```

SELECT rw.rescode , printname(rw.name)
FROM
      (SELECT r.rescode AS rescode , r.name AS name ,
              w.projcode AS projcode
       FROM researcher AS r,works_in AS w
       WHERE w.rescode=r.rescode) AS rw
WHERE rw.projcode = X

```

- Επερώτηση 21: Για το project με κωδικό X να βρεθούν τα όνομα των ιδρυμάτων, των οποίων ερευνητές απασχολούνται στο project.

```

SELECT DISTINCT(a.ins_name)
FROM
      (SELECT ins.ins_name , occ.rescode
       FROM institution_schools AS ins , school_occupys AS occ
       WHERE occ.schcode=ins.schcode) AS a
JOIN
      (SELECT rescode
       FROM works_in
       WHERE projcode=X) AS b
ON a.rescode=b.rescode

```

5.6 Δημοσίευση

- Επερώτηση 22: Για κάθε δημοσίευση να βρεθεί ο κωδικός και ο τίτλος της.

```

SELECT pubcode , title
FROM publication

```

- Επερώτηση 23: Να βρεθεί ο κωδικός, το όνομα και ο αριθμός αναφορών για κάθε δημοσίευση με παραπάνω από X δημοσιεύσεις.

```

SELECT pubcode , title , refers
FROM publication
WHERE (refers > X)

```

- Επερώτηση 24: Να βρεθεί ο κωδικός, το όνομα και ο αριθμός δημοσιεύσεων για τον ερευνητή με τις περισσότερες δημοσιεύσεις.

```
SELECT res.rescode, printname(res.name), pp.COUNT
FROM researcher AS res,
     (SELECT rescode,COUNT(*)
      FROM published GROUP BY rescode) AS pp
WHERE pp.rescode=res.rescode
ORDER BY COUNT DESC
```

6 Παράρτημα

Η DDL που χρησιμοποιήσαμε για τη δημιουργία της βάσης καθώς και το Makefile που χρησιμοποιήθηκε για την αυτοματοποίηση της διαδικασίας αυτής.

6.1 Makefile

```
# vim: set noet

.PHONY: all schema data clean

DB := acad_world

all: clean schema data

schema: schema.sql
      psql $(DB) -f $<

data: data.sql
      psql $(DB) -f $<

clean:
      dropdb $(DB)
      createdb $(DB)
```

Listing 1: Makefile

6.2 DDL

```
drop table if exists institution cascade;
drop table if exists school cascade;
drop table if exists researcher cascade;
drop table if exists degree cascade;
drop table if exists institution_phone cascade;
drop table if exists school_phone cascade;
drop table if exists dep cascade;
drop table if exists lab_researcher cascade;
drop table if exists occupy cascade;
drop table if exists suggests cascade;
```

```
drop table if exists works_in cascade;
drop table if exists published cascade;
drop table if exists sponsor cascade;
drop table if exists funds cascade;
drop table if exists project cascade;
drop table if exists publication cascade;

drop type if exists location cascade;
drop type if exists address cascade;
drop type if exists fullname cascade;
drop domain if exists year cascade;

CREATE TYPE location AS (
    country varchar(50) ,
    city varchar(50)
);
CREATE TYPE address AS (
    street varchar(50),
    num integer ,
    zipcode varchar(10)
);
CREATE TYPE fullname AS (
    first varchar(20),
    last varchar(20)
);
CREATE DOMAIN year AS integer
    CHECK (VALUE<=2010 AND VALUE>1800);

CREATE TABLE institution (
    inscode serial PRIMARY KEY,
    name varchar(50) NOT NULL,
    yfounded year,
    site varchar(60),
    loc location NOT NULL
—    addr address
);

CREATE TABLE institution_phone (
    inscode integer REFERENCES institution(inscode) ON DELETE
        CASCADE ON UPDATE CASCADE,
    phone varchar(20) NOT NULL,
    PRIMARY KEY(inscode , phone)
);

CREATE TABLE researcher (
```



```

    rescode serial PRIMARY KEY,
    name fullname NOT NULL,
    phone varchar(20),
    site varchar(60),
    email varchar(60),
    loc location NOT NULL,
    addr address
);

CREATE TABLE dep (
    rescode integer PRIMARY KEY REFERENCES researcher(rescode) ON DELETE CASCADE ON UPDATE CASCADE,
    rank varchar(20) — Maybe needs constraint
);

CREATE TABLE school (
    schcode serial UNIQUE NOT NULL,
    inscode integer REFERENCES institution(inscode) ON DELETE CASCADE ON UPDATE CASCADE,
    name varchar(50) NOT NULL,
    head integer REFERENCES dep(rescode) ON DELETE CASCADE ON UPDATE CASCADE,
    yelect year,
    yfounded year,
    PRIMARY KEY(schcode , inscode)
);

CREATE TABLE school_phone (
    schcode integer REFERENCES school(schcode) ON DELETE CASCADE ON UPDATE CASCADE,
    phone varchar(20) NOT NULL,
    PRIMARY KEY(schcode , phone)
);

CREATE TABLE degree (
    rescode integer REFERENCES researcher(rescode) ON DELETE CASCADE ON UPDATE CASCADE,
    schcode integer REFERENCES school(schcode) ON DELETE CASCADE ON UPDATE CASCADE,
    profcode integer REFERENCES dep(rescode) ON DELETE CASCADE ON UPDATE CASCADE,
    title varchar(50) NOT NULL,
    grade numeric(4,2) NOT NULL,
    year year,
    PRIMARY KEY(rescode , schcode , profcode , title),

```

```
        CHECK (grade >= 5 AND grade <= 10)
    );

CREATE TABLE lab_researcher (
    rescode integer PRIMARY KEY REFERENCES researcher(rescode) ON
        DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE occupy (
    schcode integer REFERENCES school(schcode) ON DELETE CASCADE ON
        UPDATE CASCADE,
    rescode integer REFERENCES researcher(rescode) ON DELETE
        CASCADE ON UPDATE CASCADE,
    started date NOT NULL,
    ended date NOT NULL,
    sallary integer NOT NULL,
    rank varchar(20) NOT NULL, -- xreiazetai elegxo sto api
    PRIMARY KEY(schcode, rescode, started),
    CHECK (started < ended)
);

CREATE TABLE sponsor (
    sponcode serial PRIMARY KEY,
    name varchar(50) NOT NULL,
    sptype varchar(50)
);

CREATE TABLE project (
    projcode serial PRIMARY KEY,
    title varchar(100) NOT NULL,
    started date,
    ended date,
    check (started < ended)
);

CREATE TABLE funds (
    sponcode integer REFERENCES sponsor(sponcode) ON DELETE CASCADE
        ON UPDATE CASCADE,
    projcode integer REFERENCES project(projcode) ON DELETE CASCADE
        ON UPDATE CASCADE,
    started date NOT NULL,
    amount integer NOT NULL,
    ended date,
    PRIMARY KEY(sponcode, projcode, started)
);

CREATE TABLE publication (
    pubcode serial PRIMARY KEY,
```

```
        title varchar(50) NOT NULL,
        magaz varchar(50),
        date_publ date NOT NULL,
        refers integer
    );

CREATE TABLE suggests (
    sponcode integer REFERENCES sponsor(sponcode) ON DELETE CASCADE
        ON UPDATE CASCADE,
    rescode integer REFERENCES researcher(rescode) ON DELETE
        CASCADE ON UPDATE CASCADE,
    code serial UNIQUE NOT NULL,
    reason varchar(200) NOT NULL,
    amount integer NOT NULL,
    state varchar(50) NOT NULL,
    PRIMARY KEY(sponcode ,rescode ,code)
);

CREATE TABLE works_in (
    rescode integer REFERENCES researcher(rescode) ON DELETE
        CASCADE ON UPDATE CASCADE,
    projcode integer REFERENCES project(projcode) ON DELETE CASCADE
        ON UPDATE CASCADE,
    started date NOT NULL,
    sallary integer NOT NULL,
    ended date ,
    PRIMARY KEY (rescode , projcode , started)
);

CREATE TABLE published (
    rescode integer REFERENCES researcher(rescode) ON DELETE
        CASCADE ON UPDATE CASCADE,
    pubcode integer REFERENCES publication(pubcode) ON DELETE
        CASCADE ON UPDATE CASCADE,
    PRIMARY KEY (rescode ,pubcode)
);

CREATE VIEW institution_schools as
    select ins.inscode , ins.name as ins_name ,sch.schcode ,sch.name
        as sch_name
        from institution as ins
            join
                school as sch
            on (ins.inscode=sch.inscode);

CREATE VIEW school_occupys as
```

```
select sch.schcode , sch.name, occ.rescode ,
      occ.sallary ,occ.rank ,occ.started ,occ.ended
from school as sch
      join
      occupy as occ
      on sch.schcode=occ.schcode;

CREATE VIEW sponsor_funds_project as
select
  spon_occ.sponcode , spon_occ.name, spon_occ.amount , proj.title
from ( select
      sponsor.sponcode , sponsor.name, funds.projcode , funds.amount
      from sponsor join funds on
      sponsor.sponcode=funds.sponcode) as spon_occ
      natural join
  project as proj;

CREATE function printname(fullname) returns varchar
as $$ select ($1).first || ' ' || ($1).last $$ language sql;

CREATE function printlocation(location) returns varchar
as $$ select ($1).country || ' ' || ($1).city $$ language sql;
```

Listing 2: Data Definition Language