

**ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ**

**ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ**

**ΑΣΚΗΣΗ 1η**

**Ονοματεπώνυμα:**

Βάρδας Εμμανουήλ ΑΜ:03113024

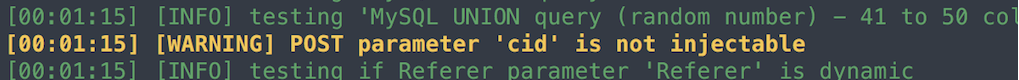
Βουτσινάς Ιωάννης ΑΜ:03113071

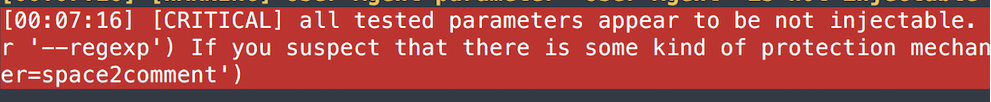
Πέτρου Γεώργιος ΑΜ:03113145

Ο σχεδιασμός της βάσης δεδομένων βασίστηκε στην πρότυπη λύση που δόθηκε στα πλαίσια της 1ης άσκησης.

**A)** Εργαστήκαμε στο DBMS της MySQL, ενώ το User Interface υλοποιήθηκε μέσω HTML,CSS και PHP. Όλα αυτά συντονίστηκαν μέσω του πακέτου λογισμικού MAMP. Γενικά οι δυσκολίες που συναντήσαμε ήταν κυρίως στο κομμάτι της HTML και της PHP για την σωστή αναπαράσταση της βάσης. Στο κομμάτι της SQL, παρατηρήσαμε πως οι περιορισμοί που θέταμε στις σχέσεις δεν υλοποιούνταν από το RDBMS της MySQL (βλ. παρακάτω) οπότε και οι περιορισμοί έπρεπε να υλοποιηθούν μέσω της PHP, ώστε κάθε φόρμα συμπλήρωσης στοιχείων στη σελίδα να απαγορεύει στο χρήστη την προσθήκη μη έγκυρων τιμών (δεν προτιμήθηκε η JavaScript για το λόγο ότι ορισμένοι browsers μπορεί να την είχαν απενεργοποιημένη και να υπήρχαν θέματα συμβατότητας ). Επιπλέον, καταλάβαμε πως η σύνταξη των triggers εξυπηρετεί καταστάσεις στις οποίες μία ενέργεια σε έναν πίνακα (insert, update, delete) πυροδοτεί μία ενέργεια σε έναν άλλο πίνακα. Για να πετύχουμε η ενέργεια που πυροδοτείται από το trigger να αφορά τον ίδιο πίνακα, απαιτείται μία πιο πολύπλοκη σύνταξη trigger (κλείδωμα – ξεκλείδωμα πίνακα κατά τη δράση μίας ενέργειας σε αυτόν) και για αυτό αποφεύχθηκε. Ωστόσο, αυτό μας περιόρισε στο να κατασκευάσουμε triggers που θα προσέδιδαν μία σημασιολογία στη βάση (π.χ. Μεταξύ μίας εταιρείας και ενός φαρμακείου να υπάρχει ένα συμβόλαιο και αν εισάγαμε δεύτερο να διαγραφόταν η καταχώρηση του παλιού).

Στα θετικά, η PHP όπως αναφέραμε και παραπάνω βοήθησε στην υλοποίηση των περιορισμών που θεωρήσαμε αναγκαίο να τεθούν στα πεδία των σχέσεων. Πάρα πολύ σημαντικό θεωρήσαμε επίσης ότι μέσω της PHP μπορούσαμε να συντάξουμε queries εξαρτώμενα από είσοδο που έδινε ο χρήστης (π.χ query 7!). Επιπλέον, κάναμε χρήση βιβλιοθήκης Bootstrap η οποία εξασφάλισε δυνατότητες στα γραφικά που έκαναν την εφαρμογή φιλική και άρτια προς το χρήστη (το περιεχόμενο είναι πιο responsive δηλαδή η εφαρμογή ανταποκρίνεται καλύτερα αναλογα με τη συσκευή του χρήστη). Επίσης, η εντολή ON DELETE CASCADE ON UPDATE CASCADE της SQL στα foreign keys, όπου δεν χρησιμοποιήθηκαν triggers, μας βοήθησε στην εξασφάλιση αναφορικής ακεραιότητας στη βάση μας (όταν διαγράφαμε μία πλειάδα από έναν πίνακα, διαγράφονταν αυτόματα πλειάδες από πίνακες που είχαν ως foreign key το primary key αυτού του πίνακα). Τέλος όσον αφορά την ασφάλεια της εφαρμογής έγιναν διάφορα αυτοματοποιημένα test με την χρήση του open source εργαλείου SQLMAP. Έγινα διάφορα test όσο αναφορά τις GET παραμέτρους που χρησιμοποιεί η εφαρμογή ωστόσο δεν βρέθηκε κάποια ευπάθεια. Όσο αφορά κάποια POST request υπήρχε αρχικά κάποιο vulnerabilityσε μία εκ των παραμέτρων που στέλνονταν στο server προς επεξεργασία (και πιο συγκεκριμένα η μεταβλητή cid στο update.php που είναι τύπου integer ) .Τα POST request γίνανε captured με τη χρήση του BurbSuite (free edition) σε συνδιασμό με τον firefox. Το injection point που βρέθηκε και αφορούσε την cid παράμετρο θα επέτρεπε σε κάποιο πραγματικό σενάριο πλήρη πρόσβαση στη βάση. Ύστερα από αναθεώρηση του php κώδικα και με τη πληροφορία ότι η μόνη ευπαθής παράμετρος ήταν η cid η οποία η μόνη διαφορά που είχε από τις υπόλοιπες παραμέτρους είναι ότι ήτα τύπου integer, φάνηκε ότι η ευπάθεια είχε να κάνει με το ότι στον κώδικα δεν γινόταν πουθενά έλεγχος για το αν αυτή η παράμετρος που δινόταν ως είσοδο από το χρήστη ήταν όντως τύπου integer. Έτσι υλοποιώντας κατάλληλο έλεγχο σε κάθε php script που χειρίζεται ακεραίους η βάση είναι πλέον πιο ασφαλής .Για τα αλφαριθμητικά χρησιμοποιήθηκε η συνάρτηση mysqli\_real\_escape\_string η οποία κάνει escape περίεργους-κακόβουλους χαρακτήρες που μπορεί να χρησιμοποιηθούν ως είσοδο από το χρήστη. Ενδεικτικά κάποιοι έλεγχοι που έγιναν στη συνέχεια και επιβεβαιώνουν ότι η βάση είναι πλέον **πιο** ασφαλής φαίνονται παρακάτω:



**B)**

Ο πλήρης SQL κώδικας σχεδιασμού της βάσης φαίνεται στο τμήμα C της εργασίας. Να σημειωθεί ότι στη σελίδα ορίσαμε τις σχέσεις Doctor, Drug, Pharmaceutical Company ως προσβάσιμες από το χρήστη για insert, update, delete. Επίσης, η όψη 1 που είναι ενημερώσιμη και παρουσιάζεται παρακάτω προσφέρεται για τροποποίηση από το χρήστη.

Οι περιορισμοί που θεωρήσαμε είναι οι ακόλουθοι:

* Στα primary keys προφανώς τέθηκε ο περιορισμός not null, ενώ παράλληλα τα ορίσαμε auto increment, ώστε ο χρήστης να μην εισάγει id (αποφεύγοντας έτσι και λάθη στη συνέπεια της βάσης) και το id να ορίζεται αυτόματα από το σύστημα.
* Στις σχέσεις που υπάρχουν foreign keys, τα ορίζουμε ως not null για να υπάρχει συνέπεια στη βάση και να συμφωνούν με τα primary keys των σχέσεων στις οποίες ανήκουν.
* Τα ονόματα των διαφόρων σχέσεων (π.χ Firstname, Lastname) τέθηκαν ως not null, καθώς θεωρήθηκαν βασικό γνώρισμα στις αντίστοιχες σχέσεις. Ομοίως, not null τέθηκαν γνωρίσματα όπως ονόματα πόλεων, δρόμων, τηλεφώνων, προσδιορισμοί ποσοτήτων (π.χ Quantity στη σχέση Prescriptions) και Dates (π.χ στη σχέση Contract), καθώς αποτελούν βασικούς προσδιορισμούς των αντίστοιχων σχέσεων.
* Δεν ορίσαμε not null στα πεδία που θεωρήθηκαν δευτερεύοντα για τον προσδιορισμό των σχέσεων, όπως τα χρόνια εμπειρίας του γιατρού, ηλικία και στοιχεία κατοικίας ασθενή, κείμενο στο συμβόλαιο κτλ. Τα παραπάνω αποτελούν μια σύμβαση, η οποία θα μπορούσε να έχει θεωρηθεί διαφορετικά.
* Βάλαμε περιορισμό για θετικές τιμές σε όλα τα πεδία που η αρνητική τιμή δεν έχει καμία φυσική σημασία (π.χ χρόνια εμπειρίας γιατρού, αριθμός οδού, ηλικία, ποσότητα φαρμάκου κλπ). Επιπλέον ορίσαμε τα τηλέφωνα να είναι δεκαψήφια (περιοχή τιμών 1000000000 – 9999999999) και το EndDate να είναι μεγαλύτερο από το StartDate στο συμβόλαιο.

Ωστόσο οι περιορισμοί που ορίστηκαν δεν υλοποιούνται από τη MySQL στην οποία φτιάξαμε τη βάση δεδομένων. Όποιοι περιορισμοί (εκτός από not null) τέθηκαν μέσω κώδικα PHP οπότε ο χρήστης επιβάλλεται να εισάγει έγκυρες τιμές στις φόρμες, καθώς στη MySQL οι περιορισμοί check μπορούν να οριστούν και να διαβαστούν από το σύστημα, ωστόσο αγνοούνται.

Τέλος, δεν υλοποίησαμε κάποιο ευρετήριο στη βάση μας και επομένως τα υπάρχοντα ευρετήρια είναι αυτά που χρησιμοποιεί το DBMS της MySQL (κυρίως B+ trees indexes ως προς το primary key, Hash indexes).

**C)**

**Σχεδιασμός Βάσης**

Ο SQL κώδικας για τη δημιουργία της βάσης παρατίθεται παρακάτω:

**create database DrugStore;**

**use DrugStore;**

**show tables;**

**create table Doctor**(

DoctorId int not null AUTO\_INCREMENT,

FirstName varchar(15) not null,

LastName varchar(15) not null,

Speciality varchar(20) not null,

ExperienceYears int check(ExperienceYears>=0),

primary key(DoctorId) );

**create table Patient**(

PatientId int not null AUTO\_INCREMENT ,

FirstName varchar(15) not null,

LastName varchar(15) not null,

Town varchar(20) not null,

StreetName varchar(20) ,

Number int check(Number>0),

PostalCode int check(PostalCode>0),

Age int check(Age>0),

DoctorId int not null,

foreign key(DoctorId) references Doctor(DoctorId),

primary key(PatientId) );

**create table Pharmacy**(

PharmacyId int not null AUTO\_INCREMENT,

Name varchar(15) not null,

Town varchar(20) not null,

StreetName varchar(20) not null,

Number int not null check(Number>0),

PostalCode int not null check(PostalCode>0),

PhoneNumber bigint not null check(PhoneNumber between 1000000000 AND 9999999999),

primary key(PharmacyId) );

**create table PharmaceuticalCompany**(

PharmaceuticalCompanyId int not null AUTO\_INCREMENT,

Name varchar(15) not null,

PhoneNumber bigint not null check(PhoneNumber between 1000000000 AND 9999999999),

primary key(PharmaceuticalCompanyId) );

**create table Drug**(

DrugId int not null AUTO\_INCREMENT,

Name varchar(15) not null,

Formula varchar(15) not null,

PharmaceuticalCompanyId int not null,

foreign key(PharmaceuticalCompanyId) references PharmaceuticalCompany(PharmaceuticalCompanyId) ON DELETE CASCADE ON UPDATE CASCADE,

primary key(DrugId) );

**create table Prescription**(

PatientId int not null,

DoctorId int not null,

DrugId int not null,

Date date not null,

Quantity int not null check(Quantity>0),

foreign key(PatientId) references Patient(PatientId) ON DELETE CASCADE ON UPDATE CASCADE,

foreign key(DoctorId) references Doctor(DoctorId),

foreign key(DrugId) references Drug(DrugId) ,

constraint presc\_key primary key(PatientId,DoctorId,DrugId) );

**create table Sell**(

PharmacyId int not null,

DrugId int not null,

Price int check(Price>0),

foreign key(PharmacyId) references Pharmacy(PharmacyId) ON DELETE CASCADE ON UPDATE CASCADE,

foreign key(DrugId) references Drug(DrugId) ,

constraint sl\_key primary key(PharmacyId,DrugId) );

**create table Contract**(

PharmacyId int not null,

PharmaceuticalCompanyId int not null,

StartDate date not null,

EndDate date not null,

Text varchar(20),

Supervisor varchar(15),

foreign key(PharmacyId) references Pharmacy(PharmacyId) ON DELETE CASCADE ON UPDATE CASCADE,

foreign key(PharmaceuticalCompanyId) references PharmaceuticalCompany(PharmaceuticalCompanyId) ON DELETE CASCADE ON UPDATE CASCADE,

constraint cntr\_key primary key(PharmacyId,PharmaceuticalCompanyId) ,

constraint DatesCompatibility check(EndDate>StartDate));

**commit;**

**Queries**

Τα queries που υλοποιήθηκαν φαίνονται παρακάτω

***1st Query: Γιατρός ανά ασθενή (query με Join)***

select Patient.LastName, Patient.FirstName, Doctor.LastName, Doctor.FirstName

from Patient, Doctor

where Patient.DoctorId = Doctor.DoctorId

*Επεξήγηση****:*** Εμφανίζεται το ονοματεπώνυμο κάθε ασθενή και του γιατρού που τον παρακολουθεί. Πρόκειται για βασικό ερώτημα που λόγω της σημασιολογίας του θεωρήσαμε σημαντικό να συμπεριλαμβάνεται στη βάση που κατασκευάσαμε.

***2nd Query: Φάρμακα ανά ασθενή (Query με Join και Order By)***

select Patient.PatientId, Patient.LastName, Patient.FirstName, Drug.Name

from Patient, Drug, Prescription

where Patient.PatientId = Prescription.PatientId AND Drug.DrugId = Prescription.DrugId

order by Patient.PatientId

*Επεξήγηση****:*** Εμφανίζεται το ονοματεπώνυμο κάθε ασθενή και τα φάρμακα που λαμβάνει. Η σειρά εμφάνισης είναι αύξουσα ως προς το Id κάθε ασθενή.

***3rd Query: Συμβόλαια εταιρείων με φαρμακεία που λήγουν πριν την Χ ημερομηνία λήξης (Query με Join και Order By)***

select Pharmacy.PharmacyId, Pharmacy.Name, PharmaceuticalCompany.Name

from Pharmacy, Contract, PharmaceuticalCompany

where Pharmacy.PharmacyId = Contract.PharmacyId AND Contract.PharmaceuticalCompanyId = PharmaceuticalCompany.PharmaceuticalCompanyId AND Contract.EndDate < **‘Χ’**

order by Pharmacy.PharmacyId

*Επεξήγηση****:*** Από το χρήστη δίνεται μέσω HTML φόρμας μία ημερομηνία και εμφανίζονται τα συμβόλαια που λήγουν πριν την δοθείσα ημερομηνία.

***4th Query: Γιατροί με περισσότερα από Χ χρόνια υπηρεσίας (Query με Order By)***

select \*

from Doctor

where Doctor.ExperienceYears > **‘Χ’**

order by Doctor.ExperienceYears desc

*Επεξήγηση****:*** Από το χρήστη δίνεται μέσω HTML φόρμας μία θετική τιμή και εμφανίζονται τα στοιχεία γιατρών με τουλάχιστον τόσα χρόνια υπηρεσίας σε φθίνουσα σειρά ως προς αυτά. Το συγκεκριμένο query αποτελεί μία ένδειξη για την εμπειρία των γιατρών με βάση τις απαιτήσεις του χρήστη.

***5th Query: Αναζήτηση φαρμακείων στην Χ πόλη (Query που χρησιμοποιεί όψη)***

select \*

from v1

where v1.Town = **‘Χ’**

*Επεξήγηση****:*** Από το χρήστη δίνεται μέσω HTML φόρμας ένα όνομα πόλης και εμφανίζονται τα φαρμακεία που βρίσκονται σε αυτήν. Για την υλοποίηση χρησιμοποιήθηκε η όψη v1 που φαίνεται πιο κάτω, με την οποία εμφανίζονται οι πόλεις και τα φαρμακεία που βρίσκονται σε αυτές.

***6th Query: Ποσοστό γιατρών με Χ ειδικότητα (Query με Aggregate function και Nested Query)***

select Count(Doctor.DoctorId) \* 100 / ( select Count(Doctor.DoctorId) from Doctor) as Speciality\_Percentage

from Doctor

where Doctor.speciality = ***‘Χ’***

*Επεξήγηση****:*** Από το χρήστη δίνεται μέσω HTML φόρμας μία ειδικότητα και εμφανίζεται το ποσοστό των γιατρών αυτής της ειδικότητας. Το συγκεκριμένο query μπορεί να αναδείξει τους τομείς στους οποίους ειδικεύονται περισσότεροι/λιγότεροι γιατροί.

***7th Query: Φαρμακεία στην Χ πόλη που πωλείται το Υ φάρμακο σε τιμή μικρότερη από Ζ (Query με Join και χρήση όψης)***

select v1.PharmacyId, v1.PharmacyName

from v1,Sell,Drug

where (v1.Town = **‘Χ’**) AND (Sell.DrugId = Drug.DrugId) AND (Drug.Name = **‘Υ’**) AND (v1.PharmacyId = Sell.PharmacyId) AND (Sell.Price < **‘Ζ’**)

*Επεξήγηση****:*** Από το χρήστη δίνεται μέσω HTML φόρμας τρία στοιχεία: Το όνομα X μίας πόλης, το όνομα Y ενός φαρμάκου και μία θετική τιμή Ζ. Εμφανίζονται τα φαρμακεία που βρίσκονται στην Χ πόλη και διαθέτουν το Υ φάρμακο σε τιμή μικρότερη από Ζ. Πρόκειται για ένα πολύ εξυπηρετικό query, καθώς αναδεικνύει τα φαρμακεία που πωλούν σε οικονομική τιμή τα φάρμακα που ενδιαφέρουν το χρήστη και βρίσκονται στην πόλη ενδιαφέροντος του.

***8th Query: Φαρμακεία με τις μέσες τιμές φαρμάκων που πουλάνε να είναι μικρότερες από την αντίστοιχη μέση τιμή όλων των φαρμακείων (Query με Group By, Having και Aggregate function)***

select \*

from v2

group by v2.PharmacyId

having v2.Average\_Price < (select avg(v2.Average\_Price)

from v2)

*Επεξήγηση****:*** Αξιοποιείται η όψη v2, που φαίνεται πιο κάτω, με την οποία εμφανίζονται τα φαρμακεία και οι αντίστοιχες μέσες τιμές πώλησης των φαρμάκων τους. Το συγκεκριμένο query αποτελεί ένα μέτρο για την ακρίβεια στις τιμές που παρέχουν τα φαρμακεία.

***9th Query: Φάρμακα που έχουν συνταγογραφηθεί από τουλάχστον ένα γιατρό (Query με Nested Query)***

select Drug.name

from Drug

where Drug.DrugId in (select Drug.DrugId

from Drug, Prescription

where Drug.DrugId= Prescription.DrugId)

*Επεξήγηση****:*** Εδώ αναδεικνύονται τα φάρμακα που έχουν συνταγογραφηθεί από τουλάχιστον ένα γιατρό. Το συγκεκριμένο query δίνει μία αίσθηση των συνταγογραφούμενων φαρμάκων.

***10th Query: Γιατροί σε φθίνουσα σειρά με βάση των αριθμό πελατών τους (Query με Join, Group By, Order By)***

select Doctor.DoctorId, Doctor.LastName, Doctor.FirstName, count(Patient.PatientId) as Patients

from Doctor, Patient

where (Doctor.DoctorId= Patient.DoctorId)

group by Doctor.DoctorId

order by count(Patient.PatientId) desc

*Επεξήγηση****:*** Με αυτό το query μπορόυμε να δούμε τους πιο δημοφιλείς γιατρούς βάσει πελατείας.

**Views**

Τα views που υλοποίηθηκαν είναι τα ακόλουθα:

***View 1: Φαρμακεία και οι πόλες στις οποίες βρίσκονται (Ενημερώσιμο)***

create view v1(Town,PharmacyId,PharmacyName) as

(select Pharmacy.Town,Pharmacy.PharmacyId, Pharmacy.Name

from Pharmacy

order by Pharmacy.Town)

*Επεξήγηση****:***  Το συγκεκριμένο view παρουσιάζει όλα τα φαρμακεία με τις αντίστοιχες πόλεις στις οποίες βρίσκεται. Αξιοποείται για την διευκόλυνση στην κατασκευή των queries 5 και 7, που παρουσιάστηκαν παραπάνω. Επίσης, είναι ενημερώσιμο καθώς ορίζεται σε μία σχέση βάσης και περιέχει το κύριο κλειδί της.

***View 2: Φαρμακεία με τη μέση τιμή που πουλούν τα φάρμακα τους (Μη ενημερώσιμο)***

create view v2(PharmacyId, PharmacyName, Average\_Price) as

(select Pharmacy.PharmacyId, Pharmacy.Name, avg(Sell.Price)

from Sell, Pharmacy

where Pharmacy.PharmacyId=Sell.PharmacyId

group by Pharmacy.PharmacyId

order by avg(Sell.Price) desc )

*Επεξήγηση****:*** Το view αυτό παρουσιάζει τα φαρμακεία με τη μέση τιμή πώλησης των φαρμάκων τους και είναι μη ενημερώσιμο αφού περιέχει τη συναθροιστική συνάρτηση avg. Χρησιμοποίηθηκε για την κατασκευή του query 8.

**Triggers**

Παρακάτω παρουσιάζονται τα triggers που υλοποίησαμε:

***Trigger 1:*** Όταν διαγραφεί φάρμακο να διαγράφονται οι καταχωρήσεις πωλήσεων (σχέση Sell) και συνταγογραφήσεων (σχέση Prescription) στις οποίες συμμετέχει.

DELIMITER $$

create trigger Drug\_Delete BEFORE DELETE ON Drug

FOR EACH ROW BEGIN

DELETE FROM Sell WHERE Sell.DrugId = old.DrugId;

DELETE FROM Prescription WHERE Prescription.DrugId = old.DrugId;

END$$

DELIMITER ;

***Trigger 2***: Όταν διαγράφεται Φαρμακευτική Εταιρεία να διαγράφονται οι καταχωρήσεις φαρμάκων (σχέση Drug) που παρήχθησαν από αυτήν.

DELIMITER $$

create trigger PhC\_Delete BEFORE DELETE ON PharmaceuticalCompany

FOR EACH ROW BEGIN

DELETE FROM Drug WHERE old.PharmaceuticalCompanyId = Drug.PharmaceuticalCompanyId;

END$$

DELIMITER ;

*Σχόλιο*: Όταν διαγράφονται οι καταχωρήσεις των φαρμάκων της εταιρείας, ενεργοποιείται το Trigger 1 που διαγράφει με τη σειρά του τις αντίστοιχες καταχωρήσεις στη σχέσεις Prescription και Sell. Τέλος, διαγράφονται τα συμβόλαια στα οποία συμμετέχει η εταιρεία με την εντολή ON DELETE CASCADE που έχει συμπεριληφθεί για το foreign key PharmaceuticalCompanyId στη σχέση Contract.

***Trigger 3***: Κατά τη διαγραφή ενός γιατρού διαγράφονται οι πελάτες που παρακολουθούνται από αυτόν (σχέση Patient) και οι συνταγογραφήσεις που έχει γράψει (σχέση Prescription).

DELIMITER $$

create trigger Doctor\_Delete BEFORE DELETE ON Doctor

FOR EACH ROW BEGIN

DELETE FROM Patient WHERE old.DoctorId = Patient.DoctorId;

DELETE FROM Prescription WHERE old.DoctorId = Prescription.DoctorId;

END$$

DELIMITER ;

*Σχόλιο*: Θεωρήσαμε πως όταν ένας γιατρός διαγράφεται από τη βάση, οι πελάτες του θα πρέπει επίσης να διαγραφούν ώστε να καταχωρηθούν εκ νέου με άλλον επιβλέποντα γιατρό (αν και εφόσον εισαχθούν από το χρήστη), καθώς και οι συνταγογραφήσεις που έγραψε από τη στιγμή που ο ίδιος δεν ανήκει στη βάση πια.

*Επεξήγηση*: Τα triggers αυτά υλοποιούν απλούς και λειτουργικούς περιορισμούς, ώστε να υπάρχει ακεραιότητα στη βάση μας. Οι υπόλοιποι περιορισμοί τέτοιου είδους υλοποιήθηκαν με την εντολή ON DELETE CASCADE στα αντίστοιχα foreign keys των υπόλοιπων σχέσεων. ∎