ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ: « ΜΙΚΡΟΒΙΟΛΟΓΙΚΟ ΕΡΓΑΣΤΗΡΙΟ »

Φοιτήτρια: Πολύζου Αικατερίνη , Α. Μ . : 600

Εισηγητής: Παπαδάκης Νικόλαος

Έτος 2013-2014

Πρόλογος :

Σκοπός αυτής της εργασίας είναι η κατασκευή ενός πληροφοριακού συστήματος για ένα μικροβιολογικό εργαστήριο. Η βάση δεδομένων που χρησιμοποιείται είναι η postgres και η γλώσσα για την διασύνδεση του web interface είναι η php.

Περιγραφή :

Θέλουμε να αποθηκεύουμε την παρακάτω πληροφορία για ένα μικροβιολογικό εργαστήριο.

- Τα στοιχεία κάθε ασθενή. Όνομα, επίθετο τηλέφωνο διεύθυνση και αριθμό κοινωνικών ασφαλίσεων.
- Τα διάφορα είδη εξετάσεων που υπάρχουν. Όνομα, κωδικό, αρρώστια για την οποία γίνεται.
- Τα διάφορα αντιδραστήρια που υπάρχουν. Όνομα , κωδικός και αποθέματα που υπάρχουν. Επίσης και το ελάχιστο αποθεματικό που πρέπει να έχουμε από το κάθε αντιδραστήριο.
- Οι παραγγελίες που πραγματοποιούνται για την προμήθεια νέων ποσοτήτων αντιδραστηρίων. Έχουν κωδικό, ημερομηνία που έγινε και ημερομηνία που παραλείφθηκε.
- Οι πελάτες κάνουν πολλές εξετάσεις και μια εξέταση γίνεται από πολλούς πελάτες. Ένας πελάτη μπορεί να κάνει την ίδια εξέταση σε διαφορετικές ημερομηνίες. Θέλουμε να αποθηκεύομε την ημερομηνία και τα αποτελέσματα αυτής. Επίσης θέλομε να αποθηκεύομε αν ο πελάτης έχει πάρει ή όχι τα αποτελέσματα των εξετάσεων.
- Μια εξέταση χρησιμοποιεί κάποια αντιδραστήρια. Ένα αντιδραστήριο μπορεί να χρησιμοποιείται από πολλές εξετάσεις. Κάθε εξέταση χρησιμοποιεί διαφορετική ποσότητα αντιδραστηρίου από κάποια άλλη εξέταση που χρησιμοποιεί το ίδιο αντιδραστήριο.
- Μια παραγγελία μπορεί να αφορά πολλά αντιδραστήρια. Θέλουμε να αποθηκεύομε την ποσότητα του καθενός.
- Το σύστημα πρέπει να υποστηρίζει τις παρακάτω διεργασίες :
    1. Εισαγωγή όλης της παραπάνω πληροφορίας τμηματικά. Πρέπει να υποστηρίζεται η εισαγωγή εξετάσεων που θέλει να πραγματοποιήσει κάποιος ασθενής. Πρέπει να δίνεται η δυνατότητα ενημέρωση για τα αποτελέσματα αυτών των εξετάσεων. Επίσης κάθε φορά που γίνεται ενημέρωση για εξετάσεις που θέλει να πραγματοποιήσει κάποιος πελάτης να γίνεται αυτόματη μείωση των αποθεμάτων των αντιδραστηρίων.
    2. Συγκεντρωτική αναφορά των πελατών που δεν έχουν πάρει ακόμα τα αποτελέσματα κάποιων εξετάσεων που έκαναν.
    3. Κάθε φορά που θα τα αποθέματα κάποιου αντιδραστηρίου πέφτουν κάτω από τα ελάχιστα όρια να εμφανίζεται το μήνυμα αμέσως μετά την εισαγωγή των εξετάσεων που προκάλεσαν αυτήν την μείωση.
    4. Ενημέρωση για παραλαβή κάποιας παραγγελίας. Αυτό έχει ως αποτέλεσμα την αυτόματη αύξηση των αποθεμάτων των αντιδραστηρίων.

Περιεχόμενα

Εισαγωγή

Οι βάσεις δεδομένων αποτελούν ένα τόσο σημαντικό κομμάτι της σημερινής ζωής που συχνά αγνοούμε ότι χρησιμοποιούμε κάποια. Με σκοπό την καλύτερη κατανόηση των βάσεων δεδομένων μπορούμε να φανταστούμε μία βάση σαν μία συλλογή από δεδομένα και το σύστημα διαχείρισης της βάσης δεδομένων σαν το λογισμικό το οποίο διαχειρίζεται και ελέγχει την πρόσβαση σε αυτή.

Όταν για παράδειγμα αγοράζουμε προϊόντα από κάποιο supermarket είναι πολύ πιθανό ότι θα υπάρξει πρόσβαση σε κάποια βάση. Ο ταμίας είναι πολύ πιθανό να επεξεργαστεί με κάποια συσκευή το bar code του προϊόντος που αγοράσαμε. Αυτή θα συνδεθεί με κάποια εφαρμογή η οποία θα συνδεθεί σε κάποια βάση και θα πάρει την τιμή του προϊόντος που αγοράσαμε. Στη συνέχεια η ίδια εφαρμογή θα ελαττώσει στο stock του supermarket τον αριθμό του προϊόντος κατά μία μονάδα. Αν ο αριθμός των προϊόντων πέσει κάτω από κάποια τιμή , τότε η εφαρμογή θα μπορούσε αυτόματα να κάνει παραγγελία για τα προϊόντα που στο stock ο αριθμός τους έχει πέσει κάτω από κάποια συγκεκριμένη τιμή.

Ανάλογα όταν αγοράζουμε κάτι μέσω πιστωτικής κάρτας πάντα γίνεται έλεγχος για το αν έχουμε αρκετό πιστωτικό όριο για την αγορά μας ή όταν κλείνουμε εισιτήρια για διακοπές σε κάποιο ταξιδιωτικό γραφείο πάλι ο πράκτορας συνδέεται σε κάποια βάση για να κάνει την κράτηση των εισιτηρίων και για να δει αν υπάρχουν ελεύθερα εισιτήρια ή είναι όλα δεσμευμένα. Παρόμοιες ενέργειες συμβαίνουν π.χ. στην περίπτωση που δανειζόμαστε κάποιο βιβλίο από κάποια βιβλιοθήκη αλλά και σε πολλές άλλες περιπτώσεις που μπορεί να μην γνωρίζουμε την ύπαρξη κάποιας βάσης δεδομένων.

ΙΣΤΟΡΙΚΗ ΑΝΑΔΡΟΜΗ

Η τεχνολογία των υπολογιστών έχει επιφέρει μια μόνιμη αλλαγή στο τρόπο με τον οποίο λειτουργούν οι επιχειρήσεις σε ολόκληρο τον κόσμο. Πληροφορίες που παλαιότερα βρίσκονταν αποθηκευμένες σε συρτάρια μπορούν πλέον να είναι προσβάσιμες άμεσα και ταχύτατα με το πάτημα ενός κουμπιού. Παραγγελίες που δίνονται από πελάτες σε άλλες χώρες μπορούν στιγμιαία να επεξεργαστούν στο εργοστάσιο κατασκευής.
Εκτός από την ανάπτυξη του σχεσιακού μοντέλου βάσης δεδομένων, δύο άλλες τεχνολογίες έχουν οδηγήσει στην ταχύτατη άνοδο των Συστημάτων Βάσεων Δεδομένων Πελάτη/Εξυπηρετητή (client/server datbase systems).
Η πρώτη σημαντική τεχνολογία ήταν ο προσωπικός υπολογιστής (Personal Computer, PC). Φθηνές, ευκολόχρηστες εφαρμογές όπως το Lotus 1-2-3 και το Word Perfect έδωσαν την δυνατότητα σε εργαζόμενους αλλά και χρήστες οικιακών υπολογιστών να δημιουργούν έγγραφα και να επεξεργάζονται δεδομένα εύκολα και με ακρίβεια. Οι χρήστες εξοικειώθηκαν με την συνεχή αναβάθμιση των συστημάτων τους σε ολοένα και πιο προηγμένα συστήματα, λόγω του ταχύτατου ρυθμού αλλαγών αλλά και της πτώσης των τιμών.
Η δεύτερη σημαντική τεχνολογία ήταν τα τοπικά δίκτυα (Local Area Network - LAN) και η ενσωμάτωση τους στα γραφεία των επιχειρήσεων σε όλο τον κόσμο. Παλαιότερα, οι χρήστες ήταν εξοικειωμένοι με συνδέσεις μέσω τερματικού (terminal) στα κεντρικά γραφεία των επιχειρήσεων τα οποία συνήθως ήταν εξοπλισμένα με υπολογιστές mainframe. Καθώς όμως τα έγγραφα των εφαρμογών γραφείου (επεξεργασία κειμένου, λογιστικά φύλλα κλπ) θα μπορούσαν να αποθηκευθούν σε ένα κεντρικό σημείο ώστε να είναι προσβάσιμα από οποιοδήποτε υπολογιστή που ήταν συνδεμένος στο δίκτυο, η χρήση των τοπικών δικτύων διαδόθηκε ραγδαία. Από την στιγμή που η εταιρεία υπολογιστών Apple κυκλοφόρησε τον Macintosh και εισήγαγε ένα νέο γραφικό περιβάλλον χρήσης (Graphical User Interface - GUI), οι προσωπικοί υπολογιστές δεν ήταν πλέον μόνο πανίσχυροι και φθηνοί, αλλά έγιναν και εύκολοι στην χρήση.
Στην διάρκεια αυτής της εποχής των αλλαγών και της προόδου, ένας νέος τύπος συστήματος έκανε την εμφάνισή του. Λεγόταν σύστημα πελάτη/εξυπηρετητή (client/server) επειδή η επεξεργαστική απαίτηση μοιραζόταν πλέον μεταξύ του υπολογιστή-πελάτη και του εξυπηρετητή όπου βρίσκονταν το σύστημα βάσης δεδομένων. Η τεχνολογία αυτή ερχόταν να ανατρέψει την επικρατούσα αρχιτεκτονική mainframe, όπου όλη η επεξεργασία γινόταν σε κάποιον κεντρικό υπολογιστή mainframe που χρησιμοποιούνταν μέσω τερματικού (terminal), δηλαδή ενός πληκτρολογίου και μιας οθόνης χαρακτήρων.
Αυτός ο νέος τρόπος λειτουργίας των συστημάτων απαιτεί την χρήση νέων εργαλείων ανάπτυξης. Το περιβάλλον χρήσης είναι πλέον σχεδόν αποκλειστικά γραφικό, μέσω των λειτουργικών συστημάτων MS Windows, Apple Macintosh ή X-Window συστήματα UNIX. Με τη χρήση της SQL και μιας σύνδεσης δικτύου, οι εφαρμογές πλέον μπορούν να έχουν πρόσβαση στο σύστημα DBMS που βρίσκεται εγκατεστημένο σε έναν απομακρυσμένο server. Η ολοένα και αυξανόμενη ισχύς των προσωπικών υπολογιστών επιτρέπει την χρήση δεδομένων τα οποία βρίσκονται καταχωρημένα σε ένα φθηνό, σε σύγκριση με τα mainframe, υπολογιστικό σύστημα. Επιπρόσθετα, με την αντικατάσταση του hardware του server μπορεί να αναβαθμιστεί η απόδοση ολόκληρου του συστήματος client/server χωρίς τα υπάρχει ανάγκη για την παραμικρή αλλαγή στους πελάτες-clients.

Με την εξέλιξη και την ολοένα μεγαλύτερη διάδοση του Internet και των νέων τεχνολογιών που έχουν αναπτυχθεί πάνω και γύρω από αυτό, οι βάσεις δεδομένων παίζουν πλέον κυρίαρχο ρόλο. Η εμφάνιση και η ανάπτυξη του ηλεκτρονικού εμπορίου και οι νέες συνθήκες ανταγωνισμού στην παγκόσμια αγορά δημιουργούν νέα πεδία εφαρμογών για τις βάσεις δεδομένων και αποτελούν κινητήριο μοχλό για την ενσωμάτωση ολοένα και περισσοτέρων δυνατοτήτων στα συστήματα DBMS.

ΚΕΦΑΛΑΙΟ 1

1. Τι είναι μία Βάση Δεδομένων.

Ορισμός : Αποτελεί μία διαμοιρασμένη συλλογή λογικά συσχετισμένων στοιχείων και μία περιγραφή αυτών σχεδιασμένη να ικανοποιεί τις πληροφοριακές ανάγκες κάποιου οργανισμού

Βάση δεδομένων είναι ένας και μοναδικός χώρος αποθήκευσης των δεδομένων ο οποίος ορίζεται μία φορά και μπορεί να χρησιμοποιηθεί ταυτόχρονα από πολλούς τελικούς χρήστες .Έτσι αντί να έχουμε ξεχωριστά αρχεία με πλεονάζουσα πληροφορία, η βάση αποτελεί πλέον μία διαμοιρασμένη συλλογική πηγή .Μία βάση δεδομένων κρατάει εκτός από τα στοιχεία κάποιου οργανισμού και μία περιγραφή των στοιχείων αυτών .

Η προσέγγιση αυτή των βάσεων δεδομένων, να διατηρούνται δηλαδή ξεχωριστά τα στοιχεία της βάσης από τα διάφορα προγράμματα εφαρμογής, μοιάζει πολύ με την προσέγγιση των σύγχρονων γλωσσών προγραμματισμού οπού για κάθε αντικείμενο παρέχεται ένα εσωτερικός ορισμός, αλλά και ένας ξεχωριστός εξωτερικός ορισμός. Έτσι οι τελικοί χρήστες ενός αντικειμένου γνωρίζουν μόνο τον εξωτερικό ορισμό του και έχουν άγνοια για το πώς έχει οριστεί αυτό το αντικείμενο και πως αυτό λειτουργεί .Έτσι με αυτή την προσέγγιση μπορούμε να αλλάξουμε τον εσωτερικό ορισμό ενός αντικειμένου χωρίς να επηρεαστεί ο τελικός χρήστης εφόσον ο εξωτερικός ορισμός του αντικειμένου παραμείνει ο ίδιος .Με το ίδιο σκεπτικό η βάση δεδομένων ξεχωρίζει την δομή των δεδομένων από τα προγράμματα εφαρμογών και την αποθηκεύει στη βάση .Εάν νέες δομές δεδομένων προστεθούν ή παλιές αλλαχθούν τα προγράμματα εφαρμογών δεν επηρεάζονται (παραμένουν τα ίδια) εφόσον οι δομές που αλλάχθηκαν δεν επηρεάζουν άμεσα τα προγράμματα εφαρμογών

Τι είναι ένα Σύστημα Διαχείρισης Βάσης Δεδομένων (DBMS)

Ορισμός : Είναι ένα πρόγραμμα λογισμικού (Software) το οποίο επιτρέπει στους χρήστες να ορίζουν να δημιουργούν και να διαχειρίζονται τη βάση, ενώ παράλληλα παρέχει και έλεγχο της πρόσβασης στη βάση .

Τα τελευταία χρόνια, τα Συστήματα Διαχείρισης Βάσεων Δεδομένων (DBMS-DataBase Management System) έχουν καθιερωθεί σαν το πρωταρχικό μέσο

καταχώρησης δεδομένων για συστήματα πληροφοριών που κυμαίνονται από τα μεγαλύτερα τραπεζικά συστήματα συναλλαγών μέχρι μικροεφαρμογές για συστήματα προσωπικών υπολογιστών. Στην καρδιά των περισσοτέρων σημερινών πληροφοριακών συστημάτων υπάρχει ένα Σύστημα Διαχείρισης Σχεσιακών Βάσεων Δεδομένων (RDBMS-Relational DataBase Management System). Τα συστήματα RDBMS είναι η κινητήρια δύναμη για συστήματα διαχείρισης πληροφοριών εδώ και περισσότερο από μια δεκαετία και εξακολουθούν να εξελίσσονται και να προσφέρουν όλο και πιο προηγμένα συστήματα αποθήκευσης, ανάκτησης και διανομής δεδομένων.

Το σύστημα διαχείρισης της βάσης στην ουσία παρεμβάλλεται ανάμεσα στα διάφορα προγράμματα εφαρμογών που χρησιμοποιούν οι τελικοί χρήστες και στην ίδια τη βάση .Συνήθως ένα σύστημα διαχείρισης βάσης παρέχει τις παρακάτω ευκολίες :

- Επιτρέπει στους χρήστες να ορίσουν μία βάση συνήθως μέσω κάποιας γλώσσας ορισμού δεδομένων (DDL : Data Definition Language). Η γλώσσα αυτή επιτρέπει στους χρήστες να ορίσουν διάφορους τύπους δεδομένων της βάσης αλλά και περιορισμούς πάνω στα δεδομένα που θα αποθηκευτούν στη βάση

- Δίνει τη δυνατότητα στους χρήστες να εισάγουν , να αλλάξουν , να σβήσουν και να τραβήξουν δεδομένα από τη βάση συνήθως μέσω κάποιας γλώσσας διαχείρισης δεδομένων (DML : Data Manipulation Language) όπως για παράδειγμα είναι η SQL

- Αποτελεί ένα σύστημα προστασίας το οποίο προστατεύει και δεν επιτρέπει μη εξουσιοδοτημένη πρόσβαση στους χρήστες.

- Είναι ένα σύστημα ακεραιότητας το οποίο διατηρεί την ακεραιότητα των αποθηκευμένων δεδομένων της βάσης .

- Αποτελεί ένα σύστημα σύγχρονου ελέγχου το οποίο επιτρέπει την από κοινού, (ταυτόχρονη) πρόσβαση στους τελικούς χρήστες.

- Είναι ένα σύστημα αποκατάστασης το οποίο επαναφέρει τη βάση σε μία προηγούμενη σταθερή κατάσταση μετά από κάποια hardware ή software αποτυχία .

- Είναι ένα κατάλογο προσβάσιμο από τους χρήστες ο οποίος περιέχει μία περιγραφή των δεδομένων της βάσης .

- Παρέχει ένα μηχανισμό όψεων ο οποίος επιτρέπει σε κάθε χρήστη να έχει την δικιά του όψη στη βάση δηλ. να βλέπει τα δεδομένα όπως αυτός θέλει ή συνήθως όπως κρίνεται αναγκαίο για λόγους ασφαλείας της βάσης από τον διαχειριστή της βάσης.

Να διευκρινιστεί ότι η παραπάνω αναφορά για τα συστήματα διαχείρισης βάσεων είναι γενική .Το πραγματικό επίπεδο διευκόλυνσης που παρέχει το

εκάστοτε σύστημα διαχείρισης δεν είναι πάντα το ίδιο και διαφέρει από προϊόν σε προϊόν .

Πλεονεκτήματα και Μειονεκτήματα των Συστημάτων Διαχείρισης Βάσεων Δεδομένων

Πλεονεκτήματα :

- **Έλεγχος του πλεονασμού δεδομένων :** Όπως αναφέρθηκε παραπάνω τα παραδοσιακά συστήματα αρχείων σπαταλούσαν αρκετό χώρο με το να αποθηκεύουν τα ίδια δεδομένα σε περισσότερα από ένα αρχεία . Αντιθέτως, τα συστήματα βάσεων δεδομένων προσπαθούν να εξαλείψουν τον πλεονασμό τελείως ενσωματώνοντας τα αρχεία έτσι ώστε να μην υπάρχουν πολλά αντίγραφα των ιδίων δεδομένων .Παρ` όλα αυτά οι βάσεις δεδομένων δεν εξαφανίζουν τελείως τον πλεονασμό των δεδομένων, αφού σε πολλές περιπτώσεις χρειάζεται να έχουμε επανάληψη των ιδίων δεδομένων όπως για παράδειγμα στην υλοποίηση σύνθετων σχέσεων (relationships) ανάμεσα στα στοιχεία της βάσης.
- **Συνεκτικότητα των Δεδομένων :** Με την εξαφάνιση ή τον έλεγχο του πλεονασμού των δεδομένων ελαττώνουμε τον κίνδυνο εμφάνισης μη συνεκτικών δεδομένων .Εάν τα δεδομένα είναι αποθηκευμένα μονάχα μία φορά στη βάση, οποιαδήποτε ενημέρωση στις τιμές τους εκτελείται μία φορά και η νέα τιμή είναι κατευθείαν διαθέσιμη σε όλους τους τελικούς χρήστες . Εάν πάλι τα ίδια δεδομένα είναι αποθηκευμένα περισσότερες από μία φορές στη βάση και το σύστημα διαχείρισης είναι ενήμερο, μπορεί να εγγυηθεί ότι όλα τα αντίγραφα θα κρατηθούν ενημερωμένα .Δυστυχώς όμως μέχρι και σήμερα δεν μπορούν όλα τα υπάρχοντα στο εμπόριο συστήματα διαχείρισης βάσεων να εγγυηθούν αυτή τη συνεκτικότητα των δεδομένων.
- **Επιπλέον Πληροφορίες Από Τα Ίδια Δεδομένα :** Μέσω της ενσωμάτωσης των δεδομένων καθίσταται δυνατό για έναν οργανισμό να αντλήσει από τα δεδομένα της βάσης επιπλέον πληροφορίες, είτε μέσω

συναρτήσεων στατιστικών του συστήματος διαχείρισης της βάσης, είτε μέσω της συνένωσης πινάκων .

- **Κοινοποίηση Δεδομένων :** Τυπικά, τα αρχεία ανήκουν σε όλους τους εξουσιοδοτημένους χρήστες και έτσι οι περισσότεροι χρήστες μπορούν να μοιραστούν τα δεδομένα .Επιπλέον οι εφαρμογές μπορούν να επεκτείνουν τα υπάρχοντα δεδομένα προσθέτοντας απλά τα νέα δεδομένα στη βάση, χωρίς να χρειάζεται να ορίσουν ξανά όλα τα δεδομένα. Οι εφαρμογές επίσης μπορούν να βασίζονται στις συναρτήσεις του συστήματος διαχείρισης χωρίς να χρειάζεται να έχουν τις δικές τους συναρτήσεις.
- **Βελτιωμένη Ακεραιότητα Δεδομένων :** Η ακεραιότητα εκφράζει συνήθως τους διάφορους περιορισμούς , οι οποίοι είναι στην ουσία κανόνες, τους οποίους η βάση δεν πρέπει να παραβαίνει . Οι περιορισμοί αυτοί μπορεί να εφαρμόζονται στα δεδομένα ενός πεδίου (γνωρίσματος), ενός πίνακα, ή μπορεί να εφαρμόζονται και στις σχέσεις μεταξύ των πινάκων .Για παράδειγμα, στο πεδίο (γνώρισμα) μιας email διευθύνσεως θα θέλαμε να υπάρχει το σύμβολο @ υποχρεωτικά .
- **Βελτιωμένη Ασφάλεια :** Η ασφάλεια μίας βάσης δεδομένων αποτελεί την προστασία της απέναντι σε μη εξουσιοδοτημένους χρήστες .Χωρίς τα απαραίτητα μέτρα η συνένωση των αρχείων κάνει τα δεδομένα ακόμα πιο επιρρεπή και ευάλωτα σε σχέση με τα συστήματα αρχείων .Έτσι τα συστήματα διαχείρισης βάσεων επιτρέπουν στον administrator να ορίσει και να επιβάλλει την ασφάλεια της βάσης . Αυτό μπορεί να γίνει με τη μορφή ονόματος χρήστη και κωδικού έτσι ώστε να ορισθούν οι εξουσιοδοτημένοι χρήστες . Επιπλέον ορίζονται και τα δικαιώματα που μπορεί να έχει ένας χρήστης ή ένα γκρουπ χρηστών στους διάφορους πίνακες της βάσης. Αξίζει διευκρινιστεί ότι δίνεται και η δυνατότητα ορισμού διαφορετικών δικαιωμάτων για τον ίδιο χρήστη σε κάθε πίνακα της βάσης .
- **Βελτιωμένη Διαθεσιμότητα και Απόκριση :** Σαν αποτέλεσμα της ενσωμάτωσης των αρχείων τα δεδομένα είναι απευθείας προσβάσιμα από τον τελικό χρήστη . Τα περισσότερα συστήματα διαχείρισης βάσεων παρέχουν στον τελικό χρήστη γλώσσες υποβολής ερωτήσεων στη βάση, έτσι ώστε ο κάθε χρήστης να μπορεί να λάβει τα στοιχεία που αυτός θέλει,

χωρίς να είναι απαραίτητη η παρουσία κάποιου προγραμματιστή ο οποίος θα γράψει κάποια εφαρμογή για την εξαγωγή στοιχείων από τη βάση .

- **Αυξημένη Παραγωγικότητα :** Όπως αναφέρθηκε και πριν τα διάφορα συστήματα διαχείρισης παρέχουν έτοιμες συναρτήσεις στους προγραμματιστές εφαρμογών ώστε να μην χρειάζεται να ανησυχούν για πολύ χαμηλού επιπέδου λεπτομέρειες . Αυτό έχει ως αποτέλεσμα την αύξηση της παραγωγικότητας των προγραμματιστών και την μείωση του χρόνου ανάπτυξης των διαφόρων εφαρμογών με τελικό αποτέλεσμα και την μείωση του κόστους μίας τέτοιας εφαρμογής .
- **Βελτιωμένη Συντήρηση :** Στα παλαιότερα συστήματα αρχείων η περιγραφή των δεδομένων ήταν ενσωματωμένη μέσα σε κάθε εφαρμογή , κάνοντας έτσι την κάθε εφαρμογή να εξαρτάτε από τα δεδομένα. Μία οποιαδήποτε αλλαγή στη δομή των δεδομένων απαιτούσε και την ανάλογη αλλαγή και στα προγράμματα εφαρμογών που επηρεάζονταν από αυτήν. Αντίθετα στα συστήματα διαχείρισης απομονώνεται η περιγραφή των δεδομένων από τις εφαρμογές με αποτέλεσμα αυτές να μένουν απρόσβλητες από οποιαδήποτε αλλαγή.
- **Αυξημένος Συγχρονισμός :** Σε πολλά από τα παλιά συστήματα αρχείων όταν δύο ή περισσότεροι χρήστες προσπαθούσαν να έχουν πρόσβαση στο ίδιο αρχείο συγχρόνως ήταν πιθανό οι προσβάσεις αυτές να ανακατεύονταν με αποτέλεσμα την απώλεια των πληροφοριών ή ακόμα και την απώλεια της ακεραιότητας . Τα σημερινά συστήματα διαχείρισης όμως εξασφαλίζουν ότι κάτι τέτοιο δεν θα συμβεί .

Μειονεκτήματα :

- **Πολυπλοκότητα :** Η παροχή όλων των λειτουργιών που απαιτούμε από ένα καλό σύστημα διαχείρισης γίνεται από ένα πολύ σύνθετο πρόγραμμα . Οι σχεδιαστές , οι προγραμματιστές , οι διαχειριστές, ακόμα και οι τελικοί χρήστες θα πρέπει να αντιληφθούν τις λειτουργίες το συστήματος διαχείρισης για να μπορέσουν να το εκμεταλλευτούν. Αποτυχία στο να μπορέσουν να αντιληφθούν τις λειτουργίες του συστήματος διαχείρισης θα μπορούσε να οδηγήσει σε λανθασμένες αποφάσεις σχεδίασης με πολλαπλές συνέπειες .

- **Μέγεθος :** Η πολυπλοκότητα και το εύρος των λειτουργιών του συστήματος διαχείρισης το κάνουν ένα πολύ μεγάλο πρόγραμμα με αρκετές απαιτήσεις σε αποθηκευτικό χώρο και μνήμης για να τρέξει ικανά .
- **Κόστος Αγοράς Συστήματος Διαχείρισης :** Το κόστος ενός τέτοιου συστήματος είναι υψηλό.
- **Επιπρόσθετο Κόστος Υλικού (Hardware) :**Οι απαιτήσεις σε αποθηκευτικό χώρο για το σύστημα διαχείρισης είναι πιθανόν να αυξηθούν με αποτέλεσμα την αγορά επιπλέον δίσκων για την κάλυψη των αναγκών της βάσης . Επιπλέον πολλές φορές για να επιτύχουμε την επιθυμητή απόδοση σε χρόνους απόκρισης ίσως χρειαστεί η αγορά νέου υπολογιστή .
- **Κόστος Μετατροπής :** Πολλές φορές το κόστος αλλαγής hardware εξ` αιτίας των παραπάνω λόγων είναι ασήμαντο συγκριτικά με το κόστος μετατροπής των διαφόρων προγραμμάτων εφαρμογών έτσι ώστε αυτά να μπορούν να συνεργαστούν με ένα καινούργιο σύστημα διαχείρισης της βάσης ή με καινούργιο hardware .
- **Επιδόσεις Συστήματος :** Τυπικά ένα παλιό σύστημα αρχείων είναι γραμμένο για μία συγκεκριμένη εφαρμογή με αποτέλεσμα να έχει καλές επιδόσεις . Αντιθέτως ένα σύστημα διαχείρισης είναι γραμμένο πιο γενικά με σκοπό να καλύπτει τις ανάγκες πολλών εφαρμογών και όχι μίας μονάχα . Αυτό έχει ως αποτέλεσμα οι εφαρμογές να μην τρέχουν τόσο γρήγορα όπως θα έτρεχαν σε ένα σύστημα αρχείων .
- **Μεγαλύτερες Επιπτώσεις Σε Αποτυχία :** Η συγκέντρωση όλων των πόρων έχει ως αποτέλεσμα να γίνεται στο σύστημα πιο ευάλωτο . Από τη στιγμή πού όλοι οι χρήστες και οι εφαρμογές βασίζονται στην διαθεσιμότητα του συστήματος διαχείρισης η αποτυχία οποιουδήποτε μέρους μπορεί να οδηγήσει το σύστημα σε προσωρινή παύση .

Το περιβάλλον των βάσεων δεδομένων

Βασικός σκοπός των συστημάτων βάσεων δεδομένων είναι να παρέχει στους χρήστες μία αφαιρετική εικόνα των δεδομένων κρύβοντας συγκεκριμένες λεπτομέρειες για το πώς είναι αυτά αποθηκευμένα και πώς γίνεται η διαχείρισή

τους. Για το λόγο αυτό, η αρχική σχεδίαση της βάσης πρέπει να είναι μία αφαιρετική και γενική περιγραφή των πληροφοριακών αναγκών του οργανισμού που θα αναπαρασταθούν στη βάση.

Αρχιτεκτονική Τριών Επιπέδων (Ansi – Spark)

Το 1975 έγινε μία πρόταση από την American National Standards Institute (ANSI) Standards Planning and Requirements Committee (SPARK) , για την δημιουργία ενός προτύπου αρχιτεκτονικής και ορολογίας, όπου αναγνωρίστηκε η ανάγκη για μία τριών επιπέδων αρχιτεκτονική με κατάλογο συστήματος . Παρά το γεγονός ότι το ANSI-SPARK μοντέλο δεν έγινε πρότυπο, παρέχει μία βάση για την κατανόηση μερικών λειτουργιών του συστήματος διαχείρισης (DBMS).

Η αρχιτεκτονική τριών επιπέδων συμπεριλαμβάνει τα εξής τρία επίπεδα : Ένα εξωτερικό, ένα εσωτερικό και ένα νοητό. Σκοπός της αρχιτεκτονικής τριών επιπέδων είναι να διαχωρίσει την εικόνα που έχουν οι χρήστες για τη βάση από τον τρόπο με τον οποίο η βάση είναι σε φυσικό επίπεδο απεικονισμένη. Υπάρχουν πολλοί λόγοι για τους οποίους ο παραπάνω διαχωρισμός είναι επιθυμητός :

- Κάθε χρήστης πρέπει για τα ίδια δεδομένα να μπορεί να έχει μια διαφορετική προσαρμόσιμη εικόνα. Κάθε χρήστης θα πρέπει να μπορεί να αλλάξει τον τρόπο με τον οποίο βλέπει τα δεδομένα χωρίς αυτό να επηρεάζει τους υπόλοιπους χρήστες.
- Οι χρήστες δεν θα πρέπει να έρχονται αντιμέτωποι απευθείας με λεπτομέρειες σε φυσικό επίπεδο, που αφορούν τη βάση δεδομένων όπως τη δημιουργία ευρετηρίου κλπ. Με άλλα λόγια η αλληλεπίδραση του χρήστη με τη βάση θα πρέπει να είναι ανεξάρτητη από θεωρήσεις αποθήκευσης.
- Ο διαχειριστής της βάσης θα πρέπει να μπορεί να αλλάζει τα δεδομένα της σε φυσικό επίπεδο, χωρίς να επηρεάζει τον τελικό χρήστη και την εικόνα που έχει αυτός για τα δεδομένα.
- Η εσωτερική δομή της βάσης θα πρέπει να μένει ανεπηρέαστη σε αλλαγές των αποθηκευτικών μέσων (σκληρών δίσκων).
- Ο διαχειριστής της βάσης θα πρέπει να μπορεί να αλλάξει τη λογική δομή, χωρίς να επηρεάζει τους υπόλοιπους χρήστες.

Ο τρόπος με τον οποίο οι χρήστες αντιλαμβάνονται τα δεδομένα ονομάζεται εξωτερικό επίπεδο. Ο τρόπος με τον οποίο το σύστημα διαχείρισης της βάσης και το λειτουργικό σύστημα αντιλαμβάνονται τα δεδομένα αποτελεί το εσωτερικό επίπεδο όπου βρίσκονται στην πραγματικότητα αποθηκευμένα τα δεδομένα σύμφωνα με τις αρχές των δομών δεδομένων και της οργάνωσης αρχείων. Το

νοητό επίπεδο είναι αυτό το οποίο παρέχει τόσο τη χαρτογράφηση, όσο και την επιθυμητή ανεξαρτησία ανάμεσα στο εξωτερικό και το εσωτερικό επίπεδο.

Εξωτερικό επίπεδο :

Είναι η εικόνα (view) που έχουν οι χρήστες για τη βάση. Αυτό το επίπεδο περιγράφει το κομμάτι της βάσης που σχετίζεται με κάθε χρήστη .

Το εξωτερικό επίπεδο αποτελείται από ένα αριθμό από διαφορετικές όψεις της βάσης δεδομένων. Το εξωτερικό επίπεδο αποτελείται από εκείνα τα στοιχεία της βάσης που ενδιαφέρουν το χρήστη. Επιπλέον στοιχεία μπορεί να υπάρχουν στη βάση, αλλά ο χρήστης βλέπει μονάχα αυτά που τον ενδιαφέρουν**.**

Νοητό επίπεδο :

Είναι το επίπεδο αυτό περιγράφει το είδος των δεδομένων που είναι αποθηκευμένα στη βάση και τις σχέσεις ανάμεσά τους.

Το ενδιάμεσο επίπεδο στην αρχιτεκτονική τριών επιπέδων είναι το νοητό επίπεδο. Αυτό το επίπεδο περιέχει τις λογικές δομές ολόκληρης της βάσης, όπως τις βλέπει ο διαχειριστής. Αποτελεί μία ολοκληρωμένη εικόνα των δεδομένων ενός οργανισμού και είναι ανεξάρτητο από το φυσικό επίπεδο. Το νοητό επίπεδο αναπαριστά :

- Όλους τους πίνακες , τα πεδία των πινάκων, και τις σχέσεις τους. ((Relationships) .
- Τους περιορισμούς των δεδομένων. Με άλλα λόγια, αυτοί οι περιορισμοί αποτελούν τα πεδία ορισμού των τιμών των δεδομένων της βάσης, δηλαδή τις τιμές που μπορεί ή δεν μπορεί να πάρει ένα πεδίο (γνώρισμα) της βάσης.
- Σημασιολογικές πληροφορίες για τα δεδομένα.
- Πληροφορίες ασφαλείας και ακεραιότητας των δεδομένων.

Το νοητό επίπεδο υποστηρίζει κάθε εξωτερική όψη, δηλαδή όλα τα δεδομένα που πρέπει να παρέχονται σε κάποιο χρήστη πρέπει να περιλαμβάνονται ή να προέρχονται από το νοητό επίπεδο. Παρ` όλα αυτά αυτό το επίπεδο δεν πρέπει να περιλαμβάνει λεπτομέρειες αποθήκευσης των δεδομένων στους δίσκους του συστήματος όπως για παράδειγμα τον αριθμό των bytes που χρησιμοποιούνται ή το που βρίσκονται τα αρχεία της βάσης ή τα δεδομένα.

Εσωτερικό επίπεδο :

Αποτελεί μία φυσική αναπαράσταση της βάσης δεδομένων στον υπολογιστή. Αυτό το επίπεδο περιγράφει τον τρόπο με τον οποίο τα δεδομένα είναι αποθηκευμένα στη βάση.

Το εσωτερικό επίπεδο καλύπτει την φυσική υλοποίηση της βάσης με σκοπό την επίτευξη της βέλτιστης απόδοσης και χρησιμοποίησης χώρου. Καλύπτει τις δομές των δεδομένων και την οργάνωση των αρχείων που χρησιμοποιούνται για την αποθήκευση των δεδομένων στους δίσκους. Συνεργάζεται με τις μεθόδους προσπέλασης του λειτουργικού συστήματος για να τοποθετήσει τα δεδομένα στους δίσκους του συστήματος, να δημιουργήσει τα περιεχόμενα, να προσπελάσει τα ζητούμενα δεδομένα και ούτω καθ`εξής. Το εσωτερικό επίπεδο ασχολείται με τα εξής :

- Δέσμευση χώρου στους δίσκους για τα δεδομένα και τα περιεχόμενα.
- Καταγραφή περιγραφών των δίσκων σχετικών με το μέγεθος των αποθηκευμένων δεδομένων.
- Τοποθέτηση των καταγραφών.
- Τεχνικές συμπίεσης και κρυπτογράφησης δεδομένων.

Κάτω από το εσωτερικό επίπεδο βρίσκεται το φυσικό επίπεδο το οποίο διαχειρίζεται από το λειτουργικό σύστημα κάτω από τις οδηγίες του συστήματος διαχείρισης της βάσης. Το σύστημα διαχειρίσης της βάσης είναι υπεύθυνο για την χαρτογράφηση ανάμεσα στα τρία αυτά επίπεδα. Το σύστημα διαχείρισης πρέπει να ελέγχει ότι κάθε εξωτερικό σχήμα προέρχεται από το νοητό επίπεδο και πρέπει να χρησιμοποιεί τις πληροφορίες του νοητού επιπέδου για να κάνει την αντιστοίχηση (χαρτογράφηση) ανάμεσα σε κάθε εσωτερικό και εξωτερικό σχήμα. Το νοητό επίπεδο σχετίζεται με το εσωτερικό επίπεδο, μέσω της αντιστοίχησης ανάμεσά τους. Αυτό επιτρέπει στο σύστημα διαχείρισης να βρει την ακριβή εγγραφή ή συνδυασμό εγγραφών στο φυσικό επίπεδο, το οποίο συνιστά μία λογική εγγραφή στο λογικό επίπεδο, μαζί με όποιους περιορισμούς εφαρμόζονται στις ενέργειες για αυτή τη λογική εγγραφή. Τέλος, κάθε εξωτερικό σχήμα σχετίζεται με το νοητό επίπεδο μέσω της αντιστοίχησης ανάμεσα στο εξωτερικό και εσωτερικό επίπεδο.

Ανεξαρτησία Δεδομένων

Ο βασικός σκοπός της αρχιτεκτονικής τριών επιπέδων είναι η επίτευξη ανεξαρτησίας δεδομένων το οποίο σημαίνει ότι τα ανώτερα επίπεδα παραμένουν ανεπηρέαστα από αλλαγές που τυχόν θα συμβούν στα κατώτερα επίπεδα. Υπάρχουν δύο είδη ανεξαρτησίας δεδομένων : λογική και φυσική.

Σαν **Λογική** ανεξαρτησία δεδομένων αναφέρεται η άγνοια του εξωτερικού επιπέδου σχετικά με αλλαγές του νοητού επιπέδου. Αλλαγές στο νοητό επίπεδο, όπως για παράδειγμα η προσθήκη νέων πινάκων δεδομένων ή σχέσεων, πρέπει να είναι δυνατή χωρίς να χρειάζεται να γίνουν αλλαγές στο εξωτερικό σχήμα ή να χρειάζεται να ξαναγραφτούν οι διάφορες εφαρμογές.

Η **Φυσική Ανεξαρτησία Δεδομένων** αναφέρεται στη άγνοια του νοητού επιπέδου σχετικά με αλλαγές που συμβαίνουν στο εσωτερικό επίπεδο. Αλλαγές στο εσωτερικό επίπεδο όπως αλλαγή στην οργάνωση αρχείων ή στις δομές των δίσκων ή και αντικατάστασή τους με καινούργιους, θα πρέπει να είναι δυνατές χωρίς να χρειάζεται να γίνουν αλλαγές στο νοητό ή στο εξωτερικό επίπεδο.

Τι είναι το Σχεσιακό Μοντέλο Δεδομένων

Το σχεσιακό μοντέλο είναι σήμερα το βασικό μοντέλο δεδομένων για εμπορικές εφαρμογές επεξεργασίας δεδομένων. Έχει κερδίσει την πρωτεύουσα θέση, εξαιτίας της απλότητάς του, που διευκολύνει τη δουλειά των προγραμματιστών σε σύγκριση με άλλα μοντέλα δεδομένων. Μια σχεσιακή βάση δεδομένων αποτελείται από ένα σύνολο από πίνακες, καθένας εκ των οποίων έχει ένα μοναδικό όνομα. Πίνακας είναι μία δισδιάστατη δομή δεδομένων. Κάθε στήλη του πίνακα αφορά ένα συγκεκριμένο γνώρισμα (attribute). Μία γραμμή ενός πίνακα αντιπροσωπεύει μία σχέση ενός συνόλου από τιμές για τα αντίστοιχα γνωρίσματα. Αφού ένας πίνακας είναι ένα σύνολο από τέτοιες σχέσεις υπάρχει μια στενή σχέση μεταξύ ενός πίνακα και της μαθηματικής ιδέας της σχέσης (relation), από την οποία παίρνει το όνομά του το σχεσιακό μοντέλο δεδομένων. Δεν είναι απαραίτητο για μια σχέση να έχει γραμμές για να θεωρείται σχέση. Ακόμα και αν η σχέση δεν περιέχει δεδομένα, αυτή παραμένει ορισμένη με το σετ των ιδιοτήτων της

Οι 13 κανόνες του Dr.Codd για το Σχεσιακό Μοντέλο Βάσης Δεδομένων

Το Σχεσιακό Μοντέλο Βάσης Δεδομένων στηρίχθηκε σε μια εργασία του Dr. E.F.Codd από το 1970 με τίτλο «Ένα Σχεσιακό Μοντέλο Δεδομένων για Μεγάλες Κοινόχρηστες Τράπεζες Δεδομένων» (A Relational Model of Data for Large Shared Data Banks). Η γλώσσα SQL αναπτύχθηκε για να εξυπηρετήσει το πρότυπα του σχεσιακού μοντέλου βάσης δεδομένων.
Οι 13 κανόνες, που κατά περίεργο τρόπο ονομάζονται «Δωδεκάλογος του Codd», λένε τα εξής :

0. Ένα σύστημα για να χαρακτηρίζεται ως RDBMS πρέπει να είναι ικανό να διαχειρίζεται βάσεις δεδομένων αποκλειστικά μέσω των σχεσιακών δυνατοτήτων του.
1. Ο κανόνας της Πληροφορίας : Όλες οι πληροφορίες σε μια σχεσιακή βάση δεδομένων (συμπεριλαμβανομένων των ονομάτων των πινάκων και των στηλών τους) θα πρέπει να αναπαριστώνται άμεσα σαν τιμές σε πίνακες.

2. Εγγυημένη Πρόσβαση : Κάθε τιμή σε μια σχεσιακή βάση δεδομένων είναι εγγυημένο ότι θα είναι προσβάσιμη με τη χρήση του ονόματος του πίνακα, της τιμής του πρωτεύοντος κλειδιού και του ονόματος της στήλης.

3. Υποστήριξη της τιμής NULL : Το DBMS θα πρέπει να προσφέρει συστηματική υποστήριξη για τον χειρισμό των τιμών NULL ανεξάρτητα από τις προκαθορισμένες τιμές (default values) ή τους τύπους δεδομένων (data types).

4. Ενεργός, online σχεσιακός κατάλογος : Η περιγραφή της βάσης δεδομένων και των περιεχομένων της θα αναπαριστάται σε λογικό επίπεδο σαν πίνακες στους οποίους θα μπορούν να υποβληθούν ερωτήματα μέσω της γλώσσας της βάσης δεδομένων.

5. Περιεκτική γλώσσα δεδομένων : Τουλάχιστον μία από τις υποστηριζόμενες γλώσσες θα πρέπει να έχει αυστηρά καθορισμένη μορφή και να είναι περιεκτική. Θα πρέπει να υποστηρίζει Ορισμό Δεδομένων (Data Definition), χειρισμό δεδομένων (Data Manipulation), κανόνες ακεραιότητας (Interity Rules), εξουσιοδότηση (Authorization) και Συναλλαγές (Transactions).

6. Κανόνας ενημέρωσης όψεων : Όλες οι όψεις (Views) που είναι θεωρητικά ενημερώσιμες θα πρέπει να μπορούν να ενημερωθούν μέσω του συστήματος.

7. Εισαγωγή, ενημέρωση, διαγραφή σε επίπεδο set : Το DBMS θα πρέπει να υποστηρίζει όχι μόνο ανάκτηση σε επίπεδο set αλλά και εισαγωγή, ενημέρωση και διαγραφή.

8. Ανεξαρτησία των φυσικών δεδομένων : Η εφαρμογή δεν θα πρέπει να επηρεάζεται σε λογικό επίπεδο όταν γίνονται μεταβολές στις δομές αποθήκευσης.

9. Ανεξαρτησία των λογικών δεδομένων : Τα προγράμματα εφαρμογών δεν θα πρέπει να επηρεάζονται όταν γίνονται μεταβολές σε λογικό επίπεδο π.χ. σε πίνακες, στήλες, σειρές κλπ.

10. Ανεξαρτησία της ακεραιότητας : Η γλώσσα της βάσης δεδομένων θα πρέπει να είναι ικανή για τον καθορισμό των κανόνων ακεραιότητας. Οι κανόνες αυτοί θα πρέπει να βρίσκονται καταχωρημένοι στον κατάλογο της βάσης δεδομένων και δεν θα μπορούν να παρακαμφθούν.

11. Ανεξαρτησία της διανομής : Τα προγράμματα εφαρμογών δεν θα πρέπει να επηρεάζονται σε λογικό επίπεδο όταν τα δεδομένα διανέμονται για πρώτη φορά ή όταν επαναδιανέμονται.

12. Μη αντιστρεψιμότητα : Δεν θα πρέπει να είναι δυνατή η παράκαμψη των κανόνων ακεραιότητας που καθορίζονται από την γλώσσα της βάσης δεδομένων με την χρήση γλωσσών χαμηλότερου επιπέδου.

Τι είναι κλειδιά και αναφορική ακεραιότητα

Οι ιδιότητες ομαδοποιούνται με βάση την εξάρτησή τους από ένα πρωτεύον κλειδί (primary key). Πρωτεύον κλειδί είναι μια ιδιότητα (ή μια ομάδα ιδιοτήτων) η οποία ταυτοποιεί μια γραμμή σε έναν πίνακα. Ένας πίνακας μπορεί να έχει το πολύ ένα πρωτεύον κλειδί. Επειδή οι τιμές των κλειδιών αυτών χρησιμοποιούνται για την ταυτοποίηση της γραμμής δεν μπορούν να περιέχουν τιμή NULL.
Μπορεί να υπάρχουν και άλλες ιδιότητες σε μία σχέση των οποίων οι τιμές πρέπει να είναι επίσης μοναδικές. Σε αντίθεση όμως με το πρωτεύον κλειδί αυτές μπορούν να περιέχουν τιμή NULL. Στην πράξη, οι ιδιότητες αυτές, που ονομάζονται "μοναδικά κλειδιά" (unique keys) χρησιμοποιούνται για να αποκλείσουμε την πιθανότητα καταχώρησης δύο γραμμών με την ίδια τιμή για την ίδια ιδιότητα, και όχι για να ταυτ2οποιήσουμε την γραμμή.
Η σύνδεση μιας σχέσης με μια δεύτερη συνήθως απαιτεί μια κοινή ιδιότητα, η οποία υπάρχει και στις δύο σχέσεις. Οι κοινές ιδιότητες αυτές είναι συνήθως ένα πρωτεύον κλειδί (primary key) της μίας σχέσης και ένα ξένο κλειδί (foreign key) της δεύτερης. Ο κανόνας της αναφορικής ακεραιότητας (referential integrity) επιβάλλει ότι τιμές για ένα ξένο κλειδί της δεύτερης σχέσης αναφέρονται σε τιμές του πρωτεύοντος κλειδιού της πρώτης.

ΚΕΦΑΛΑΙΟ 2

Τι είναι η Γλώσσα Προγραμματισμού Php

Η **PHP** είναι μια γλώσσα προγραμματισμού που σχεδιάστηκε για τη δημιουργία δυναμικών σελίδων στο Δυαδίκτυο και είναι επισήμως γνωστή ως: **HyperText Preprocessor.**Είναι μια Server - side (εκτελείται δηλαδή στον διακομιστή) scripting γλώσσα που γράφεται συνήθως πλαισιωμένη από HTML κώδικα, για την μορφοποίηση των αποτελεσμάτων.
Αντίθετα από μια συνηθισμένη HTML σελίδα η σελίδα PHP δεν στέλνεται άμεσα σε έναν πελάτη (Client), άντ' αυτού πρώτα αναλύεται και μετά αποστέλλεται το παραγόμενο αποτέλεσμα. Μεταφράζεται στην πλευρά του διαδικτυακού διακομιστή και δημιουργεί HTML ή άλλη έξοδο, την οποία θα δει ο επισκέπτης. Ο επισκέπτης θα δει μόνο την HTML έξοδο που παράγει η PHP από την πλευρά του διακομιστή και δε χρειάζεται να έχει εγκατεστημένη την PHP στο δικό του υπολογιστή. Τα στοιχεία HTML στον πηγαίο κώδικα μένουν ως έχουν, αλλά ο PHP κώδικας ερμηνεύεται και εκτελείται. Ο κώδικας PHP μπορεί να θέσει ερωτήματα σε Βάσεις Δεδομένων, να δημιουργήσει εικόνες, να διαβάσει και να γράψει αρχεία, να συνδεθεί με απομακρυσμένους υπολογιστές κ.ο.κ. Είναι ένα προϊόν ανοιχτού κώδικα, γεγονός που σημαίνει ότι διανέμεται δωρεάν. Επίσης, έχουμε πρόσβαση στον κώδικα προέλευσής του. Έτσι, μπορούμε να τον τροποποιήσουμε και να τον επαναχρησιμοποιήσουμε, σύμφωνα με τις ανάγκες και τις απαιτήσεις μας. Σε γενικές γραμμές οι δυνατότητες που μας δίνει είναι απεριόριστες .

Ιστορία της γλώσσας Php

Η ιστορία της PHP ξεκινά από το 1995, όταν ένας φοιτητής, ο Rasmus Lerdorf δημιούργησε χρησιμοποιώντας τη γλώσσα προγραμματισμού Perl ένα απλό script με το όνομα php.cgi, για προσωπική του χρήση. Το script αυτό είχε σαν σκοπό να διατηρεί μια λίστα στατιστικών για τα άτομα που έβλεπαν το online βιογραφικό του σημείωμα. Αργότερα αυτό το script το διέθεσε και σε φίλους του, οι οποίοι άρχισαν να του ζητούν να προσθέσει περισσότερες δυνατότητες. Η γλώσσα τότε ονομαζόταν PHP/FI από τα αρχικά Personal Home Page/Form Interpreter. Αυτά τα scripts δεν άργησε να τα εμπλουτίσει με λειτουργίες επεξεργασίας δεδομένων με SQL, αλλά τα σημαντικά βήματα που έφεραν και την μεγάλη αποδοχή της PHP ήταν αρχικά η μετατροπή τους σε κώδικα γλώσσας προγραμματισμού C και μετέπειτα η δωρεάν παροχή του πηγαίου κώδικα μέσω της σελίδας του ώστε να επωφεληθούν όλοι από αυτό που είχε φτιάξει, αλλά και να τον βοηθήσουν στην περαιτέρω ανάπτυξή της. Το 1997 η PHP/FI έφθασε στην έκδοση 2.0, βασιζόμενη αυτή τη φορά στη γλώσσα προγραμματισμού C και αριθμώντας περισσότερα από 50.000 Web sites που τη χρησιμοποιούσαν, ενώ αργότερα την ίδια χρονιά οι Andi Gutmans και Zeev Suraski ξαναέγραψαν τη γλώσσα από την αρχή, βασιζόμενοι όμως αρκετά στην PHP/FI 2.0.

Έτσι η PHP έφθασε στην έκδοση 3.0 η οποία θύμιζε περισσότερο τη σημερινή μορφή της. Στη συνέχεια, οι Zeev και Andi δημιούργησαν την εταιρεία Zend (από τα αρχικά των ονομάτων τους), η οποία συνεχίζει μέχρι και σήμερα την ανάπτυξη και εξέλιξη της γλώσσας προγραμματισμού PHP. Ακολούθησε το 1998 η έκδοση 4 της PHP, τον Ιούλιο του 2004 διατέθηκε η έκδοση 5, ενώ αυτή τη στιγμή έχουν ήδη διατεθεί και τα πρώτα snapshots της επερχόμενης PHP 6, για οποιονδήποτε προγραμματιστή θέλει να τη χρησιμοποιήσει.
Σήμερα περισσότερα από 16.000.000 Web sites, ποσοστό μεγαλύτερο από το 35% των ιστοσελίδων του Διαδικτύου, χρησιμοποιούν scripts γραμμένα με τη γλώσσα PHP, ενώ το υπόλοιπο 65% το μοιράζονται στατικές σελίδες HTML και όλες οι άλλες γλώσσες προγραμματισμού. Πρόκειται για μια εξέλιξη που ο ίδιος ο Rasmus Lerdorf σε πρόσφατη συνέντευξή του δήλωσε ότι δεν περίμενε όταν, πριν από 10 χρόνια, δημιουργούσε τις πρώτες γραμμές κώδικα PHP. Τόνισε όμως ότι η PHP δεν θα είχε γίνει τόσο δημοφιλής αν η εξέλιξή της είχε παραμείνει προσωπική του προσπάθεια και δεν είχε βοηθηθεί από τους Andi Gutmans, Zeev Suraski και την εθελοντική συμμετοχή προγραμματιστών από ολόκληρο τον κόσμο. Τα περισσότερα Web sites επί του παρόντος χρησιμοποιούν κυρίως τις εκδόσεις 4 και 5 της PHP.

Τι μπορεί να κάνει η γλώσσα PHP

Η απάντηση στο ερώτημα αυτό είναι οτιδήποτε. Όπως είπαμε και πιο πάνω η γλώσσα προγραμματισμού PHP επικεντρώνεται κυρίως στο Server - side scripting, έτσι μπορεί να δημιουργήσει οτιδήποτε ,μπορεί να μαζέψει δεδομένα, να παράγει δυναμικό περιεχόμενο σελίδων, ή να στείλει και να πάρει cookies **.**Αλλά η PHP μπορεί να κάνει ακόμα περισσότερα πράγματα.
Υπάρχουν τρεις κύριοι τομείς που χρησιμοποιείται ένα PHP script.

· **Server - side scripting.** Αυτό είναι το πιο παραδοσιακό και το κύριο πεδίο για την PHP. Χρειάζονται τρία πράγματα για να δουλέψει αυτό. Τον PHP μεταγλωττιστή (parser) (CGI ή server module), ένα Webserver (εξυπηρετητή σελίδων) και ένα Web Browser. Πρέπει να τρέξει ο Webserver, με μια συνδεδεμένη εγκατάσταση της PHP. Τα αποτελέσματα του PHP προγράμματος μπορούν να προσπελαστούν με ένα Web browser, βλέποντας την σελίδα PHP μέσα από τον Server.

· **Command line scripting.** Δημιουργείται ένα PHP script για να τρέξει χωρίς Server ή Browser. Απαιτείται μόνο ένας PHP μεταγλωττιστής για να την χρησιμοποιήσει με αυτό τον τρόπο. Αυτός ο τύπος είναι ιδανικός για script που εκτελούνται συχνά με τη χρήση της cron (σε *nix ή Linux) ή με τον Task Scheduler (στα Windows). Αυτά τα scripts μπορούν επίσης να χρησιμοποιηθούν για απλές εργασίες επεξεργασίες κειμένου.

· **Εγγραφή Client - side GUI εφαρμογών** (Γραφικά περιβάλλοντα χρηστών). Η PHP ίσως να μην είναι η πιο καλή γλώσσα για να γράψει κανείς παραθυριακές εφαρμογές, αλλά αν γνωρίζει κάποιος PHP πολύ καλά και θέλει να χρησιμοποιήσει κάποια προχωρημένα χαρακτηριστικά της PHP στις Client - side εφαρμογές του, μπορεί επίσης να χρησιμοποιήσει το PHP-GTK για αυτού του είδους τα προγράμματα. Υπάρχει επίσης η δυνατότητα να γραφούν cross - platform εφαρμογές με αυτό τον τρόπο. Το PHP-GTK είναι μια επέκταση της PHP και δεν συμπεριλαμβάνεται στην κύρια διανομή.

Η PHP μπορεί να χρησιμοποιηθεί σε όλα τα κύρια λειτουργικά συστήματα, συμπεριλαμβανομένου του Linux, πολλών εκδοχών του Unix (HP-UX, Solaris και OpenBSD), Microsoft Windows, Mac OS X, RISC OS και πιθανώς σε άλλα. Η PHP υποστηρίζει επίσης τους Apache, Microsoft Internet Information Server, Personal Web Server, Netscape και iPlanet Servers, Oreilly Website Pro Server, Caudium, Xitami, OmniHTTPd, και πολλούς άλλους Webserver. Για την πλειοψηφία των Server η PHP έχει ένα module, για τους υπόλοιπους η PHP μπορεί να λειτουργήσει ως ένας CGI επεξεργαστής.Έτσι με την PHP υπάρχει η ελευθερία επιλογής ενός λειτουργικού συστήματος και ενός Web Server. Επιπλέον, υπάρχει η ελευθερία να χρησιμοποιήσει κάποιος συναρτησιακό (procedural) ή αντικειμενοστραφή (object oriented) προγραμματισμό ή μια ανάμειξη τους. Αν και η παρούσα έκδοση δεν υποστηρίζει όλα τα πρότυπα χαρακτηριστικά, μεγάλες βιβλιοθήκες κώδικα και μεγάλες εφαρμογές (συμπεριλαμβανομένης και της βιβλιοθήκης PEAR) είναι γραμμένες μόνο με αντικειμενοστραφή κώδικα. Με την PHP δεν είναι απαραίτητο να εξαχθεί HTML κώδικας. Οι δυνατότητες της PHP συμπεριλαμβάνουν την εξαγωγή εικόνων, αρχείων PDF, ακόμη και ταινίες Flash (χρησιμοποιώντας τα libswf και Ming) όπου παράγονται αμέσως. Είναι επίσης εύκολο να εξαχθεί οποιοδήποτε κείμενο όπως XHTML και οποιοδήποτε άλλο XML αρχείο. Η PHP μπορεί να δημιουργεί αυτόματα αυτά τα αρχεία και να τα αποθηκεύει στο σύστημα αρχείων , αντί να τα εκτυπώνει, αποτελώντας έτσι μια Server - side cache για το δυναμικό περιεχόμενο. Ένα αρχείο με κώδικα PHP θα πρέπει να έχει την κατάλληλη επέκταση (π.χ. *.php, *.php4, *.phtml κ.ά.). Η ενσωμάτωση κώδικα σε ένα αρχείο επέκτασης .html δεν θα λειτουργήσει και θα εμφανίσει στον Browser τον κώδικα χωρίς καμία επεξεργασία, εκτός αν έχει γίνει η κατάλληλη ρύθμιση στα MIME types του Server. Επίσης ακόμη κι όταν ένα αρχείο έχει την επέκταση .php, θα πρέπει ο server να είναι ρυθμισμένος για να επεξεργάζεται κώδικα PHP.

Ένα από τα πιο δυνατά και σημαντικά χαρακτηριστικά της PHP είναι η υποστήριξη που έχει για ένα μεγάλο σύνολο Βάσεων Δεδομένων. Η συγγραφή μιας σελίδας που υποστηρίζει Βάσεις Δεδομένων είναι εξαιρετικά απλή. Οι Βάσεις Δεδομένων οι οποίες υποστηρίζονται μέχρι στιγμής είναι:

· Adabas D
· dBase
· Empress
· FilePro (read-only)
· Hyperwave
· IBM DB2
· Informix
· Ingres
· InterBase
· FrontBase
· mSQL
· Direct MS-SQL
· MySQL
· ODBC
· Oracle (OCI7 and OCI8)
· Ovrimos
· PostgreSQL

· Solid
· Sybase
· Velocis
· Unix dbm

Υπάρχει επίσης μια αφαιρετική επέκταση DBX Βάσεων Δεδομένων (DBX Database Abstraction Extension) που επιτρέπει διάφανα να χρησιμοποιηθεί οποιαδήποτε Βάση Δεδομένων υποστηρίζεται από αυτή την επέκταση. Επιπλέον η PHP υποστηρίζει το ODBC, το Open Database Connection Standard (Ανοιχτό πρότυπο Σύνδεσης Βάσεων Δεδομένων) και έτσι είναι εφικτή η σύνδεση σε οποιαδήποτε Βάση Δεδομένων που υποστηρίζει αυτό το παγκόσμιο πρότυπο.
Η PHP έχει επίσης υποστήριξη για επικοινωνία με άλλες υπηρεσίες χρησιμοποιώντας πρωτόκολλα όπως τα LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (στα Windows) και αμέτρητα άλλα. Η PHP έχει ακόμη υποστήριξη για την περίπλοκη ανταλλαγή δεδομένων WDDX μεταξύ σχεδόν όλων των Web Programming γλωσσών. Μιλώντας για δια επικοινωνία, η PHP υποστηρίζει instantiation αντικειμένων Java και τα χρησιμοποιεί διάφανα σαν αντικείμενα PHP.
Με την χρησιμοποίηση της PHP στον τομέα του e-commerce, θα συναντήσει κάποιος τις Cybercash payment, CyberMUT, VeriSign Payflow Pro και CCVS συναρτήσεις χρήσιμες για τα online προγράμματα πληρωμής.
Τελευταίο αλλά σημαντικό, υπάρχουν πολλές άλλες ενδιαφέρουσες επεκτάσεις, τις mnoGoSearch search engine συναρτήσεις, πολλά εργαλεία συμπίεσης (gzip, bz2), μετατροπές ημερολογίου, μεταφράσεις.

Πλεονεκτήματα και Δυνατότητες της γλώσσας Php

- Να εκτελεί οποιοδήποτε ερώτημα σε μία συμβατή Βάση Δεδομένων.
- Να δημιουργεί εικόνες.
- Να γράφει και να διαβάζει αρχεία.
- Να έχει επικοινωνία με απομακρυσμένους εξυπηρετητές.
- Να εκτελεί εντολές σε απομακρυσμένο υπολογιστή.
- Το βασικό πλεονέκτημα της PHP είναι ότι λειτουργεί δυναμικά. Αυτό σημαίνει ότι τα αποτελέσματα που παράγει, αλλάζουν σύμφωνα με τις ανάγκες του χρήστη. Ωστόσο, ο δυναμικός τρόπος λειτουργίας δεν παύει να εφαρμόζεται ακόμα και μέσα στο εσωτερικό της PHP. Για παράδειγμα, έχει τη δυνατότητα να αλλάζει τον τύπο των μεταβλητών δυναμικά, σύμφωνα με τα δεδομένα που κάθε χρονική στιγμή είναι αποθηκευμένα σε αυτές.
- Σε σύγκριση με τους βασικούς της ανταγωνιστές (Perl, ASP και JSP), η PHP έχει πολλά πλεονεκτήματα που την καθιστούν επικρατούσα.:

    1. Υψηλή απόδοση. Είναι πολύ αποτελεσματική. Λόγω της δυναμικότητας της Zend Engine που χρησιμοποιεί η PHP, μπορεί να συγκριθεί με την Asp. Έχουνγίνει κάποιες δοκιμές όσον αφορά τη σύγκριση της PHP και της Asp. Ταποτελέσματα αυτών των δοκιμών

κατέληξαν στο πόρισμα πως η PHPμεταγλωττίζεται αρκετά πιο γρήγορα από ότι η Asp.

2. Διασυνδέσεις με πολλά διαφορετικά συστήματα Βάσεων Δεδομένων. Έχει εγγενείς συνδέσεις για πολλά συστήματα Βάσεων Δεδομένων. Εκτός από τη MySQL, είναι δυνατή η σύνδεση με πολλές Βάσεις Δεδομένων μερικές από τις οποίες είναι οι mSQL, Oracle, Hyperwave, Informix, InterBase, PostgreSQL και πολλές άλλες. Χρησιμοποιώντας Open Database Connectivity Standard (ODCB), μπορούμε να συνδεθούμε σε οποιαδήποτε Βάση Δεδομένων παρέχει ένα πρόγραμμα οδήγησης ODCB.

3. Φορητότητα. Είναι διαθέσιμη για πολλά λειτουργικά συστήματα. Είναι δυνατή η συγγραφή κώδικα για πολλές διαφορετικές εκδόσεις των Microsoft Windows και συστήματα τύπου Unix

ΚΕΦΑΛΑΙΟ 3

Τι είναι η PostgreSQL

Η PostgreSQL αποτελεί μια ανοιχτού κώδικα σχεσιακή βάση δεδομένων με πολλές δυνατότητες. Η ανάπτυξη της ήδη διαρκεί πάνω από 20 χρόνια και βασίζεται σε μια αποδεδειγμένα καλή αρχιτεκτονική η οποία έχει δημιουργήσει μια ισχυρή αντίληψη των χρηστών της γύρω από την αξιοπιστία, την ακεραιότητα δεδομένων και την ορθή λειτουργία.
Η PostgreSQL τρέχει σε όλα τα βασικά λειτουργικά συστήματα, περιλαμβάνοντας Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), και Windows. Είναι ACID συμβατή (ACID compliant), έχει ολοκληρωμένη υποστήριξη για foreign keys, joins, views, triggers, και stored procedures (σε διάφορες γλώσσες προγραμματισμού). Συμπεριλαμβάνει τα περισσότερα SQL92 και SQL99 data types, συμπεριλαμβανομένων INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, και TIMESTAMP. επίσης υποστηρίζει αποθήκευση binary large objects, όπως εικόνες, ήχοι ή video. Διαθέτει native programming interfaces για C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, κ.α. καθως και εξαιρετικό εγχειρίδιο χρήσης. Επίσης, ο πηγαίος κώδικας της PostgreSQL είναι διαθέσιμος κάτω από την πιο ελεύθερη open source άδεια: το BSD license. Αυτή η άδεια δίνει την δυνατότητα χρήσης, μετατροπής και διανομής της PostgreSQL σε οποιαδήποτε μορφή, ανοιχτού ή κλειστού κώδικα. Η PostgreSQL δεν είναι μόνο μια δυνατή βάση δεδομένων ικανή να τρέχει μέσα σε επιχειρήσεις, είναι μια πλατφόρμα ανάπτυξης πάνω στην οποία δύναται να γίνει ανάπτυξη in-house, web ή εμπορικών εφαρμογών τα οποία χρειάζονται RDBMS.

Ιστορική Αναδρομή

Η PostgreSQL προέρχεται από το πακέτο POSTGRES, το οποίο γράφτηκε στο Πανεπιστήμιο του Berkeley στην Καλιφόρνια των Η.Π.Α..Το σύνθετο πρόγραμμα της POSTGRES, που πραγματοποιήθηκε υπό τη βασική καθοδήγηση του καθηγητή Michael Stonebraker, χρηματοδοτήθηκε από την προηγμένη αντιπροσωπεία ερευνητικών προγραμμάτων. άμυνας(DARPA), το ερευνητικό γραφείο στρατού (ARO), το εθνικό ίδρυμα επιστήμης (NSF), αλλά και από τα ESL και INC. Η εφαρμογή POSTGRES άρχισε το 1986. Οι πρώτες ιδέες για το σύστημα παρουσιάστηκαν στο σχέδιο POSTGRES και ο καθορισμός του αρχικού δεδομενικού μοντέλου εμφανίστηκε στο μοντέλο δεδομένων POSTGRES. Το σχέδιο των κανόνων του συστήματος εκείνη την περίοδο, περιγράφηκε στο «Σχέδιο του συστήματος κανόνων POSTGRES». Η λογική και η αρχιτεκτονική του διευθυντή(manager) αποθήκευσης περιγράφονται λεπτομερώς στο «σχέδιο του συστήματος αποθήκευσης της POSTGRES». Η POSTGRES έχει υποβληθεί σε διάφορες σημαντικές επεκτάσεις από τότε. Το πρώτο «demoware» σύστημα κατέστη λειτουργικό το 1987 και παρουσιάστηκε στη διάσκεψη ACM- SIGMOD του 1988. Η έκδοση 1, που

περιγράφηκε στην υλοποίηση της POSTGRES, κυκλοφόρησε σε μερικούς εξωτερικούς χρήστες τον Ιούνιο του 1989.
Ανταποκρινόμενο σε μια αρνητική κριτική του αρχικού συστήματος κανόνων, το σύστημα κανόνων της POSTGRES ξανασχεδιάστηκε σε ότι αφορά στους κανόνες, τις διαδικασίες, την εναποθήκευση και τις όψεις στα συστήματα βάσεων δεδομένων και η δεύτερη έκδοσή της κυκλοφόρησε τον Ιούνιο του 1990 με το νέο σύστημα κανόνων. Στην έκδοση 3 της POSTGRES, που πραγματοποιήθηκε το 1991, προστέθηκε η υποστήριξη πολλαπλών διευθυντών αποθήκευσης. Επιπροσθέτως, διαθέτει τώρα και ένα βελτιωμένο εκτελεστή επερωτήσεων, και ένα νέο σύστημα επανεγγράψιμων κανόνων. Ως επί το πλείστον, οι επόμενες εκδόσεις μέχρι και την Postgres95 εστίασαν στη μεταφερσιμότητα και την αξιοπιστία.
Η POSTGRES έχει χρησιμοποιηθεί για να υλοποιήσει πολλές διαφορετικές εφαρμογές έρευνας και παραγωγής. Σε αυτές περιλαμβάνονται: ένα οικονομικό σύστημα ανάλυσης δεδομένων, μια συσκευασία ελέγχου απόδοσης αεριωθούμενων μηχανών, μια ιατρική βάση δεδομένων πληροφοριών και διάφορα γεωγραφικά συστήματα πληροφοριών. Η POSTGRES έχει χρησιμοποιηθεί επίσης ως εκπαιδευτικό εργαλείο σε διάφορα πανεπιστήμια. Εν κατακλείδι, τεχνολογίες πληροφοριών Illustra (που συγχωνεύονται αργότερα σε Informix, το οποίο είναι κύριο τώρα από την IBM) πήραν τον κώδικα και τον εμπορευματοποίησαν. Στα τέλη του 1992, η POSTGRES αποτέλεσε τον πρωταρχικό διαχειριστή (manager) δεδομένων για το sequoia2000, ένα επιστημονικό πρόγραμμα υπολογισμού. Το μέγεθος της εξωτερικής κοινότητας χρηστών διπλασιάστηκε σχεδόν κατά τη διάρκεια του 1993. Έγινε όλο και περισσότερο προφανές ότι η συντήρηση του πρωτότυπου κώδικα και η υποστήριξή του απαιτεί μεγάλο χρονικό διάστημα, που θα έπρεπε να έχει αφιερωθεί στην έρευνα βάσεων δεδομένων. Σε μια προσπάθεια να μειωθεί αυτό το βάρος της υποστήριξης, το πρόγραμμα του Berkeley, POSTGRES, τελείωσε επίσημα με την έκδοση 4.2.

Postgres95

Το 1994, ο Andrew Yu και ο Jolly Chen πρόσθεσαν τον γλωσσικό διερμηνέα της SQL στην POSTGRES. Με ένα νέο όνομα, η Postgres95 απελευθερώθηκε στη συνέχεια στον Ιστό, ως απόγονος του αρχικού κώδικα POSTGRES. Στην Postgres95 ο κώδικας ήταν όλος σε Ansi C και μικρότερος στο μέγεθος κατά 25%. Πολλές εσωτερικές αλλαγές βελτίωσαν την απόδοση και τη συντηρησιμότητα του νέου αυτού λογισμικού. Η απελευθέρωση 1.0.x Postgres95 έτρεξε περίπου 30-50% γρηγορότερα στη συγκριτική μέτρηση επιδόσεων του Wisconsin σε σύγκριση με την έκδοση 4.2 της POSTGRES. Εκτός από κάποιες μικρές διορθώσεις, τα εξής ήταν οι σημαντικότερες επεκτάσεις:

- Η γλώσσα επερωτήσεων PostQUEL αντικαταστάθηκε από την SQL, που ενσωματώνεται στον εξυπηρετητή(server). Επερωτήσεις δεν υποστηρίχθηκαν μέχρι την εμφάνιση της PostgreSQL (που θα αναλυθεί παρακάτω), αλλά θα μπορούσαν να υλοποιηθούν στην Postgres95 με τις καθορισμένες από το χρήστη συναρτήσεις SQL. Οι αθροιστικές συναρτήσεις υλοποιήθηκαν από την αρχή. Επίσης προστέθηκε και το στοιχείο GROUP BY της εντολής SELECT των επερωτήσεων.

- Εκτός από το πρόγραμμα παρατηρητή (monitor), ένα νέο πρόγραμμα (psql) παρείχε τη δυνατότητα πραγματοποίησης διαλογικών επερωτήσεων SQL, οι οποίες χρησιμοποίησαν το GNU Readline.

- Μια νέα front-end βιβλιοθήκη (libpgtcl) καθιστούσε δυνατή την υποστήριξη πελατών Tcl. Ένας αντιπροσωπευτικός φλοιός, pgtclsh, παρείχε νέες εντολές Tcl για να διασυνδέσει τα προγράμματα Tcl με τον εξυπηρετητή της Postgres95.

- Η διεπαφή μεγάλο-αντικείμενο(large-object) εξετάστηκε λεπτομερώς. Τα μεγάλα αντικείμενα αντιστροφής ήταν ο μόνος μηχανισμός για την αποθήκευση μεγάλων αντικειμένων. Το σύστημα αρχείων αντιστροφής αφαιρέθηκε αργότερα.

- Το σε επίπεδο στιγμιότυπων σύστημα κανόνων αφαιρέθηκε. Κανόνες εξακολούθησαν να είναι διαθέσιμοι με τη μορφή επανεγγράψιμων κανόνων.

- Ένα σύντομο διδακτικό βοήθημα που κάνει μια εισαγωγή στα συνηθισμένα χαρακτηριστικά γνωρίσματα της SQL, καθώς επίσης και σε όλα εκείνα τα γνωρίσματα της Postgres95 που διανεμήθηκαν με τον πηγαίο κώδικα.

- GNU make (αντί του BSD make) χρησιμοποιήθηκε για το κτίσιμο (build). Επίσης, η Postgres95 θα μπορούσε να μεταγλωττιστεί με το GCC.

PostgreSQL

Το 1996 έγινε σαφές ότι η ονομασία Postgres95 δεν θα αντέξει με την πάροδο του χρόνου, γιαυτό και επιλέχθηκε η ονομασία PostgreSQL ώστε να αντικατοπτρίσει την σχέση ανάμεσα στην αρχική POSTGRES και τις πιο πρόσφατες εκδόσεις με συμβατότητα SQL. Την ίδια χρονιά ορίστηκε η αρίθμηση των εκδόσεων να ξεκινά από το νούμερο 6.0, έτσι ώστε να συνεχίζεται από την αρίθμηση που ξεκίνησε από το αρχικό σχέδιο POSTGRES του πανεπιστημίου Berkeley. Η έμφαση κατά τη διάρκεια της ανάπτυξης των εκδόσεων v1.0.x της Postgres95 ήταν στην σταθεροποίηση του backend. Με τη σειρά εκδόσεων v6.x της PostgreSQL η έμφαση μεταφέρθηκε από τον προσδιορισμό και την κατανόηση των υπαρχόντων προβλημάτων στο backend, στην προσπάθεια αύξησης των δυνατοτήτων.

Η PostgreSQL υλοποιεί εξεζητημένα χαρακτηριστικά όπως Multi-Version Concurrency Control (MVCC), point in time recovery, tablespaces, asynchronous replication, nested transactions (savepoints), online/hot backups, a sophisticated query planner/optimizer, write ahead logging for fault tolerance. Υποστηρίζει διεθνή σετ χαρακτήρων, κωδικοποίηση χαρακτήρων σε πολλά byte, Unicode καθώς και δυνατότητα ταξινόμησης δεδομένων ανεξάρτητα από το locale. Η PostgreSQL μπορεί να διαχειριστεί εύκολα μεγάλους αριθμούς ταυτόχρονων χρηστών καθώς και μεγάλο όγκο δεδομένων. Υπάρχουν ενεργές εγκαταστάσεις σε περιβάλλοντα παραγωγής που διαχειρίζονται πάνω από 4 terabytes δεδομένων.

Μερικές γενικές οριακές τιμές συμπεριλαμβάνονται στον παρακάτω πίνακα:

Limit	Value
Maximum Database Size	Unlimited
Maximum Table Size	32 TB
Maximum Row Size	1.6 TB
Maximum Field Size	1 GB
Maximum Rows per Table	Unlimited
Maximum Columns per Table	250 - 1600 depending on column types
Maximum Indexes per Table	Unlimited

Η PostgreSQL διαθέτει μια ευρεία ποικιλία τύπων δεδομένων. Ο χρήστης έχει την δυνατότητα να προσθέσει ένα νέο τύπο δεδομένων χρησιμοποιώντας την εντολή CREATE TYPE

Κατηγορίες	Τύποι
Λογικοί και δυαδικοί (Boolean and binary types)	boolean, bool, bit(n), bit varying(n), varbit(n)
Χαρακτήρες (Character types)	character (n), char(n), character varying(n), varchar(n), text
Αριθμητικοί (Numeric types)	smallint, int2, integer, int, int4, bigint, int8, real, float4, double precision, float8, float, numeric(p,s), decimal(p,s), money, serial
Ημερομηνία και ώρα (Date and time types)	date, time, time with time zone, timestamp (includes time zone), interval
Γεωμετρικοί (Geometric types)	box, line, lseg, circle, path, point, polygon
Δικτυακοί (Network types)	cidr, inet, macaddr
Συστήματος (System types)	oid, xid

Η PostgreSQL απολαμβάνει αναγνώριση από τους χρήστες της και την βιομηχανία πληροφορικής, συμπεριλαμβανομένων των Linux New Media Award for Best Database System, και έχει υπάρξει 3 φορές νικήτρια στο Linux Journal Editors' Choice Award for best DBMS.
Η PostgreSQL είναι συνεπής με τις προδιαγραφές . Η υλοποίησή της είναι απολύτως σύμφωνη με τις προδιαγραφές ANSI-SQL 92/99. Έχει ολοκληρωμένη υποστήριξη για subqueries (συμπεριλαμβανομένων subselects μέσα από το FROM), read-committed και serializable transaction isolation levels. Η PostgreSQL αποτελεί ένα πλήρες σχεσιακό σύστημα που υποστηρίζει πολλαπλά σχήματα ανά database, ο κατάλογος (πληροφορίες σχετικά με τους πίνακες, στήλες, views κλπ) είναι διαθέσιμος διαμέσου του Information Schema όπως ορίζεται στο SQL standard.
Στα data integrity χαρακτηριστικά συμπεριλαμβάνονται: primary keys, foreign keys με υποστήριξη restricting και cascading updates/deletes, check constraints, unique constraints, και not null constraints.
Η PostgreSQL έχει αρκετά προηγμένα χαρακτηριστικά όπως: auto-increment columns μέσω sequences, LIMIT/OFFSET που επιτρέπουν την επιστροφή partial result sets. Όσον αφορά τα indexes υποστηρίζει compound, unique, partial, και functional indexes τα οποία μπορούν να χρησιμοποιήσουν οποιονδήποτε από τους B-tree, R-tree, hash, ή GiST αλγόριθμους.
Άλλα προηγμένα χαρακτηριστικά της PostgreSQL είναι: table inheritance, rules systems και database events .

- Το **table inheritance** (κληρονομικότητα πινάκων) προσθέτει μια αντικειμενοστραφή διάσταση στην δημιουργία πινάκων, επιτρέποντας στους σχεδιαστές database να δημιουργούν νέους πίνακες από άλλους πίνακες χρησιμοποιώντας τους ως βάση. Ακόμα καλύτερα η PostgreSQL υποστηρίζει και μονή και πολλαπλή κληρονομικότητα με τον δικό της τρόπο.
- Το **rules system**, που επίσης καλείται the query rewrite system, επιτρέπει στον σχεδιαστή βάσεων να δημιουργήσει κανόνες που ορίζουν συγκεκριμένες λειτουργίες για έναν πίνακα ή view, και να μετατρέπει δυναμικά λειτουργίες, την ώρα που εκτελούνται, σε άλλες εναλλακτικές.
- Το **events system** αποτελεί ένα interprocess communication system στο οποίο μηνύματα και events μπορούν να μεταδοθούν μεταξύ πελατών (clients) χρησιμοποιώντας τις LISTEN και NOTIFY εντολές, επιτρέποντας από την απλή peer to peer επικοινωνία ως ένα εξελιγμένο συντονισμό βασισμένο σε database events. Εφόσον τα notifications μπορεί να προέρχονται από triggers και stored procedures, PostgreSQL clients μπορούν να επιβλέπουν λειτουργίες όπως: updates, inserts ή deletes πινάκων όταν αυτά γίνονται.

Συμβατότητα

Η PostgreSQL τρέχει stored procedures σε πολλές γλώσσες προγραμματισμού συμπεριλαμβανομένων Java, Perl, Python, Ruby, Tcl, C/C++, και της PL/pgSQL η οποία είναι παρόμοια με την PL/SQL της Oracle. Στην βασική βιβλιοθήκη συναρτήσεων της PostgreSQL συμπεριλαμβάνονται εκατοντάδες built-in συναρτήσεις οι οποίες καλύπτουν από βασικές μαθηματικές συναρτήσεις και διαχείριση Συμβολοσειρών ως κρυπτογραφία και Oracle compatibility. Triggers και stored procedures μπορούν να γράφουν σε c και να φορτωθούν μέσα στη βάση ως βιβλιοθήκη, επιτρέποντας μεγάλη ευελιξία στην επέκταση των δυνατοτήτων της βάσης. Παρομοίως η PostgreSQL περιλαμβάνει framework που επιτρέπει τον ορισμό και την δημιουργία custom data types καθώς και βοηθητικές συναρτήσεις και τελεστές (operators) που θα περιγράφουν την λειτουργία τους. Σαν αποτέλεσμα ένα πλήθος από εξελιγμένα data types έχουν δημιουργηθεί από γεωμετρικά και spatial δεδομένα ως διευθύνσεις δικτύων και ISBN/ISSN (International Standard Book Number/International Standard Serial Number), τα οποία μπορούν κατ' επιλογή να προστεθούν στο σύστημα.
Η PostgreSQL, όπως διαθέτει πολλές procedure languages, έτσι διαθέτει και πολλά library interfaces , επιτρέποντας πολλές γλώσσες προγραμματισμού είτε compiled είτε interpreted να επικοινωνούν με την PostgreSQL. Υπάρχουν interfaces για Java (JDBC), ODBC, Perl, Python, Ruby, C, C++, PHP, Lisp, Scheme, Qt, κ.α.

PgAdmin

Ως user interface της PostgreSQL χρησιμοποιήσαμε την διαδικτυακή εφαρμογή PgAdmin στην τελευταία της έκδοση 5.1.

Χαρακτηριστικά :

- Ταυτόχρονη διαχείριση πολλών servers
- Συμβατότητα με τις εκδόσεις 8.4.x, 9.0.x, 9.1.x, 9.2.x , 9.3x της PostgreSQL

- Παροχή όλων των δυνατοτήτων σχετικά με:
    1. Users & groups
    2. Databases
    3. Schemas
    4. Tables, indexes, constraints, triggers, rules & privileges
    5. Views, sequences & functions
    6. Advanced objects
    7. Reports

- Εύκολη διαχείριση δεδομένων :
    1. Δυνατότητα για εμφάνιση όλων των δεδομένων(browse) σε tables, views & reports
    2. Δυνατότητα απευθείας εκτέλεσης εντολών SQL select, insert, update, delete καθώς και SQL scripts.

- Εξαγωγή αρχείο ανάκτησης δεδομένων σε πολλαπλές μορφές : SQL, COPY, XML, XHTML, CSV, Tabbed, pg_dump
- Εισαγωγή δεδομένων επίσης με πολλαπλή συμβατότητα, μέσω SQL scripts, COPY data, XML, CSV, Tabbed
- Slony master-slave replication
- Διάθεση σε 27 γλώσσες χωρίς προβλήματα κωδικοποίησης και συμβατότητας μεταξύ δεδομένων και interface
- Εύκολη εγκατάσταση και εκμάθηση.

Τι είναι ο Apache Web Server

Ο Apache HTTP γνωστός και απλά ως Apache είναι ένας σταθερός και ευέλικτος web server. Όποτε επισκέπτεστε έναν ιστότοπο ο πλοηγός σας επικοινωνεί με έναν διακομιστή HTTP. Αποτελεί τα τελευταία χρόνια τον πιο δημοφιλή διακομιστή του διαδικτύου. Αρχικά σχεδιάστηκε για τους κεντρικούς υπολογιστές Unix. Πλέον όμως έχει την δυνατότητα να εφαρμοστεί σε μεγάλη ποικιλία λειτουργικών συστημάτων όπως Unix, Linux , Solaris , Novell NetWare , AmigaOS , Mac OS X ,Microsoft Windows όπως επίσης και να υποστηρίξει πολλές γλώσσες προγραμματισμού όπως PHP, Perl, Python κ.α. και τέλος συνεργάζεται με συστήματα δεδομένων όπως MySQL και Oracle.

Είναι ένα λογισμικό ανοιχτού κώδικα του οποίου την επιτήρηση(εποπτεία, διάθεση και υποστήριξη) την έχει το Ίδρυμα Λογισμικού Apache (Apache Software Foundation). Ο Apache WebServer υποστηρίζει plug in ενότητες και έχει την δυνατότητα και παρέχει μια πλήρη σειρά προηγμένων χαρακτηριστικών γνωρισμάτων των διακομιστών συμπεριλαμβανομένων των CGI, SSL και των εικονικών περιοχών.Δημιουργός ήταν ο Robert McColl. Η πρώτη έκδοση του ήταν γνωστή ως NCSA Http και κυκλοφόρησε το 1993. Εκ τότε η πορεία αγοράς του ήταν ανοδική και μάλιστα από τον Μάρτιο του 2011 βρίσκεται στην πρώτη θέση αγοράς των web servers, σύφμωνα με τα στατιστικά στοιχεία της εταιρείας Net craft. Μάλιστα σε έρευνα της ίδιας εταιρείας που ολοκληρώθηκε προ λίγων ημερών(για τα έτη 1995 έως 2011) βλέπουμε καθαρά την ανοδική της πορεία στην αγορά και των servers, το ποσοστό της οποίας ανέρχεται σε 66,8%.

Συνεργασία των MySQL, Apache και PHP

Είναι τρία προϊόντα που συνεργάζονται με απόλυτη επιτυχία.
Τα βήματα που ακολουθεί μία αίτηση του browser περιγράφονται παρακάτω:
1) Ο Web browser κάνει μία HTTP αίτηση για μία συγκεκριμένη σελίδα στον Web Server (Apache)
2) Έπειτα λαμβάνει την αίτηση, βρίσκει την σελίδα και την περνά στην PHP για επεξεργασία.
3) Η PHP κάνει ανάλυση του script. Αν μέσα στο script υπάρχει ερώτημα προς τη βάση δεδομένων τότε η
PHP ανοίγει μία σύνδεση με τον Mysql Server και στέλνει το ερώτημα.
4) Ο Mysql Server λαμβάνει το ερώτημα το επεξεργάζεται και στέλνει το αποτέλεσμα στην PHP.
5)Η PHP κάνει μορφοποίηση του αποτελέσματος σε HTML και επιστρέφει την τελική HTML
σελίδα στον Web Server.
6) Ο Web Server περνά την σελίδα στον Web browser Server.

ΚΕΦΑΛΑΙΟ 4

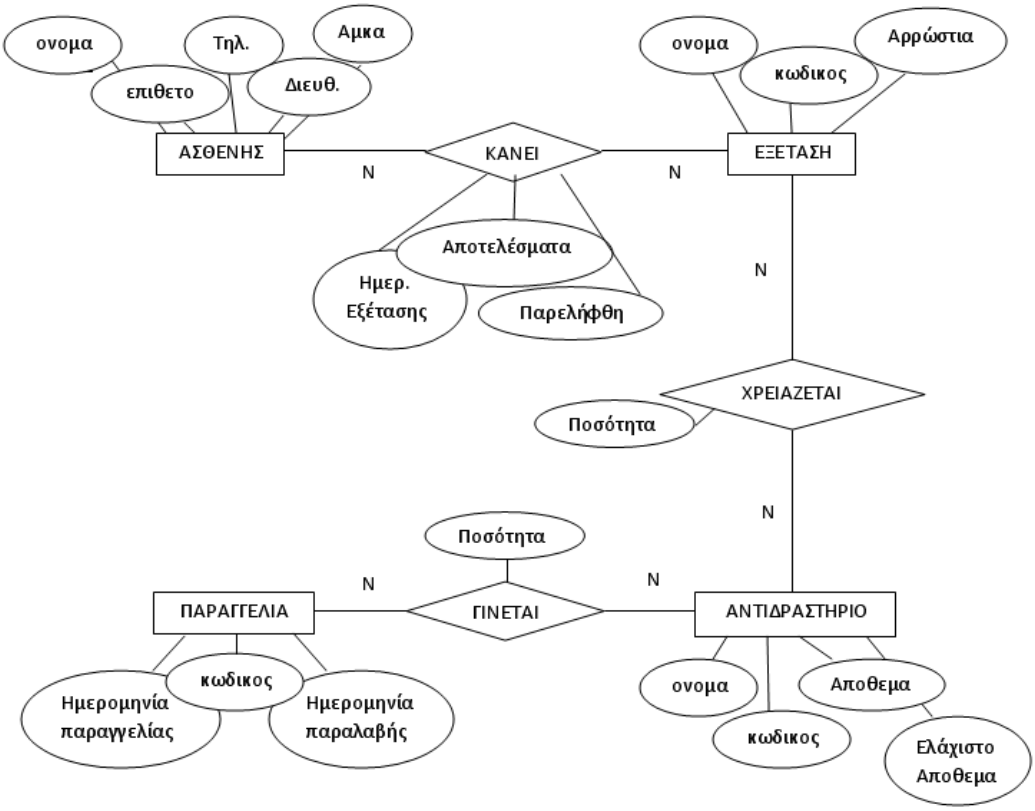
Σχεδιασμός Βάσης

1.Λογικός σχεδιασμός βάσης

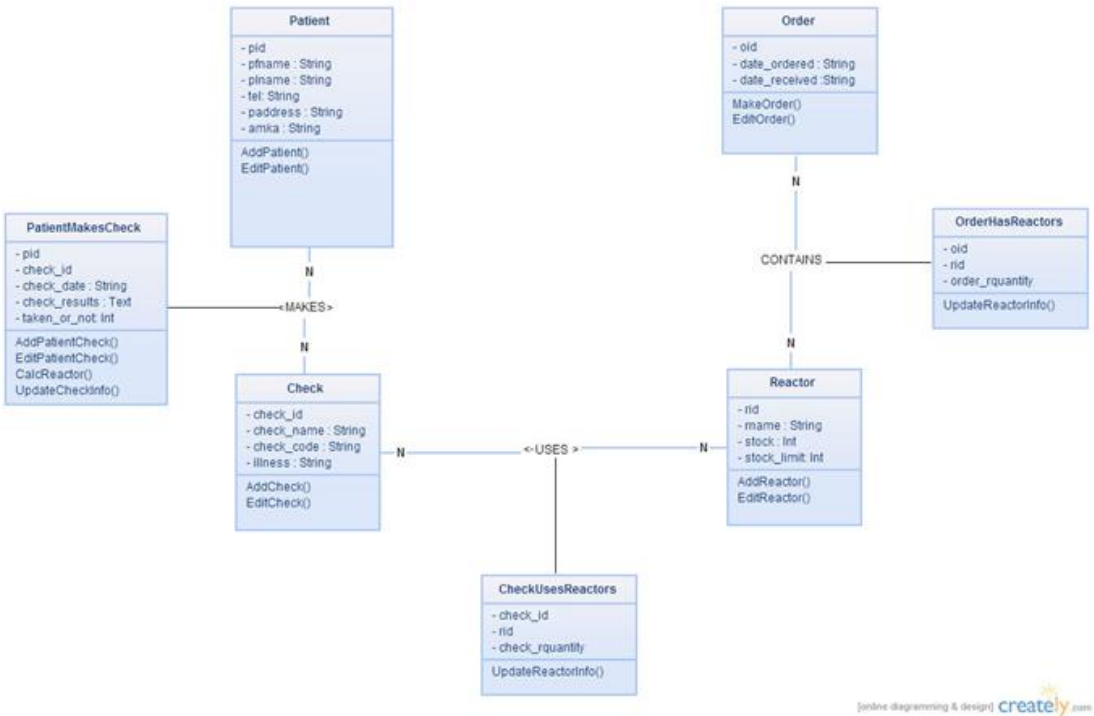
Για τη βέλτιστη λειτουργία ενός συστήματος βάσεων δεδομένων, είναι απαραίτητο να έχει γίνει πρώτα ένας προσεκτικός λογικός σχεδιασμός της βάσης. Είναι ίσως ο σημαντικότερος παράγοντας όσον αφορά την απόδοση του συστήματος. Αυτός που σχεδιάζει το σχήμα της βάσης, οφείλει να γνωρίζει άριστα το σκοπό για τον οποίο αυτή πρόκειται να χρησιμοποιηθεί. Βασισμένος στις ανάγκες που θα εξυπηρετεί η εφαρμογή και στους ανθρώπους που θα τη χρησιμοποιούν, πρέπει να μοντελοποιήσει σωστά τις απαιτήσεις της και να σχεδιάσει με βέλτιστο τρόπο τις οντότητες που θα απαρτίζουν τη βάση, τα γνωρίσματα τους και τις σχέσεις μεταξύ τους

Αρχικό βήμα :

Το πρώτο βήμα πριν το σχεδιασμό της βάσης ,είναι η κατανόηση των δεδομένων και ο σχεδιασμός των οντοτήτων και σχέσεων.
Αυτό πετυχαίνεται με τη σχεδίαση **ένός πλήρους διαγράμματος οντοτήτων-σχέσεων για την εταιρία** (εικόνα 1) **και την σχεδίαση** του **αντίστοιχου διάγραμμα κλάσεων σε UML** με τα γνωρίσματα (όνομα, τύπος) όλων των οντοτήτων και σχέσεων μαζι με τους περιορισμούς πληθικότητας (εικόνα 2).



Εικόνα 1.



Εικόνα 2.

Δεύτερο βήμα :

Το σύστημα πρέπει να υποστηρίζει τις παρακάτω διεργασίες

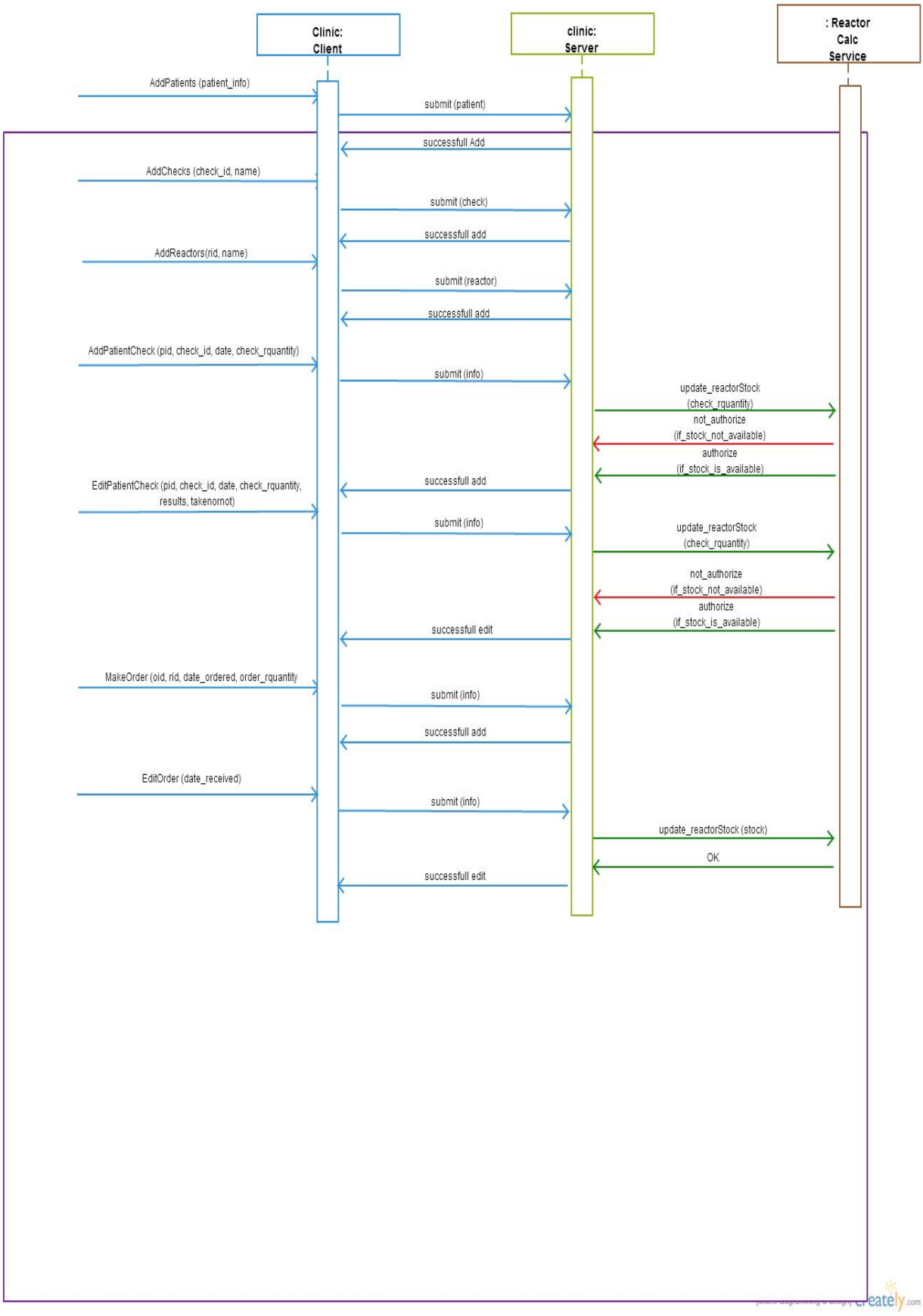
1. Εισαγωγή όλης της παραπάνω πληροφορίας τμηματικά. Πρέπει να υποστηρίζεται η εισαγωγή εξετάσεων που θέλει να πραγματοποιήσει κάποιος ασθενής. Πρέπει να δίνεται η δυνατότητα ενημέρωση για τα αποτελέσματα αυτών των εξετάσεων. Επίσης κάθε φορά που γίνεται ενημέρωση για εξετάσεις που θέλει να πραγματοποιήσει κάποιος πελάτης να γίνεται αυτόματη μείωση των αποθεμάτων των αντιδραστηρίων.

2. Συγκεντρωτική αναφορά των πελατών που δεν έχουν πάρει ακόμα τα αποτελέσματα κάποιων εξετάσεων που έκαναν.

3. Κάθε φορά που θα τα αποθέματα κάποιου αντιδραστηρίου πέφτουν κάτω από τα ελάχιστα όρια να εμφανίζεται το μήνυμα αμέσως μετά την εισαγωγή των εξετάσεων που προκάλεσαν αυτήν την μείωση.

4. Ενημέρωση για παραλαβή κάποιας παραγγελίας. Αυτό έχει ως αποτέλεσμα την αυτόματη αύξηση των αποθεμάτων των αντιδραστηρίων.

Φτιάχνουμε το παρακάτω διάγραμμα ακολουθίας UML για την υλοποίηση των διεργασιών 1 εως 4 .


Clinic:
Client
clinic:
Server
: Reactor
Calc
Service
AddPatients (patient_info)
submit (patient)
successfull Add
AddChecks (check_id, name)
submit (check)
successfull add
AddReactors(rid, name)
submit (reactor)
successfull add
AddPatientCheck (pid, check_id, date, check_rquantity)
submit (info)
update_reactorStock
(check_rquantity)
not_authorize
(if_stock_not_available)
authorize
(if_stock_is_available)
EditPatientCheck (pid, check_id, date, check_rquantity,
results, takenormot)
successfull add
submit (info)
update_reactorStock
(check_rquantity)
not_authorize
(if_stock_not_available)
authorize
(if_stock_is_available)
successfull edit
MakeOrder (oid, rid, date_ordered, order_rquantity
submit (info)
successfull add
EditOrder (date_received)
submit (info)
update_reactorStock (stock)
OK
successfull edit

Τρίτο βήμα :

Ύστερα απαραίτητη είναι η μετάφραση του μοντέλου μας ,στο σχεσιακό μοντέλο μαζί με τα πρωτεύοντα κλειδιά

PATIENT

pid	**pfname**	**plname**	**tel**	**paddress**	**amka**

CHECK

check_id	**check_name**	**check_code**	**illness**

ORDER

oid	**date_ordered**	**date_received**

REACTOR

rid	**rname**	**stock**	**stock_limit**

PATIENT_MAKES_CHECK

pid	**check_id**	**check_date**	**check_results**	**taken_or_not**

CHECK_USES_REACTORS

check_id	**rid**	**check_rquantity**

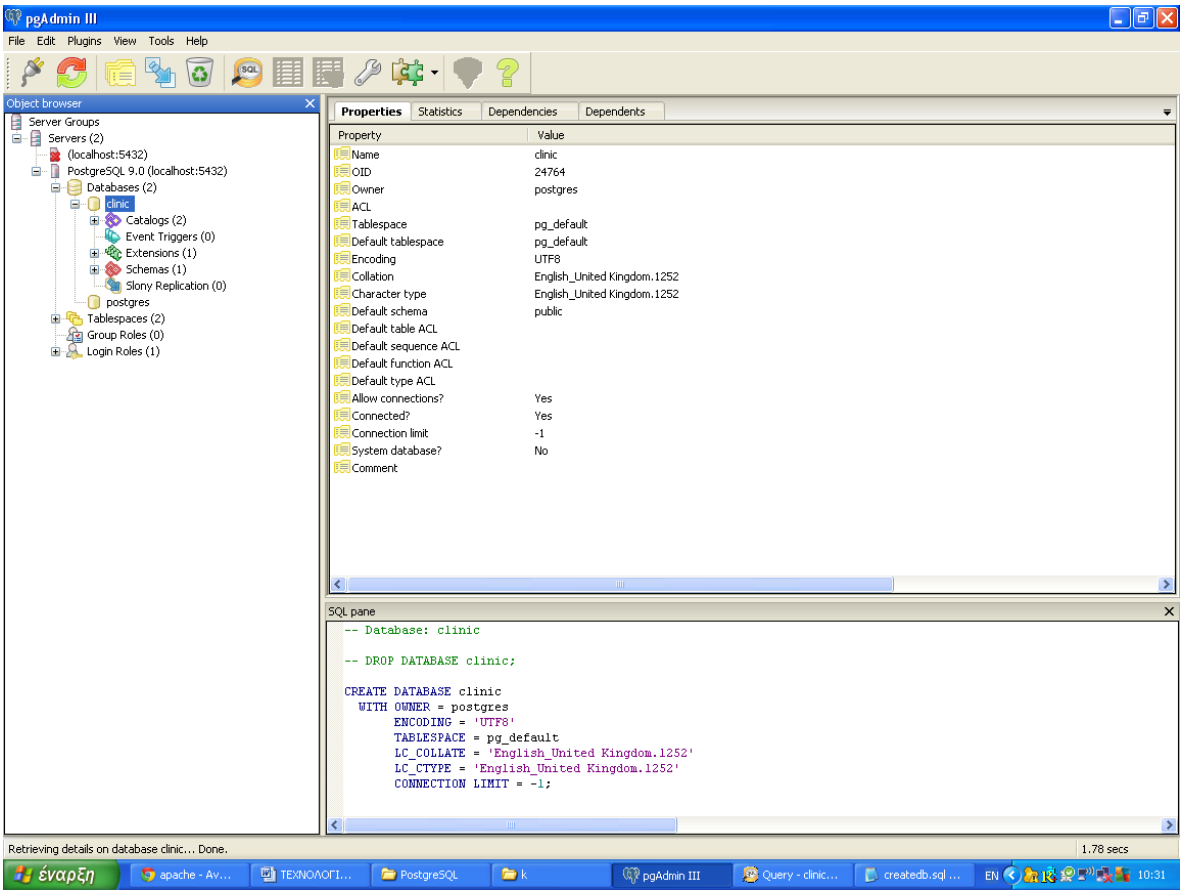
ORDER_HAS_REACTORS

oid	**rid**	**order_rquantity**

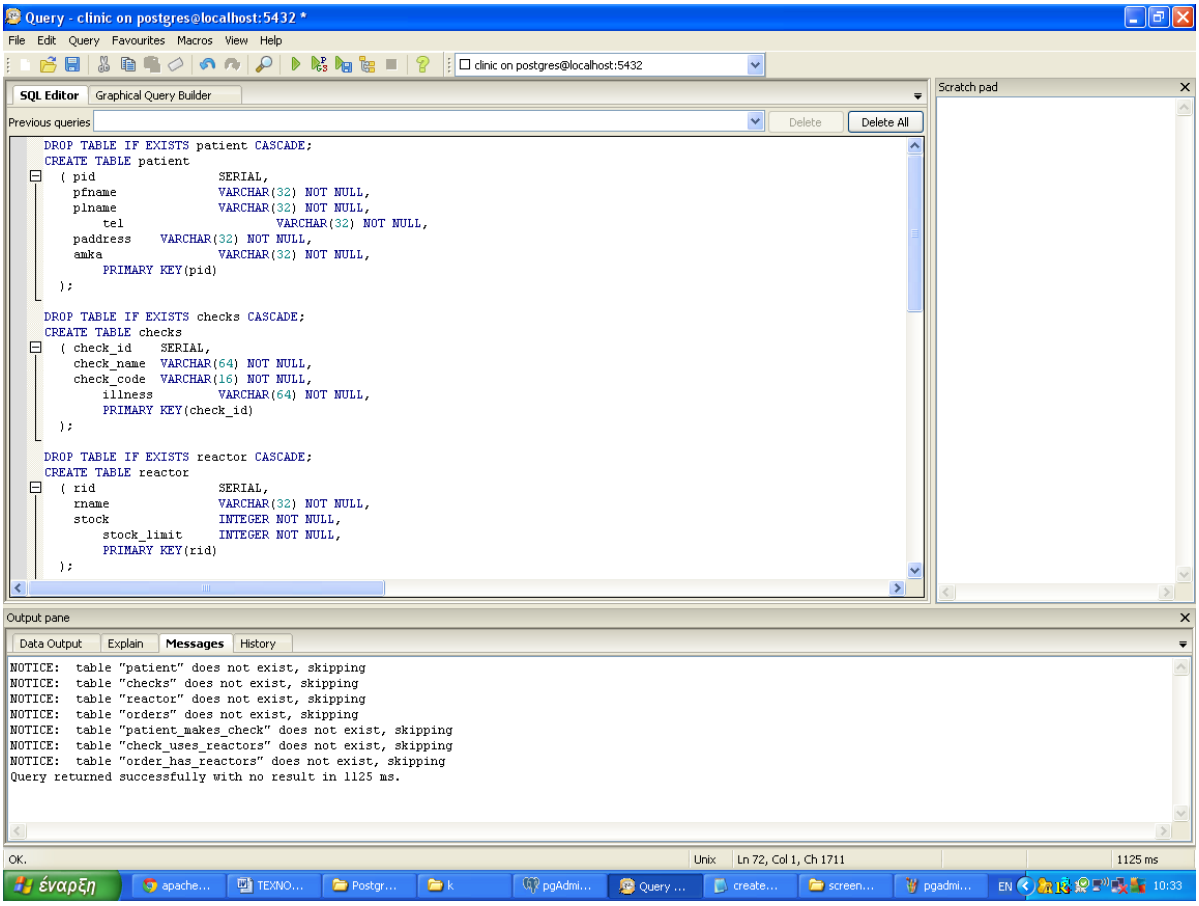
2.Σχεδιασμός με PostgreSQL

Για τη συγκεκριμένη βάση έγινε εγκατάσταση του PostgreSQL συστήματος διαχείρισης βάσης δεδομένων. Η έκδοση που εγκαταστάθηκε είναι η 9.3.Κατά την εγκατάσταση της ,έγινε και η εγκατάσταση του εξυπηρετητή (server) Apache .Ως web διαχειριστικό της βάσης χρησιμοποιήσαμε το pgAdmin .

Με το pgAdmin αφου συνδεθούμε με το localhost μας (5432),δημιουργούμε την βάση μας την οποία την ονομάζουμε clinic .



Τις εντολές της γλώσσας ορισμού δεδομένων για τις σχέσεις που προκύπτουν τις γράφουμε στο query :



Οι οποίες πιο αναλυτικά για τον σχηματισμό πινάκων φαίνονται παρακάτω :

Πίνακας Ασθενή :

```
DROP TABLE IF EXISTS patient CASCADE;
CREATE TABLE patient
 ( pid                  SERIAL,
   pfname               VARCHAR(32) NOT NULL,
   plname               VARCHAR(32) NOT NULL,
        tel                     VARCHAR(32) NOT NULL,
   paddress    VARCHAR(32) NOT NULL,
   amka                 VARCHAR(32) NOT NULL,
        PRIMARY KEY(pid)
 );
```

Πίνακας Εξετάσεων :

```
DROP TABLE IF EXISTS checks CASCADE;
CREATE TABLE checks
 ( check_id     SERIAL,
   check_name          VARCHAR(64) NOT NULL,
   check_code VARCHAR(16) NOT NULL,
        illness          VARCHAR(64) NOT NULL,
        PRIMARY KEY(check_id)
 );
```

Πίνακας αντιδραστηρίων :

```
DROP TABLE IF EXISTS reactor CASCADE;
CREATE TABLE reactor
 ( rid                  SERIAL,
   rname                VARCHAR(32) NOT NULL,
   stock                INTEGER NOT NULL,
        stock_limit     INTEGER NOT NULL,
        PRIMARY KEY(rid)
 );
```

Πίνακας Παραγγελειών :

```
DROP TABLE IF EXISTS orders CASCADE;
CREATE TABLE orders
 ( oid                          SERIAL,
   date_ordered         VARCHAR(32) NOT NULL,
   date_received        VARCHAR(32) NOT NULL,
        PRIMARY KEY(oid)
 );
```

Επείδη στην βάση μας όλες οι σχέσεις είναι M : N ,πέρα από τους πίνακες που έχω για τις οντότητες , αναγκαστικά έχω πίνακες (3) και για τις αναθέσεις , όπου υπάρχουν όλα τα ζεύγη που έχουν δημιουργηθεί .

```
DROP TABLE IF EXISTS patient_makes_check CASCADE;
CREATE TABLE patient_makes_check
 (
        first_id SERIAL,
        check_id        INTEGER REFERENCES checks (check_id),
        pid                     INTEGER REFERENCES patient (pid),
        check_date      VARCHAR(32) NOT NULL,
        check_results TEXT NOT NULL,
        taken_or_not INTEGER NOT NULL,
        PRIMARY KEY (first_id)
 );
```

```
DROP TABLE IF EXISTS check_uses_reactors CASCADE;
CREATE TABLE check_uses_reactors
 (
        second_id       SERIAL,
   check_id     INTEGER REFERENCES checks (check_id),
        rid                     INTEGER REFERENCES reactor (rid),
        check_rquantity INTEGER NOT NULL,
        PRIMARY KEY (second_id)
 );


DROP TABLE IF EXISTS order_has_reactors CASCADE;
CREATE TABLE order_has_reactors
 (
        third_id        SERIAl,
        oid     INTEGER REFERENCES orders (oid),
        rid     INTEGER REFERENCES reactor (rid),
        order_rquantity INTEGER NOT NULL,
        PRIMARY KEY (third_id)
 );
```

3.SQL QUERIES – WEB INTERFACE GUIDELINE

Τα queries υπάρχουν στα αντίστοιχα αρχεία που επισυνάπτονται.Είναι php αρχεία τα οποία για να τα δημιουργήσουμε χρησιμοποιήσαμε το πρόγραμμα Notepad++.
Το πρόγραμμα αυτό είναι ένας εύχρηστος text editor για τους χρήστες των Windows. Το Notepad++ χρησιμοποιείται ακόμα και ως source code editor μιας και μας βοηθά να γράψουμε ή να επεξεργαστούμε κώδικα σε PHP , HTML και δεκάδες ακόμα γλώσσες .

Υπάρχει μενού στο αρχείο **index2.php** μέσω του οποίου ο χρήστης καθοδηγείται στις επιλογές που διαθέτει

```
<?php

echo '
Choose one of the following options: </br>
(The first six options may not be used, if dummy data are entered)</br>

1. <a href="addpatient.php">Add Patient</a></br>
2. <a href="editpatient.php">Edit Patient</a></br>
3. <a href="addcheck.php">Add Check</a></br>
4. <a href="editcheck.php">Edit Check</a></br>
5. <a href="addreactor.php">Add Reactor</a></br>
6. <a href="editreactor.php">Edit Reactor</a></br>
7. <a href="addorder.php">Add order</a></br>
```

```
8. <a href="editorder.php">Edit order</a></br>
</br>-----------------------------------------</br>
9. <a href="addpatientmakescheck.php">Add a check for a patient</a></br>
10. <a href="editpatientmakescheck.php">Edit a check for a patient</a></br>
</br>-----------------------------------------</br>
11. <a href="addcheckusesreactors.php">Add reactor quantity for a check</a></br>
12. <a href="editcheckusesreactors.php">Edit reactor quantity for a check</a></br>
</br>-----------------------------------------</br>
13. <a href="addorderhasreactors.php">Add reactor quantity for an order</a></br>
14. <a href="editorderhasreactors.php">Edit reactor quantity for an order</a></br>
</br>-----------------------------------------</br>
15. <a href="patientsnotreceivedchecks.php">List of patients not received their check
results</a></br>
16. <a href="reactorslist.php">List of reactors</a></br>
17. <a href="checksreactors.php">List of reactors associated with checks</a></br>
';
?>
```

1. Το Query για την Δημιουργία Ασθενούς είναι -> addpatient.php

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";


if ($_SERVER["REQUEST_METHOD"] == "POST"){

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");

  $pfname = $_POST["pfname"];
  $plname = $_POST["plname"];
  $tel = $_POST["tel"];
  $paddress = $_POST["paddress"];
  $amka= $_POST["amka"];

$query = "INSERT INTO patient (pfname, plname, tel, paddress, amka) VALUES
('$pfname', '$plname', '$tel', '$paddress', '$amka')";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
//$row = pg_fetch_row($rs);

echo  'PATIENT ADDED !</br>
<a href="addpatient.php">Add another one</a> or return to the <a
href="index2.php">starting menu</a>.
';
```

```
pg_close($con);

}
else{
?>

<h2>ADD PATIENT</h2>

<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  FirstName: <input type="text" name="pfname">
  <br>
  LastName: <input type="text" name="plname">
  <br>
  Telephone: <input type="text" name="tel">
  <br>
  Address: <input type="text" name="paddress">
  <br>
  AMKA: <input type="text" name="amka">

  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
}

?>
```

2. **Τα Queries για την Ενημέρωση ασθενούς είναι -> editpatient.php, editpatient2.php**

editpatient.php:

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");


$query = "select * from patient";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
```

```
?>

<label>Select the patient you want to edit:</label><br>
<form method="post" action="editpatient2.php">
<select name="patientlist">
<?php

while ($row = pg_fetch_assoc($rs)) {
?>
<option value="<?php echo $row['pid'];?>">
<?php
  echo $row['pid'];
  echo ' ';
  echo $row['pfname'];
  echo ' ';
  echo $row['plname'];
  echo '</br>';
?>
  </option>
<?php
}
?>
</select>
<input name="editpatientform" type="submit" value="Submit">
</form>
<?php




pg_close($con);
?>
```

editpatient2.php :

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");



if ($_SERVER["REQUEST_METHOD"] == "POST" && !isset($_POST['editpatientform'])){
```

```
$pid = $_POST['patientlist'];

$query = "select * from patient where pid=$pid";
$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
$row = pg_fetch_assoc($rs);
?>



<h2>EDIT PATIENT</h2>

<form method="post" action="editpatient2.php">
  FirstName: <input type="text" name="pfname" value="<?php echo $row['pfname'];?>">
  <br>
  LastName: <input type="text" name="plname" value="<?php echo $row['plname'];?>">
  <br>
  Telephone: <input type="text" name="tel" value="<?php echo $row['tel'];?>">
  <br>
  Address: <input type="text" name="paddress" value="<?php echo $row['paddress'];?>">
  <br>
  AMKA: <input type="text" name="amka" value="<?php echo $row['amka'];?>">

        <input type="hidden" name="pid" value="<?php echo $row['pid'];?>">
  <br><br>
  <input type="submit" name="editpatientform" value="Submit">
</form>

<?php

}
else if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['editpatientform'])){

  $pid = $_POST["pid"];
  $pfname = $_POST["pfname"];
  $plname = $_POST["plname"];
  $tel = $_POST["tel"];
  $paddress = $_POST["paddress"];
  $amka= $_POST["amka"];

$query = "update patient set pfname='$pfname', plname='$plname', tel='$tel', paddress='$paddress', amka='$amka' where pid=$pid ";
//echo $query;
$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
//$row = pg_fetch_row($rs);
```

```
echo  'PATIENT EDITED !</br>
<a href="editpatient.php">Edit another one</a> or return to the <a
href="index2.php">starting menu</a>.
';
}

pg_close($con);
?>
```

3. Το Query για την Δημιουργία Εξέτασης είναι -> addcheck.php

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";


if ($_SERVER["REQUEST_METHOD"] == "POST"){

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");

  $check_name = $_POST["check_name"];
  $check_code = $_POST["check_code"];
  $illness = $_POST["illness"];


$query = "INSERT INTO checks (check_name, check_code, illness) VALUES
('$check_name', '$check_code', '$illness')";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
//$row = pg_fetch_row($rs);

echo  'CHECK ADDED !</br>
<a href="addcheck.php">Add another one</a> or return to the <a
href="index2.php">starting menu</a>.
';

pg_close($con);

}
else{
?>

<h2>ADD CHECK</h2>
```

```
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  CheckName: <input type="text" name="check_name">
  <br>
  CheckCode: <input type="text" name="check_code">
  <br>
  Illness: <input type="text" name="illness">


  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
}


?>
```

4. Τα Queries για την Ενημέρωση Εξέτασης είναι -> editcheck.php, editcheck2.php

editcheck.php :

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");


$query = "select * from checks";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
?>

<label>Select the check you want to edit:</label><br>
<form method="post" action="editcheck2.php">
<select name="checklist">
<?php

while ($row = pg_fetch_assoc($rs)) {
?>
<option value="<?php echo $row['check_id'];?>">
<?php
  echo $row['check_id'];
```

```
  echo ' ';
  echo $row['check_name'];
  echo '</br>';
?>
  </option>
<?php
}
?>
</select>
<input name="editche" type="submit" value="Submit">
</form>
<?php




pg_close($con);
?>
```

editcheck2.php :

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";
$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
    or die ("Could not connect to server\n");



if ($_SERVER["REQUEST_METHOD"] == "POST" &&
!isset($_POST['editcheckform'])){


$check_id = $_POST['checklist'];

$query = "select * from checks where check_id=$check_id";
$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
$row = pg_fetch_assoc($rs);
?>



<h2>EDIT CHECK</h2>

<form method="post" action="editcheck2.php">
```

```
  CheckName: <input type="text" name="check_name" value="<?php echo
$row['check_name'];?>">
  <br>
  CheckCode: <input type="text" name="check_code"  value="<?php echo
$row['check_code'];?>">
  <br>
  Illness: <input type="text" name="illness" value="<?php echo $row['illness'];?>">


        <input type="hidden" name="check_id" value="<?php echo
$row['check_id'];?>">
  <br><br>
  <input type="submit" name="editcheckform" value="Submit">
</form>

<?php

}
else if ($_SERVER["REQUEST_METHOD"] == "POST" &&
isset($_POST['editcheckform'])){

 $check_id = $_POST["check_id"];
 $check_name = $_POST["check_name"];
 $check_code = $_POST["check_code"];
 $illness = $_POST["illness"];


$query = "update checks set check_name='$check_name', check_code='$check_code',
illness='$illness' where check_id=$check_id";
//echo $query;
$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
//$row = pg_fetch_row($rs);

echo  'CHECK EDITED !</br>
<a href="editcheck.php">Edit another one</a> or return to the <a
href="index2.php">starting menu</a>.
';
}

pg_close($con);
?>
```

5.Το Query για την Δημιουργία Αντιδραστηρίου είναι -> addreactor.php

```
        <?php

        $host = "localhost";
        $user = "postgres";
        $pass = "1234";
        $db = "clinic";
```

```
if ($_SERVER["REQUEST_METHOD"] == "POST"){

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
    or die ("Could not connect to server\n");

  $rname = $_POST["rname"];
  $stock = $_POST["stock"];
  $stock_limit = $_POST["stock_limit"];


$query = "INSERT INTO reactor (rname, stock, stock_limit) VALUES ('$rname', '$stock', '$stock_limit')";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
//$row = pg_fetch_row($rs);

echo  'REACTOR ADDED !</br>
<a href="addreactor.php">Add another one</a> or return to the <a href="index2.php">starting menu</a>.
';

pg_close($con);

}
else{
?>

<h2>ADD REACTOR</h2>

<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
   ReactorName: <input type="text" name="rname">
   <br>
   Stock: <input type="text" name="stock">
   <br>
   StockLimit: <input type="text" name="stock_limit">


   <br><br>
   <input type="submit" name="submit" value="Submit">
</form>

<?php
}


?>
```

6. Τα Queries για την Ενημέρωση Αντιδραστηρίου είναι -> editreactor.php, editreactor2.php

editreactor.php :

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");


$query = "select * from reactor";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
?>

<label>Select the reactor you want to edit:</label><br>
<form method="post" action="editreactor2.php">
<select name="reactorlist">
<?php

while ($row = pg_fetch_assoc($rs)) {
?>
<option value="<?php echo $row['rid'];?>">
<?php
  echo $row['rid'];
  echo ' ';
  echo $row['rname'];
  echo '</br>';
?>
  </option>
<?php
}
?>
</select>
<input name="editreact" type="submit" value="Submit">
</form>
<?php




pg_close($con);
?>
```

editreactor2.php :

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");


$query = "select * from reactor";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
?>

<label>Select the reactor you want to edit:</label><br>
<form method="post" action="editreactor2.php">
<select name="reactorlist">
<?php

while ($row = pg_fetch_assoc($rs)) {
?>
<option value="<?php echo $row['rid'];?>">
<?php
  echo $row['rid'];
  echo ' ';
  echo $row['rname'];
  echo '</br>';
?>
  </option>
<?php
}
?>
</select>
<input name="editreact" type="submit" value="Submit">
</form>
<?php




pg_close($con);
?>
```

7. Το Query για την Δημιουργία Παραγγελίας είναι -> addorder.php

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";


if ($_SERVER["REQUEST_METHOD"] == "POST"){

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");


 $date_ordered = $_POST["date_ordered"];


$query = "INSERT INTO orders (date_ordered,date_received) VALUES
('$date_ordered', '')";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
//$row = pg_fetch_row($rs);

echo  'ORDER ADDED !</br>
<a href="addorder.php">Add another one</a> or return to the <a
href="index2.php">starting menu</a>.
';

pg_close($con);

}
else{
?>

<h2>ADD PATIENT</h2>
<form method="post" action="<?php echo
htmlspecialchars($_SERVER["PHP_SELF"]);?>">
  DateOrdered: <input type="text" name="date_ordered">

  <br><br>
  <input type="submit" name="submit" value="Submit">
</form>

<?php
}


?>
```

8. Τα Queries για την Ενημέρωση Παραγγελίας είναι -> editorder.php, editorder2.php

editorder.php:

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");


$query = "select * from orders";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
?>

<label>Select the order you want to edit:</label><br>
<form method="post" action="editorder2.php">
<select name="orderedlist">
<?php

while ($row = pg_fetch_assoc($rs)) {
?>
<option value="<?php echo $row['oid'];?>">
<?php
  echo $row['oid'];
  echo ' ';
  echo $row['date_ordered'];
  echo '</br>';
?>
  </option>
<?php
}
?>
</select>
<input name="editorder" type="submit" value="Submit">
</form>
<?php




pg_close($con);
?>
```

editorder2.php:

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");



if ($_SERVER["REQUEST_METHOD"] == "POST" &&
!isset($_POST['editorderform'])){


$oid = $_POST['orderedlist'];

$query = "select * from orders where oid=$oid and date_received="";
$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
$row = pg_fetch_assoc($rs);
?>



<h2>EDIT ORDER</h2>

<form method="post" action="editorder2.php">
  DateOrdered: <input type="text" name="date_ordered" value="<?php echo
$row['date_ordered'];?>"><br>
  DateReceived: <input type="text" name="date_received" value="<?php
echo $row['date_received'];?>">


        <input type="hidden" name="oid" value="<?php echo
$row['oid'];?>">
  <br><br>
  <input type="submit" name="editorderform" value="Submit">
</form>

<?php

}
else if ($_SERVER["REQUEST_METHOD"] == "POST" &&
isset($_POST['editorderform'])){
```

```
 $oid = $_POST["oid"];
 $date_ordered = $_POST["date_ordered"];
 $date_received = $_POST["date_received"];

 //an prosthesame hmeromhnia, paralavhs, phgainoume na ananewsoume to stock
 if($date_received!=""){

$query2 = "select * from order_has_reactors where oid=$oid";
$rs2 = pg_query($con, $query2) or die("Cannot execute query: $query\n");
$row2 = pg_fetch_assoc($rs2);


while ($row2 = pg_fetch_assoc($rs2)) {

$order_rquantity = $row2['order_rquantity'];
$rid = $row2['rid'];


$query3 = "update reactor set stock=stock+$order_rquantity where rid=$rid ";
$rs3 = pg_query($con, $query3) or die("Cannot execute query: $query\n");


} //end of while


 }


$query = "update orders set date_ordered='$date_ordered', date_received='$date_received' where oid=$oid";
//echo $query;
$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
//$row = pg_fetch_row($rs);

echo  'ORDER EDITED !</br>
<a href="editorder.php">Edit another one</a> or return to the <a href="index2.php">starting menu</a>.
';
}

pg_close($con);
?>
```

9.Τα Queries για την Ανάθεση Εξέτασης σε Ασθενή είναι -> addpatientmakescheck.php,addpatientmakescheck2.php, addpatientmakescheck3.php

addpatientmakescheck.php :

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");


$query = "select * from patient";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");


?>

<label>Select the patient you want to associate with a check:</label><br>
<form method="post" action="addpatientmakescheck2.php">
<select name="patientlist">
<?php

while ($row = pg_fetch_assoc($rs)) {
?>
<option value="<?php echo $row['pid'];?>">
<?php
  echo $row['pid'];
  echo ' ';
  echo $row['pfname'];
  echo ' ';
  echo $row['plname'];
  echo '</br>';
?>
  </option>
<?php
}
?>
</select>
<input name="addcheck2patient" type="submit" value="Submit">
</form>
<?php




pg_close($con);
?>
```

addpatientmakescheck2.php :

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");

$pid = $_POST['patientlist'];



$query = "select * from checks";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
?>

<label>Select the check you want to edit:</label><br>
<form method="post" action="addpatientmakescheck3.php">
<select name="checklist">
<?php

while ($row = pg_fetch_assoc($rs)) {
?>
<option value="<?php echo $row['check_id'];?>">
<?php
  echo $row['check_id'];
  echo ' ';
  echo $row['check_name'];
  echo '</br>';
?>
  </option>
<?php
}
?>
</select>
</br>
Date of Check <input type="text" name="check_date"></br>
Check Results <input type="text" name="check_results"></br>

<select name="taken_or_not">
<option value="0" selected>NO</option>
<option value="1">YES</option>
</select>
</br>
```

```
<input type="hidden" name="pid" value="<?php echo $pid?>">
<input name="editche" type="submit" value="Submit">

</form>
<?php



pg_close($con);
?>
```

addpatientmakescheck3.php :

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
    or die ("Could not connect to server\n");

  $check_id = $_POST['checklist'];

  $pid = $_POST['pid'];
  $check_date = $_POST["check_date"];
  $check_results = $_POST["check_results"];
  $taken_or_not = $_POST["taken_or_not"];



$query = "INSERT INTO patient_makes_check (pid, check_id, check_date,
check_results, taken_or_not) VALUES ( $pid, $check_id, '$check_date',
'$check_results', $taken_or_not)";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");



$query2 = "select * from check_uses_reactors where check_id=$check_id";
$rs2 = pg_query($con, $query2) or die("Cannot execute query: $query\n");
$row2 = pg_fetch_assoc($rs2);


while ($row2 = pg_fetch_assoc($rs2)) {

$check_rquantity = $row2['check_rquantity'];
$rid = $row2['rid'];
```

```
$query3 = "update reactor set stock=stock-$check_rquantity where rid=$rid ";
$rs3 = pg_query($con, $query3) or die("Cannot execute query: $query\n");


$query4 = "select * from reactor where rid=$rid";
$rs4 = pg_query($con, $query4) or die("Cannot execute query: $query\n");
$row4 = pg_fetch_assoc($rs4);

$rname=$row4['rname'];
$stock=$row4['stock'];
$stock_limit=$row4['stock_limit'];


if ($stock < $stock_limit){
   echo '</br></br><b>----> ALERT: Stock for ';
         echo $rname;
         echo' has gone below the stock_limit <-----</b></br></br>';
}

} //end of while

echo  'CHECK ASSOCIATED WITH A PATIENT !</br>
<a href="addpatientmakescheck.php">Associate another check to patient</a> or
return to the <a href="index2.php">starting menu</a>.
';


pg_close($con);
?>
```

10. Τα Queries για την Ενημέρωση Εξέτασης σε Ασθενή είναι-> editpatientmakescheck.php,editpatientmakescheck2.php, editpatientmakescheck3.php

editpatientmakescheck.php :

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");

echo 'lala';
```

```
$query = "select * from patient_makes_check";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");


?>


<label>Select the check association with patient you want to edit:</label><br>
<form method="post" action="editpatientmakescheck2.php">
<select name="associationlist">
<?php

while ($row = pg_fetch_assoc($rs)) {

$pid=$row['pid'];
$query1 = "select * from patient where pid=$pid";
$rs1 = pg_query($con, $query1) or die("Cannot execute query: $query\n");
$row1 = pg_fetch_assoc($rs1);


$check_id=$row['check_id'];
$query2 = "select * from checks where check_id=$check_id";
$rs2 = pg_query($con, $query2) or die("Cannot execute query: $query\n");
$row2 = pg_fetch_assoc($rs2);

?>
<option value="<?php echo $row['first_id'];?>">
<?php
  echo $row1['pfname'];
  echo ' ';
  echo $row1['plname'];
  echo ' - ';
  echo $row2['check_name'];
  echo '</br>';
?>
  </option>
<?php
}
?>
</select>
</br></br>
<input name="editassociation" type="submit" value="Submit">
</form>

<?php


pg_close($con);
?>
```

editpatientmakescheck2.php :

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");

$first_id=$_POST['associationlist'];

$query = "select * from patient_makes_check where first_id=$first_id";
$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
$row = pg_fetch_assoc($rs);

$pid=$row['pid'];
$query1 = "select * from patient where pid=$pid";
$rs1 = pg_query($con, $query1) or die("Cannot execute query: $query\n");
$row1 = pg_fetch_assoc($rs1);


$check_id=$row['check_id'];
$query2 = "select * from checks where check_id=$check_id";
$rs2 = pg_query($con, $query2) or die("Cannot execute query: $query\n");
$row2 = pg_fetch_assoc($rs2);

?>

<h2>EDIT CHECK ASSOCIATION FOR A PATIENT</h2>
<form method="post" action="editpatientmakescheck3.php">
<?php
  echo 'Patient named ';
  echo $row1['pfname'];
  echo ' ';
  echo $row1['plname'];
  echo ' with check associated named ';
  echo $row2['check_name'];
  echo 'is now edited!';
  echo '</br>';
?>
</br></br>
Date of Check <input type="text" name="check_date" value="<?php echo
$row['check_date'];?>"></br>
Check Results <input type="text" name="check_results" value="<?php echo
$row['check_results'];?>"></br>
```

```
<?php if ($row['taken_or_not']==0) {?>
<select name="taken_or_not">
<option value="0" selected>NO</option>
<option value="1">YES</option>
</select>
<?php
}
else if ($row['taken_or_not']==1) {?>
<select name="taken_or_not">
<option value="1" selected>YES</option>
<option value="0">NO</option>
</select>
<?php
}
?>

</br>
<input type="hidden" name="first_id" value="<?php echo $first_id?>">
<input name="editassociation" type="submit" value="Submit">

</form>
<?
pg_close($con);
?>
```

editpatientmakescheck3.php :

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
    or die ("Could not connect to server\n");

  $first_id = $_POST['first_id'];

  $check_date = $_POST["check_date"];
  $check_results = $_POST["check_results"];
  $taken_or_not = $_POST["taken_or_not"];



$query = "UPDATE patient_makes_check set check_date='$check_date',
check_results='$check_results', taken_or_not=$taken_or_not where
first_id=$first_id";
//echo $query;
$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
```

```
//$row = pg_fetch_row($rs);

echo  'CHECK ASSOCIATED WITH A PATIENT EDITED!</br>
<a href="editpatientmakescheck.php">Edit another check association to patient</a>
or return to the <a href="index2.php">starting menu</a>.
';


pg_close($con);
?>
```

11. Τα Queries για την Ανάθεση Αντιδραστηρίου σε Εξέταση είναι -> addcheckusesreactors.php,addcheckusesreactors2.php, addcheckusesreactors3.php

addcheckusesreactors.php:

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");


$query = "select * from reactor";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");


?>



<label>Select the reactor you want to associate with a check:</label><br>
<form method="post" action="addcheckusesreactors2.php">
<select name="reactorlist">
<?php

while ($row = pg_fetch_assoc($rs)) {
?>
<option value="<?php echo $row['rid'];?>">
<?php
  echo $row['rid'];
  echo ' ';
  echo $row['rname'];
  echo '</br>';
```

```
?>
  </option>
<?php
}
?>
</select>
<input name="addreactor2check" type="submit" value="Submit">
</form>
<?php




pg_close($con);
?>
```

addcheckusesreactors2.php:

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
    or die ("Could not connect to server\n");

$rid = $_POST['reactorlist'];



$query = "select * from checks";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
?>

<label>Select the check you want to edit:</label><br>
<form method="post" action="addcheckusesreactors3.php">
<select name="checklist">
<?php

while ($row = pg_fetch_assoc($rs)) {
?>
<option value="<?php echo $row['check_id'];?>">
<?php
  echo $row['check_id'];
  echo ' ';
  echo $row['check_name'];
```

```
  echo '</br>';
?>
  </option>
<?php
}
?>
</select>
</br>
Reactor Quantity for this Check <input type="text"
name="check_rquantity"></br>

</br>
<input type="hidden" name="rid" value="<?php echo $rid?>">
<input name="editche" type="submit" value="Submit">

</form>
<?php



pg_close($con);
?>
```

addcheckusesreactors3.php:

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");

  $check_id = $_POST['checklist'];

  $rid = $_POST['rid'];

  $check_rquantity = $_POST["check_rquantity"];



$query = "INSERT INTO check_uses_reactors (check_id, rid, check_rquantity)
VALUES ( $check_id, $rid, $check_rquantity)";
//echo $query;
$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
//$row = pg_fetch_row($rs);

echo  'CHECK ASSOCIATED WITH A REACTOR !</br>
```

```
<a href="addcheckusesreactors.php">Associate another check to reactor</a> or return to the <a href="index2.php">starting menu</a>.
';


pg_close($con);
?>
```

12. Τα Queries για την Ενημέρωση Αντιδραστηρίου σε Εξέταση είναι -> editcheckusesreactors.php,editcheckusesreactors2.php, editcheckusesreactors3.php

editcheckusesreactors.php:

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");

echo 'lala';

$query = "select * from check_uses_reactors";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");


?>


<label>Select the check association with reactor you want to edit:</label><br>
<form method="post" action="editcheckusesreactors2.php">
<select name="associationlist">
<?php

while ($row = pg_fetch_assoc($rs)) {

$rid=$row['rid'];
$query1 = "select * from reactor where rid=$rid";
$rs1 = pg_query($con, $query1) or die("Cannot execute query: $query\n");
$row1 = pg_fetch_assoc($rs1);


$check_id=$row['check_id'];
$query2 = "select * from checks where check_id=$check_id";
```

```
$rs2 = pg_query($con, $query2) or die("Cannot execute query: $query\n");
$row2 = pg_fetch_assoc($rs2);

?>
<option value="<?php echo $row['second_id'];?>">
<?php
  echo $row1['rname'];
  echo ' - ';
  echo $row2['check_name'];
  echo '</br>';
?>
  </option>
<?php
}
?>
</select>
</br></br>
<input name="editassociation" type="submit" value="Submit">
</form>

<?php


pg_close($con);
?>
```

editcheckusesreactors2.php:

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");

$second_id=$_POST['associationlist'];

$query = "select * from check_uses_reactors where second_id=$second_id";
$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
$row = pg_fetch_assoc($rs);

$rid=$row['rid'];
$query1 = "select * from reactor where rid=$rid";
$rs1 = pg_query($con, $query1) or die("Cannot execute query: $query\n");
$row1 = pg_fetch_assoc($rs1);
```

```
$check_id=$row['check_id'];
$query2 = "select * from checks where check_id=$check_id";
$rs2 = pg_query($con, $query2) or die("Cannot execute query: $query\n");
$row2 = pg_fetch_assoc($rs2);

?>

<h2>EDIT CHECK ASSOCIATION FOR A REACTOR</h2>
<form method="post" action="editcheckusesreactors3.php">
<?php
  echo 'Reactor named ';
  echo $row1['rname'];
  echo ' with check associated named ';
  echo $row2['check_name'];
  echo 'is now edited!';
  echo '</br>';
?>
</br></br>
Check Reactor Quantity <input type="text" name="check_rquantity"
value="<?php echo $row['check_rquantity'];?>"></br>

</br>
<input type="hidden" name="second_id" value="<?php echo $second_id?>">
<input name="editassociation" type="submit" value="Submit">

</form>
<?
pg_close($con);
?>
```

editcheckusesreactors3.php:

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");

  $second_id = $_POST['second_id'];

  $check_rquantity = $_POST["check_rquantity"];
```

```
$query = "UPDATE check_uses_reactors set check_rquantity='$check_rquantity' where second_id=$second_id";
//echo $query;
$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
//$row = pg_fetch_row($rs);

echo  'CHECK ASSOCIATED WITH A REACTOR EDITED!</br>
<a href="editcheckusesreactors.php">Edit another check association to reactor</a> or return to the <a href="index2.php">starting menu</a>.
';


pg_close($con);
?>
```

13. Τα Queries για την Ανάθεση Αντιδραστηρίου σε Παραγγελία είναι -> addorderhasreactors.php,addorderhasreactors2.php, addorderhasreactors3.php

addorderhasreactors.php:

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");


$query = "select * from reactor";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");


?>



<label>Select the reactor you want to associate with an order:</label><br>
<form method="post" action="addorderhasreactors2.php">
<select name="reactorlist">
<?php

while ($row = pg_fetch_assoc($rs)) {
?>
<option value="<?php echo $row['rid'];?>">
<?php
```

```
  echo $row['rid'];
  echo ' ';
  echo $row['rname'];
  echo '</br>';
?>
  </option>
<?php
}
?>
</select>
<input name="addreactor2check" type="submit" value="Submit">
</form>
<?php




pg_close($con);
?>
```

addorderhasreactors2.php:

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
    or die ("Could not connect to server\n");

$rid = $_POST['reactorlist'];



$query = "select * from orders  where date_received="";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
?>

<label>Select the order you want to edit:</label><br>
<form method="post" action="addorderhasreactors3.php">
<select name="orderedlist">
<?php

while ($row = pg_fetch_assoc($rs)) {
?>
<option value="<?php echo $row['oid'];?>">
<?php
```

```
  echo $row['oid'];
  echo ' ';
  echo $row['date_ordered'];
  echo '</br>';
?>
  </option>
<?php
}
?>
</select>
</br>
Order Reactor Quantity <input type="text" name="order_rquantity"></br>

</br>
<input type="hidden" name="rid" value="<?php echo $rid?>">
<input name="editche" type="submit" value="Submit">

</form>
<?php



pg_close($con);
?>
```

addorderhasreactors3.php:

```
    <?php

    $host = "localhost";
    $user = "postgres";
    $pass = "1234";
    $db = "clinic";

    $con = pg_connect("host=$host dbname=$db user=$user password=$pass")
       or die ("Could not connect to server\n");

      $oid = $_POST['orderedlist'];

      $rid = $_POST['rid'];
      $order_rquantity = $_POST["order_rquantity"];



    $query = "INSERT INTO order_has_reactors (oid, rid, order_rquantity)
    VALUES ( $oid, $rid, $order_rquantity)";
    //echo $query;
    $rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
    //$row = pg_fetch_row($rs);
```

```
    echo  'REACTOR ASSOCIATED WITH AN ORDER!</br>
    <a href="addorderhasreactors.php">Associate another reactor to an order</a>
    or return to the <a href="index2.php">starting menu</a>.
    ';


    pg_close($con);
    ?>
```

14. Τα Queries για την Ενημέρωση Αντιδραστηρίου σε Παραγγελία είναι-> editorderhasreactors.php,editorderhasreactors2.php, editorderhasreactors3.php

editorderhasreactors.php:

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");


$query = "select * from order_has_reactors";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");


?>


<label>Select the order associated with reactor you want to edit:</label><br>
<form method="post" action="editorderhasreactors2.php">
<select name="associationlist">
<?php

while ($row = pg_fetch_assoc($rs)) {

$rid=$row['rid'];
$query1 = "select * from reactor where rid=$rid";
$rs1 = pg_query($con, $query1) or die("Cannot execute query: $query\n");
$row1 = pg_fetch_assoc($rs1);
```

```
$oid=$row['oid'];
$query2 = "select * from orders where oid=$oid";
$rs2 = pg_query($con, $query2) or die("Cannot execute query: $query\n");
$row2 = pg_fetch_assoc($rs2);

?>
<option value="<?php echo $row['third_id'];?>">
<?php
  echo $row1['rname'];
  echo ' - ';
  echo $row2['date_ordered'];
  echo '</br>';
?>
  </option>
<?php
}
?>
</select>
</br></br>
<input name="editassociation" type="submit" value="Submit">
</form>

<?php


pg_close($con);
?>
```

editorderhasreactors2.php:

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
    or die ("Could not connect to server\n");

$third_id=$_POST['associationlist'];

$query = "select * from order_has_reactors where third_id=$third_id";
$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
$row = pg_fetch_assoc($rs);

$rid=$row['rid'];
$query1 = "select * from reactor where rid=$rid";
$rs1 = pg_query($con, $query1) or die("Cannot execute query: $query\n");
$row1 = pg_fetch_assoc($rs1);
```

```
$oid=$row['oid'];
$query2 = "select * from orders where oid=$oid";
$rs2 = pg_query($con, $query2) or die("Cannot execute query: $query\n");
$row2 = pg_fetch_assoc($rs2);

?>

<h2>EDIT ORDER ASSOCIATED WITH A REACTOR</h2>
<form method="post" action="editorderhasreactors3.php">
<?php
  echo 'Reactor named ';
  echo $row1['rname'];
  echo ' with order associated made on date ';
  echo $row2['date_ordered'];
  echo 'is now edited!';
  echo '</br>';
?>
</br></br>
Order Reactor Quantity <input type="text" name="order_rquantity" value="<?php
echo $row['order_rquantity'];?>"></br>

</br>
<input type="hidden" name="third_id" value="<?php echo $third_id?>">
<input name="editassociation" type="submit" value="Submit">

</form>
<?
pg_close($con);
?>
```

editorderhasreactors3.php:

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");

  $third_id = $_POST['third_id'];

  $order_rquantity = $_POST["order_rquantity"];
```

```
$query = "UPDATE order_has_reactors set order_rquantity='$order_rquantity' where
third_id=$third_id";
//echo $query;
$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");
//$row = pg_fetch_row($rs);

echo  'ORDER ASSOCIATED WITH A REACTOR EDITED!</br>
<a href="editorderhasreactors.php">Edit another order associated with a reactor</a>
or return to the <a href="index2.php">starting menu</a>.
';


pg_close($con);
?>
```

15. Το Query για την Λίστα ατόμων που δεν παρέλαβαν εξέταση είναι -> patientsnotreceivedchecks.php

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");


$query = "select * from patient_makes_check where taken_or_not=0";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");

echo '<h2>PATIENT LIST NOT RECEIVED THEIR CHECKS</h2></br></br>';

while ($row = pg_fetch_assoc($rs)) {

$pid=$row['pid'];
$query1 = "select * from patient where pid=$pid";
$rs1 = pg_query($con, $query1) or die("Cannot execute query: $query\n");
$row1 = pg_fetch_assoc($rs1);


$check_id=$row['check_id'];
$query2 = "select * from checks where check_id=$check_id";
$rs2 = pg_query($con, $query2) or die("Cannot execute query: $query\n");
$row2 = pg_fetch_assoc($rs2);
```

```
 echo 'Patient with name ';
 echo '<b>';
 echo $row1['pfname'];
 echo ' ';
 echo $row1['plname'];
 echo '</b>';
 echo ' has not received yet the check ';
 echo '<b>';
 echo $row2['check_name'];
 echo '</b>';
 echo ' </br>';

}
echo '</br></br>Return to the <a href="index2.php">starting menu</a>.';
pg_close($con);
?>
```

16. Το Query για την Αναλυτική Λίστα αντιδραστηρίων είναι -> reactorslist.php

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");


$query = "select * from reactor";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");

echo '<h2>REACTOR LIST NOT RECEIVED THEIR CHECKS</h2></br></br>';

while ($row = pg_fetch_assoc($rs)) {

$rid=$row['rid'];
$rname=$row['rname'];
$stock=$row['stock'];
$stock_limit=$row['stock_limit'];

  echo $rid;
  echo ' : Reactor with name ';
  echo '<b>';
  echo $rname;
```

```
  echo '</b>';
  echo ' and stock = ';
  echo '<b>';
  echo $stock;
  echo '</b>';
  echo ' has stock limit = ';
  echo '<b>';
  echo $stock_limit;
  echo '</b>';
  echo ' </br>';

 }

 echo '</br></br>Return to the <a href="index2.php">starting menu</a>.';

 pg_close($con);
 ?>
```

17. Το Query για την Αναλυτική Λίστα συσχετισμών αντιδραστηρίων-εξέτασεων είναι -> checksreactors.php

```
<?php

$host = "localhost";
$user = "postgres";
$pass = "1234";
$db = "clinic";

$con = pg_connect("host=$host dbname=$db user=$user password=$pass")
   or die ("Could not connect to server\n");


$query = "select * from check_uses_reactors";

$rs = pg_query($con, $query) or die("Cannot execute query: $query\n");

echo '<h2>REACTORS ASSOCIATION LIST WITH CHECKS</h2></br></br>';

while ($row = pg_fetch_assoc($rs)) {

$rid=$row['rid'];
$query1 = "select * from reactor where rid=$rid";
$rs1 = pg_query($con, $query1) or die("Cannot execute query: $query\n");
$row1 = pg_fetch_assoc($rs1);


$check_id=$row['check_id'];
$query2 = "select * from checks where check_id=$check_id";
$rs2 = pg_query($con, $query2) or die("Cannot execute query: $query\n");
```

```
$row2 = pg_fetch_assoc($rs2);

  echo 'Reactor with name ';
  echo '<b>';
  echo $row1['rname'];
  echo '</b>';
  echo ' is associated with the check ';
  echo '<b>';
  echo $row2['check_name'];
  echo '</b>';
  echo ' </br>';

}
echo '</br></br>Return to the <a href="index2.php">starting menu</a>.';
pg_close($con);
?>
```

Υποσημειώσεις:

Κάθε φορά που τα αποθέματα κάποιου αντιδραστηρίου πέφτουν κάτω από τα ελάχιστα όρια ,εμφανίζεται το μήνυμα αμέσως μετά την εισαγωγή των εξετάσεων που προκάλεσαν αυτήν την μείωση

Στο αρχείο addcheckusesreactors3.php γίνονται οι απαραίτητοι έλεγχοι και εκτυπώνεται το αντίστοιχο μήνυμα εάν η ανάθεση εξέτασης ρίξει το stock κάτω από το stock_limit του αντιδραστηρίου.

Η Ενημέρωση για παραλαβή κάποιας παραγγελίας έχει ως αποτέλεσμα την αυτόματη αύξηση των αποθεμάτων των αντιδραστηρίων

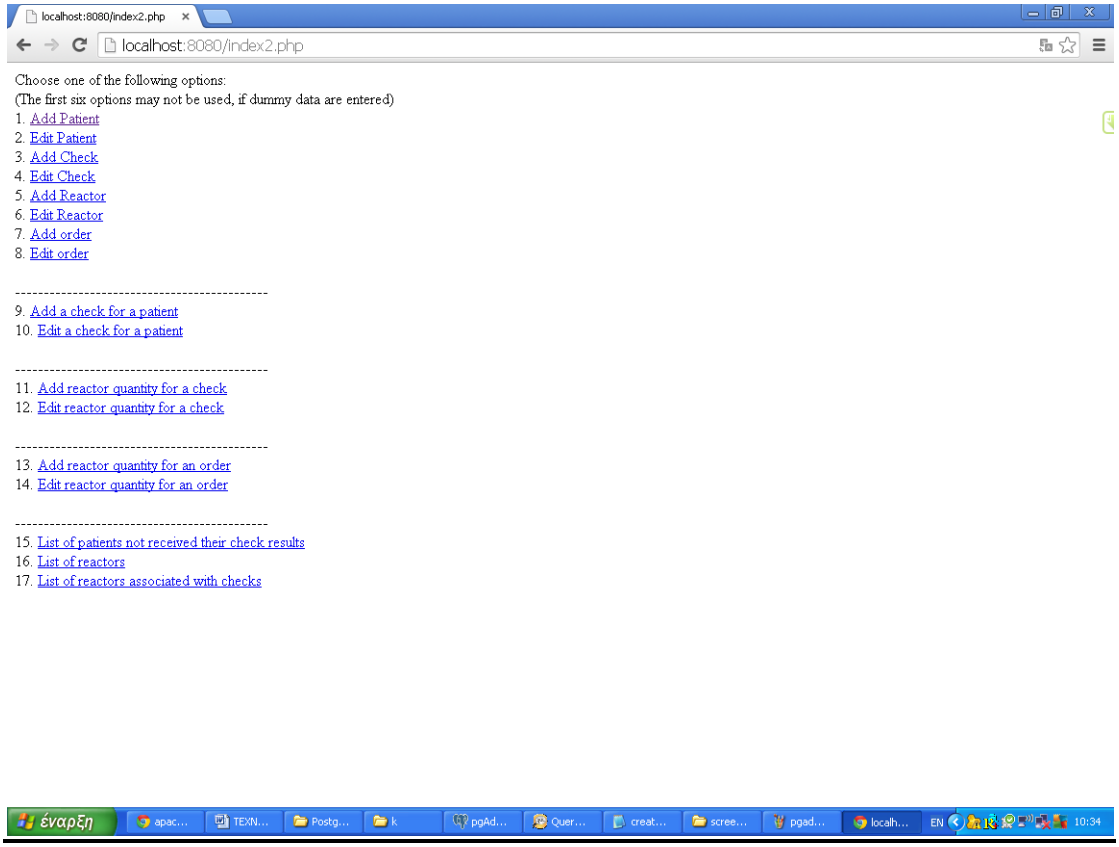
Στο αρχείο editorder2.php μόλις εισαχθεί ημερομηνία παραλαβής της παραγγελίας τότε προστίθενται οι ποσότητες (order_rquantity) των αντιδραστηρίων που υπάρχουν στην παραγγελία στο stock του καθενός αντιδραστηρίου**.**

4.Σύνδεση με τον server Apache

Για να λειτουργήσει πρακτικά ο apache και να τον τρέξουμε σε υπολογιστή στο σπίτι μας πρέπει να έχουμε κάνει κάποιες ρυθμίσεις. Έχουμε εξαιρέσει το πρόγραμμα από τον firewall (για windows) και να έχουμε ρυθμίσει το router ώστε όταν δέχεται αίτηση σύνδεσης στην θύρα 80 (που χρησιμοποιείται για σύνδεση http) να την προωθεί στον υπολογιστή .Και φυσικά πρέπει να έχουμε ανοιχτό τον υπολογιστή με το πρόγραμμα να τρέχει.

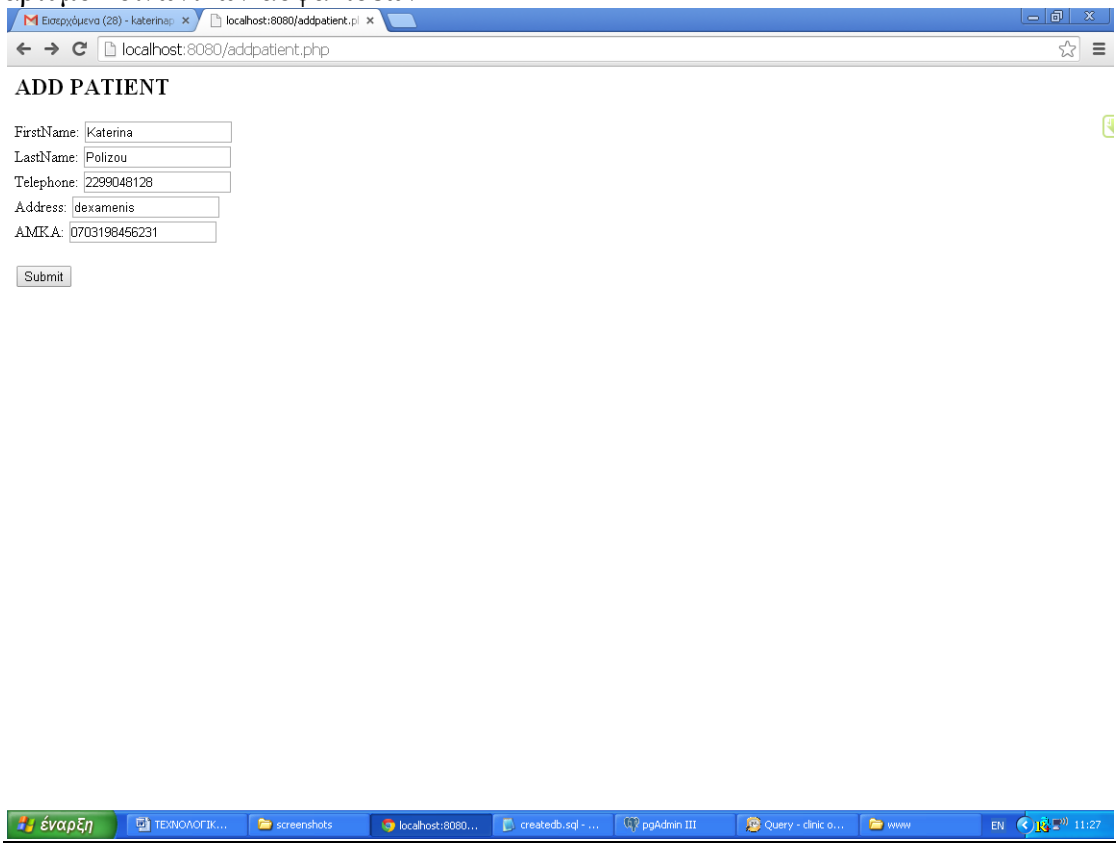
Web Interface

Σύνδεση βάσης με server:

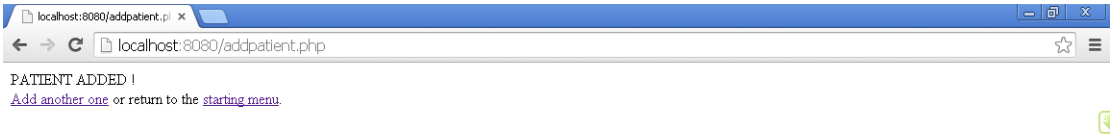


Καταχώρηση Ασθενή:

Καταχωρούμε τα στοιχεία κάθε ασθενή. Όνομα, επίθετο τηλέφωνο διεύθυνση και αριθμό κοινωνικών ασφαλίσεων



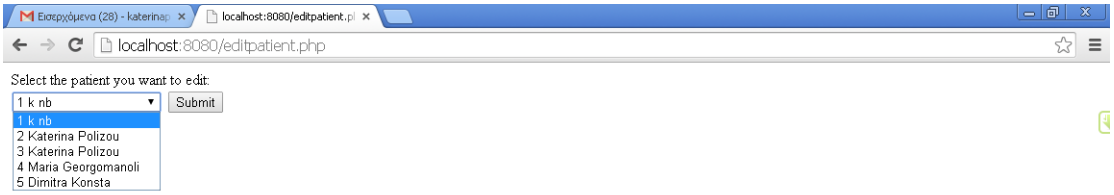
Πατώντας το **Submit** :



Και είτε καταχωρούμε άλλο ασθενή ,είτε επιστρέφουμε στο αρχικό μενού για άλλες καταχωρήσεις.

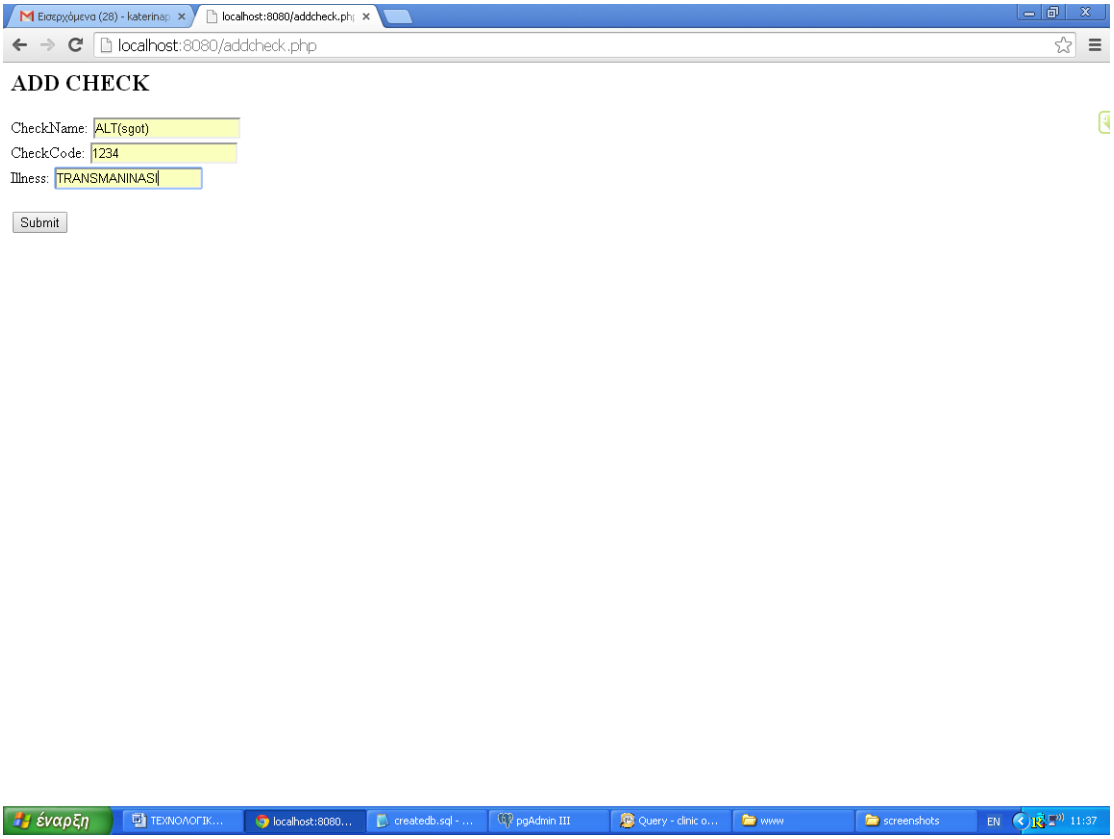
Διόρθωση Ασθενή:

Αν τυχόν θέλουμε να διορθώσουμε κάποιο στοιχείο στην καρτέλα του ασθενούς, ή να προσθέσουμε κάποια από τα ζητούμενα στοιχεία πατάμε την επιλογή Edit Patient του αρχικού μενού και επιλέγουμε όποιον ασθενή από την λίστα θέλουμε.Πατώντας παλι την εντολή Submit, μπαίνουμε στην καρτέλα που επιθυμούμε.



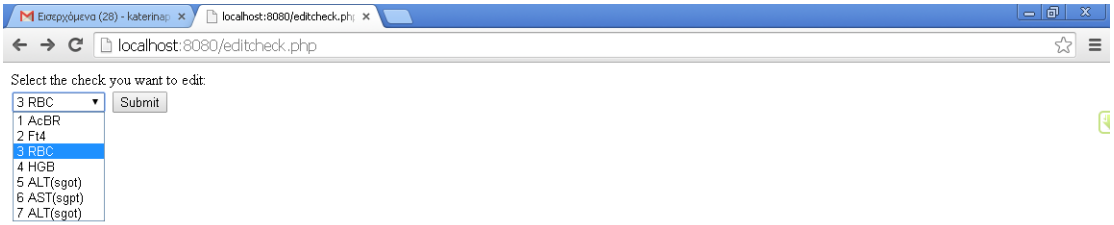
Καταχώρηση εξετάσεων:

Με την επιλογή Add Check γίνεται καταχώρηση εξέτασης.Ζητείται όνομα, κωδικός και η αρρώστια για την οποία γίνεται η εξέταση.



Πάλι η εντολή Submit λειτουργεί ως το OK της καταχώρησης.

Δίορθωση Εξέτασης :Επιλέγοντας στο αρχικό μενού το Edit Check διαλέγουμε ποία εξέταση θέλουμε να να τροποποιήσουμε .



Και πατάμε την επιλογή Submit.Έτσι ,επιστρέφουμε στην καρτέλα της εξέτασης και κάνουμε όποιες αλλαγές η τροποποιήσεις επιθυμούμε.

Καταχώρηση και Διόρθωση Αντιδραστηρίου :

Με τον ίδιο τρόπο ακριβώς γίνεται και η προσθήκη Αντιδραστηρίου (Add Reactor) , όπου ζητούνται όνομα , κωδικός και αποθέματα που υπάρχουν. Επίσης και το ελάχιστο αποθεματικό που πρέπει να έχουμε από το κάθε αντιδραστήριο.Με την επιλογή Submit γίνεται η καταχώρηση και αν τυχόν θέλουμε να διορθώσουμε η προσθέσουμε κάτι ,στη σελίδα του αρχικού μενού επιλέγουμε το Edit Reactor.

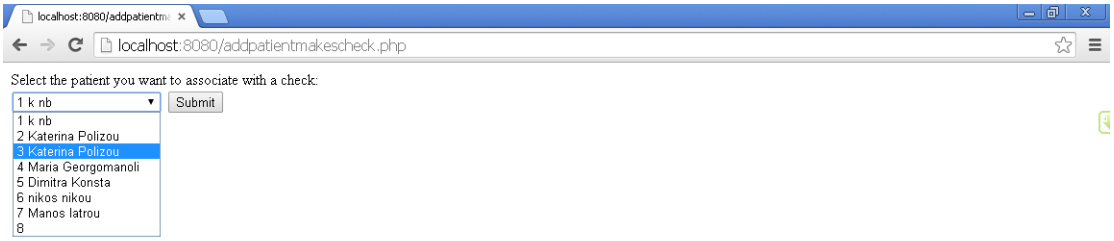
Καταχώρηση και Διόρθωση Παραγγελειών :

Η καταχώρηση γίνεται με το Add Order όπου ζητείται ημερομηνία παραγγελείας και με το Edit Order μπορούμε να επιλέξουμε κάποια συκεκριμένη ημερομηνία ώστε να κάνουμε όποιες αλλαγές θέλουμε.

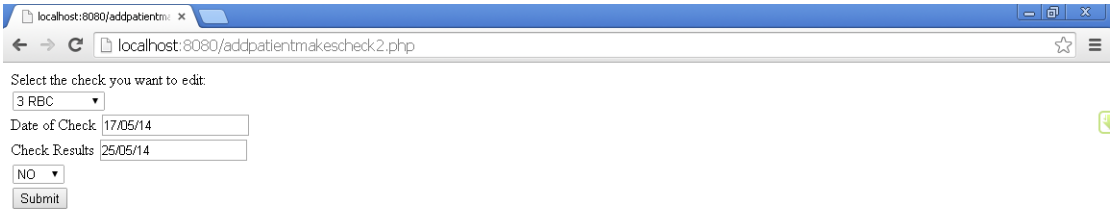
(Υπάρχει επισυναπτόμενος φάκελος με screenshots ώστε να δείτε και σε αυτές τις περιπτώσεις ,τις επιλογές)

Καταχώρηση και Διόρθωση εξέτασης με ασθενή:

Στα παρακάτω screenshots φαίνονται τα εξής ζητούμενα:
Οι πελάτες κάνουν πολλές εξετάσεις και μια εξέταση γίνεται από πολλούς πελάτες. Ένας πελάτης μπορεί να κάνει την ίδια εξέταση σε διαφορετικές ημερομηνίες. Θέλουμε να αποθηκεύομε την ημερομηνία και τα αποτελέσματα αυτής. Επίσης θέλουμε να αποθηκεύουμε αν ο πελάτης έχει πάρει ή όχι τα αποτελέσματα των εξετάσεων.



Επιλέγουμε το όνομα του ασθενή όπου θέλουμε .
Πατώντας το Submit , προχωράμε στο επόμενο screenshot



Στην παραπάνω εικόνα πατώντας το πρώτο βελάκι επιλέγουμε το όνομα της εξέτασης που θέλει ο ε[ιλεγμένος ασθενής να πραγματοποιήσει.
Υστερα στο Date of Check γράφουμε την ημερομηνία που πραγματοποιήθηκε η εξέταση και στο Check Results την ημερομηνία που θα είναι έτοιμες για παράδοση οι εξετάσεις.
Με το δεύτερο βελακι επιλεγουμε αν παραδόθηκαν ή όχι οι εξετάσεις στον ασθενή.

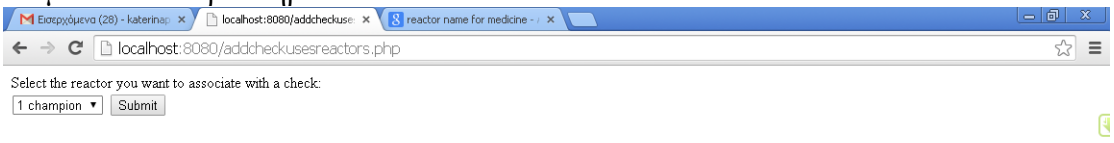
Αν θέλουμε στα παραπάνω δεδομένα να κάνουμε κάποια αλλαγή επιλέγουμε από το αρχικό μενού το Edit a check with a patient και βήμα βήμα μας καθοδηγεί να κάνουμε όποιες τροποποιήσεις είναι αναγκαίες.

(Υπάρχει επισυναπτόμενος φάκελος με screenshots ώστε να δείτε και σε αυτές τις περιπτώσεις ,τις επιλογές)

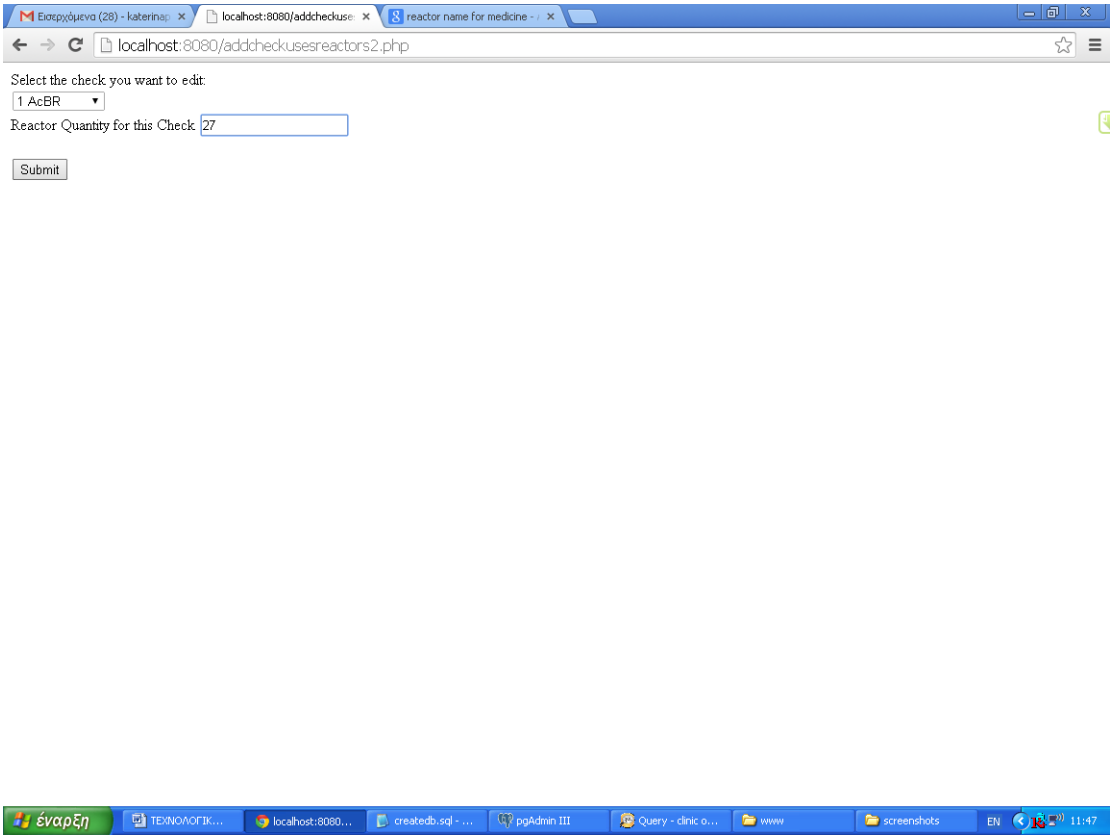
Καταχώρηση και Διόρθωση αντιδραστηρίου με εξέταση:

Μια εξέταση χρησιμοποιεί κάποια αντιδραστήρια. Ένα αντιδραστήριο μπορεί να χρησιμοποιείται από πολλές εξετάσεις. Κάθε εξέταση χρησιμοποιεί διαφορετική ποσότητα αντιδραστηρίου από κάποια άλλη εξέταση που χρησιμοποιεί το ίδιο αντιδραστήριο

Με την επιλογή από το αρχικό μενού Add reactor quantity for a check επιλέγουμε το όνομα του αντιδραστηρίου.



Πατώντας το βελάκι αριστερά εμφανίζονται όλα τα αντιδραστήρια του μικροβιολογικού εργαστηρίου.Επιλέγουμε αυτό που θέλουμε και πατώντας το Submit μας πηγαίνει στην επόμενη εικόνα

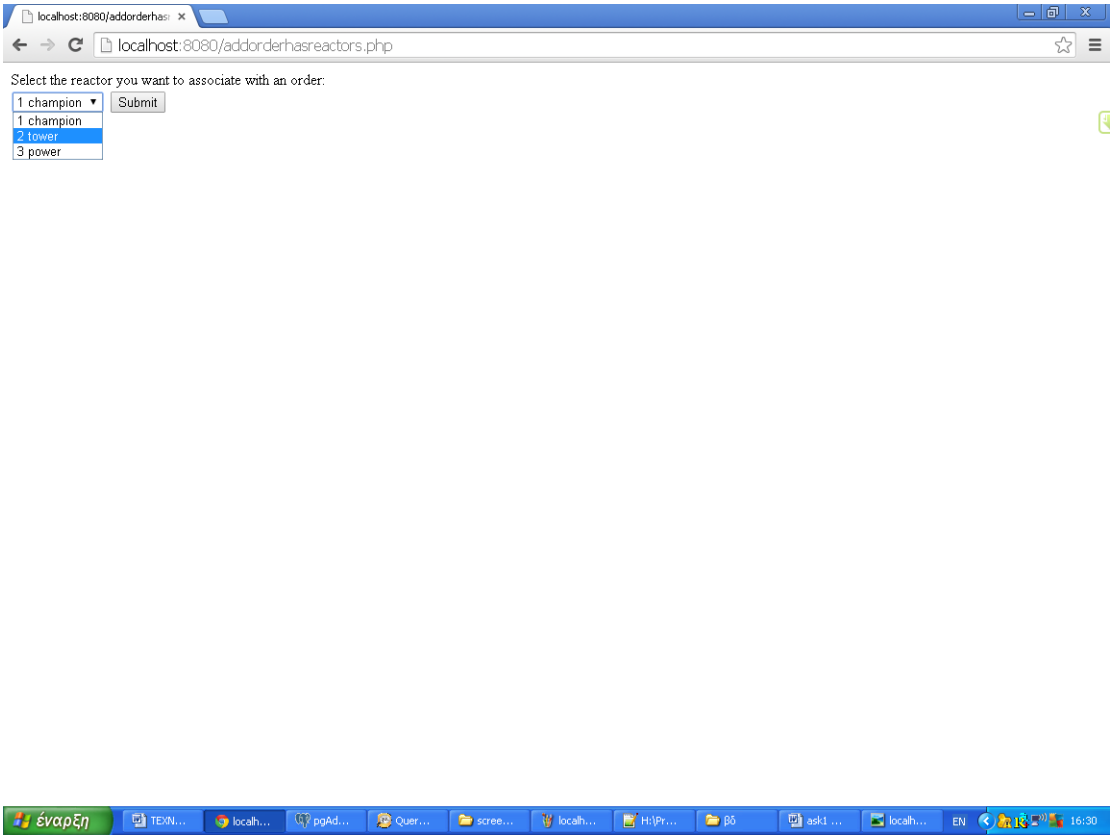


Πρώτα επιλέγουμε την ασθένεια και ύστερα την ποσότητα του αντιδραστηρίου που χρειάζεται η συγκεκριμένη εξέταση.

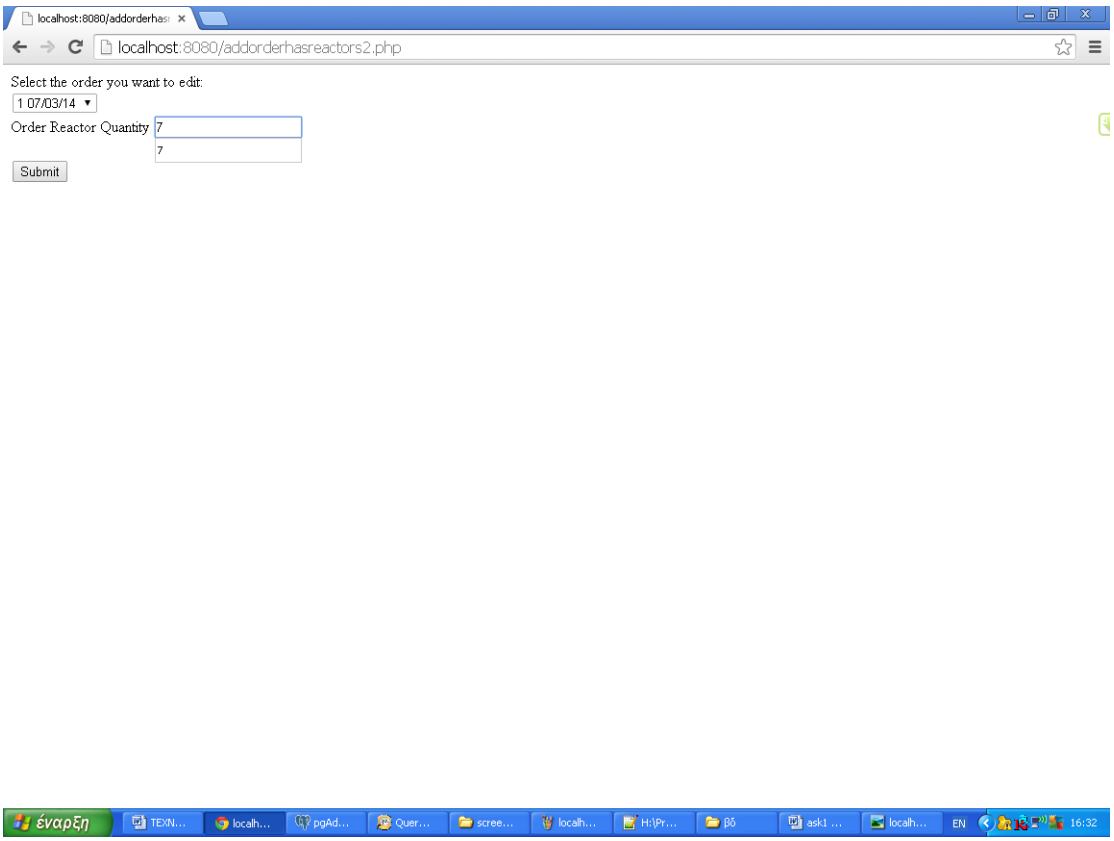
Για διορθώσεις και αλλαγές επιλέγουμε το Edit reactor quantity for a check του αρχικού μενού.

<u>Καταχώρηση και Διόρθωση αντιδραστηρίου με παραγγελία:</u>

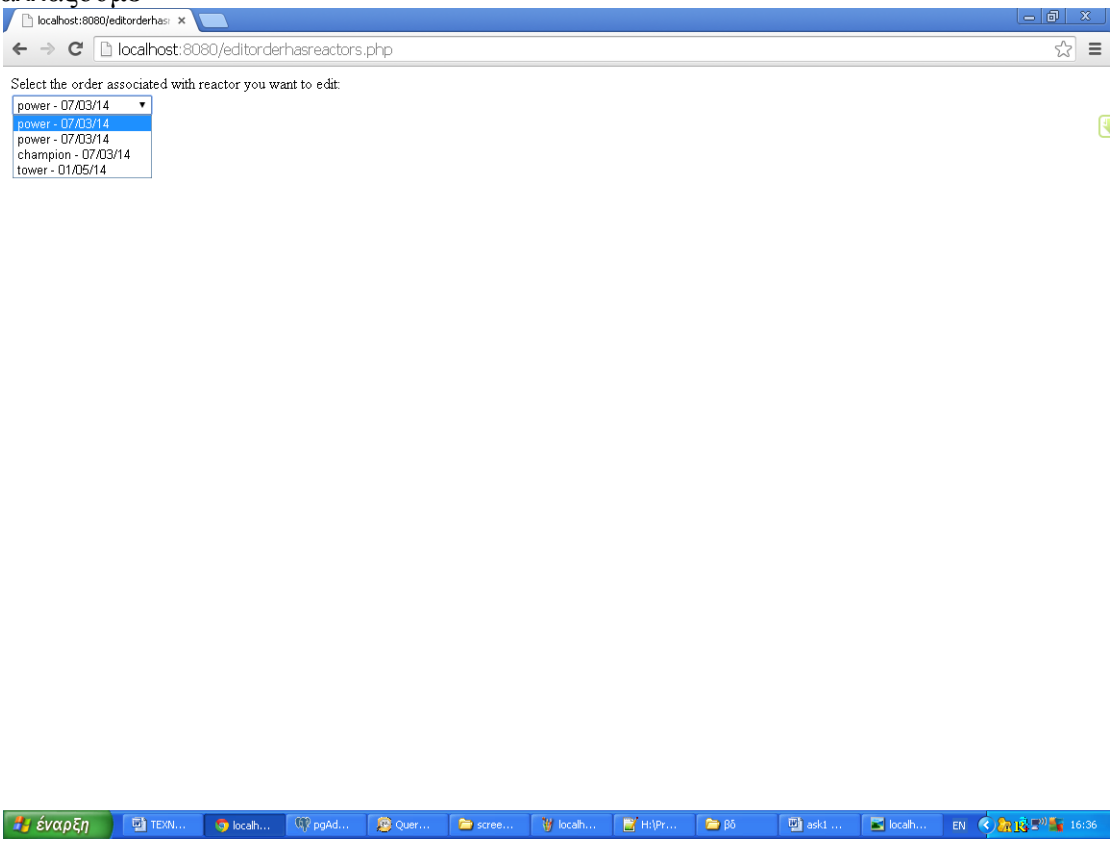
Μια παραγγελία μπορεί να αφορά πολλά αντιδραστήρια. Θέλουμε να αποθηκεύομε την ποσότητα του καθενός
Με την επιλογή Add reactor quantity for an order επιλέγουμε το αντιδραστήριο που θέλουμε

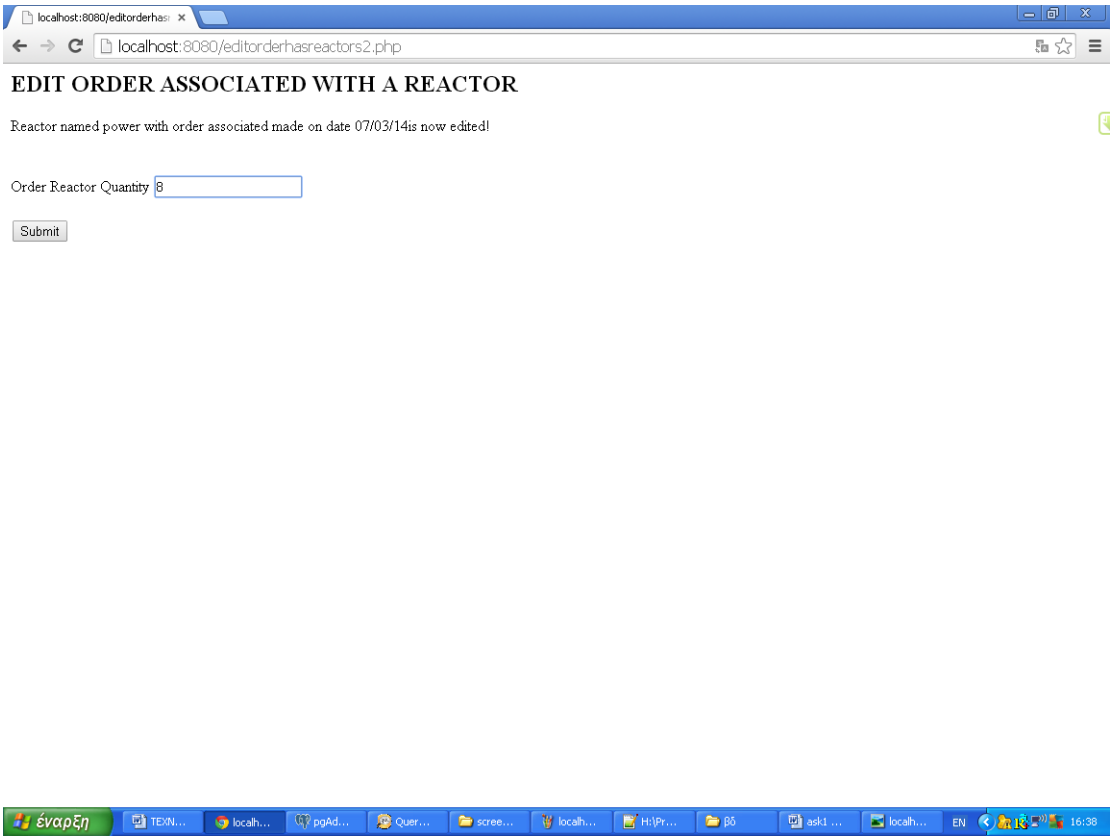


Και πατώντας το Submit επιλέγουμε την ημερομηνία της παραγγελίας και ύστερα γράφουμε την ποσότητα του αντιδραστηρίου.Με το Submit γίνεται η καταχώρηση,



Με το Edit reactor quantity for an order του αρχικού μενού γίνεται η διόρθωση.Επιλέγουμε ποιο από τα ζευγάρια αντιδραστήριο- παραγγελία θέλουμε να αλλάξουμε



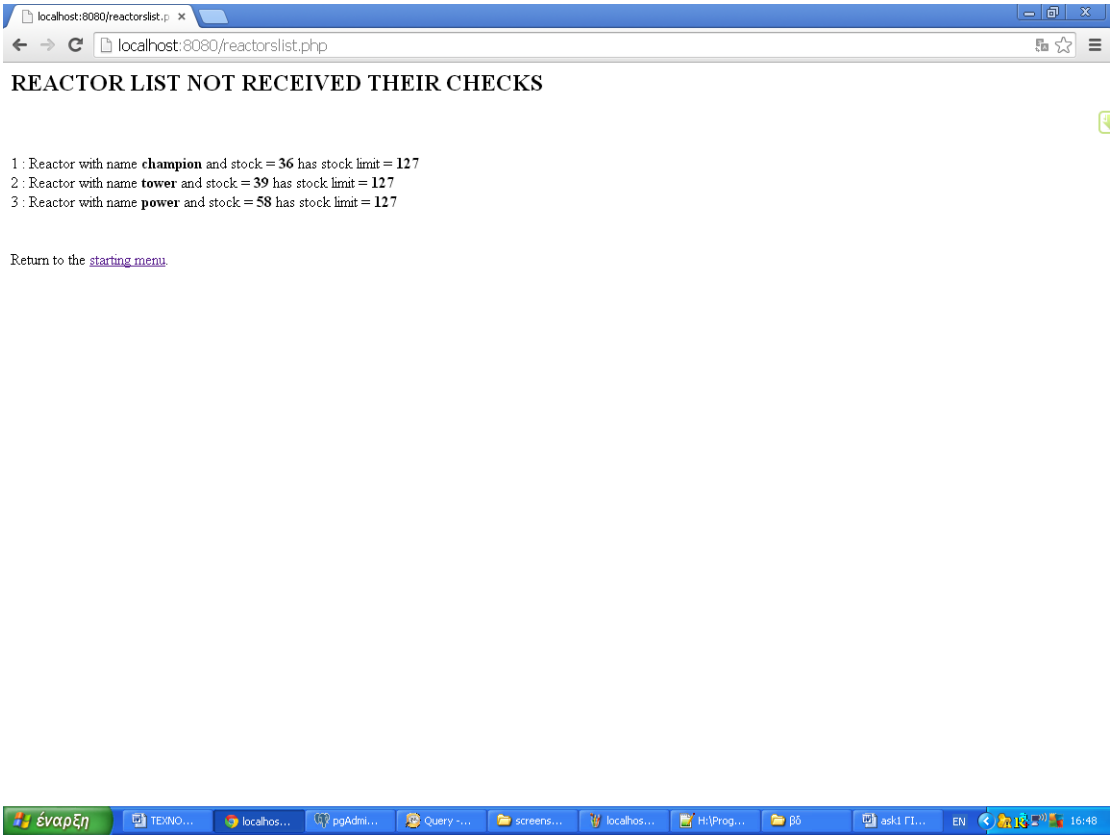


Να σημειωθεί ότι με την ενημέρωση για παραλαβή οποιασδήποτε παραγγελίας γίνεται αυτόματη αύξηση των αποθεμάτων των αντιδραστηρίων.

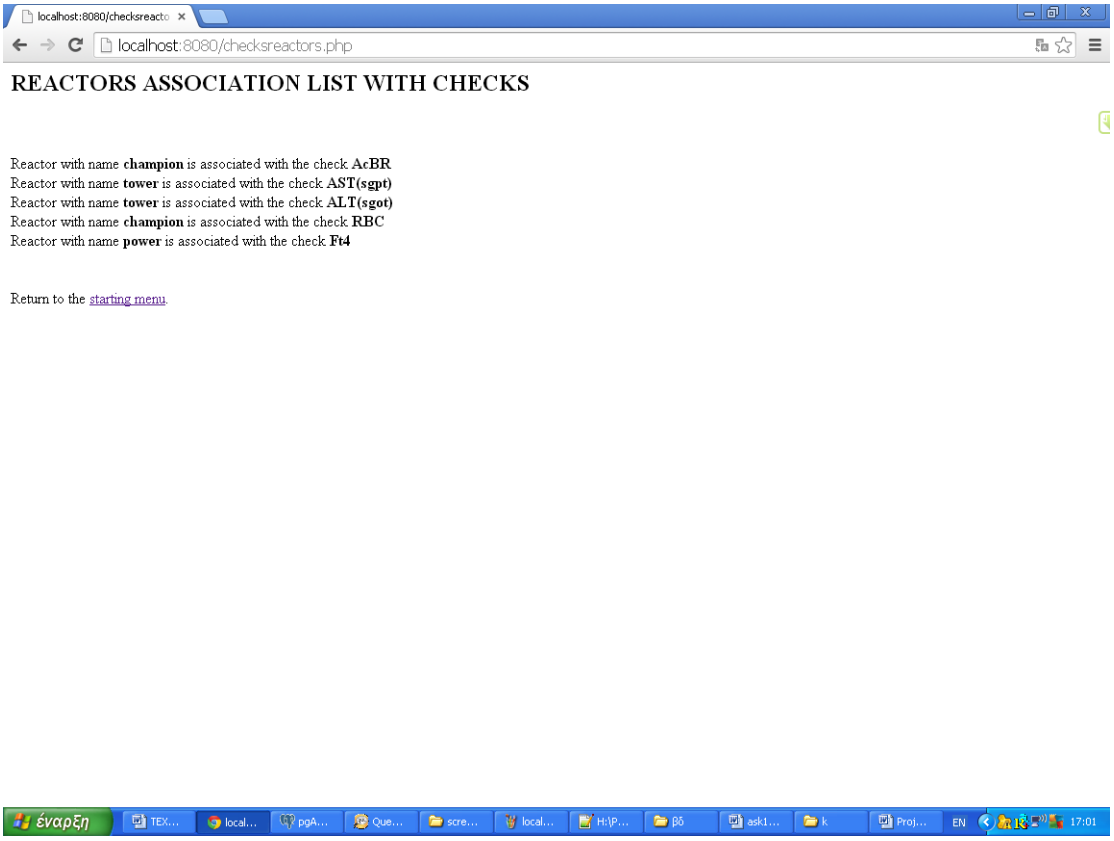
Επιπλέον Δυνατότητες Βάσης :

Με τη βάση αυτή υπάρχει η δυνατότητα να μας εμφανίζονται κάποια συγκεντρωμένα αποτελέσματα.Ενα από αυτά είναι μία λίστα με τα ονόματα των ασθενων όπου δεν έχουν λάβει τα αποτελέσματα των εξετάσεων που πραγματοποίησαν.
Αυτο το βλέπουμε με την επιλογή List of patients not received their check results.

Επίσης μπορεί να εμφανιστεί λίστα με τους αντιδραστήρες ,ποσο stock έχει χρησιμοποιηθεί και ποιο είναι το όριο του stock .Αυτό γίνεται με την επιλογή List of Reactors του αρχικού μενού ,όπου φαίνεται στην παρακάτω εικόνα
Να σημειωθεί ότι κάθε φορά που τα αποθέματα κάποιου αντιδραστηρίου πέφτουν κάτω από τα ελάχιστα όρια εμφανίζεται το μήνυμα αμέσως μετά την εισαγωγή των εξετάσεων που προκάλεσαν αυτήν την μείωση.



Και τέλος υπάρχει η δυνατότητα να εμφανίζονται σε λίστα όλοι οι αντιδραστήρες του μικροβιολογικού εργαστηρίου ,με ποιες εξετάσεις συσχετίζονται,επιλέγοντας το List of Reactors acssociated with checks από το αρχικό μενού .



ΒΙΒΛΙΟΓΡΑΦΙΑ

1. http://en.wikipedia.org/wiki/Database

2. http://www.scribd.com/doc/6562060/Part-1-1

3. http://en.wikipedia.org/wiki/Codd's_12_rules

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

5. Γλώσσα προγραμματισμού PHP:
http://php.net/manual/en/history.php.php

http://el.wikipedia.org/wiki/PHP

6. PostgreSQL database server :
http://en.wikipedia.org/wiki/PostgreSQL

7. Lockhart Th., The PostgreSQL Development Team (1998). *PostgreSQL Tutorial* (release 6.3).Indianapolis

8. http://www.postgresql.org/about/history

9. http://wiki.postgresql.org/wiki/Tuning_Your_PostgreSQL_Server

10. http://phppgadmin.sourceforge.net/doku

11. The Apache HTTP server project :
http://httpd.apache.org/ABOUT_APACHE.html