ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Βάσεις Δεδομένων

Εξαμηνιαία Εργασία

7ο Εξάμηνο

Συνεργάτες:

Ζαϊρές Σωτήριος 03113305

Περράκης Γεώργιος 03113511

Σοφιανίδης Γεώργιος 03113179

2016-2017

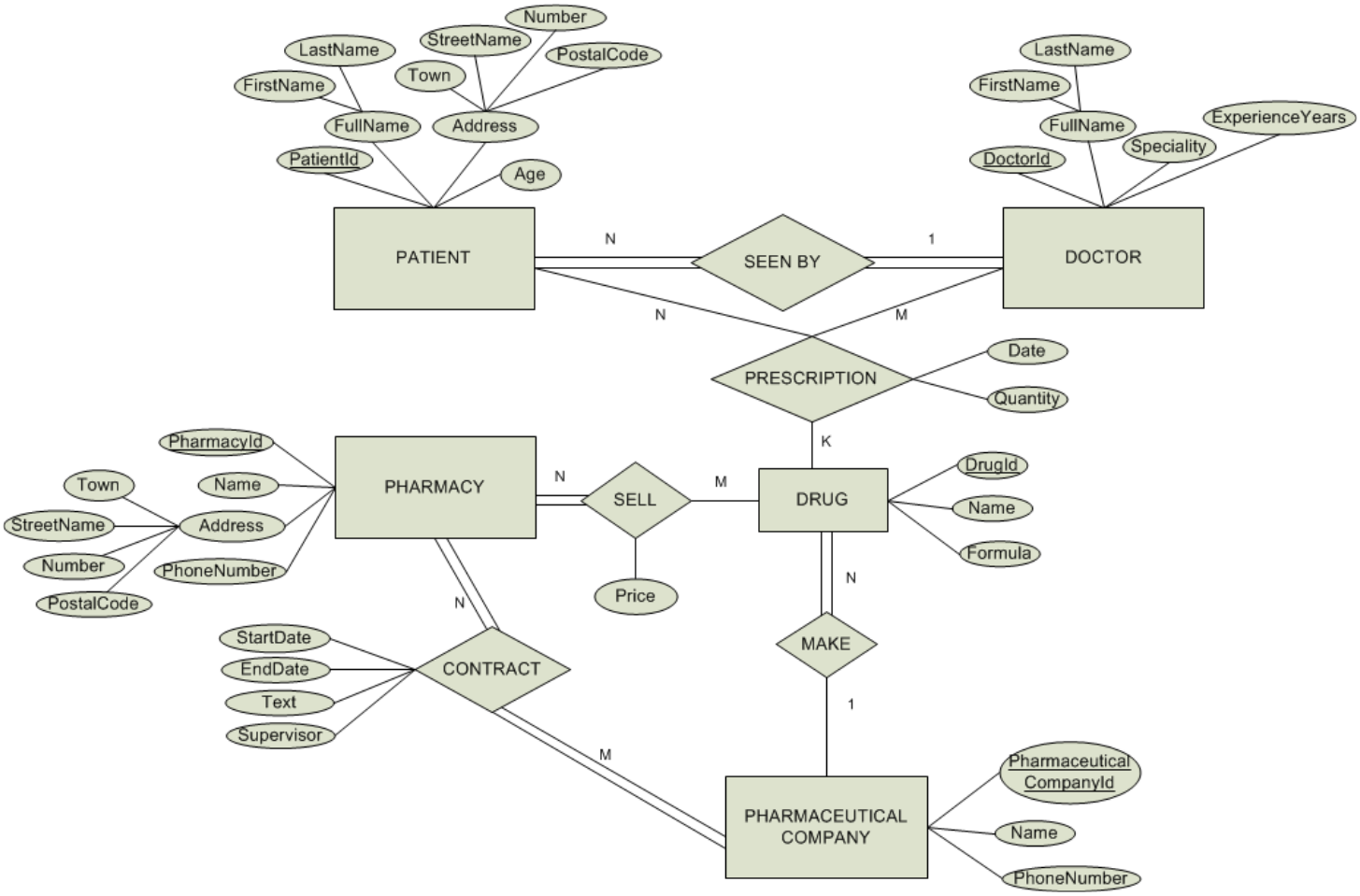
ΕΙΣΑΓΩΓΗ

Η αλυσίδα φαρμακείων Prescription-R-X, έχοντας αναπτύξει τις εργασίες της, χρειάζεται τη δημιουργία μια Βάσης Δεδομένων, με σκοπό τη συνεχή καταγραφή και έλεγχο όλων των δραστηριοτήτων μεταξύ των ασθενών, των γιατρών, των φαρμακείων, των φαρμάκων και των φαρμακευτικών εταιρειών που εμπλέκονται. Για το λόγο αυτό, ζητήθηκε η δημιουργία της Βάσης για την υποστήριξη όλου του συστήματος. Μετά από μια σειρά συνεντεύξεων με στελέχη της αλυσίδας Prescription-R-X συλλέχθηκαν όλα τα απαραίτητα στοιχεία και καταλήξαμε στις παρακάτω διαπιστώσεις/εξελίξεις:

- Κάθε ασθενείς αναγνωρίζεται από ένα κωδικό ασθενούς, και επιπλέον καταγράφεται για αυτόν το όνομα, η διεύθυνση και η ηλικία.
- Κάθε γιατρός αναγνωρίζεται από ένα κωδικό γιατρού, και καταγράφεται για αυτόν το όνομα, η ειδικότητα και τα χρόνια εμπειρίας του.
- Κάθε φαρμακευτική εταιρία από την οποία προμηθεύονται φάρμακα αναγνωρίζεται από έναν κωδικό εταιρίας, και επιπλέον καταγράφεται το όνομα και ένα τηλέφωνο επικοινωνίας.
- Κάθε φάρμακο κατασκευάζεται από μια φαρμακευτική εταιρία, και ένας κωδικός φαρμάκου χαρακτηρίζει το φάρμακο μοναδικά. Επιπλέον, καταγράφεται το όνομα και η φόρμουλα(σύσταση) του φαρμάκου. Εάν η φαρμακευτική εταιρία διαγραφεί, δε χρειάζεται να καταγράφονται πλέον τα προϊόντα της εταιρίας.
- Κάθε φαρμακείο έχει ένα κωδικό φαρμακείου, όνομα, διεύθυνση και ένα τηλέφωνο επικοινωνίας.
- Κάθε ασθενής έχει ένα γιατρό που τον παρακολουθεί. Κάθε γιατρός έχει τουλάχιστον έναν ασθενή.
- Κάθε φαρμακείο έχει προς πώληση πολλά φάρμακα και έχει μια τιμή για το καθένα. Ένα φάρμακο μπορεί να πωλείται σε πολλά φαρμακεία, και η τιμή μπορεί να ποικίλει από το ένα φαρμακείο στο άλλο.
- Οι γιατροί συνταγογραφούν φάρμακα για τους ασθενείς. Ένας γιατρός μπορεί να συνταγογραφεί ένα ή περισσότερα φάρμακα για πολλούς ασθενείς, και ένα ασθενείς μπορεί να λάβει συνταγή από πολλούς γιατρούς. Κάθε συνταγογράφηση χαρακτηρίζεται από την ημερομηνία την οποία δόθηκε και την ποσότητα του φαρμάκου. Μπορείτε να θεωρήσετε ότι αν ένας γιατρός συνταγογραφεί το ίδιο φάρμακο για τον ίδιο ασθενή πάνω από μια φορά, μόνο η τελευταία συνταγογράφηση χρειάζεται να καταγράφεται.
- Οι φαρμακευτικές εταιρίες έχουν μακροχρόνια συμβόλαια με τα φαρμακεία. Μια φαρμακευτική εταιρία μπορεί να συνεργάζεται με πολλά φαρμακεία, και ένα φαρμακείο μπορεί να συνεργάζεται με πολλές φαρμακευτικές εταιρίες. Για κάθε συμβόλαιο, πρέπει να καταγράφεται η ημερομηνία έναρξης και η ημερομηνία λήξης του συμβολαίου, καθώς και το κείμενο του συμβολαίου.
- Τα φαρμακεία διορίζουν έναν επόπτη για κάθε συμβόλαιο. Θα πρέπει πάντα να υπάρχει ένας επόπτης για κάθε συμβόλαιο, αλλά ο επόπτης μπορεί να αλλάξει όσο είναι σε ισχύ το συμβόλαιο.

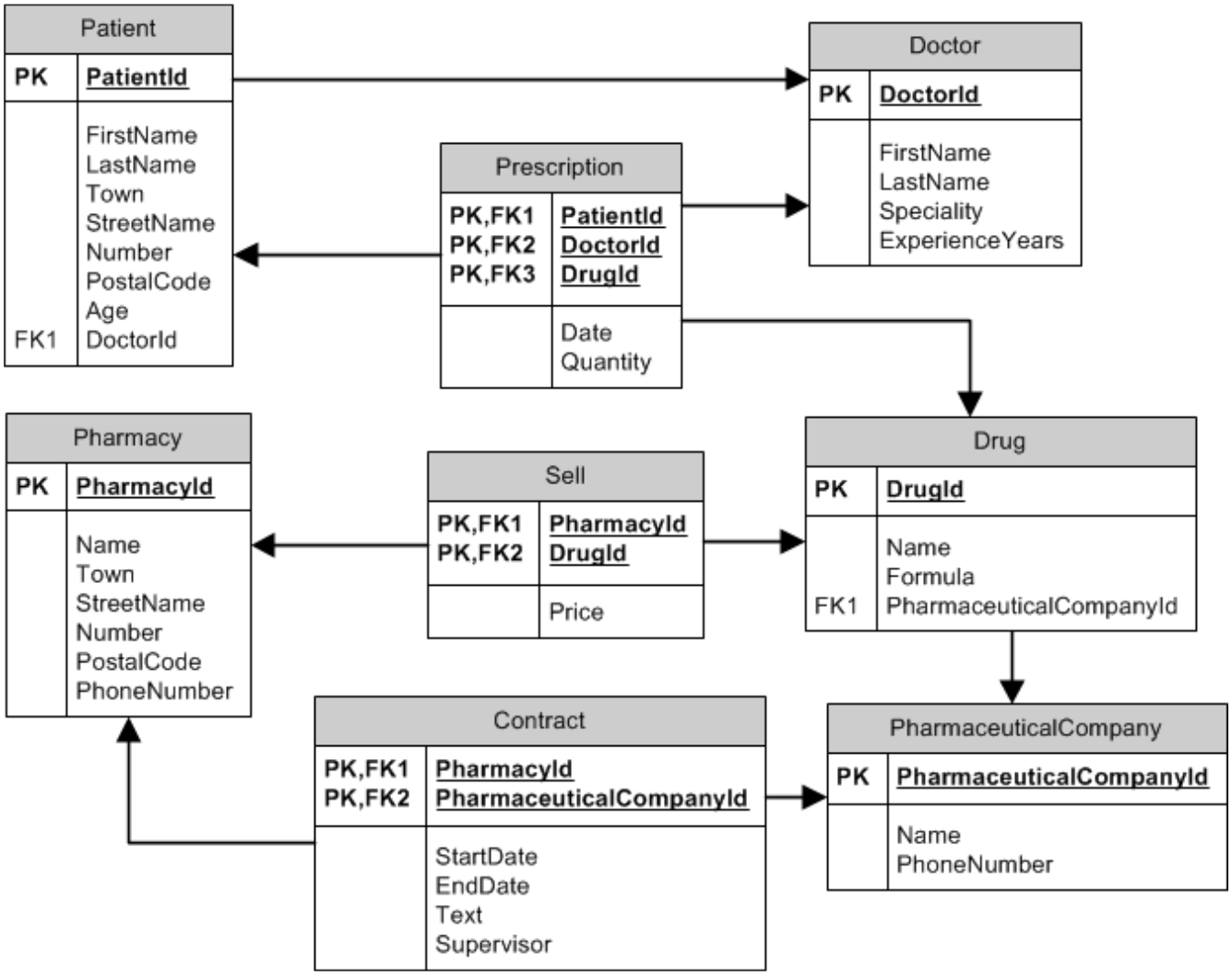
Λαμβάνοντας υπόψη όλες αυτές τις βασικές απαιτήσεις και προδιαγραφές για τη σωστή λειτουργία του συστήματος, και κάνοντας κάποιες λογικές υποθέσεις/παραδοχές, ξεκινήσαμε το σχεδιασμό της Βάσης Δεδομένων. Το πρώτο βήμα είναι η δημιουργία του συνολικού εννοιολογικού διαγράμματος Οντοτήτων-Συσχετίσεων της Βάσης(E-R Diagram).

Σημείωση: Χρησιμοποιούμε τη προτεινόμενη βάση δεδομένων.



Στη συνέχεια μετατρέπουμε το προηγούμενο διάγραμμα οντοτήτων-συσχετίσεων στο αντίστοιχο σχεσιακό σχήμα. Το πρωτεύον κλειδί κάθε σχέσης σημειώνεται με έντονα γράμματα και υπογράμμιση.

Προκύπτει το εξής Σχεσιακό Διάγραμμα (Relational Model).



ΣΧΟΛΙΑ ΓΙΑ ΤΗΝ ΔΙΑΔΙΚΑΣΙΑ ΑΝΑΠΤΥΞΗΣ ΤΗΣ ΒΑΣΗΣ ΔΕΔΟΜΕΝΩΝ

Για την υλοποίηση της Βάσης Δεδομένων και την ανάπτυξη της Εφαρμογής χρησιμοποιήθηκαν τα εργαλεία MySQL Server, με χρήση του XAMPP για τη σύνδεση στο server Apache και τη χρησιμοποίηση του phpMyAdmin, με σκοπό την άμεση και γραφική δημιουργία της βάσης. Επίσης, χρησιμοποιήθηκε η γλώσσα PHP για την σύνδεση του Interface με τη βάση και τη μεταφορά δεδομένων, καθώς και την εκτέλεση των εντολών και των ερωτημάτων στη βάση από το χρήστη. Για το Interface χρησιμοποιήθηκαν HTML/CSS τεχνικές για τη δημιουργία βασικού γραφικού περιβάλλοντος.

Όλα τα εργαλεία είναι διαθέσιμα δωρεάν και έτσι η απόκτηση και η πρόσβαση σε αυτά ήταν άμεση και χωρίς ιδιαίτερα προβλήματα.

Η χρήση του MySQL Server και ιδιαίτερα το εργαλείο phpmyAdmin μας βοήθησε σημαντικά στην κατασκευή της Βάσης, κυρίως λόγω της ευκολίας που παρουσιάζει στη χρήση και συντήρηση μιας βάσης δεδομένων, μέσα από ένα γραφικό περιβάλλον, της ταχύτητας του αλλά και στην πληθώρα αναφορών και βοηθημάτων που υπάρχουν για αυτό, ιδιαίτερα στο διαδίκτυο. Επίσης, εξαιτίας του λογισμικού είχαμε στη διάθεση μας, άμεσα, όλα τα κατάλληλα εργαλεία για το σχεδιασμό, την ανανέωση, τη συντήρηση, και την επίβλεψη της Βάσης Δεδομένων μας.

Ο συνδυασμός PHP + MySQL Database System αποτελεί μια πλατφόρμα ανάπτυξης που μπορεί να αναπτυχθεί και χρησιμοποιηθεί σε όλα τα λειτουργικά συστήματα (Windows και UNIX) και αποτελεί τη στάνταρ επιλογή για πολύ γνωστές εφαρμογές με μεγάλο όγκο δεδομένων και χρηστών (πχ Facebook), γεγονός που δείχνει τη σημασία της χρήσης της. Συγκεκριμένα, κάναμε χρήση του MySQLi extension με procedural API. Βασικό μειονέκτημα του MySQLi είναι ότι λειτουργεί μόνο για MySQL databases και όχι σε διαφορετικούς τύπους (όπως το PDO), που σημαίνει ότι για αλλαγή βάσης, απαιτείται αλλαγή του κώδικα.

Για τη δημιουργία του User Interface χρησιμοποιήσαμε HTML/CSS, για το σχεδιασμό φορμών και λιστών ώστε ο χρήστης να επιλέγει τα ερωτήματα και να βλέπει τα αντίστοιχα αποτελέσματα-πίνακες χωρίς την απαραίτητη γνώση SQL queries, αλλά με γνώση σε μερικές περιπτώσεις των αντίστοιχων αποδεκτών τιμών (όπως κωδικός κάποιου γιατρού ή φαρμάκου, που είναι μοναδικοί).

Συνοπτικά, η ροή ανάπτυξης είναι η εξής: δημιουργία βάσης μέσω phpMyAdmin, MySQL Server, ορισμός πινάκων και περιορισμών της βάσης, γέμισμα πινάκων, δημιουργία του αντίστοιχου interface, φορμών και μεθόδων μεταφοράς δεδομένων και δημιουργία όλων των ερωτημάτων και απαραίτητων λειτουργιών της βάσης.

Περιορισμοί που ορίσαμε στη Βάση Δεδομένων

Κατά τον σχεδιασμό και κατασκευή της Βάσης Δεδομένων στο Σχεσιακό Μοντέλο θέσαμε κάποιους απαραίτητους **δομικούς περιορισμούς** (constraints), άμεσους και έμμεσους, ώστε τα δεδομένα να έχουν συνέπεια και συνάφεια. Αρχικά, θέσαμε τους βασικότερους περιορισμούς, αυτούς που είναι έμφυτοι στο μοντέλο (περιορισμοί **κλειδιών** (key), **ακεραιότητα οντότητας** (entity integrity) και **αναφορική ακεραιότητα** (referential integrity)). Κατόπιν, προχωρήσαμε στον ορισμό των ρητών δομικών περιορισμών (**πεδίο τιμών** (domain), **πεδίο στηλών** (column) και **περιορισμοί οριζόμενοι από το χρήστη** (user-defined)).

Περιορισμοί κλειδιών: Τα διαφορετικά κλειδιά, όπως ορίστηκαν στο μοντέλο E-R, ισχύουν και στο Σχεσιακό Μοντέλο. Γενικά, υπάρχουν τριών ειδών κλειδιά: το υπρ-κλειδί (superkey) – ένα σύνολο γνωρισμάτων μιας σχέσης για το οποίο κάθε πλειάδα σε στιγμιότυπο πρέπει να έχει μοναδική τιμή (τιμές), υποψήφιο κλειδί (candidate key) – ένα ελάχιστο υπέρ-κλειδί, δηλαδή δεν υπάρχει υποσύνολο του που να είναι και αυτό υπέρ-κλειδί (συνήθως ονομάζεται κλειδί) και κύριο κλειδί (primary key) – ένα από τα υποψήφια κλειδιά που συμφωνείται να παίξει το ρόλο του προσδιοριστή για τις πλειάδες της σχέσης. Είναι κοινή τακτική τα κύρια κλειδιά να αποτελούνται από αριθμητικούς και όχι αλφαριθμητικά στοιχεία, καθώς είναι γρηγορότερη, ευκολότερη και γενικά αποδοτικότερη η διαχείριση μιας Βάσης Δεδομένων αν οι πλειάδες προσδιορίζονται από αριθμούς.

Ακεραιότητα οντότητας: Το κύριο κλειδί (primary key) στο σχήμα κάθε σχέσης δεν μπορεί να έχει κενές (INSERT) τιμές σε πλειάδες αυτών των σχέσεων. Αυτό οφείλεται στο γεγονός ότι κάθε κύριο κλειδί έχει το ρόλο του προσδιορισμού των πλειάδων στη σχέση, έτσι δεν είναι δυνατόν να είναι κενό μέσα σε μία πλειάδα.

Αναφορική ακεραιότητα: Αυτός ο περιορισμός εμπλέκει δύο σχέσεις (για αυτό και θα τον περιγράψουμε πώς εφαρμόζεται στις σχέσεις μας στο τέλος των περιορισμών) και χρησιμοποιείται για να καταγράψει τη συνέπεια σε μια συσχέτιση μεταξύ πλειάδων των δύο σχέσεων. Η πλέον συνήθης μορφή είναι αυτή των εξωτερικών κλειδιών (foreign key), δηλαδή ενός συνόλου γνωρισμάτων σε μια σχέση που αποτελεί κύριο κλειδί σε μια άλλη σχέση. Αλλιώς, λέγεται ότι αυτό το σύνολο γνωρισμάτων αναφέρεται σε μια άλλη πλειάδα (στη σχέση που είναι κύριο κλειδί).

Περιορισμού πεδίου τιμών: Είναι οι κανόνες που ορίζονται για το πεδίο τιμών και κληρονομούνται από τις στήλες (γνωρίσματα) που παίρνουν τιμές από το πεδίο.

Περιορισμοί στηλών: είναι επιπρόσθετοι των περιορισμών πεδίου τιμών και αναφέρονται στις τιμές για τα γνωρίσματα.

Περιορισμοί οριζόμενοι από το χρήστη: Κάθε περιορισμός ακεραιότητας, πέραν αυτών που έχουν ήδη αναφερθεί και συνήθως αποσκοπούν στην υποστήριξη επιχειρηματικών κανόνων, και εξυπηρετούν την πολιτική της εταιρείας, ανάλογα αν κρίνεται απαραίτητη η γνώση κάποιων χαρακτηριστικών ή όχι.

Περιορισμοί ακεραιότητας οντότητας

- **Patient**
  Αυτός πίνακας έχει τα υπερκλειδιά «Patient_Id, First_Name, Last_Name, Town, Street_Name, Street_Number, Postal_Code, Age, Doctor_Id». Από αυτά επιλέξαμε το Patient_Id ως primary key, γιατί καθώς είναι ένα μόνο γνώρισμα και είναι και αριθμός, γεγονός που διευκολύνει γενικά τη διαχείριση τις σχέσεις στη Βάση αφού ως γνωστό indices και joins δουλεύουν αρκετά καλύτερα πάνω σε αριθμητικές τιμές. Για να τηρείται ο περιορισμός ακεραιότητας οντότητας το γνώριμα PatientId δεν πρέπει να έχει κενή τιμή. Παράλληλα, όλα τα γνωρίσματα δεν έχουν μοναδική (unique) τιμή, καθώς μπορούν να υπάρχουν άλλα γνωρίσματα με ίδιες τιμές. Επίσης, οι τιμές First_Name, Last_Name, Age είναι υποχρεωτικό να τις εισάγει ο χρήστης προκειμένου να έχει νόημα η καταχώρηση. Τέλος, το γνώρισμα Doctor_Id δεν μπορεί να έχει τιμή NULL, μιας και όλοι ο ασθενείς πρέπει να έχουν ένα γιατρό.

- **Doctor**
  Αυτός ο πίνακας έχει τα υπερκλειδιά «Doctor_Id, First_Name, Last_Name, Speciality, Experience_Years». Από αυτά επιλέξαμε το Doctor_Id ως primary key, το οποίο δεν πρέπει να έχει κενές τιμές. Θεωρήσαμε ότι όλα τα γνωρίσματα έχουν μοναδικές τιμές, καθώς επίσης και ότι τα γνωρίσματα First_Name, Last_Name, Speciality, Experience_Years είναι υποχρεωτικό να τα συμπληρώσει ο χρήστης για να έχει νόημα η καταχώρηση.

- **Pharmacy**
  Αυτός ο πίνακας έχει τα υπερκλειδιά «Pharmacy_Id, Name, Town, Street_Name, Number, Postal_Code, Phone_Number». Από αυτά επιλέξαμε το Pharmacy_Id ως primary key, το οποίο δεν πρέπει να έχει κενές τιμές. Θεωρήσαμε ότι όλα τα γνωρίσματα έχουν μοναδικές τιμές εκτός από το Phone_Number, μιας και δεν μπορεί να υπάρχουν φαρμακεία με ίδια τηλέφωνα. Παράλληλα, τα γνωρίσματα Name, Town, Street_Name, Number, Postal_Code είναι υποχρεωτικό να τα συμπληρώσει ο χρήστης για να έχει νόημα η καταχώρηση.

- **Drug**
  Αυτός ο πίνακας έχει τα υπερκλειδιά «Drug_Id, Name, Formula, Pharmaceutical_Company_Id». Από αυτά επιλέξαμε το Drug_Id ως primary key, το οποίο δεν πρέπει να έχει κενές τιμές. Θεωρήσαμε ότι τα γνωρίσματα Name, Formula πρέπει να έχουν μοναδική τιμή, σε αντίθεση με το Pharmaceutical_Company_Id που δεν πρέπει να είναι μοναδικό, μιας και πολλά φάρμακα μπορούν να έχουν την ίδια φαρμακευτική εταιρία. Επίσης, το γνώρισμα Pharmaceutical_Company_Id είναι υποχρεωτικό να το συμπληρώσει ο χρήστης για να έχει νόημα η καταχώρηση.

- **PharmaceuticalCompany**
  Αυτός ο πίνακας έχει τα υπερκλειδιά «Pharmaceutical_Company_Id , Name, Phone_Number». Από αυτά επιλέξαμε το Pharmaceutical_Company_Id ως primary key, το οποίο δεν πρέπει να έχει κενές τιμές. Θεωρήσαμε ότι τα γνωρίσματα Name και Phone_Number πρέπει να έχουν μοναδική τιμή, ότι δηλαδή δεν μπορεί να υπάρχουν εταιρίες με το ίδιο όνομα ή το ίδιο τηλέφωνο. Επίσης, το γνώρισμα Name είναι υποχρεωτικό να το συμπληρώσει ο χρήστης για να έχει νόημα η καταχώρηση, σε αντίθεση με το Phone_Number που είναι προαιρετικό.

- **Prescription**
  Αυτός ο πίνακας έχει τα υπερκλειδιά «Patient_Id, Doctor_Id, Drug_Id, Date, Quantity». Από αυτά επιλέξαμε τα Patient_Id, Doctor_Id, Drug_Id ως primary keys, τα οποία δεν πρέπει να έχουν κενές τιμές. Θεωρήσαμε ότι όλα τα γνωρίσματα πρέπει να έχουν μοναδική τιμή, καθώς επίσης και ότι είναι υποχρεωτικό να τα συμπληρώσει ο χρήστης για να έχει νόημα η καταχώρηση.

- **Sell**
  Αυτός ο πίνακας έχει τα υπερκλειδιά «Parmacy_Id, Drug_Id, Price». Από αυτά επιλέξαμε τα Parmacy_Id, Drug_Id ως primary keys, τα οποία δεν πρέπει να έχουν κενές τιμές. Θεωρήσαμε ότι όλα τα γνωρίσματα πρέπει να έχουν μοναδική τιμή, καθώς επίσης και ότι είναι υποχρεωτικό να τα συμπληρώσει ο χρήστης για να έχει νόημα η καταχώρηση.

- **Contract**
  Αυτός ο πίνακας έχει τα υπερκλειδιά «Parmacy_Id, Pharmaceutical_Company_Id, Start_Date, End_Date, Text, Supervisor». Από αυτά, επιλέξαμε τα Parmacy_Id, Pharmaceutical_Company_Id ως primary keys, τα οποία δεν πρέπει να έχουν κενές τιμές. Θεωρήσαμε ότι όλα τα γνωρίσματα πρέπει να έχουν μοναδική τιμή εκτός από το Text που πρέπει να είναι μοναδικό. Επίσης, είναι υποχρεωτικό ο χρήστης να συμπληρώσει όλα τα γνωρίσματα για να έχει νόημα η καταχώρηση.

Περιορισμοί αναφορικής ακεραιότητας

- **Patient**
  Περιλαμβάνει το γνώρισμα Doctor_Id που αποτελεί κύριο κλειδί στη σχέση Doctor, συνεπώς εξυπηρετεί τη σύνδεση κάθε ασθενούς με τον αντίστοιχο γιατρό ως εξωτερικό κλειδί στη σχέση Patient. Ως εκ τούτου, το Doctor_Id θέτει συγκεκριμένους περιορισμούς: κατά την εισαγωγή / ενημέρωση ασθενών πρέπει να παίρνει τιμή γιατρού που ήδη υπάρχει. Επίσης, αν διαγραφεί ένας γιατρός τότε διαγράφονται και όλοι οι ασθενείς οι οποίοι παρακολουθούνται από το συγκεκριμένο γιατρό, αφού δεν μπορεί να υπάρξει ασθενής που να μην έχει γιατρό. Αντίθετα, αν υπάρξει κάποια αλλαγή στο γιατρό δεν αλλάζει κάτι στους ασθενείς που παρακολουθούνται από αυτό το γιατρό.

- **Drug**
  Περιλαμβάνει το γνώρισμα Pharmaceutical_Company_Id που αποτελεί κύριο κλειδί στη σχέση PharmaceuticalCompany, συνεπώς εξυπηρετεί τη σύνδεση κάθε φαρμάκου με την αντίστοιχη φαρμακευτική εταιρία ως εξωτερικό κλειδί στη σχέση Drug. Ως εκ τούτου, το Pharmaceutical_Company_Id θέτει συγκεκριμένους περιορισμούς: κατά την εισαγωγή / ενημέρωση φαρμάκων πρέπει να παίρνει τιμή εταιριών που ήδη υπάρχουν. Επίσης, αν διαγραφεί μία εταιρία τότε διαγράφονται και όλα τα φάρμακα τα οποία παράγονται από τη συγκεκριμένη εταιρία, αφού δεν μπορεί να υπάρξει φάρμακο από εταιρία που δεν υπάρχει. Αντίθετα, αν υπάρξει κάποια αλλαγή στην εταιρία δεν αλλάζει κάτι στα φάρμακα που παράγονται από αυτή την εταιρία.

- **Contract**
  Περιλαμβάνει το γνώρισμα Pharmacy_Id που αποτελεί κύριο κλειδί στη σχέση Pharmacy και εξυπηρετεί τη σύνδεση κάθε φαρμακείου με το αντίστοιχο συμβόλαιο ως εξωτερικό κλειδί στη σχέση Contract. Παράλληλα, περιλαμβάνει και το γνώρισμα Drug_Id που αποτελεί κύριο κλειδί στη σχέση Drug και εξυπηρετεί τη σύνδεση κάθε φαρμάκου με το αντίστοιχο συμβόλαιο ως εξωτερικό κλειδί στη σχέση Contract. Ως εκ τούτου, τα γνωρίσματα Pharmacy_Id, Pharmaceutical_Company_Id θέτουν συγκεκριμένους περιορισμούς: κατά την εισαγωγή / ενημέρωση συμβολαίων πρέπει να παίρνουν τιμές φαρμακείων και εταιριών που ήδη υπάρχουν. Επίσης, αν διαγραφεί μία εταιρία ή ένα φαρμακείο, τότε διαγράφονται και όλα τα αντίστοιχα συμβόλαια, αφού δεν μπορεί να υπάρξει συμβόλαιο αν δεν υπάρχει μία εταιρία ή ένα φαρμακείο. Αντίθετα, αν υπάρξει κάποια αλλαγή σε μία εταιρία ή σε ένα φαρμακείο τότε δεν αλλάζει κάτι στα αντίστοιχα συμβόλαια.

- **Sell**
  Περιλαμβάνει το γνώρισμα Pharmacy_Id που αποτελεί κύριο κλειδί στη σχέση Pharmacy και εξυπηρετεί τη σύνδεση κάθε φαρμακείου με την αντίστοιχη πώληση ως εξωτερικό κλειδί στη σχέση Sell. Παράλληλα, περιλαμβάνει και το γνώρισμα Drug_Id που αποτελεί κύριο κλειδί στη σχέση Drug και εξυπηρετεί τη σύνδεση κάθε φαρμάκου με την αντίστοιχη πώληση ως εξωτερικό κλειδί στη σχέση Sell. Ως εκ τούτου, τα γνωρίσματα Pharmacy_Id, Drug_Id θέτουν συγκεκριμένους περιορισμούς: κατά την εισαγωγή / ενημέρωση πωλήσεων πρέπει να παίρνουν τιμές φαρμακείων

και φαρμάκων που ήδη υπάρχουν. Επίσης, αν διαγραφεί ένα φάρμακο ή ένα φαρμακείο, τότε διαγράφονται και όλες οι αντίστοιχες πωλήσεις, αφού δεν μπορεί να υπάρξει πώληση αν δεν υπάρχει αντίστοιχο φάρμακο ή φαρμακείο. Αντίθετα, αν υπάρξει κάποια αλλαγή σε ένα φάρμακο ή σε ένα φαρμακείο τότε δεν αλλάζει κάτι στην αντίστοιχη πώληση.

- **Prescription**
  Περιλαμβάνει το γνώρισμα Patient_Id που αποτελεί κύριο κλειδί στη σχέση Patient και εξυπηρετεί τη σύνδεση κάθε συνταγογράφησης με τον αντίστοιχο ασθενή ως εξωτερικό κλειδί στη σχέση Prescription. Επίσης, έχει το γνώρισμα Doctor_Id που αποτελεί κύριο κλειδί στη σχέση Doctor και εξυπηρετεί τη σύνδεση κάθε συνταγογράφησης με τον αντίστοιχο γιατρό ως εξωτερικό κλειδί στη σχέση Prescription. Παράλληλα, περιλαμβάνει και το γνώρισμα Drug_Id που αποτελεί κύριο κλειδί στη σχέση Drug και εξυπηρετεί τη σύνδεση κάθε φαρμάκου με την αντίστοιχη πώληση ως εξωτερικό κλειδί στη σχέση Prescription. Ως εκ τούτου, τα γνωρίσματα Pharmacy_Id, Doctor_Id , Drug_Id θέτουν συγκεκριμένους περιορισμούς: κατά την εισαγωγή / ενημέρωση συνταγογραφήσεων πρέπει να παίρνουν τιμές ασθενών, γιατρών και φαρμάκων που ήδη υπάρχουν. Αν διαγραφεί ένας ασθενής ή ένα φάρμακο, τότε διαγράφονται και όλες οι αντίστοιχες συνταγογραφήσεις, αφού δεν μπορεί να υπάρξει συνταγογράφηση αν δεν υπάρχει αντίστοιχο φάρμακο ή ασθενής. Αντίθετα, αν διαγραφεί ένας γιατρός τότε δεν αλλάζει κάτι στη συνταγογράφηση. Τέλος, αν υπάρξει κάποια αλλαγή σε ασθενή, σε φάρμακο ή σε γιατρό, δεν αλλάζει κάτι στην αντίστοιχη συνταγογράφηση.

Περιορισμοί πεδίου τιμών

- **Ονόματα:** Όλα τα ονόματα πρέπει να είναι χαρακτήρες και δεν επιτρέπονται κενά
- **Τηλεφωνικοί αριθμοί:** Πρέπει να είναι αριθμοί και μέχρι 15 ψηφία
- **Ταχυδρομικοί κώδικες:** Πρέπει να είναι αριθμοί και μέχρι 5 ψηφία
- **Αριθμοί οδών:** Πρέπει να είναι αριθμοί και από 0 μέχρι 1000
- **Ηλικίες:** Πρέπει να είναι αριθμοί και από 0 μέχρι 150
- **Χρόνια εμπειρίας:** Πρέπει να είναι αριθμοί και από 0 μέχρι 100
- **Ποσότητα:** Πρέπει να είναι θετικός αριθμός
- **Τιμή:** Πρέπει να είναι θετικός αριθμός (float)

User Interface

Στην άσκησή μας, δημιουργήσαμε κατάλληλο user interface με τη βοήθεια των γλωσσών php και html για να γίνονται οι κατάλληλες ενημερώσεις στη βάση μας. Οι επιλογές αυτές γίνονται μέσω ενός αρχικού menu. Για την πραγματοποίηση των βασικών λειτουργιών της βάσης δεδομένων (insert, update, delete) χρησιμοποιούμε την γλώσσα sql. Παρακάτω φαίνονται μερικά παραδείγματα χρήσης της sql για τις παραπάνω λειτουργίες:

Εισαγωγή στον πίνακα doctors

```
INSERT INTO `doctor` (`Doctor_Id`, `First_Name`, `Last_Name`, `Speciality`, `Experience_Years`) VALUES (NULL, 'Giannis', 'Maniatis', 'Psychiatrist', '12');
```

Εισαγωγή στον πίνακα pharmaceuticalcompany

```
INSERT INTO `pharmaceuticalcompany` (`Pharmaceutical_Company_Id`, `Name`, `Phone_Number`) VALUES (NULL, 'Bayer', '2105245698');
```

Ενημέρωση Στοιχείου στον πίνακα patient

```
UPDATE `patient` SET `First_Name` = 'Ioannis', `Last_Name` = 'Papapetrou', `Town ` = 'Athens', `Street_Name` = 'LVouliagmenhs', `Street_Number` = '56', `Age` = '24' WHERE`patient`.`Patient_Id` = 12;
```

Διαγραφή φαρμακείου από τον πίνακα φαρμακείο

```
DELETE FROM `pharmacy` WHERE `pharmacy`.`Pharmacy_Id` = 27
```

Εισαγωγή νέου Συμβολαίου

```
INSERT INTO `contract` (`Pharmacy_Id`, `Pharmaceutical_Company_Id`, `Start_Date`, `End_Date`, `Text`, `Supervisor`) VALUES ('23', '16', '2017-03-25', '2017-03-31', 'πρώωρη λύση συμβολαίου ισοδυναμεί με πρόστιμο 10000 ευρώ', 'JOHN');
```

SQL Queries

Παρακάτω παραθέτουμε τα SQL ερωτήματα που χρησιμοποιήσαμε στη βάση μας σύμφωνα με τις απαιτήσεις της εκφώνησης:

- **<u>Ερώτημα με συνενώσεις (join) :</u>**

  - *ID, Τηλέφωνο των εταιριών και τα ονόματα των φαρμάκων που παράγουν*

    ```
    SELECT pharmaceuticalcompany.Pharmaceutical_Company_Id,
    pharmaceuticalcompany.Phone_Number, drug.Name

    FROM drug

    INNER JOIN pharmaceuticalcompany

    ON pharmaceuticalcompany.Pharmaceutical_Company_Id =
    drug.Pharmaceutical_Company_Id

    ORDER BY pharmaceuticalcompany.Pharmaceutical_Company_Id
    ```

    Στο παραπάνω ερώτημα χρησιμοποιούμε την συνένωση ώστε να βρούμε τα φάρμακα που παράγουν οι φαρμακευτικές εταιρείες και να τυπώσουμε κάποια στοιχεία και των δύο. Η ομαδοποίηση γίνεται μέσω των μοναδικών ids.

  - *Όλα τα φαρμακεία με τις ημερομηνίες των συμβολαίων τους με ημερομηνία έναρξης μετά την 1-1-2014 (αν έχουν), ταξινομημένα κατά όνομα*

    ```
    SELECT pharmacy.Name, contract.Start_Date, contract.End_Date

    FROM pharmacy

    LEFT JOIN contract

    ON pharmacy.Pharmacy_Id = contract.Pharmacy_Id

    WHERE contract.Start_Date >= '2014-01-01'

    ORDER BY pharmacy.Name
    ```

Στο παραπάνω ερώτημα χρησιμοποιούμε τη συνένωση ώστε να βρούμε τις ημερομηνίες συμβολαίων για κάθε φαρμακείο (αν αυτές υπάρχουν) μετά την 1-1-2014. Η ομαδοποίηση γίνεται με βάση το όνομα του φαρμακείου.

- **<u>Ερωτήματα με συναθροιστικές συναρτήσεις (aggregate query):</u>**

  - *Ο αριθμός των γιατρών με ειδικότητα την καρδιολογία*

    ```
    SELECT count(Speciality)
    FROM doctor
    ```

```
WHERE Speciality = 'Cardiologist'
```

Εδώ χρησιμοποιούμε την συναθροιστική συνάρτηση count για να υπολογίσουμε τον αριθμό των γιατρών με ειδικότητα την καρδιολογία

- **Ερωτήματα Ομαδοποίησης (group by):**

➢ *Όνομα, επίθετο και ηλικία των ασθενών που τους παρακολουθεί ο γιατρός με Id = 45,ταξινομημένοι κατά επίθετο*

```
SELECT First_Name, Last_Name, Age
FROM patient
WHERE Doctor_Id = 45
GROUP BY Last_Name
```

- **Ερωτήματα με Ταξινόμηση (order by):**

➢ *Όλα τα στοιχεία του πίνακα πωλήσεων ταξινομημένα κατα αύξουσα τιμή*

```
SELECT *
FROM sell
ORDER BY Price asc
```

- **Ερωτήματα Ομαδοποίησης με Περιορισμό (group by με having):**

➢ *Όνομα, επίθετο και ηλικία των ασθενών που μένουν στην Αθήνα με ηλικία μεγαλύτερη των 20 χρόνων*

```
SELECT First_Name, Last_Name, Age
FROM patient
WHERE Town = 'Athens'
GROUP BY Last_Name
HAVING Age > 20
```

- **Εμφωλευμένα Ερωτήματα (nested query):**

➢ *Όλα τα ονόματα των φαρμάκων με τιμή μεγαλύτερη του μέσου όρου*

```
SELECT d.Name
FROM drug as d, sell as s
WHERE d.Drug_Id = s.Drug_Id and (d.Drug_Id, s.Price) in
        (SELECT Drug_Id, Price
        FROM sell
        WHERE Price > (SELECT AVG(Price) FROM sell))
```

Στο παραπάνω ερώτημα χρησιμοποιούμε εμφωλευμένες εντολές ώστε αφού βρούμε την μέση τιμή των πωλήσεων των φαρμάκων, να βρούμε το id και την τιμή αυτών που την ξεπερνούν. Στη συνέχεια από αυτά επιλέγουμε όσα έχουν κοινό id στις δύο συσχετίσεις sell και drug.

- *Τα φαρμακεία με το όνομα Παπανικολάου που στα κείμενά τους περιέχουν τη συμβολοσειρά Y*

```
SELECT Pharmacy_Id, Town, Street_Name, Number, Postal_Code,
Phone_Number
FROM pharmacy
WHERE Name = 'Papanikolaou'
AND (Pharmacy_Id) IN
        (SELECT Pharmacy_Id
        FROM contract
        WHERE Text LIKE '%Y%')
```

Στο παραπάνω ερώτημα χρησιμοποιούμε εμφωλευμένες εντολές, ώστε αφού βρούμε τα συμβόλαια που περιέχουν στο κείμενο τους την συμβολοσειρά Y να επιλέξουμε τα φαρμακεία με τα συγκεκριμένα συμβόλαια.

- *Επώνυμο, όνομα ασθενή, όνομα γιατρού, όνομα και φόρμουλα φαρμάκου για συνταγογραφήσεις μετά την 1 Μαρτίου και ποσότητα μικρότερη του μέσου όρου*

```
SELECT p.Last_Name, p.First_Name, dr.Doctor_Id, d.Name, d.Formula
FROM patient as p, prescription as pr, doctor as dr, drug as d
WHERE p.Patient_Id = pr.Patient_Id and dr.Doctor_Id = pr.Doctor_Id and
d.Drug_Id = pr.Drug_Id
AND (pr.Date >= '2017-03-01') and (pr.Quantity < (Select avg(Quantity)
FROM prescription))
GROUP BY d.Name
ORDER BY p.Last_Name
```

- **<u>Ερωτήματα Σύγκρισης Συμβολοσειρών (like):</u>**

- *Όλα τα φάρμακα που στη σύσταση τους περιέχουν το συστατικό X*

```
SELECT Name
FROM drug
WHERE Formula LIKE '%X%'
```

- **<u>Ερωτήματα με Πράξεις Συνόλων:</u>**

➢ *Διεύθυνση και Πόλη Ασθενών και Φαρμακείων με ΤΚ μεταξύ 45100 και 45200*

```
(SELECT Street_Number, Street_Name, Town
FROM patient
WHERE Postal_Code between 45100 and 45200)
UNION
(SELECT Number, Street_Name, Town
FROM pharmacy
WHERE Postal_Code between 45100 and 45200)
```

Στο παραπάνω ερώτημα αφού βρούμε τα στοιχεία των φαρμακείων και των ασθενών με ταχυδρομικό κώδικα μεταξύ 45100 και 45200 τα συνενώνουμε με τη χρήση της εντολής union. Με τον τρόπο αυτό αποφεύγουμε επίσης τα duplicates.

Ευρετήρια (Indexes)

Προσπαθήσαμε να ορίσουμε ευρετήρια (indexes) σε γνωρίσματα που χρησιμοποιούνται από την Βάση για να εκτελούνται οι αναζητήσεις αποδοτικότερα και ταχύτερα. Ορίσαμε ευρετήρια στους πίνακες Patient(First_Name, Last_Name) και Doctor(First_Name, Last_Name). Επίσης, τα primary, unique keys των πινάκων είναι indexes. Το είδος των indexes είναι BTREE, λόγω χρήσης InnoDB (phpMyAdmin).

```
ALTER TABLE `doctor` ADD INDEX( `First_Name`, `Last_Name`);

ALTER TABLE ‘patient’ ADD INDEX(`First_Name`, `Last_Name`);
```

Όψεις (Views)

Παρακάτω αναφέρουμε πληροφορίες για τη δημιουργία των δύο όψεων, μίας ενημερώσιμης και μίας μη:

- **Ενημερώσιμη Όψη:**

```
create or replace view pharmacy_view_date as
select Pharmacy_Id, Pharmaceutical_Company_Id, End_Date
 from contract
order by End_Date
```

Στην παραπάνω όψη επιλέγουμε όλα τα γνωρίσματα της σχέσης «συμβόλαιο» και τα ταξινομούμε σύμφωνα με την ημερομηνία τερματισμού τους. Ο χρήστης μπορεί να κάνει αλλαγές σε αυτή.

- **Μη Ενημερώσιμη Όψη:**

```
create or replace view postal_code_ph_p as
select p.Postal_Code, p.First_Name, p.Last_Name, p.Age, ph.Name
from patient as p, pharmacy as ph
where p.Postal_Code = ph.Postal_Code and p.Postal_Code!=0
        and ph.Postal_Code != 0
group by p.Postal_Code
having p.Age > 18
```

Στην παραπάνω όψη επιλέγουμε τα στοιχεία του ασθενή (ταχυδρομικός κώδικας, Όνομα, Επίθετο, Ηλικία) και τα ονόματα των φαρμακευτικών εταιρειών που βρίσκονται στην ίδια περιοχή και κατά συνέπεια έχουν ίδιο ταχυδρομικό κώδικα με την προϋπόθεση ότι αυτός έχει εισαχθεί στη βάση μας. Η ομαδοποίηση γίνεται σύμφωνα με τον Τ.Κ. Επίσης αναζητούμε τους ασθενείς με ηλικία μεγαλύτερη των 18 χρόνων. Ο χρήστης δεν μπορεί να κάνει αλλαγές σε αυτή την όψη.

DDL της Βάσης μας

```
CREATE DATABASE ‘testdb’;

CREATE TABLE `contract` (
  `Pharmacy_Id` int(11) NOT NULL,
  `Pharmaceutical_Company_Id` int(11) NOT NULL,
  `Start_Date` date NOT NULL,
  `End_Date` date NOT NULL,
  `Text` text NOT NULL,
  `Supervisor` varchar(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


CREATE TABLE `doctor` (
  `Doctor_Id` int(11) NOT NULL,
  `First_Name` varchar(255) NOT NULL,
  `Last_Name` varchar(255) NOT NULL,
  `Speciality` varchar(255) NOT NULL,
  `Experience_Years` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;


CREATE TABLE `drug` (
  `Drug_Id` int(11) NOT NULL,
  `Name` varchar(255) NOT NULL,
  `Formula` varchar(255) NOT NULL,
  `Pharmaceutical_Company_Id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
CREATE TABLE `patient` (
  `Patient_Id` int(11) NOT NULL,
  `First_Name` varchar(255) NOT NULL,
  `Last_Name` varchar(255) NOT NULL,
  `Town` varchar(255) DEFAULT NULL,
  `Street_Name` varchar(255) DEFAULT NULL,
  `Street_Number` int(11) DEFAULT NULL,
  `Postal_Code` int(11) DEFAULT NULL,
  `Age` int(11) NOT NULL,
  `Doctor_Id` int(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `pharmaceuticalcompany` (
  `Pharmaceutical_Company_Id` int(11) NOT NULL,
  `Name` varchar(255) NOT NULL,
  `Phone_Number` bigint(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
CREATE TABLE `pharmacy` (
  `Pharmacy_Id` int(11) NOT NULL,
  `Name` varchar(255) NOT NULL,
  `Town` varchar(255) NOT NULL,
  `Street_Name` varchar(255) NOT NULL,
  `Number` int(11) NOT NULL,
  `Postal_Code` int(11) NOT NULL,
  `Phone_Number` bigint(11) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `pharmacy_view_date` (
`Pharmacy_Id` int(11)
,`Pharmaceutical_Company_Id` int(11)
,`End_Date` date
);
```

```
CREATE TABLE `postal_code_ph_p` (
`Postal_Code` int(11)
,`First_Name` varchar(255)
,`Last_Name` varchar(255)
,`Age` int(11)
,`Name` varchar(255)
);

CREATE TABLE `prescription` (
  `Patient_Id` int(11) NOT NULL,
  `Drug_Id` int(11) NOT NULL,
  `Date` date NOT NULL,
  `Quantity` int(11) NOT NULL,
  `Doctor_Id` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `sell` (
  `Pharmacy_Id` int(11) NOT NULL,
  `Drug_Id` int(11) NOT NULL,
  `Price` float NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

ALTER TABLE `contract`
  ADD PRIMARY KEY (`Pharmacy_Id`,`Pharmaceutical_Company_Id`),
  ADD KEY `Pharmaceutical_Company_Id` (`Pharmaceutical_Company_Id`);

ALTER TABLE `doctor`
  ADD PRIMARY KEY (`Doctor_Id`);
```

```
ALTER TABLE `drug`
  ADD PRIMARY KEY (`Drug_Id`),
  ADD UNIQUE KEY `Name` (`Name`),
  ADD UNIQUE KEY `Formula` (`Formula`),
  ADD KEY `drug_ibfk_1` (`Pharmaceutical_Company_Id`);

ALTER TABLE `patient`
  ADD PRIMARY KEY (`Patient_Id`),
  ADD UNIQUE KEY `Patient_Id` (`Patient_Id`),
  ADD UNIQUE KEY `Patient_Id_2` (`Patient_Id`),
  ADD KEY `patient_ibfk_1` (`Doctor_Id`);

ALTER TABLE `pharmaceuticalcompany`
  ADD PRIMARY KEY (`Pharmaceutical_Company_Id`),
  ADD UNIQUE KEY `Name` (`Name`),
  ADD UNIQUE KEY `Phone_Number` (`Phone_Number`);

ALTER TABLE `pharmacy`
  ADD PRIMARY KEY (`Pharmacy_Id`),
  ADD UNIQUE KEY `Phone_Number` (`Phone_Number`);

ALTER TABLE `prescription`
  ADD PRIMARY KEY (`Patient_Id`,`Drug_Id`,`Doctor_Id`),
  ADD KEY `Drug_Id` (`Drug_Id`),
  ADD KEY `Doctor_Id` (`Doctor_Id`);

ALTER TABLE `sell`
  ADD PRIMARY KEY (`Pharmacy_Id`,`Drug_Id`),
  ADD KEY `Drug_Id` (`Drug_Id`);

ALTER TABLE `doctor`
  MODIFY `Doctor_Id` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=50;
```

```
ALTER TABLE `drug`

  MODIFY `Drug_Id` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=12;


ALTER TABLE `patient`

  MODIFY `Patient_Id` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=17;

ALTER TABLE `pharmaceuticalcompany`

  MODIFY `Pharmaceutical_Company_Id` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=17;


ALTER TABLE `pharmacy`

  MODIFY `Pharmacy_Id` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=28;


ALTER TABLE `contract`

  ADD CONSTRAINT `contract_ibfk_1` FOREIGN KEY
(`Pharmaceutical_Company_Id`) REFERENCES `pharmaceuticalcompany`
(`Pharmaceutical_Company_Id`) ON DELETE CASCADE ON UPDATE NO ACTION,

  ADD CONSTRAINT `contract_ibfk_2` FOREIGN KEY (`Pharmacy_Id`)
REFERENCES `pharmacy` (`Pharmacy_Id`) ON DELETE CASCADE ON UPDATE
NO ACTION;


ALTER TABLE `drug`

  ADD CONSTRAINT `drug_ibfk_1` FOREIGN KEY (`Pharmaceutical_Company_Id`)
REFERENCES `pharmaceuticalcompany` (`Pharmaceutical_Company_Id`) ON
DELETE CASCADE ON UPDATE NO ACTION;


ALTER TABLE `patient`

  ADD CONSTRAINT `patient_ibfk_1` FOREIGN KEY (`Doctor_Id`) REFERENCES
`doctor` (`Doctor_Id`) ON DELETE CASCADE ON UPDATE NO ACTION;


ALTER TABLE `prescription`

  ADD CONSTRAINT `prescription_ibfk_1` FOREIGN KEY (`Patient_Id`)
REFERENCES `patient` (`Patient_Id`) ON DELETE CASCADE ON UPDATE NO
ACTION,
```

```
  ADD CONSTRAINT `prescription_ibfk_2` FOREIGN KEY (`Drug_Id`)
REFERENCES `drug` (`Drug_Id`) ON DELETE CASCADE ON UPDATE NO
ACTION,

  ADD CONSTRAINT `prescription_ibfk_3` FOREIGN KEY (`Doctor_Id`)
REFERENCES `doctor` (`Doctor_Id`) ON DELETE NO ACTION ON UPDATE NO
ACTION;


ALTER TABLE `sell`

  ADD CONSTRAINT `sell_ibfk_1` FOREIGN KEY (`Drug_Id`) REFERENCES `drug`
(`Drug_Id`) ON DELETE CASCADE ON UPDATE NO ACTION,

  ADD CONSTRAINT `sell_ibfk_2` FOREIGN KEY (`Pharmacy_Id`) REFERENCES
`pharmacy` (`Pharmacy_Id`) ON DELETE CASCADE ON UPDATE NO ACTION;
```