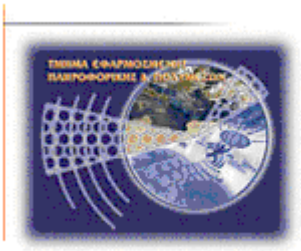
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων



Πτυχιακή εργασία

Τίτλος: *Ικανοποίηση περιορισμών σε χρονικές βάσεις δεδομένων μέσω Διαδικτύου*

Κοσμά Αικατερίνη (Α.Μ. : 2008)
Στιβακτάκη Μαρία (Α.Μ. 2033)

Επιβλέπων καθηγητής : Παπαδάκης Νικόλαος

Ημερομηνία παρουσίασης: 25/11/2010

Ευχαριστίες

Αρχικά θέλουμε να ευχαριστήσουμε τον καθηγητή μας, κ. Νικόλαο Παπαδάκη για την εμπιστοσύνη που μας έδειξε αναθέτοντας μας την συγκεκριμένη πτυχιακή εργασία καθώς και για τις πολύτιμες συμβουλές που μας έδωσε ώστε να την ολοκληρώσουμε.

Τέλος, ένα πολύ μεγάλο ευχαριστώ στις οικογένειες μας που μας στήριξαν καθ' όλη την διάρκεια των σπουδών μας και εξακολουθούν να μας στηρίζουν σε όλα μας τα βήματα.

Abstract

The aim of this thesis is the addressing of the ramification problem in temporal databases and the deal of problems that arising of them. The creation is done using PHP and MySQL.

Chapter 1 gives an indication of the problem. In chapter 2 and 3 we present and we analyze the programming languages used for it. Then, in chapter 4, we present the tools we used to complete the thesis. Chapter 5, shows the code which we create for this purpose. Finally, in chapter 6, we make an example of implementation.

Σύνοψη

Στόχος αυτής της πτυχιακής εργασίας είναι η ικανοποίηση των περιορισμών σε χρονικές βάσεις δεδομένων και η αντιμετώπιση των προβλημάτων που προκύπτουν από αυτούς. Η δημιουργία της γίνεται χρησιμοποιώντας PHP και MySQL .

Στο κεφάλαιο 1 γίνεται η αναφορά του προβλήματος. Στο κεφάλαιο 2 και 3 παρουσιάζονται και αναλύονται οι γλώσσες προγραμματισμού που χρησιμοποιήθηκαν. Στην συνέχεια, κεφάλαιο 4, παρουσιάζονται τα εργαλεία που χρησιμοποιήθηκαν για την ολοκλήρωση της πτυχιακής εργασίας ενώ στο κεφάλαιο 5 προβάλλεται ο κώδικας που δημιουργήθηκε για τον σκοπό αυτό. Τέλος, κεφάλαιο 6, πραγματοποιείται ένα παράδειγμα υλοποίησης.

Πίνακας Περιεχομένων

Πίνακας Εικόνων

ΚΕΦΑΛΑΙΟ 1ο

ΚΕΦΑΛΑΙΟ 2ο

ΚΕΦΑΛΑΙΟ 3ο

ΚΕΦΑΛΑΙΟ 4ο

ΚΕΦΑΛΑΙΟ 6ο

Λίστα Πινάκων

ΚΕΦΑΛΑΙΟ 1ο

Ένα εργαλείο για την αντιμετώπιση του προβλήματος διακλάδωσης σε Χρονικές βάσεις Δεδομένων

Σκοπός αυτής της πτυχιακής εργασίας είναι να μελετήσουμε το πρόβλημα διακλάδωσης στον καθορισμό των χρονικών βάσεων δεδομένων.

1. 1 Εισαγωγή

Υπάρχουν πολλοί τομείς στη φύση οι οποίοι είναι δυναμικοί. Ο συλλογισμός σχετικά με την δράση ασχολείται με τον καθορισμό της φύσης του κόσμου μετά την εκτέλεση μιας δράσης σε μια γνωστή κατάσταση του κόσμου. Παράδειγμα τέτοιας εφαρμογής είναι η γνωστική ρομποτική.

Έστω το πρόβλημα διακλάδωσης σε μια χρονική ρύθμιση. Σε αυτό το πλαίσιο, οι δράσεις μπορούν να έχουν επιπτώσεις οι οποίες μπορούν να αρχίσουν σε χρόνο διαφορετικό από το επόμενο χρονικό σημείο και τα αποτελέσματα μπορεί να κρατάνε μόνο για ορισμένο χρονικό διάστημα. Για παράδειγμα, ένα συγκεκριμένο παράπτωμα μπορεί να έχει ως αποτέλεσμα την αναστολή ενός εργαζομένου για ορισμένο χρονικό διάστημα, μετά το οποίο ο εργαζόμενος είναι και πάλι ενεργός και μπορεί να πάρει ένα μισθό.

Η εγγύηση της συνοχής των δεδομένων που αποθηκεύονται σε μια βάση δεδομένων είναι ένα πολύ σημαντικό και δύσκολο πρόβλημα. Η συνοχή των δεδομένων καθορίζεται από την ικανοποίηση των περιορισμών ακεραιότητας σε διαφορετικές καταστάσεις βάσεων δεδομένων (situations). Μια κατάσταση της βάσης δεδομένων θεωρείται ότι είναι έγκυρη (συνεπής) μόνο εάν πληρούνται όλοι οι περιορισμοί ακεραιότητας. Όταν μια συναλλαγή εκτελείται, το πλαίσιο της βάσης δεδομένων είναι τροποποιημένο. Στη νέα κατάσταση (που περιλαμβάνει τις άμεσες επιπτώσεις των συναλλαγών) η βάση δεδομένων μπορεί να είναι ασυνεπής επειδή ορισμένοι από τους περιορισμούς ακεραιότητας δεν είναι ικανοποιημένοι. Ως εκ τούτου, είναι αναγκαίο να παραχθούν επιπλέον δράσεις (έμμεσες) για να ικανοποιούνται οι περιορισμοί ακεραιότητας.

Στο πλαίσιο αυτό, το πρόβλημα της διακλάδωσης αφορά τις έμμεσες επιπτώσεις των ενεργειών με την παρουσία περιορισμών. Το πρόβλημα της διακλάδωσης έχει μεγάλη σημασία στο σύστημα της βάσης δεδομένων. Οι χρήστες και οι σχεδιαστές των βάσεων δεδομένων δεν μπορούν να γνωρίζουν ακριβώς όλες τις έμμεσες επιπτώσεις των συναλλαγών τους. Έτσι, οι χρήστες μπορούν να εκτελέσουν μια πράξη που θα μπορούσε να έχει ως αποτέλεσμα την παραβίαση των περιορισμών ακεραιότητας. Η πιο προφανής λύση είναι να προσδιοριστούν χειροκίνητα όλα τα έμμεσα αποτελέσματα. Το πρόβλημα που εμφανίζεται για τη θέσπιση αυτής της λύσης είναι ότι σε μια μεγάλη βάση δεδομένων με μεγάλο αριθμό περιορισμών και συναλλαγών, οι έμμεσες επιπτώσεις, που παράγονται από την αξιολόγηση των συναλλαγών, μπορεί να είναι πάρα πολλές για να υπολογισθούν χειροκίνητα. Επίσης, γνωστοποιείται ότι η ίδια ακολουθία συναλλαγών ενδέχεται να έχει διαφορετικές έμμεσες επιπτώσεις, εάν είναι διαφορετικό το πλαίσιο της βάσης δεδομένων στην έναρξη της εκτέλεσης.

Ως συνέπεια των παραπάνω σκέψεων, θα ήταν πολύ επιθυμητό ένα εργαλείο για την υποστήριξη της διαδικασίας του προσδιορισμού των επιπτώσεων των ενεργειών.

Έτσι, ο σχεδιαστής μπορεί να ανακαλύψει λανθασμένες προδιαγραφές που είναι πολύ δύσκολο ή αδύνατο να ανακαλυφθούν χειροκίνητα. Ένα τέτοιο εργαλείο επιτρέπει στο σχεδιαστή να ανακαλύψει τα σφάλματα στο σχεδιασμό της βάσης δεδομένων και να επανασχεδιάσει τη βάση δεδομένων αν είναι απαραίτητο. Επίσης, αυτό το εργαλείο επιτρέπει τον αυτόματο προσδιορισμό των έμμεσων επιπτώσεων αφού πραγματοποιηθεί μία συναλλαγή, χωρίς να απαιτεί από τους χρήστες να την καθορίσουν.

1.2 Ένα κίνητρο για παράδειγμα

Έστω ότι ένας δημόσιος υπάλληλος διαπράττει ένα παράπτωμα. Τότε για τους επόμενους πέντε μήνες θεωρείται «παράνομος», εκτός κι αν «λάβει συγχώρηση». Όταν ένας δημόσιος υπάλληλος είναι «παράνομος», τότε πρέπει να ανασταλεί και να μην μπορεί να πάρει προαγωγή για το σύνολο του χρονικού διαστήματος κατά το οποίο αυτός/ή θεωρείται «παράνομος/η». Επίσης, όταν

ένας δημόσιος υπάλληλος έχει ανασταλεί, δεν μπορεί να λάβει αποζημίωση μέχρι το τέλος της περιόδου αναστολής. Κάθε δημόσιος υπάλληλος βαθμολογείται για το έργο του. Σε περίπτωση που λάβει ένα κακό βαθμό, τότε θεωρείται κακός υπάλληλος. Σε περίπτωση που λάβει ένα καλό βαθμό, τότε θεωρείται καλός υπάλληλος και μπορεί να πάρει ένα μπόνους αν δεν έχει ανασταλεί.

Μπορούμε να προσδιορίσουμε τέσσερις δράσεις:

- misdemeanor (πλημμέλημα)
- take pardon (να λάβει συγχώρηση)
- good grade (καλό βαθμό) και
- bad grade (κακό βαθμό)

και πέντε fluents

- good employee (καλός υπάλληλος)
- illegal (παράνομος)
- take salary (να λάβει μισθό)
- take bonus (λαμβάνει μπόνους) και
- suspended (έχει ανασταλεί).

Οι άμεσες επιπτώσεις από τις τέσσερις δράσεις εκφράζονται σε προτασιακή μορφή από τους ακόλουθους κανόνες:

Occur (misdemeanor (p), t)	→	illegal (p, t1) ^ t1 ∈ [t, t+5]	(1)
Occur (take_pardon (p), t)	→	illegal (p, t1) ^ t1 ∈ [t, ∞]	(2)
Occur (bad_grade (p), t)	→	good_employee (p, t1) ^ t1 ∈ [t, ∞]	(3)
Occur (good_grade (p), t)	→	good_employee (p, t1) ^ t1 ∈ [t, ∞]	(4)

όπου t είναι μία χρονική μεταβλητή και το κατηγόρημα occur(misdemeanor(p), t) υποδηλώνει ότι η ενέργεια misdemeanor(p), εκτελείται στο χρόνο t. Επίσης υπάρχουν οι ακόλουθοι περιορισμοί ακεραιότητας που οδηγούν σε έμμεσες επιδράσεις από τις τέσσερις ενέργειες.

Illegal (p, t1)	→	suspended (p, t1)	(5)
Suspended (p, t1)	→	¬take_salary(p, t1)	(6)
¬Suspended(p, t) ^ good_employee (p, t)	→	take_bonus (p, t)	(7)
¬good_employee (p, t1)	→	¬take_bonus (p, t1)	(8)
¬Suspended (p, t1)	→	take_salary(p, t1)	(9)

1.3 Τεχνικά Προκαταρκτικά

1.3.1 Επεκτάσεις για την κατάσταση λογισμού

Η λύση στο πρόβλημα διακλάδωσης βασίζεται στην κατάσταση του διαφορικού λογισμού. Ωστόσο, είναι απαραίτητο να επεκταθεί η κατάσταση λογισμού για να κυριεύσει τα χρονικά φαινόμενα.

• Για κάθε fluent f, ένα επιχείρημα L προστίθεται. L είναι μια λίστα των χρονικών διαστημάτων [α, b], α <β. Το fluent f είναι αληθές στα χρονικά διαστήματα που περιέχονται στη λίστα L.

• Ορίζονται λειτουργίες start(α) και στο end(α), όπου a είναι μια ενέργεια. Η προηγούμενη συνάρτηση επιστρέφει τη χρονική στιγμή κατά την οποία η δράση ενός ξεκινά, ενώ η τελευταία επιστρέφει τη χρονική στιγμή κατά την οποία τελειώνει.

• Οι ενέργειες (υποτίθεται ότι είναι στιγμιαίες) ταξινομούνται ως εξής: α1 <α2 <· · · <αn, όταν start(α1) <start (α2) <· · · <start (αn).

• Το κατηγόρημα occur (α, t) σημαίνει ότι η δράση εκτελείται σε χρονική στιγμή t.

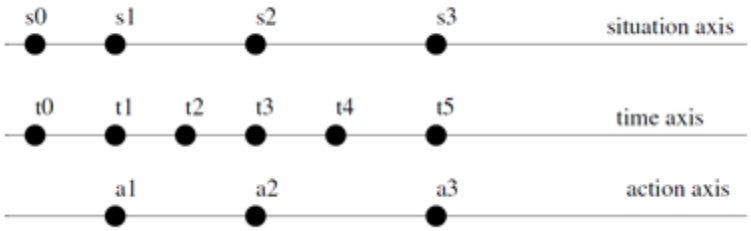
• Ορίζονται λειτουργίες start(S) και end(S), όπου S είναι μια κατάσταση. Η προηγούμενη συνάρτηση επιστρέφει τη χρονική στιγμή κατά την οποία η κατάσταση S ξεκινά, ενώ η δεύτερη επιστρέφει τη χρονική στιγμή κατά την οποία τελειώνει.

• Η λειτουργία FluentHold (S, t) επιστρέφει το σύνολο όλων των fluents που ισχύουν στη χρονική στιγμή t.

• Ορίζεται ως νόμιμη (συνεπής), μια κατάσταση κατά την οποία όλοι οι περιορισμοί ακεραιότητας ικανοποιούνται.

1.3.2 Ανταπόκριση μεταξύ του χρόνου, των δράσεων και των καταστάσεων

Οι προηγούμενες προσεγγίσεις για την επίλυση του προβλήματος των διακλαδώσεων είναι ανεπαρκείς στην περίπτωση των χρονικών βάσεων δεδομένων. Οι δυσκολίες ξεπερνιούνται με την time — actions — situations αντιστοιχία που εμφανίζεται στο σχήμα 1 . Υπάρχουν τρεις παράλληλοι άξονες: ο άξονας των θέσεων, ο άξονας του χρόνου και ο άξονας των ενεργειών. Όταν πραγματοποιείται μια ενέργεια ,η βάση αλλάζει σε νέα θέση.



***Εικ 1.1** : Η αλληλογραφία μεταξύ των Time-Actions-Situations.*

1.3.3 Στατικοί και δυναμικοί κανόνες

Η υλοποίηση στηρίζεται στις ιδέες του McCain και Turner3 που προτείνουν τη χρήση στατικών κανόνων προκειμένου να συλληφθούν οι έμμεσες επιπτώσεις των ενεργειών και των δυναμικών κανόνων για να αναπαρασταθούν τα άμεσα αποτελέσματα των ενεργειών. Για κάθε ενέργεια A υπάρχει ένας δυναμικός κανόνας του τύπου $A \rightarrow \Lambda$ Fi(L'i) όπου κάθε Fi(L'i) είναι fi(L'i) ή $\neg$fi(L'i) για ένα fluent f. Οι παραπάνω κανόνες περιγράφουν τα άμεσα αποτελέσματα μιας ενέργειας. Επιπλέον, για κάθε fluent f καθορίζονται δύο κανόνες, $G(L) \rightarrow f(L)$ και $B(L) \rightarrow \neg f(L)$. Ο G(L) είναι ένας fluent τύπος ο οποίος όταν είναι αληθής (στη λίστα L), προκαλεί το fluent f να γίνει αληθής κατά τα χρονικά διαστήματα που περιέχονται στη λίστα L (αντίστοιχα για τον B(L)). Οι κανόνες αυτοί ενσωματώνουν τις έμμεσες επιπτώσεις μιας ενέργειας. Οι προηγούμενοι κανόνες είναι δυναμικοί επειδή αξιολογούνται μετά την εκτέλεση μιας ενέργειας, ενώ οι επόμενοι είναι στατικοί γιατί αξιολογούνται σε κάθε χρονική στιγμή κατά την οποία το αντίστοιχο fluent είναι ψευδές.

1.3.4 Βασικές παραδοχές

Υποθέτουμε ότι δίνονται οι δυναμικοί κανόνες, που αντιπροσωπεύουν τα άμεσα αποτελέσματα των ενεργειών. Παρακάτω παρουσιάζεται ένας αλγόριθμος ο οποίος, παράγει ένα σύνολο στατικών κανόνων που αντιπροσωπεύουν τα στατικά αποτελέσματα των ενεργειών και ικανοποιούν την ακόλουθη ιδιότητα: Όταν ένας στατικός κανόνας είναι εκτελέσιμος σε μια κατάσταση, τότε η κατάσταση αυτή είναι παράνομη (δηλαδή, τουλάχιστον ένας περιορισμός ακεραιότητας παραβιάζεται).

Αλγόριθμος 1

Αλγόριθμος 1

- Κάθε περιορισμός ακεραιότητας μεταμορφώνεται στην CNF μορφή του. Τώρα κάθε περιορισμός ακεραιότητας έχει τη μορφή $C_1 \wedge C_2 \wedge C_3 \cdots \wedge C_n$ *όπου κάθε* C_i *είναι η* διάζευξη όλων των fluents.
- Θέστε $R = \{False \rightarrow f, False \rightarrow \neg f : \text{για κάθε fluent } f\}$
- Για κάθε i από 1 μέχρι n κάνε: έστω ότι $C_i = f_1 \vee \cdots \vee f_m$
  Για κάθε j από 1 μέχρι n κάνε
  Για κάθε k από 1 μέχρι ν και k≠j κάνε :
  if $(f_j, f_k) \in I$ then $R = R \cup (\neg f_j \text{ causes } f_k \text{ if } \wedge \neg f_l), l \neq j, k.$
- Για κάθε fluent fk οι κανόνες στο R έχουν την ακόλουθη μορφή :
  $\wedge f_i \text{ causes } f_k \text{ if } \Phi, \quad \wedge f'_i \text{ causes } \neg f_k \text{ if } \Phi'.$
  Οι στατικοί κανόνες αλλάζουν από $G \rightarrow f_k$, $K \rightarrow \neg f_k$

σε $(G \vee (\wedge f_i \wedge \Phi)) \rightarrow f_k$, $(K \vee (\wedge f'_i \wedge \Phi')) \rightarrow \neg f_k$.

- Τη χρονική στιγμή t, για κάθε στατικό κανόνα $G^s(t, t1) \rightarrow f$ do
  (a) let $G^s = G_1 \vee \cdots \vee G_{s\_n}$
  (b) For each j from 1 to s_n do
  (i) let $G_j = f_1([..]) \wedge \cdots \wedge f_n([..])$

A. για κάθε fluent $f_i(L)$ (s.t $f_i(L) \in G_j$) πάρε το πρώτο στοιχείο $[t', t'']$ της λίστας L.
B. if $t' > t$ then G is false and terminates.
C. else $t_i = t'' - t$ and remove $[t', t'']$ from L.

(ii) let $t_{min} = min(t_1, \ldots, t_{sn})$
(iii) replace G_i with $G_i(t, t + t_{min})$

Αλγόριθμος 2

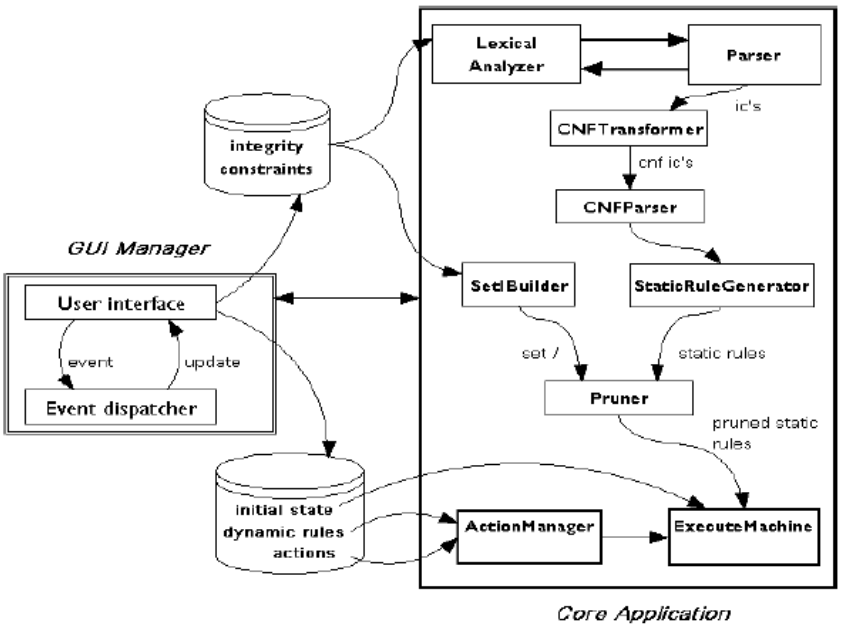
- Μετά την εκτέλεση μιας ενέργειας αξιολογείται ο δυναμικός κανόνας αναφερόμενος σ' αυτήν την ενέργεια.
- Αξιολογούνται όλοι οι κανόνες μέχρι να μη συμβαίνει καμία αλλαγή.

Μια άλλη βασική υπόθεση είναι ότι το σύνολο των περιορισμών ακεραιότητας στον τομέα είναι ικανοποιήσιμο και υπάρχουν δύο είδη περιορισμών ακεραιότητας.
(a) $G_f \rightarrow K_f$ και (b) $G_f \equiv K_f$ όπου G_f και K_f είναι fluent προτάσεις.

1.4 Η αρχιτεκτονική του εργαλείου (of the tool).

Στην ενότητα αυτή, παρουσιάζεται η αρχιτεκτονική και η λειτουργικότητα του εργαλείου που χρησιμοποιήθηκε. Το Σχήμα 2 δείχνει την αρχιτεκτονική του, ως συστατικό διάγραμμα με συνιστώσες. Το εργαλείο αποτελείται από ένα σύνολο στοιχείων που επικοινωνούν μεταξύ τους με σαφώς καθορισμένους ρόλους.

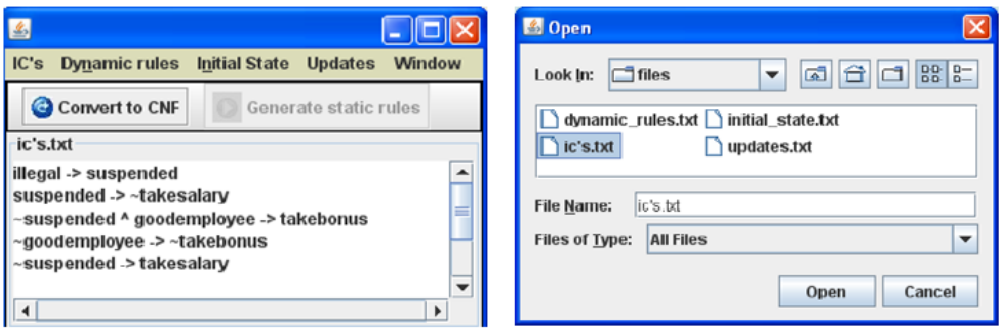
Η αλληλεπίδραση μεταξύ του χρήστη και του εργαλείου επιτυγχάνεται αποτελεσματικά μέσω μιας γραφικής διεπαφής χρήστη. Αρχικά, ο χρήστης πρέπει να παρέχει τους περιορισμούς ακεραιότητας, την αρχική κατάσταση, τους δυναμικούς κανόνες και τις ενέργειες, χρησιμοποιώντας το μενού της διεπαφής χρήστη, με τη μορφή αρχείων.



***Εικ. 1.2**: Η αρχιτεκτονική του εργαλείου.*

Τα Lexer and Parser ευθύνονται για την εκτέλεση της λεξιλογικής και συντακτικής ανάλυσης των περιορισμών ακεραιότητας, αντίστοιχα. Κάθε περιορισμός ακεραιότητας στη συνέχεια μετατρέπεται σε CNF μορφή από τον CNFTransformer και θα αναλυθεί από τον CNFParser. Ο StaticRuleGenerator δημιουργεί τους στατικούς κανόνες από τους (CNF) περιορισμούς ακεραιότητας, καθώς ο SetIBuilder κατασκευάζει το set I , το οποίο χρησιμοποιείται για την ανακάλυψη εξαρτήσεων μεταξύ των fluents όπως αναφέραμε στο Τμήμα 2.

Ο Pruner «κλαδεύει» μερικούς από τους παραγόμενους στατικούς κανόνες και παρέχει αυτούς τους κανόνες στο ExecuteMachine.



***Εικ. 1.3**: Φόρτωση ακέραιων σταθερών*

Στη συνέχεια, ο ActionManager επεξεργάζεται τους δυναμικούς κανόνες και τις ενημερώσεις και ταιριάζει κάθε δυναμικό κανόνα με μια ενημέρωση. Τέλος, το ExecuteMachine κατεργάζεται την αρχική κατάσταση, εκτελεί τις ενέργειες που έχουν δοθεί σε αυτή την κατάσταση και αξιολογεί όλους τους στατικούς κανόνες μέχρι να μην επέρχονται μεταβολές, δημιουργώντας μια σειρά από προσωρινές καταστάσεις. Τέλος αναλύεται η περιγραφή των συνιστωσών του εργαλείου. Τα σύμβολα ~, ↔ που απεικονίζονται στα σχήματα του εργαλείου, υποδηλώνουν την άρνηση (¬) και την ισοτιμία (≡) αντίστοιχα.

1.4.1 GUI Manager:

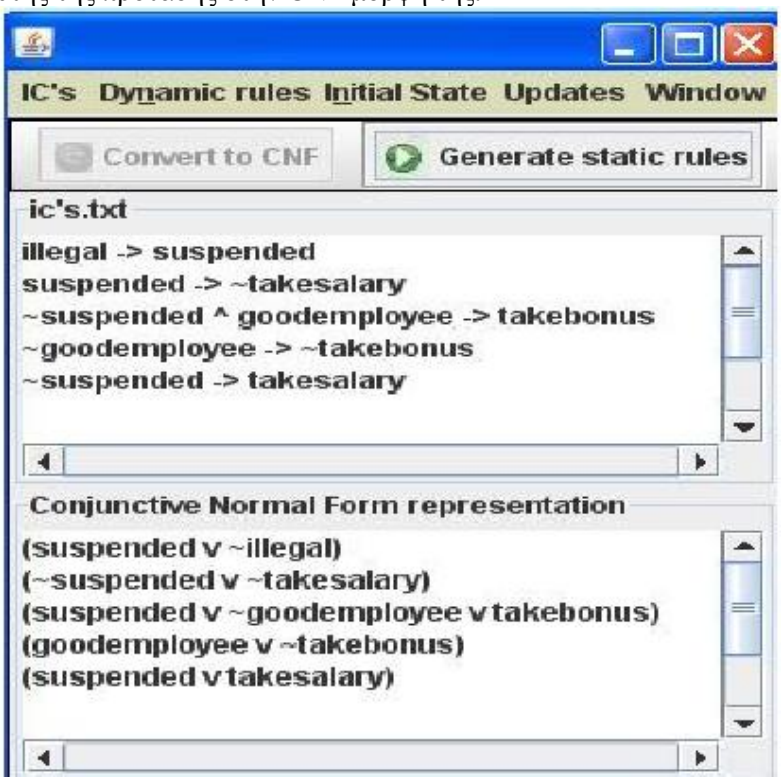
Αυτή η συνιστώσα είναι υπεύθυνη για τη διαχείριση του περιβάλλοντος εργασίας του χρήστη. Οι χρήστες μπορούν να παρέχουν εύκολα τους περιορισμούς ακεραιότητας, τους δυναμικούς κανόνες, την αρχική κατάσταση και τις ενημερώσεις, με τη χρήση των αντίστοιχων μενού από τη γραμμή μενού. Οι κοινές επιλογές, όπως "New", "Open", "Save" που παρέχονται μέσω του μενού με τις ετικέτες IC's , Dynamic Rules, Initial State, Updates από τη γραμμή μενού, επιτρέπουν στους χρήστες να δημιουργήσουν, να φορτώσουν ή να σώσουν τα αρχεία που αναφέρονται στους περιορισμούς ακεραιότητας, τους δυναμικούς κανόνες, την αρχική κατάσταση ή τις ενημερώσεις, αντίστοιχα. Όλα τα γεγονότα που προκλήθηκαν από τον χρήστη, διεκπεραιώνονται από τον αποστολέα Event (Event dispatcher). Για παράδειγμα, όταν ο χρήστης επιλέγει μια εντολή από τη γραμμή μενού, ή πατάει ένα κουμπί, συμβαίνει ένα γεγονός και ενημερώνεται η διεπαφή χρήστη.

1.4.2 Lexer :

Ο Αναλυτής Λεξικού διαβάζει τους περιορισμούς ακεραιότητας, μετατρέπει τις ακολουθίες των χαρακτήρων σε ειδικότερες ενδείξεις (tokens) και κρατάει κομμάτι από την τρέχουσα ένδειξη που έχει διαβαστεί. Οι περιορισμοί ακεραιότητας παρέχονται από το χρήστη μέσω της χρήσης της παρεχόμενης διασύνδεσης, που φαίνεται στο σχήμα 3. Όπως αναφέρεται στο τμήμα 2, έστω ότι υπάρχουν δύο είδη των περιορισμών ακεραιότητας: α) $G_f \rightarrow K_f$ και β) $G_f \equiv K_f$, όπου G_f και K_f είναι fluent προτάσεις. Οι περιορισμοί ακεραιότητας που δεν είναι σύμφωνοι με την παραπάνω μορφή απορρίπτονται. Η λειτουργία του Lexer είναι να παρέχει την επόμενη ένδειξη στον Parser, όποτε ζητηθεί.

1.4.3 Parser.

Ο ρόλος του Parser είναι να αναλύσει τους περιορισμούς ακεραιότητας, που δεν είναι στην CNF μορφή τους και να εξετάσει το είδος του κάθε περιορισμού ακεραιότητας. Ο Parser παρέχει τους περιορισμούς ακεραιότητας στον CNFTransformer, ο οποίος με τη σειρά του ευθύνεται για τον μετασχηματισμού αυτής της πρότασης στην CNF μορφή της.



***Εικ. 1.4**: Μετασχηματισμός των σταθερών ακεραιότητας στην CNF μορφή τους*

1.4.4 CNFTransformer.

Ο CNFTransformer ανακτά όλους τους περιορισμούς ακεραιότητας από τον Parser, οι οποίοι μπορεί να είναι ισοδύναμες ή έμμεσες προτάσεις και μεταμορφώνει τις προτάσεις στην CNF μορφή τους. Στο Σχήμα 4, παρουσιάζεται ένα παράδειγμα του μετασχηματισμού ενός συνόλου περιορισμών ακεραιότητας που παρέχονται από το χρήστη στη CNF μορφή τους.

1.4.5 CNFParser.

Ο CNFParser έχει το ρόλο της ανάλυσης των CNF περιορισμών ακεραιότητας, που παρέχονται από τον CNFTransformer. Κάθε περιορισμός ακεραιότητας έχει τώρα τη μορφή $C_1 \wedge C_2 \wedge C_3 \wedge \ldots \wedge C_n$ όπου κάθε C_i αποτελεί ένα διαχωρισμό των fluents (σε κατηγορήματα και σε συναρτήσεις). Ας υποθέσουμε ότι $C_i = f_1 (L_1) \vee f_2 (L_2) \vee \ldots \vee f_m (L_m)$. Κάθε fluent f συμβολίζεται ως f (L), όπου L = ([t, t'],...) είναι μια λίστα. Κάθε μέλος της λίστας είναι ένα χρονικό διάστημα [t, t'], πράγμα που σημαίνει ότι η ευχέρεια αληθεύει σε χρονικό διάστημα [t, t']. Αυτή η μονάδα ανάλυσης (Parser), χρησιμοποιείται για την κατασκευή, το διαχωρισμό και τη σύνδεση των fluents.

1.4.6 SetIBuilder.

Ο SetIBuilder όπως περιλαμβάνει το όνομά του, κατασκευάζει μια δυαδική σχέση I. Στόχος της δυαδικής σχέσης I είναι να ενσωματωθούν εξαρτήσεις μεταξύ των fluents και να διασφαλιστεί ότι όταν ένας περιορισμός ακεραιότητας δεν έχει ικανοποιηθεί, τότε υπάρχει τουλάχιστον ένας στατικός κανόνας που είναι εκτελέσιμος, και μετά την εκτέλεση ο περιορισμός ακεραιότητας θα ικανοποιηθεί. Το " set I " είναι κατασκευασμένο, σύμφωνα με τον αλγόριθμο, ο οποίος παρουσιάζεται παρακάτω. Έστω ότι υπάρχουν δύο διαφορετικά είδη περιορισμών ακεραιότητας.

Αλγόριθμος για την κατασκευή του set I:

1)Για το πρώτο είδος των περιορισμών, για κάθε $f \in G_f$ και $f' \in K_f$, ο SetIBuilder προσθέτει το ζεύγος (f, f') στο I.

2)Για το δεύτερο είδος των περιορισμών, για κάθε ζεύγος των fluents (f, f') ,τέτοιο ώστε $f \in G_f$ και $f' \in K_f$, ο SetIBuilder προσθέτει (f, f') και (f', f) στο I.

Όταν παραχθεί το set I , ο Pruner μπορεί να το χρησιμοποιήσει κατάλληλα, προκειμένου να κλαδέψει (prune) τους παραγόμενους στατικούς κανόνες. Στο παράδειγμά, set I = {(illegal, suspended), (suspended, ¯takesalary), (¯suspended, takebonus), (goodemployee, takebonus), (¯goodemployee, ¯takebonus), (¯suspended, takesalary)}.

1.4.7 StaticRuleGenerator.

Η συνιστώσα αυτή είναι υπεύθυνη για την παραγωγή των στατικών κανόνων και επικαλέστηκε έμμεσα. Για κάθε περιορισμό ακεραιότητας ,που ανακτήθηκε από τον CNFParser , οι στατικοί κανόνες, οι οποίοι πρέπει να ικανοποιηθούν, παράγονται σύμφωνα με τον Αλγόριθμο 1. Οι Στατικοί κανόνες ενσωματώνουν τις έμμεσες συνέπειες της εκτέλεσης κάθε δράσης. Οι έμμεσες συνέπειες υφίστανται λόγω της παρουσίας των περιορισμών ακεραιότητας. Είναι λογικό να παράγονται οι στατικοί κανόνες από τους περιορισμούς ακεραιότητας.

1.4.8 Pruner.

Αυτή η συνιστώσα κλαδεύει μερικούς από τους παραγομένους στατικούς κανόνες, σύμφωνα με τη δυαδική σχέση I. Στην Εικόνα 5 (β), φαίνεται ότι μόνο πέντε στατικοί κανόνες παρέμειναν, διότι

έξι στατικοί κανόνες κλαδεύτηκαν. Για παράδειγμα, ο στατικός κανόνας ¬suspended → illegal κλαδεύτηκε δεδομένου ότι (¯suspended, illegal) /∈ I. Το ίδιο συμβαίνει και για τον στατικό κανόνα takesalary →¯suspended, δεδομένου ότι (takesalary, ¯suspended) /∈ I.

(a) Static Rules (b) Pruned Rules

***Εικ. 1.5**: Στατικοί κανόνες και pruned στατικοί κανόνες*

1.4.9 ActionManager.

Η συνιστώσα αυτή είναι υπεύθυνη για την επεξεργασία των δυναμικών κανόνων και των ενημερώσεων. Οι ενημερώσεις καθορίζουν τη σειρά με την οποία εκτελούνται οι δυναμικοί κανόνες. Κάθε ενημέρωση έμμεσα συνδέεται με έναν δυναμικό κανόνα και έχει την ακόλουθη μορφή: $d_i(L) \rightarrow f_y$, όπου d_i είναι το όνομα του δυναμικού κανόνα, L = ([t, t'],...) είναι μια λίστα, που καθορίζει τα διαστήματα του χρόνου στα οποία d_i είναι αληθές και f_y είναι το fluent , το οποίο ενεργοποιείται άμεσα από αυτήν την ενημέρωση. Αυτή η συνιστώσα, εκτελεί το Βήμα 1 του αλγόριθμου 2. Στο παράδειγμα, υποτίθεται ότι ο χρήστης έχει καθορίσει το δυναμικό κανόνα "misdemeanor → illegal" και την αντίστοιχη ενημέρωση misdemeanor([3, 7]) , έτσι ώστε να προκύπτει η ακόλουθη ενημέρωση: misdemeanor([3, 7]) → illegal

1.4.10 ExecuteMachine.

Το ExecuteMachine είναι η βασική συνιστώσα που ενσωματώνει όλες τις πληροφορίες που παράγονται από τις προηγούμενες συνιστώσες και περιλαμβάνει όλες τις απαραίτητες λογικές για την εκτέλεση των στατικών κανόνων και των ενημερώσεων. Οι στατικοί κανόνες παρέχονται από τον Pruner, όσο οι δράσεις παρέχονται από τον ActionManager. Το Βήμα 2 του αλγόριθμου 2, υλοποιείται από αυτή τη συνιστώσα. Η εκτέλεση ξεκινά από μια αρχική κατάσταση που προέρχεται από το χρήστη. Στο παράδειγμά αυτό, η αρχική κατάσταση S_0 περιλαμβάνει ένα σύνολο από fluents και τα αντίστοιχα χρονικά διαστήματα που είναι σε αναμονή. Παράγεται μια σειρά από προσωρινές καταστάσεις, μέχρι να φτάσει σε μια τελική σταθερή κατάσταση. Η τελική σταθερή κατάσταση συμβαίνει μετά από αξιολόγηση όλων των στατικών κανόνων μέχρι να μην συμβαίνουν αλλαγές. Στη συνέχεια, εμφανίζονται οι προσωρινές καταστάσεις που παράγονται μέχρι να επιτευχθεί μια τελική σταθερή κατάσταση, που να βασίζεται στο παράδειγμά. Τα fluents αντανακλούν σε κάθε κατάσταση τις αλλαγές από την προηγούμενη κατάσταση.

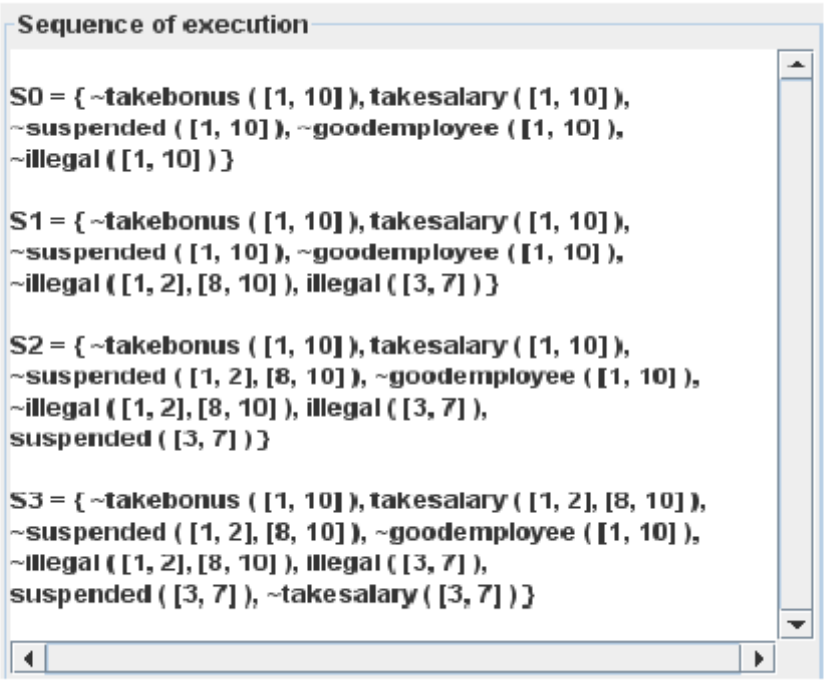
S_0 = { ¯takebonus([1, 10]), takesalary([1, 10]), ¯suspended([1, 10]), ¯goodemployee([1, 10]), ¯illegal([1, 10])}

Μετά την εκτέλεση του δυναμικού κανόνα misdemeanor([3, 7]) → illegal, θα πρέπει να ενημερωθεί η αρχική κατάσταση S_0, αλλάζοντας το χρονικό διάστημα που το fluent ¯illegal κατέχει, και προσθέτοντας το fluent illegal , οπότε η κατάσταση S_1 είναι η νέα προσωρινή κατάσταση.

S_1 = { ¯takebonus([1, 10]), takesalary([1, 10]), ¯suspended([1, 10]), ¯goodemployee([1, 10]), ¯illegal([1, 2], [8, 10]), illegal([3, 7])}

Στη συνέχεια, ο στατικός κανόνας illegal([3, 7]) → suspended ενεργοποιείται, γι' αυτό προστίθεται το fluent suspended([3, 7]) και ¯suspended([1, 2], [8, 10]) σε μια νέα προσωρινή κατάσταση S_2 όπως φαίνεται παρακάτω:

S_2 = { ¯takebonus([1, 10]), takesalary([1, 10]), ¯suspended([1, 2], [8, 10]), ¯goodemployee([1, 10]), ¯illegal([1, 2], [8, 10]), illegal([3, 7]), suspended([3, 7])}



***Εικ. 1.6**: Ακολουθία εκτέλεσης*

Στην τελευταία επανάληψη ο στατικός κανόνας suspended([3, 7])→¯ takesalary ενεργοποιείται και η τελική κατάσταση S_3 προκύπτει, όταν δεν υπάρχουν εκτελέσιμοι κανόνες.

S_3 = { ¯takebonus([1, 10]), takesalary([1, 2], [8, 10]), ¯suspended([1, 2], [8, 10]), ¯goodemployee([1, 10]), ¯illegal([1, 2], [8, 10]), illegal([3, 7]), suspended([3, 7]), ¯takesalary([3, 7])}.

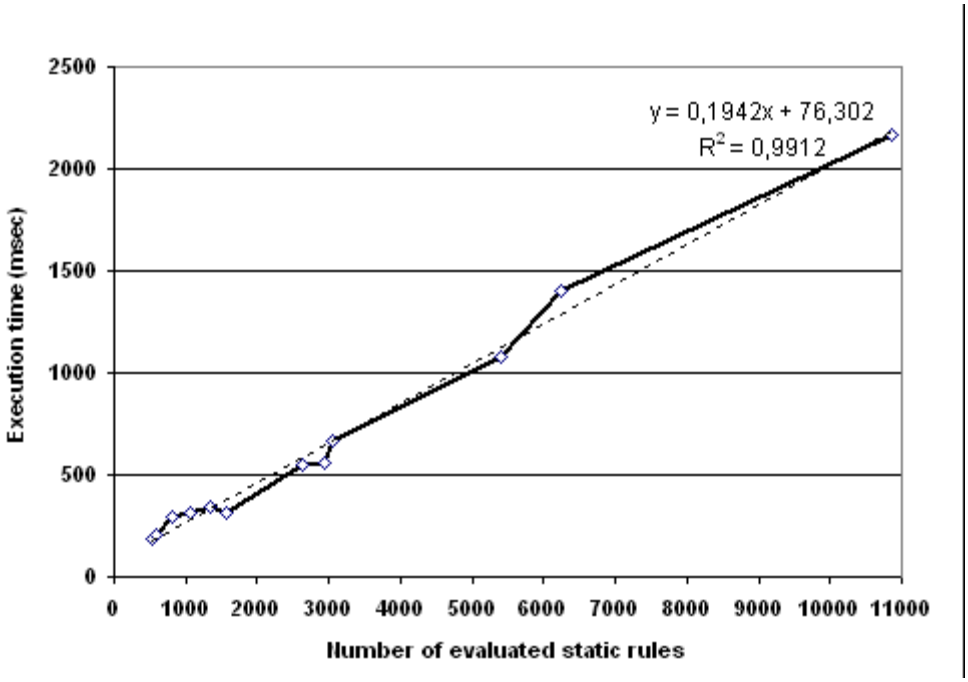
Μετά την εκτέλεση του αλγορίθμου η τελική κατάσταση S_3 παράγεται, όπως απεικονίζεται στο Σχήμα 6, γεγονός το οποίο καταδεικνύει ότι όλες οι ενημερώσεις έχουν γίνει και δεν υπάρχουν εκτελέσιμοι κανόνες .

1.5 Πολυπλοκότητα και αποτελέσματα αξιολόγησης

Η πολυπλοκότητα του υπολογισμού εξαρτάται από τη διαλειτουργικότητα της ακεραιότητας των περιορισμών. Αν υπάρχει μια σημαντική διαλειτουργικότητα τότε υπάρχει μια μεγαλύτερη ακολουθία μιας στατικής εκτέλεσης κανόνα, στην οποία κάθε κανόνας ενεργοποιεί την εκτέλεση του επόμενου. Υποθέτουμε ότι F είναι ο αριθμός των fluents και A ο αριθμός των ενεργειών. Τότε υπάρχουν A δυναμικοί κανόνες και 2*F στατικοί κανόνες.

Θεώρημα. Τα περισσότερα βήματα της εκτέλεσης του αλγόριθμου είναι O (F).

Απόδειξη. Αν υποθέσουμε ότι σε κάθε βήμα του αλγόριθμου εκτελείται ένας στατικός κανόνας, τότε ο μεγαλύτερος δυνατός αριθμός κανόνων είναι F.
$f_1 \wedge (f_1 \vee G'f_2) \rightarrow f_2, f_2 \wedge (f_2 \vee G'f_3) \rightarrow f_3, \ldots, f_n \wedge (f_n \vee G'_{\neg f1}) \rightarrow \neg f_1$
Ο μεγαλύτερος δυνατός αριθμός κανόνων που μπορούν να εκτελεστούν είναι F. Έτσι, η συνολική πολυπλοκότητα είναι O (F).



***Εικ. 1.7**: Γράφημα Χρόνος εκτέλεσης σε συνάρτηση με τον αριθμό των αξιολογηθέντων στατικών κανόνων*

Για να εκτιμηθεί ο συνολικός χρόνος εκτέλεσης σε περίπτωση που υπάρχουν πολλές εξαρτήσεις έγιναν κάποια πειράματα. Π.χ.
$\cdots \rightarrow f_1, f_1 \vee \cdots \rightarrow f_2, f_2 \cdots \rightarrow f_3, \ldots$.

Αυτοί οι περιορισμοί περιέχουν ένα σωρό από εξαρτήσεις, έτσι ο αλγόριθμος αξιολόγησης στατικού κανόνα θα εκτελέσει πολλές επαναλήψεις πριν επιστρέψει μια τελική συνεπή πρόταση. Η γραφική παράσταση που απεικονίζεται στο σχήμα 7 δείχνει ότι ο χρόνος εκτέλεσης αυξάνει σχεδόν γραμμικά. Σε αυτό το γράφημα έχει προστεθεί μια γραμμική, διακεκομμένη γραμμή παλινδρόμησης.

Η γραμμική εξίσωση που φαίνεται στον πίνακα αντιπροσωπεύει τη σχέση μεταξύ Αριθμού αξιολόγησης στατικών κανόνων(x) και Χρόνου Εκτέλεσης(y) για τη σύνθεση στο αποτέλεσμα. Η γραμμή παλινδρόμησης μπορεί να θεωρηθεί ως μία αποδεκτή εκτίμηση της πραγματικής σχέσης μεταξύ του αριθμού των στατικών κανόνων που αξιολογήθηκαν και του χρόνου εκτέλεσης. Επίσης χρησιμοποιήθηκε η R-squared τιμή που είναι το τετράγωνο του συντελεστή συσχέτισης και δίνει ένα μέτρο της αξιοπιστίας της γραμμικής σχέσης μεταξύ των x και y μεταβλητών. R-τετράγωνο τιμές κοντά στο 1 δείχνουν άριστη γραμμική αξιοπιστία. Στο γράφημα η τιμή του R-τετράγωνο είναι 0,9912.

1.6 Επίλογος

Μελετήθηκε το πρόβλημα της διακλάδωσης στον καθορισμό της χρονικής βάσης δεδομένων. Επίσης επεξηγείται η πολυπλοκότητα ανάλυσης και τα αποτελέσματα της αξιολόγησης του εργαλείου.

Με βάση την παραπάνω αναφορά και τις πληροφορίες για τις γλώσσες προγραμματισμού PHP και MySQL δημιουργήσαμε μία εφαρμογή στην οποία ικανοποιούνται και ελέγχονται όλοι οι περιορισμοί σε χρονικές βάσεις δεδομένων. Ελέγχονται όλες οι περιπτώσεις και τα αποτελέσματα εμφανίζονται σε μία σελίδα φυλλομετρητή ιστού.

ΚΕΦΑΛΑΙΟ 2ο

Η Γλώσσα Προγραμματισμού PHP

***Εικ. 2.1** : Λογότυπο της PHP*

2.1 Τι Είναι η PHP

Η PHP, όπου τα αρχικά σημαίνουν Hypertext PreProcessor, είναι μια γλώσσα συγγραφής σεναρίων (scripting language) που ενσωματώνεται μέσα στον κώδικα της HTML και εκτελείται στην πλευρά του server (server-side scripting). Ανταγωνιστικές της τεχνολογίας PHP είναι οι εξής γλώσσες προγραμματισμού: ASP (Active Server Pages), της εταιρείας Microsoft, CFML (ColdFusion Markup Language) της εταιρείας Allaire και JSP (JavaServer Pages) της εταιρείας Sun. Το μεγαλύτερο μέρος της σύνταξής της, η PHP το έχει δανειστεί από την C, την Java και την Perl και διαθέτει και μερικά δικά της μοναδικά χαρακτηριστικά. Ο σκοπός της γλώσσας είναι να δώσει τη δυνατότητα στους web developers να δημιουργούν δυναμικά παραγόμενες ιστοσελίδες. Ακολουθεί ένα εισαγωγικό παράδειγμα :

```
<html>
    <head>
        <title> Παράδειγμα </title>
    </head>
    <body>
        <?php echo "Γεια σας, είμαι ένα script της PHP!"; ?>
    </body>
</html>
```

Προσέξτε πόσο διαφέρει από ένα CGI script που γράφεται σ' άλλες γλώσσες, όπως η Perl ή η C, όπου αντί να γράψουμε ένα πρόγραμμα με πολλές εντολές για να δημιουργήσουμε κώδικα HTML, γράφουμε ένα HTML script με κάποιον ενσωματωμένο κώδικα για να κάνει κάτι, όπως στη συγκεκριμένη περίπτωση να εμφανίσει κάποιο κείμενο (μήνυμα). Ο κώδικας της PHP περικλείεται με ειδικά tags αρχής και τέλους για να μπορούμε να εισερχόμαστε και να εξαιρόμαστε από το PHP mode.

Αυτό που ξεχωρίζει την PHP από μια γλώσσα όπως η JavaScript, η οποία εκτελείται στην

πλευρά του χρήστη (client-side), είναι ότι ο κώδικάς της εκτελείται στον server. Αν είχαμε σ' έναν server ένα script παρόμοιο με το παραπάνω, ο χρήστης (client) θα λάμβανε το αποτέλεσμα της εκτέλεσης αυτού του script, χωρίς να είναι σε θέση να γνωρίζει ποιος μπορεί να είναι ο αρχικός κώδικας. Μπορούμε ακόμη να ρυθμίσουμε (configure) τον web server ώστε να επεξεργάζεται όλα τα HTML αρχεία με την PHP και τότε δεν θα υπάρχει πράγματι κανένας τρόπος να μάθουν οι χρήστες τον κώδικά μας.

2.2 Τι Μπορεί να Κάνει η PHP

Στο πιο βασικό επίπεδο, η PHP μπορεί να κάνει ό,τι και τα άλλα προγράμματα της τεχνολογίας CGI, όπως επεξεργασία των δεδομένων μιας φόρμας, δημιουργία δυναμικού περιεχομένου ιστοσελίδων ή αποστολή και λήψη cookies. Ίσως το δυνατότερο και πιο σημαντικό χαρακτηριστικό της PHP είναι η υποστήριξη που παρέχει σε μια ευρεία γκάμα από βάσεις δεδομένων. Έτσι, το να δημιουργήσουμε μια ιστοσελίδα που να παρέχει υποστήριξη σε βάσεις δεδομένων είναι απίστευτα απλό. Υποστηρίζει τις εξής βάσεις δεδομένων :

Adabas D	dBase	Empress	FilePro	Informix	InterBase	mSQL
MySQL	Oracle	PostgreSQL	Solid	Sybase	Velocis	Unix dbm

Πίνακ.1*: Υποστηριζόμενες βάσεις δεδομένων*

Η PHP παρέχει επίσης υποστήριξη για συνομιλία μ' άλλες υπηρεσίες, χρησιμοποιώντας πρωτόκολλα όπως τα IMAP, SNMP, NNTP, POP3 ή και το HTTP.

2.3 Μια Σύντομη Ιστορία της PHP

Η ιδέα για την δημιουργία της PHP ελήφθη το φθινόπωρο του 1994 από τον Rasmus Lerdorf. Οι πρώτες ανεπίσημες εκδόσεις (versions) της PHP χρησιμοποιήθηκαν στην αρχική του σελίδα (home page) για να μπορεί να παρακολουθεί αυτούς που έμπαιναν στην σελίδα. Η πρώτη έκδοση που δόθηκε για χρήση στο κοινό ήταν διαθέσιμη στις αρχές του 1995 με το όνομα Personal Home Page Tools. αποτελείτο από μια πολύ απλοϊκή μηχανή ανάλυσης (parser engine) η οποία καταλάβαινε λίγες μόνο ειδικές μακροεντολές (macros) και έναν αριθμό από utilities που βρίσκονταν σε κοινή χρήση στις home pages εκείνη την εποχή. Ένα guestbook, ένας μετρητής (counter) και κάποιο άλλο υλικό. Ο αναλυτής (parser) ξαναγράφηκε στα μέσα του 1995 και ονομάστηκε PHP/FIVersion 2. Το όνομα FI προέρχεται από ένα άλλο πακέτο που είχε γράψει ο Rasmus και το οποίο διερμήνευε (interpreted) τα δεδομένα από φόρμες της HTML. Συνδύασε τα εργαλεία scripts της Personal Home Page με τον Form Interpreter και πρόσθεσε υποστήριξη για mSQL. Έτσι γεννήθηκε η PHP/FI, η οποία αναπτύχθηκε αλματωδώς και διάφοροι χρήστες άρχισαν να συνεισφέρουν κώδικα σ' αυτήν. Υπολογίζεται ότι μέχρι τα τέλη του 1996, η PHP/FI χρησιμοποιείτο σε τουλάχιστον 15.000 web sites σ' όλον τον κόσμο και στα μέσα του 1997 αυτός ο αριθμός είχε ξεπεράσει τις 50.000. Στα μέσα του 1997 είχαμε επίσης μια αλλαγή στην ανάπτυξη της PHP. Σταμάτησε να αποτελεί το αγαπημένο αντικείμενο του Rasmus και έγινε ο στόχος μιας πιο καλά οργανωμένης ομαδικής εργασίας. Ο αναλυτής (parser) ξαναγράφηκε από την αρχή από τους Zeev Suraski και Andi Gutmans και αυτός ο νέος parser αποτέλεσε τη βάση για την PHP Version 3. Ένα μεγάλο μέρος του utility code μεταφέρθηκε από την PHP/FI στην PHP3 και ένα μεγάλο μέρος του ξαναγράφηκε από την αρχή. Σήμερα, η PHP/FI και η PHP3 έρχονται μ' έναν αριθμό εμπορικών προϊόντων όπως ο web server C2 StrongHold και το RedHat Linux. Σύμφωνα με μια συντηρητική εκτίμηση, η PHP χρησιμοποιείται από περισσότερα από 150.000 sites σ' όλον τον κόσμο.

2.4 Πώς να Ξεφύγουμε από την HTML

Υπάρχουν τέσσερις τρόποι για να μπορέσουμε να ξεφύγουμε από την HTML και να μπούμε στην μέθοδο συγγραφής κώδικα της PHP (PHP code mode) :

1ος τρόπος

```
<? echo ("Είναι η απλούστερη, μια εντολή επεξεργασίας SGML \n"); ?>
```

2ος τρόπος

```
<?php echo("Αν θέλουμε να εξυπηρετήσουμε XML έγγραφα \n"); ?>
```

3ος τρόπος

```
<script language="php">
    echo ("Σε μερικούς editors, όπως ο FrontPage, δεν αρέσουν οι εντολές επεξεργασίας");
</script>
```

4ος τρόπος

```
<% echo ("Μπορούμε να χρησιμοποιήσουμε και tags με στυλ ASP"); %>
<%= $variable;    # Είναι μια συντόμευση για το "<%echo .." %>
```

Ο πρώτος τρόπος είναι διαθέσιμος μόνο αν έχουμε ενεργοποιήσει τα σύντομα (short) tags. Αυτό μπορεί να γίνει με τη συνάρτηση short_tags(), ενεργοποιώντας το short_open_tag configuration setting στο αρχείο config της PHP ή μεταγλωττίζοντας την PHP με την επιλογή –enable-short-tags option. Ο τέταρτος τρόπος είναι διαθέσιμος μόνο αν έχουν ενεργοποιηθεί τα tags με στυλ ASP με το asp_tags configuration setting. Η υποστήριξη για τα ASP-style tags προστέθηκε στην έκδοση 3.0.4.

2.5 Τερματισμός Εντολών

Οι εντολές στην PHP τερματίζονται με τον ίδιο τρόπο όπως στην C και την Perl, δηλ. μ' έναν χαρακτήρα ; (semicolon). Μπορούμε, όμως, να δηλώσουμε το τέλος μιας εντολής και με το tag κλεισίματος (closing tag) ?>. Έτσι, τα παρακάτω είναι ισοδύναμα :

```
<?php
    echo "This is a test";
?>
```

Και

```
<?php echo "This is a test" ?>
```

2.6 Σχόλια (Comments)

Η PHP χρησιμοποιεί τον ίδιο τρόπο σχολιασμού όπως η C, η C++ και το Unix shell. Για παράδειγμα :

```
<?php
    echo "Αυτή είναι μια δοκιμή"; // Σχόλιο μίας γραμμής της C++
    /* Αυτό είναι ένα σχόλιο (comment) της C σε πολλές γραμμές
    και αυτή είναι μια άλλη γραμμή σχολίου */
    echo "Αυτή είναι άλλη μια δοκιμή";
```

```
    echo "Μια τελική δοκιμή"; # Σχόλιο της shell
?>
```

Τα σχόλια μίας γραμμής σχολιάζουν μέχρι το τέλος της γραμμής ή το τρέχον μπλοκ του PHP κώδικα, ανάλογα με το ποιο εμφανίζεται πρώτο.

```
    <h1> Αυτό είναι ένα <?# echo "απλό";?> παράδειγμα. </h1>
    <p> Το header θα εμφανίσει το 'Αυτό είναι ένα παράδειγμα.' </p>
```

Πρέπει να είμαστε προσεκτικοί για να μην φωλιάζουμε (nest) τα σχόλια τύπου C.

```
<?php
    /*
    echo "Αυτή είναι μια δοκιμή";
    /* Αυτό το σχόλιο θα δημιουργήσει πρόβλημα */
    */
?>
```

2.7 Οι Τύποι Δεδομένων της PHP

Η PHP υποστηρίζει τους εξής τύπους δεδομένων :

- array
- floating-point numbers
- integer
- object
- string

Ο τύπος δεδομένων μιας μεταβλητής δεν ορίζεται συνήθως από τον προγραμματιστή αλλά αποφασίζεται την ώρα εκτέλεσης (runtime) από την PHP ανάλογα με το περιβάλλον (context) στο οποίο χρησιμοποιείται η μεταβλητή. Αν θέλουμε να κάνουμε μια μεταβλητή να μετατραπεί σ' έναν συγκεκριμένο τύπο, μπορούμε είτε να μετατρέψουμε (cast) τη μεταβλητή ή να χρησιμοποιήσουμε τη συνάρτηση settype() σ' αυτή. Πρέπει να έχουμε υπόψη μας ότι μια μεταβλητή μπορεί να συμπεριφερθεί διαφορετικά σε συγκεκριμένες καταστάσεις, ανάλογα με το τι τύπο δεδομένων έχει εκείνη την στιγμή.

2.7.1 Οι Ακέραιοι (Integers)

Οι ακέραιοι (integers) μπορούν να καθορισθούν χρησιμοποιώντας μια από τις εξής συντάξεις :

```
    $a = 1234; # δεκαδικός αριθμός
    $a = -123; # αρνητικός αριθμός
    $a = 0123; # οκταδικός αριθμός (ισοδύναμος με τον δεκαδικό 83)
    $a = 0x12; # δεκαεξαδικός αριθμός (ισοδύναμος με τον δεκαδ. 18)
```

2.7.2 Οι Αριθμοί Κινητοί Υποδιαστολής

Οι αριθμοί κινητής υποδιαστολής (floating point numbers ή doubles), μπορούν να καθορισθούν χρησιμοποιώντας μια από τις εξής συντάξεις :

```
$a = 1.234;
$a = 1.2e3;
```

2.7.3 Τα Αλφαριθμητικά (Strings)

Τα αλφαριθμητικά (strings) μπορούν να καθορισθούν χρησιμοποιώντας ένα από δύο σύνολα οριοθετών (delimiters). Αν το string περικλείεται από διπλά εισαγωγικά (double-quotes, "), οι μεταβλητές μέσα στο string θα επεκταθούν. Όπως ισχύει στην C και την Perl, ο χαρακτήρας backslash (\) μπορεί να χρησιμοποιηθεί για να καθορίσουμε τους ειδικούς χαρακτήρες :

Ειδικός Χαρακτήρας	Νόημα
\n	Νέα γραμμή (newline)
\r	Carriage
\t	Οριζόντιο tab (στηλοθέτης)
\\	Χαρακτήρας backslash
\$	Σύμβολο του δολαρίου
\"	Διπλά εισαγωγικά
\[0-7]{1,3}	Η σειρά των χαρακτήρων που ταιριάζει με την κανονική έκφραση είναι ένας χαρακτήρας του 8δικού συστήματος (octal notation)
\x[0-9A-Fa-f]{1,2}	Η σειρά των χαρακτήρων που ταιριάζει με την κανονική έκφραση είναι ένας χαρακτήρας του 16δικού συστήματος (hexadecimal notation)

***Πίνακ.2**: Ειδικοί χαρακτήρες της PHP*

Ο δεύτερος τρόπος για να οριοθετήσουμε (delimit) ένα string χρησιμοποιεί τον χαρακτήρα μονού εισαγωγικού (single-quote, '). Όταν ένα string περικλείεται από μονά εισαγωγικά, οι μόνοι ειδικοί χαρακτήρες (escapes) που γίνονται αντιληπτοί είναι οι \\ και \'. Οι μεταβλητές δεν επεκτείνονται (αναλύονται) μέσα σ' ένα string που περικλείεται από μονά εισαγωγικά. Ένας άλλος τρόπος για να οριοθετήσουμε strings είναι να χρησιμοποιήσουμε τη σύνταξη heredoc syntax (">>>"). Θα πρέπει να υπάρχει ένα αναγνωριστικό (identifier) μετά από το >>>, μετά το string και μετά το ίδιο αναγνωριστικό για να κλείσει. Ακολουθεί ένα παράδειγμα.

```
$str = >>>EOD
Παράδειγμα ενός string που εκτείνεται σε πολλές γραμμές
```

```
    χρησιμοποιώντας τη σύνταξη heredoc syntax.
    EOD;
```

Τα strings μπορούν να ενωθούν (concatenated) με τον τελεστή '.' (dot), ενώ ο τελεστής της πρόσθεσης '+' δεν μπορεί να κάνει συνένωση. Μπορούμε να έχουμε πρόσβαση στους χαρακτήρες των strings αντιμετωπίζοντας το string σαν έναν πίνακα χαρακτήρων, χρησιμοποιώντας μια σύνταξη που θυμίζει C. Ακολουθούν παραδείγματα.

```
<?php
    /* Εκχώρηση ενός string */
    $str = "This is a string";
    /* Προσθήκη σ' ένα string */
    $str = $str . " with some more text";
    /* Ένας άλλος τρόπος προσθήκης */
    $str .= " and a newline at the end.\n";
    /* Αυτό το string θα τελειώνει ως '<p> Number :  9 </p>' */
    $num = 9;
    $str = "<p> Number : $num </p>";
    /* Και αυτό ως '<p> Number : $num </p>' */
    $num = 9;
    $str = '<p> Number : $num </p>';
    /* Διαβάζουμε τον πρώτο χαρακτήρα ενός string */
    $str = 'This is a test.';
    $first = $str[0];
    /* Διαβάζουμε τον τελευταίο χαρακτήρα ενός string */
    $str = 'This is still a test.';
    $last = $str[strlen($str)-1];
?>
```

2.7.3.1 Μετατροπή Strings

Όταν ένα string αποτιμάται σαν μια αριθμητική τιμή, η προκύπτουσα τιμή και ο τύπος δεδομένων καθορίζονται ως εξής : Το string θα αποτιμηθεί σε τύπο δεδομένων double αν περιέχει έναν από τους χαρακτήρες '.', 'e' ή 'E', αλλιώς θα αποτιμηθεί σαν ακέραιος (integer). Η τιμή δίνεται από το αρχικό τμήμα του string. Αν το string ξεκινά με έγκυρα αριθμητικά δεδομένα, αυτή θα είναι και η τιμή που θα χρησιμοποιηθεί. Διαφορετικά, η τιμή του θα είναι 0.

Έγκυρα αριθμητικά δεδομένα σημαίνει ένα προαιρετικό πρόσημο, ακολουθούμενο από ένα ή περισσότερα ψηφία, και ίσως μια υποδιαστολή, ακολουθούμενα από έναν προαιρετικό εκθέτη. Ο εκθέτης είναι ένα 'e' ή 'E' ακολουθούμενος από ένα ή περισσότερα ψηφία. Όταν η πρώτη έκφραση είναι ένα string, ο τύπος δεδομένων της μεταβλητής θα εξαρτηθεί από τη δεύτερη έκφραση.

```
$foo = 1 + "10.5";                // $foo is double   (11.5)
$foo = 1 + "-1.3e3";              //  $foo is double (-1299)
```

```
$foo = 1 + "bob-1.3e3";         //  $foo is integer     (1)
$foo = 1 + "bob3";              //  $foo is integer     (1)
$foo = 1 + "10 Small Pigs";     // $foo is integer     (11)
$foo = 1 + "10 Little Piggies";// $foo is integer     (11)
$foo = "10.0 pigs " + 1;        //  $foo is integer    (11)
$foo = "10.0 pigs " + 1.0;      // $foo is double      (11)
```

Αν θέλουμε να δοκιμάσουμε κάποια από τις παραπάνω εκφράσεις, μπορούμε να την αντιγράψουμε και να εισάγουμε την παρακάτω γραμμή για να δούμε τι γίνεται:

```
echo "\$foo==$foo; type is " . gettype($foo) . "<br>\n";
```

2.8 Πίνακες

2.8.1 Πίνακες Μίας Διάστασης (Single Dimension Arrays)

Οι πίνακες (arrays) ενεργούν και σαν πίνακες hash (associative arrays) και σαν δεικτοδοτούμενοι πίνακες (indexed arrays) ή διανύσματα (vectors). Η PHP υποστηρίζει και τους scalar και τους associative πίνακες. Στην πραγματικότητα, δεν υπάρχει καμία διαφορά ανάμεσά τους. Μπορούμε να δημιουργήσουμε έναν πίνακα με τις συναρτήσεις list() ή array() ή μπορούμε να ορίσουμε την τιμή κάθε στοιχείου του πίνακα, ως εξής :

```
$a[0] = "abc";
$a[1] = "def";
$b["foo"] = 13;
```

Μπορούμε επίσης να δημιουργήσουμε έναν πίνακα προσθέτοντας απλά τιμές στον πίνακα. Όταν εκχωρούμε μια τιμή σε μια μεταβλητή πίνακα χρησιμοποιώντας κενές αγκύλες, η τιμή θα προστεθεί στο τέλος του πίνακα.

```
$a[] = "hello"; // $a[2] == "hello"
$a[] = "world"; // $a[3] == "world"
```

Μπορούμε να ταξινομήσουμε τους πίνακες με τις συναρτήσεις asort(), arsort(), ksort(), rsort(), sort(), uasort(), usort() και uksort(), ανάλογα με τον τύπο της ταξινόμησης που θέλουμε να κάνουμε. Μπορούμε να μετρήσουμε τον αριθμό των στοιχείων ενός πίνακα με τη συνάρτηση count() και μπορούμε να διασχίσουμε έναν πίνακα με τις συναρτήσεις next() και prev() ή με τη συνάρτηση each().

2.8.2 Πίνακες Πολλών Διαστάσεων (Multi-Dimension Arrays)

Οι πίνακες πολλών διαστάσεων είναι αρκετά απλοί. Για κάθε διάσταση (dimension) του πίνακα, προσθέτουμε μια τιμή [key]. Ακολουθούν παραδείγματα.

```
$a[1] = $f;             # Πίνακες μίας διάστασης
$a["foo"] = $f;
$a[1][0] = $f;          # Πίνακες δύο διαστάσεων
$a["foo"][2] = $f;      # Μπορούμε να αναμείξουμε αριθμητικούς
```

```
$a[3]["bar"] = $f;    # και associative δείκτες (indices)
$a["foo"][4]["bar"][0] = $f;     # Πίνακας τεσσάρων διαστάσεων
```

Στην PHP3 δεν είναι δυνατό να αναφερθούμε σε πίνακες πολλών διαστάσεων απευθείας μέσα από strings. Για παράδειγμα, η επόμενη εντολή δεν θα έχει το επιθυμητό αποτέλεσμα :

```
$a[3]['bar'] = 'Bob';
echo "Αυτό δεν θα δουλέψει : $a[3][bar]";
```

Στην PHP3, το παραπάνω θα δημιουργήσει την έξοδο

```
Αυτό δεν θα δουλέψει : Array[bar].
```

Ο τελεστής ένωσης string, όμως, μπορεί να χρησιμοποιηθεί για να το ξεπεράσουμε αυτό :

```
$a[3]['bar'] = 'Bob';
echo "Αυτό θα δουλέψει : " . $a[3][bar];
```

Στην PHP4, όμως, μπορούμε να παρακάμψουμε αυτό το πρόβλημα αν περικλείσουμε την αναφορά στον πίνακα (μέσα στο string) με άγκιστρα, ως εξής :

```
$a[3]['bar'] = 'Bob';
echo "Αυτό θα δουλέψει : {$a[3][bar]}";
```

Μπορούμε να καταχωρήσουμε στοιχεία σε πίνακες με πολλούς τρόπους. Τα δύο επόμενα παραδείγματα γεμίζουν έναν πίνακα μίας διάστασης με τα ίδια στοιχεία :

```
# Παράδειγμα 1 :
$a["color"] = "red";
$a["taste"] = "sweet";
$a["shape"] = "round";
$a["name"]  = "apple";
$a[3]       = 4;

# Παράδειγμα 2 :
$a = array(
    "color" => "red",
    "taste"  => "sweet",
    "shape" => "round",
    "name"  => "apple",
    3     => 4
);
```

Μπορούμε να φωλιάσουμε (nest) τη συνάρτηση array() για πίνακες πολλαπλών διαστάσεων :

```
<?
$a = array(
          "apple"  => array(
          "color"  => "red",
          "taste"  => "sweet",
```

```
                "shape"  => "round"
            ),
                "orange"  => array(
                "color"   => "orange",
                "taste"   => "tart",
                "shape"   => "round"
            ),
                "banana"  => array(
                "color"   => "yellow",
                  taste"  => "paste-y",
                "shape"   => "banana-shaped"
            )
        );
        echo $a["apple"]["taste"];  # Θα δώσει το αποτέλεσμα "sweet"
?>
```

2.9 Τα Αντικείμενα (Objects)

Για να αρχικοποιήσουμε (initialize) ένα αντικείμενο (object), χρησιμοποιούμε την εντολή new για να δημιουργήσουμε μια μεταβλητή από το αντικείμενο.

```
class foo {
      function do_foo () {
            echo "Doing foo.";
      }
}
$bar = new foo;
$bar->do_foo();
```

2.10 Κόλπα με τους Τύπους Δεδομένων (Type Juggling)

Η PHP δεν υποστηρίζει τον σαφή (explicit) ορισμό τύπων δεδομένων στις δηλώσεις μεταβλητών και αυτό γιατί ο τύπος μιας μεταβλητής καθορίζεται από το περιβάλλον (context) στο οποίο χρησιμοποιείται αυτή η μεταβλητη. Αυτό σημαίνει ότι αν εκχωρήσουμε μια τιμή string σε μια μεταβλητή var, η var θα γίνει ένα string και αν αργότερα εκχωρήσουμε μια ακέραια τιμή στην var, αυτή θα γίνει ακέραια (integer).

Ένα παράδειγμα της αυτόματης μετατροπής τύπου στην PHP είναι ο τελεστής πρόσθεσης +. Αν κάποιος από τους τελεστέους (operands) είναι μια τιμή double, τότε όλοι οι τελεστέοι εκτιμούνται σαν τύπου doubles και το αποτέλεσμα θα είναι τύπου double. Αλλιώς, οι τελεστέοι θα θεωρηθούν ότι είναι ακέραιοι (integers) και το αποτέλεσμα θα είναι επίσης integer. Αυτό βέβαια δεν αλλάζει τους τύπους δεδομένων των ίδιων των τελεστέων και η μόνη αλλαγή είναι στο πώς εκτιμούνται οι τελεστέοι.

```
$foo = "0";                 // $foo is string          (ASCII 48)
$foo++;                     // $foo is the string "1" (ASCII 49)
```

```
$foo += 1;                  // $foo is now an integer        (2)
$foo = $foo + 1.3;          // $foo is now a double          (3.3)
$foo = 5 + "10 Little Piggies"; // $foo is integer           (15)
$foo = 5 + "10 Small Pigs";     // $foo is integer           (15)
```

Αν θέλουμε να δοκιμάσουμε κάποια από τις παραπάνω εκφράσεις, μπορούμε να την αντιγράψουμε και να εισάγουμε την παρακάτω γραμμή για να δούμε τι γίνεται :

```
echo "\$foo==$foo; type is " . gettype( $foo ) . "<br>\n";
```

2.11 Η Μετατροπή Τύπων (Type Casting)

Η μετατροπή τύπων (type casting) στην PHP εργάζεται όπως και στην C : το όνομα του επιθυμητού τύπου γράφεται μέσα σε παρενθέσεις πριν από τη μεταβλητή η οποία θα μετατραπεί.

```
$foo = 10;                  // Το $foo είναι τύπου integer
$bar = (double) $foo;       // Το $bar είναι τύπου double
```

Οι μετατροπές (casts) που επιτρέπονται είναι οι εξής :

```
(int), (integer) - cast to integer
(real), (double), (float) - cast to double
(string) - cast to string
(array) - cast to array
(object) - cast to object
```

Μπορεί να υπάρχουν κενά και tabs μέσα στις παρενθέσεις :

```
$foo = (int) $bar;
$foo = ( int ) $bar;
```

Όταν κάνουμε μετατροπή από μια μεταβλητή scalar ή string σ' έναν πίνακα (array), η μεταβλητή θα γίνει το πρώτο στοιχείο του πίνακα :

```
$var = 'ciao';
$arr = (array) $var;
echo $arr[0];         // εμφανίζει 'ciao'
```

Όταν κάνουμε μετατροπή από μια μεταβλητή scalar ή string σ' ένα αντικείμενο (object), η μεταβλητή θα γίνει μια ιδιότητα (attribute) του αντικειμένου και το όνομα (name) της ιδιότητας θα είναι τύπου scalar :

```
$var = 'ciao';
$obj = (object) $var;
echo $obj->scalar;  // εμφανίζει 'ciao'
```

2.12 Οι Μεταβλητές (Variables)

Οι μεταβλητές (variables) στην PHP παριστάνονται από το σύμβολο $ ακολουθούμενο από το όνομα της μεταβλητής. Τα ονόματα των μεταβλητών ξεχωρίζουν τα πεζά από τα κεφαλαία γράμματα (case-sensitive).

```
$var = "Bob";
```

```
$Var = "Joe";
echo "$var, $Var";  // εμφανίζει "Bob, Joe"
```

Στην PHP3, οι μεταβλητές πάντα εκχωρούνται με τιμή (by value), δηλαδή όταν εκχωρούμε μια έκφραση σε μια μεταβλητή, η τιμή της αρχικής έκφρασης αντιγράφεται στη μεταβλητή προορισμού. Αυτό σημαίνει, για παράδειγμα, ότι αφού έχουμε εκχωρήσει την τιμή μιας μεταβλητής σε μια άλλη, η αλλαγή σε μια απ' αυτές τις μεταβλητές δεν θα επηρεάσει την άλλη. Η PHP4 προσφέρει και έναν άλλον τρόπο για να εκχωρήσουμε τιμές σε μεταβλητές : με αναφορά (by reference), δηλ. η νέα μεταβλητή αναφέρεται (references) ή αποτελεί ένα ψευδώνυμο (alias) ή δείχνει (points) στην αρχική μεταβλητή. Οι αλλαγές στη νέα μεταβλητή επηρεάζουν και την αρχική και το αντίστροφο. Για να κάνουμε εκχώρηση με αναφορά, τοποθετούμε το σύμβολο & (ampersand) πριν από την αρχική μεταβλητή. Για παράδειγμα, ο επόμενος κώδικας εμφανίζει δύο φορές το μήνυμα 'My name is Bob' :

```
<?php
      $foo = 'Bob';              // Εκχώρηση της τιμής 'Bob' στην $foo
      $bar = &$foo;              // Αναφορά στην $foo μέσω της $bar.
      $bar = "My name is $bar"; // Η $bar αλλάζει
      echo $foo;                 // Η $foo αλλάζει επίσης
      echo $bar;
?>
```

Κάτι σημαντικό που πρέπει να σημειώσουμε είναι ότι μόνο ονοματισμένες μεταβλητές (named variables) μπορούν να εκχωρηθούν με αναφορά (by reference).

```
<?php
      $foo = 25;
      $bar = &$foo;              // Έγκυρη εκχώρηση
      $bar = &(24 * 7);          // Μη έγκυρη εκχώρηση
      function test() {
            return 25;
      }
      $bar = &test();            // Μη έγκυρη
?>
```

2.12.1 Οι Προκαθορισμένες Μεταβλητές

Η PHP παρέχει έναν μεγάλο αριθμό από προκαθορισμένες μεταβλητές (predefined variables) σ' οποιοδήποτε script εκτελεί. Όμως, πολλές απ' αυτές τις μεταβλητές δεν μπορούν να τεκμηριωθούν πλήρως (documented) γιατί εξαρτώνται από τον server στον οποίον εκτελούνται, την έκδοση (version) και την ρύθμιση (setup) του server καθώς και από άλλους παράγοντες. Παρ' όλα αυτά, θα δούμε μια λίστα από προκαθορισμένες μεταβλητές που είναι διαθέσιμες σε μια εγκατάσταση της PHP 3 που εκτελείται σαν ένα module σε μια εγκατάσταση του Apache 1.3.6.

2.12.2 Οι Μεταβλητές Apache

Αυτές οι μεταβλητές δημιουργούνται από τον Apache webserver. Αν χρησιμοποιούμε έναν άλλον webserver, ίσως να λείπουν μερικές ή να υπάρχουν άλλες που δεν εμφανίζονται εδώ. Ένας μεγάλος αριθμός των μεταβλητών που θα δούμε υπάρχουν στην προδιαγραφή CGI 1.1.

- ***GATEWAY_INTERFACE***

Ποια αναθεώρηση (revision) της προδιαγραφής CGI χρησιμοποιεί ο server, όπως π.χ. CGI/1.1.

- ***SERVER_NAME***

Το όνομα του server host στον οποίο εκτελείται το τρέχον script. Μπορεί να είναι και το όνομα ενός εικονικού (virtual) host.

- ***SERVER_SOFTWARE***

Το string αναγνώρισης του server (server identification string), το οποίο δίνεται στις επικεφαλίδες (headers) όταν ο server απαντάει σε αιτήσεις (requests).

- ***SERVER_PROTOCOL***

Το όνομα (name) και η αναθεώρηση (revision) του πρωτοκόλλου πληροφοριών (information protocol) μέσω του οποίου ζητήθηκε η σελίδα, δηλ. HTTP/1.0.

- ***REQUEST_METHOD***

Ποια μέθοδος αίτησης (request method) χρησιμοποιήθηκε για να έχουμε πρόσβαση στη σελίδα, δηλ. GET, HEAD, POST, PUT.

- ***QUERY_STRING***

Το query string, αν υπάρχει, μέσω του οποίου έχουμε πρόσβαση στη σελίδα.

- ***DOCUMENT_ROOT***

Το document root directory στο οποίο εκτελείται το τρέχον script, όπως ορίζεται στο αρχείο σύνθεσης (configuration file) του server.

- ***HTTP_ACCEPT***

Τα περιεχόμενα του Accept : η επικεφαλίδα (header) της τρέχουσας αίτησης (request), αν υπάρχει κάποια.

- ***HTTP_ACCEPT_CHARSET***

Τα περιεχόμενα του Accept-Charset : η επικεφαλίδα (header) της τρέχουσας αίτησης (request), αν υπάρχει κάποια. Παράδειγμα : iso-8859-1,*,utf-8'.

- ***HTTP_ENCODING***

Τα περιεχόμενα του Accept-Encoding : η επικεφαλίδα (header) της τρέχουσας αίτησης (request), αν υπάρχει κάποια. Παράδειγμα : gzip.

- ***HTTP_ACCEPT_LANGUAGE***

Τα περιεχόμενα του Accept-Language : η επικεφαλίδα (header) της τρέχουσας αίτησης (request), αν υπάρχει κάποια. Παράδειγμα : en.

- ***HTTP_CONNECTION***

Τα περιεχόμενα του Connection: η επικεφαλίδα (header) της τρέχουσας αίτησης (request), αν υπάρχει κάποια. Παράδειγμα : Keep-Alive.

- ***HTTP_HOST***

Τα περιεχόμενα του Host : η επικεφαλίδα (header) της τρέχουσας αίτησης (request), αν υπάρχει κάποια.

- ***HTTP_REFERER***

Η διεύθυνση της σελίδας, αν υπάρχει, η οποία έφερε τον φυλλομετρητή στην τρέχουσα σελίδα.

- ***HTTP_USER_AGENT***

Τα περιεχόμενα του User_Agent : η επικεφαλίδα (header) της τρέχουσας αίτησης (request), αν υπάρχει κάποια. Αυτό το string φανερώνει το λογισμικό του φυλλομετρητή που χρησιμοποιείται για να δούμε την τρέχουσα σελίδα, όπως π.χ. Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586). Μπορούμε να χρησιμοποιήσουμε αυτήν την τιμή με τη συνάρτηση get_browser() για να προσαρμόσουμε τη λειτουργικότητα της σελίδας μας σύμφωνα με τις δυνατότητες του φυλλομετρητή του χρήστη.

- ***REMOTE_ADDR***

Η IP διεύθυνση από την οποία βλέπει ο χρήστης την τρέχουσα σελίδα.

- ***REMOTE_PORT***

Η θύρα (port) που χρησιμοποιείται στο μηχάνημα του χρήστη για να επικοινωνήσει με τον web server.

- ***SCRIPT_FILENAME***

Η απόλυτη διαδρομή (pathname) του τρέχοντος εκτελούμενου script.

- ***SERVER_ADMIN***

Η τιμή που δίνεται στην οδηγία (directive) SERVER_ADMIN (για τον Apache) στο αρχείο σύνθεσης (configuration file) του web server.

- ***SERVER_PORT***

Η θύρα (port) στο μηχάνημα του server που χρησιμοποιείται από τον web server για επικοινωνία. Στις προκαθορισμένες ρυθμίσεις, έχει την τιμή 80, ενώ αν χρησιμοποιούμε το SSL, για παράδειγμα, θα αλλάξει σ' ο,τιδήποτε είναι η δική μας ορισμένη ασφαλής θύρα HTTP.

- ***SERVER_SIGNATURE***

Ένα string που περιέχει την έκδοση (version) του server και το virtual host name, τα οποία προστίθενται στις παραγόμενες από τον server σελίδες, αν είναι ενεργοποιημένες.

- ***SCRIPT_NAME***

Περιέχει τη διαδρομή (path) του τρέχοντος script. Είναι χρήσιμη για σελίδες που πρέπει να δείχνουν στον εαυτό τους.

- ***REQUEST_URI***

Το URI που δόθηκε για να μπορούμε να έχουμε πρόσβαση στη σελίδα, όπως π.χ. /index.html.

2.12.3 Οι Μεταβλητές της PHP

Αυτές οι μεταβλητές δημιουργούνται από την ίδια την PHP.

- ***Argv***

Είναι ένας πίνακας (array) από τα ορίσματα (arguments) που μεταβιβάζονται στο script. Όταν το script εκτελείται από τη γραμμή εντολών, αυτό μας δίνει μια πρόσβαση στις παραμέτρους της γραμμής εντολών, κάτι που θυμίζει την C. Όταν καλείται μέσω της μεθόδου GET, η μεταβλητή αυτή θα περιέχει το query string.

- ***argc***

Περιέχει τον αριθμό των παραμέτρων της γραμμής εντολών που μεταβιβάζονται στο script, αν αυτό εκτελείται από τη γραμμή εντολών βέβαια.

- ***PHP_SELF***

Το όνομα αρχείου (filename) του τρέχοντα εκτελούμενου script. Αν το PHP εκτελείται σαν ένας επεξεργαστής από τη γραμμή εντολών, αυτή η μεταβλητή δεν είναι διαθέσιμη.

- ***HTTP_COOKIE_VARS***

Ένας associative πίνακας (array) από μεταβλητές που μεταβιβάζονται στο τρέχον script μέσω HTTP cookies.

- ***HTTP_GET_VARS***

Ένας associative πίνακας (array) από μεταβλητές που μεταβιβάζονται στο τρέχον script μέσω της μεθόδου HTTP GET.

- ***HTTP_POST_VARS***

Ένας associative πίνακας (array) από μεταβλητές που μεταβιβάζονται στο τρέχον script μέσω της μεθόδου HTTP POST.

2.12.4 Η Εμβέλεια των Μεταβλητών

Η εμβέλεια (scope) μιας μεταβλητής είναι το περιβάλλον (context) μέσα στο οποίο ορίζεται. Οι περισσότερες από τις PHP μεταβλητές έχουν μία μόνο περιοχή εμβέλειας. Για παράδειγμα :

```
$a = 1;

include "b.inc";
```

Εδώ, η μεταβλητή $a θα είναι διαθέσιμη μέσα στο συμπεριλαμβανόμενο (included) b.inc script. Όμως, στις οριζόμενες από τον προγραμματιστή συναρτήσεις (user-defined functions) υπάρχει μια τοπική εμβέλεια. Μια μεταβλητή που χρησιμοποιείται μέσα σε μια συνάρτηση είναι εξ ορισμού περιορισμένη στην τοπική εμβέλεια αυτής της συνάρτησης. Για παράδειγμα :

```
$a = 1;              /* καθολική εμβέλεια */

Function Test () {

      echo $a;       /* αναφέρεται στην τοπική μεταβλητή */

}
```

```
Test ();
```

Αυτό το script δεν θα εμφανίσει κάποια έξοδο επειδή η εντολή echo αναφέρεται σε μια τοπική μεταβλητή (local variable) $a, η οποία δεν έχει αποκτήσει τιμή μέσα στην εμβέλειά της. Αυτό είναι διαφορετικό από τη γλώσσα C στο ότι οι καθολικές μεταβλητές (global variables) της C είναι αυτόματα διαθέσιμες στις συναρτήσεις εκτός κι αν επικαλύπτονται σαφώς από μια τοπική δήλωση. Αυτό μπορεί να προκαλέσει προβλήματα στο ότι μπορεί κάποιος άθελά του να αλλάξει μια καθολική μεταβλητή. Στην PHP οι καθολικές μεταβλητές πρέπει να δηλωθούν σαν global μέσα σε μια συνάρτηση αν πρόκειται να τις χρησιμοποιήσουμε μέσα σ' αυτή τη συνάρτηση. Ακολουθεί ένα παράδειγμα :

```
$a = 1;
$b = 2;
Function Sum () {
      global $a, $b;
      $b = $a + $b;
}
Sum ();
echo $b;
```

Το παραπάνω script θα εμφανίσει το 3. Δηλώνοντας τις $a και $b σαν global μέσα στη συνάρτηση, όλες οι αναφορές και στις δύο μεταβλητές θα αφορούν τις καθολικές τιμές. Δεν υπάρχει κάποιος περιορισμός στον αριθμό των καθολικών μεταβλητών που μπορεί να χειριστεί μια συνάρτηση. Ένας δεύτερος τρόπος για να έχουμε πρόσβαση σε μεταβλητές σε καθολική εμβέλεια είναι να χρησιμοποιήσουμε τον ειδικό πίνακα που ορίζεται στην PHP με όνομα $GLOBALS. Έτσι, το προηγούμενο παράδειγμα θα μπορεί να ξαναγραφεί ως εξής :

```
$a = 1;
$b = 2;
Function Sum () {
      $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}
Sum ();
echo $b;
```

Ο πίνακας $GLOBALS είναι ένας associative πίνακας με το όνομα της καθολικής μεταβλητής να αποτελεί το key και τα περιεχόμενα αυτής της μεταβλητής να αποτελούν την τιμή του στοιχείου του πίνακα. Ένα άλλο σημαντικό χαρακτηριστικό είναι οι στατικές μεταβλητές. Μια στατική μεταβλητή (static variable) υπάρχει μόνο στην τοπική εμβέλεια μιας συνάρτησης αλλά δεν χάνει την τιμή της όταν η εκτέλεση του προγράμματος εγκαταλείπει τη συνάρτηση. Ας δούμε το επόμενο παράδειγμα :

```
Function Test () {
      $a = 0;
      echo $a;
      $a++;
}
```

Αυτή η συνάρτηση δεν είναι και τόσο χρήσιμη εφόσον κάθε φορά που καλείται καταχωρεί στο $a το 0 και εκτυπώνει το "0". Η εντολή $a++, η οποία αυξάνει την τιμή της μεταβλητής, δεν κάνει

τίποτα εφόσον μόλις τελειώνει η συνάρτηση, εξαφανίζεται η μεταβλητή $a. Για να δημιουργήσουμε μια χρήσιμη συνάρτηση μέτρησης η οποία δεν θα χάνει τον έλεγχο της τρέχουσας μέτρησης, θα πρέπει να δηλώσουμε τη μεταβλητή $a σαν static :

```
Function Test () {
      static $a = 0;
      echo $a;
      $a++;
}
```

Τώρα, κάθε φορά που καλείται η συνάρτηση Test(), θα εκτυπώνει την τιμή της $a και θα την αυξάνει. Οι στατικές μεταβλητές παρέχουν επίσης έναν τρόπο για να ασχοληθούμε τις αναδρομικές συναρτήσεις. Μια αναδρομική συνάρτηση (recursive function) είναι αυτή που καλεί τον εαυτό της. Η επόμενη συνάρτηση μετράει αναδρομικά έως το 10 και χρησιμοποιεί τη στατική μεταβλητή $count για να ξέρει πότε να σταματήσει :

```
Function Test () {
      static $count = 0;
      $count++;
      echo $count;
      if ($count < 10) {
            Test ();
      }
      $count- -;
}
```

2.12.5 Μεταβλητές Μεταβλητές

Μερικές φορές είναι βολικό να μπορούμε να έχουμε μεταβλητά ονόματα μεταβλητών (variable variable names), δηλ. ένα όνομα μεταβλητής το οποίο μπορεί να ορισθεί και να χρησιμοποιηθεί δυναμικά. Όπως γνωρίζουμε, μια κανονική μεταβλητή ορίζεται με μια εντολή σαν την εξής :

```
$a = "hello";
```

Μια μεταβλητή μεταβλητή αποκτά την τιμή μιας μεταβλητής και την αντιμετωπίζει σαν το όνομα μιας μεταβλητής. Στο παραπάνω παράδειγμα, το hello, μπορεί να χρησιμοποιηθεί σαν το όνομα μιας μεταβλητής χρησιμοποιώντας δύο σύμβολα $, ως εξής :

```
$$a = "world";
```

Σ' αυτό το σημείο έχουμε ορίσει δύο μεταβλητές και τις έχουμε αποθηκεύσει στο συμβολικό δένδρο της PHP : η $a με περιεχόμενο "hello" και η $hello με περιεχόμενο "world". Συνεπώς, η επόμενη εντολή :

```
echo "$a ${$a}";
```

παράγει την ίδια ακριβώς έξοδο με την :

```
echo "$a $hello";
```

Δηλαδή και οι δύο παράγουν το : `hello world.`

Για να μπορέσουμε να χρησιμοποιήσουμε μεταβλητές μεταβλητές με πίνακες (arrays), θα πρέπει να λύσουμε ένα πρόβλημα ασάφειας. Δηλαδή, αν γράψουμε $$a[1], τότε ο αναλυτής (parser)

θα πρέπει να γνωρίζει αν σκοπεύαμε να χρησιμοποιήσουμε το $a[1] σαν μια μεταβλητή ή αν θέλαμε να είναι το $$a η μεταβλητή και μετά το [1] ο δείκτης (index) απ' αυτή τη μεταβλητή. Η σύνταξη για να επιλύσουμε αυτήν την αμφιβολία είναι : ${$a[1]} για την πρώτη περίπτωση και ${$a}[1] για τη δεύτερη.

2.12. 6 Μεταβλητές Εκτός της PHP

Όταν υποβάλλεται μια φόρμα σ' ένα PHP script, όλες οι μεταβλητές αυτής της φόρμας γίνονται αυτόματα διαθέσιμες στο script από την PHP. Για παράδειγμα, ας δούμε την εξής φόρμα :

```
<form action="foo.php3" method="post">
    Όνομα : <input type="text" name="name"><br>
    <input type="submit">
</form>
```

Όταν υποβληθεί η φόρμα, η PHP θα δημιουργήσει μια μεταβλητή με όνομα $name, η οποία θα περιέχει την τιμή που καταχωρήθηκε στο πεδίο name της φόρμας. Η PHP καταλαβαίνει επίσης τους πίνακες στο περιβάλλον των μεταβλητών φόρμας αλλά μόνο σε μία διάσταση. Μπορούμε, για παράδειγμα, να ομαδοποιήσουμε σχετικές μεταβλητές μαζί ή να χρησιμοποιήσουμε αυτό το χαρακτηριστικό για να ανακτήσουμε τιμές από μια λίστα πολλαπλής επιλογής (multiple select input) :

```
<form action="array.php" method="post">
    Όνομα : <input type="text" name="personal[name]"><br>
    Email : <input type="text" name="personal[email]"><br>
    Μπύρα : <br>
    <select multiple name="beer[]">
        <option value="Fix"> Fix
        <option value="Heineken"> Heineken
        <option value="Pilsen"> Pilsen
    </select>
    <input type="submit">
</form>
```

Αν το χαρακτηριστικό track_vars της PHP είναι ενεργοποιημένο, είτε με τη ρύθμιση σύνθεσης (configuration setting) track_vars ή με την οδηγία (directive) <?php_track_vars?>, τότεοι μεταβλητές που υποβάλλονται με τις μεθόδους POST ή GET θα βρίσκονται επίσης στους global associative πίνακες (arrays) $HTTP_POST_VARS και $HTTP_GET_VARS.

2.12.7 Οι Μεταβλητές Image Submit

Όταν υποβάλλουμε μια φόρμα, μπορούμε να χρησιμοποιήσουμε μια εικόνα (image) αντί για το στάνταρτ πλήκτρο submit, μ' ένα tag σαν το :

```
<input type=image src="image.gif" name="sub">
```

Όταν ο χρήστης κάνει κλικ κάπου πάνω στην εικόνα, η φόρμα θα σταλεί στον server με δύο επιπλέον μεταβλητές, τις sub_x και sub_y, οι οποίες περιέχουν τις συντεταγμένες (coordinates) του κλικ που έκανε ο χρήστης μέσα στην εικόνα.

2.12.8 Τα HTTP Cookies

Η PHP υποστηρίζει τα HTTP cookies όπως ορίζεται από τις προδιαγραφές της Netscape. Τα cookies είναι ένας μηχανισμός για να αποθηκεύονται δεδομένα στον απομακρυσμένο φυλλομετρητή και έτσι να μπορούμε να παρακολουθούμε ή να αναγνωρίζουμε τους χρήστες. Μπορούμε να ορίσουμε τα cookies με τη συνάρτηση SetCookie(). Τα cookies αποτελούν μέρος του HTTP header, έτσι η συνάρτηση SetCookie() πρέπει να κληθεί πριν σταλεί κάποια έξοδος στον φυλλομετρητή. Αυτός είναι ο ίδιος περιορισμός που ισχύει και για τη συνάρτηση Header(). Τα cookies που στέλνονται σε μας από τον client θα μετατραπούν αυτόματα σε μια μεταβλητή της PHP όπως συμβαίνει με τα δεδομένα των μεθόδων GETκαι POST. Αν θελήσουμε να εκχωρήσουμε πολλαπλές τιμές σ' ένα μόνο cookie, απλά προσθέτουμε τα σύμβολα [ ] στο όνομα του cookie, ως εξής :

```
SetCookie ("MyCookie[]", "Testing", time()+3600);
```

Ένα cookie θα αντικαταστήσει ένα ήδη υπάρχον με το ίδιο όνομα στον φυλλομετρητή μας εκτός κι αν διαφέρουν η διαδρομή (path) ή το domain. Έτσι, για μια εφαρμογή shopping cartμπορεί να θέλουμε να έχουμε έναν μετρητή (counter) και να το μεταβιβάσουμε αυτό ως εξής :

```
$Count++;

SetCookie ("Count", $Count, time()+3600);

SetCookie ("Cart[$Count]", $item, time()+3600);
```

2.12.9 Οι Μεταβλητές Περιβάλλοντος (Environment)

Η PHP κάνει αυτόματα διαθέσιμες τις μεταβλητές περιβάλλοντος (environment) σαν κανονικές PHP μεταβλητές.

```
Echo $HOME;

/* Δείχνει τη μεταβλητή environment HOME, αν έχει ορισθεί */
```

Εφόσον οι πληροφορίες που έρχονται μέσω των μηχανισμών GET, POST και Cookie δημιουργούν επίσης αυτόματα μεταβλητής PHP, είναι μερικές φορές καλύτερο να διαβάσουμε ρητά μια μεταβλητή από το environment για να είμαστε σίγουροι ότι λαμβάνουμε τη σωστή έκδοση. Η συνάρτηση getenv() μπορεί να χρησιμοποιηθεί γι' αυτόν τον σκοπό. Μπορούμε επίσης να ορίσουμε μια μεταβλητή environment με τη συνάρτηση putenv().

2.12.10 Οι Τελείες (Dots) στις Εισερχόμενες Μεταβλητές

Τυπικά, η PHP δεν αλλάζει τα ονόματα των μεταβλητών όταν αυτά μεταβιβάζονται σ' ένα script. Όμως, θα πρέπει να σημειωθεί ότι η τελεία (dot) δεν αποτελεί έναν έγκυρο χαρακτήρα σ' ένα όνομα μεταβλητής της PHP, ως εξής :

```
$varname.ext;   /* μη έγκυρο όνομα μεταβλητής */
```

Τώρα, αυτό που βλέπει ο αναλυτής (parser) είναι μια μεταβλητή με όνομα $varname ακολουθούμενη από τον τελεστή ένωσης string και από το ext που δεν ταιριάζει με κάποια γνωστή λέξη κλειδί. Γι' αυτόν τον λόγο, είναι σημαντικό να σημειώσουμε ότι η PHP αντικαθιστά αυτόματα τις τελείες στα εισερχόμενα ονόματα μεταβλητών με τον χαρακτήρα _ (underscore).

2.12.11 Καθορισμός των Τύπων Μεταβλητών

Επειδή η PHP καθορίζει τους τύπους των μεταβλητών και τους μετατρέπει όπως χρειάζεται, δεν είναι πάντα σίγουρο τι τύπο δεδομένων έχει μια δεδομένη μεταβλητή σε κάποια δεδομένη χρονική στιγμή. Η PHP περιέχει αρκετές συναρτήσεις που μπορούμε να χρησιμοποιήσουμε για να βρούμε τον τύπο δεδομένων μιας μεταβλητής. Αυτές είναι οι gettype(), is_long(), is_double(),is_string(),

is_array() και is_object().

2.13 Οι Σταθερές (Constants)

Η PHP ορίζει αρκετές σταθερές (constants) και παρέχει έναν μηχανισμό για να ορίσουμε περισσότερες κατά την ώρα εκτέλεσης (run-time). Οι σταθερές είναι σαν τις μεταβλητές, εκτός από το ότι πρέπει να ορισθούν με τη συνάρτηση define() και ότι δεν μπορούν να ξαναορισθούν αργότερα σε μια άλλη τιμή.

Οι προκαθορισμένες σταθερές, οι οποίες είναι πάντα διαθέσιμες, είναι οι εξής :

- ***__FILE__***

Το όνομα του αρχείου script που αναλύεται (parsed) αυτή τη στιγμή. Αν χρησιμοποιείται μέσα σ' ένα αρχείο το οποίο έχει συμπεριληφθεί (included), τότε δίνεται το όνομα του αρχείου που έχει συμπεριληφθεί και όχι το όνομα του πατρικού αρχείου (parent file).

- ***__LINE__***

Ο αριθμός της γραμμής μέσα στο τρέχον αρχείο script που αναλύεται (parsed). Αν χρησιμοποιείται μέσα σ' ένα αρχείο το οποίο έχει συμπεριληφθεί (included), τότε δίνεται η θέση μέσα στο αρχείο που έχει συμπεριληφθεί.

- ***PHP_VERSION***

Η αναπαράσταση σε string της έκδοσης (version) του PHP parser που χρησιμοποιείται εκείνη τη στιγμή, όπως π.χ. '3.0.8-dev'.

- ***PHP_OS***

Το όνομα του λειτουργικού συστήματος στο οποίο εκτελείται ο PHPparser, όπως π.χ. 'Linux'.

- ***TRUE***

Μια τιμή true.

- ***FALSE***

Μια τιμή false.

- ***E_ERROR***

Δηλώνει ένα λάθος (error) διαφορετικό από ένα parsing error από το οποίο είναι αδύνατη η ανάκαμψη (recovery).

- ***E_WARNING***

Δηλώνει μια συνθήκη όπου η PHP γνωρίζει ότι κάτι είναι λάθος αλλά συνεχίζει έτσι κι αλλιώς. Ένα παράδειγμα μπορεί να είναι μια μη έγκυρη κανονική έκφραση στην ereg().

- ***E_PARSE***

Ο parser σταμάτησε λόγω μη έγκυρης σύνταξης στο αρχείο script και η ανάκαμψη (recovery) είναι αδύνατη.

- ***E_NOTICE***

Κάτι συνέβη που μπορεί να είναι ένα λάθος (error) ή όχι. Η εκτέλεση συνεχίζεται. Τέτοια παραδείγματα μπορεί να είναι η χρήση ενός string χωρίς εισαγωγικά σαν hash index ή η χρήση μιας μεταβλητής που δεν έχει ορισθεί ακόμη.

Οι σταθερές του τύπου E_* χρησιμοποιούνται συνήθως με τη συνάρττηση error_reporting() για να ορίσουμε το επίπεδο αναφοράς λάθους (error reporting level). Μπορούμε να ορίσουμε επιπλέον μεταβλητές με τη συνάρτηση define(). Πρέπει να έχουμε υπόψη μας ότι αυτές είναι μεταβλητές και όχι μακροεντολές της C (C-style macros) και συνεπώς μόνο έγκυρα αριθμητικά δεδομένα (scalar data) μπορούν να παριστάνονται από μια μεταβλητή. Ακολουθεί ένα παράδειγμα με ορισμό σταθεράς.

```
<?php
      define("CONSTANT", "Hello world.");
      echo CONSTANT;              // εμφανίζει το "Hello world."
?>
```

Ακολουθεί ένα παράδειγμα με χρήση των προκαθορισμένων σταθερών

`__FILE__` και `__LINE__` .

```
<?php
      function report_error($file, $line, $message) {
            echo "An error occured in $file on line $line : $message.";
      }
      report_error(__FILE__,__LINE__, "Something went wrong!");
?>
```

2.14 Οι Εκφράσεις (Expressions)

Οι εκφράσεις (expressions) είναι από τους σημαντικότερους θεμέλιους λίθους της PHP. Στην PHP, σχεδόν ο,τιδήποτε γράφουμε αποτελεί μια έκφραση. Σαν έναν πολύ απλοϊκό αλλά ακριβή ορισμό για μια έκφραση μπορούμε να πούμε ότι «είναι ο,τιδήποτε έχει μια τιμή». Οι βασικότερες μορφές εκφράσεων είναι οι σταθερές (constants) και οι μεταβλητές (variables). Για παράδειγμα, όταν γράφουμε $a = 5, εκχωρούμε το 5 στο $a. Το 5 είναι μια έκφραση (expression) με την τιμή 5 και σ' αυτήν την περίπτωση το 5 είναι μια ακέραια σταθερά. Μετά απ' αυτήν την εκχώρηση, θα αναμέναμε η τιμή της $a να είναι ίση με 5, έτσι αν γράψουμε $b = $a, θα περιμέναμε να συμπεριφερθεί σαν να είχαμε γράψει $b = 5. Μ' άλλα λόγια, το $a είναι μια έκφραση (expression) με την τιμή 5 επίσης. Λίγο περισσότερα σύνθετα παραδείγματα για τις εκφράσεις είναι οι συναρτήσεις (functions), όπως για παράδειγμα η ακόλουθη συνάρτηση :

```
function foo () {
      return 5;
}
```

Οι συναρτήσεις είναι εκφράσεις με την τιμή της τιμής επιστροφής τους (return value). Έτσι, εφόσον η συνάρτηση foo() επιστρέφει το 5, η τιμή της έκφρασης foo() είναι 5. Συνήθως, βέβαια οι συναρτήσεις δεν επιστρέφουν απλά μια στατική τιμή αλλά κάνουν και υπολογισμούς. Η PHP υποστηρίζει τους εξής τρεις τύπους τιμών scalar : ακέραιες τιμές (integer), τιμές κινητής υποδιαστολής (floating point) και αλφαριθμητικές τιμές (string) values. Οι τιμές scalar είναι τιμές που δεν μπορούμε να διασπάσουμε σε μικρότερα κομμάτια, σ' αντίθεση με τους πίνακες (arrays), για παράδειγμα.

Η PHP υποστηρίζει επίσης δύο σύνθετους (composite, non-scalar) τύπους : τους πίνακες (arrays) και τα αντικείμενα (objects). Ο καθένας απ' αυτούς τους τύπους τιμών μπορεί να εκχωρηθεί σε μεταβλητές ή να επιστραφεί από συναρτήσεις. Η PHP αντιμετωπίζει τις εκφράσεις με τον ίδιο τρόπο που τις αντιμετωπίζουν πολλές από τις υπόλοιπες γλώσσες. Η PHP είναι μια προσανατολισμένη σε εκφράσεις (expression-oriented) γλώσσα, με την έννοια ότι σχεδόν ο,τιδήποτε αποτελεί μια έκφραση. Στο παράδειγμα που είδαμε προηγουμένως, το $a = 5 είναι μια έκφραση που έχει την τιμή 5. Έτσι, το να γράψουμε κάτι σαν το $b = ($a = 5) είναι το ίδιο με το να γράψουμε $a = 5; $b= 5;. Εφόσον οι εκχωρήσεις γίνονται από δεξιά προς τα αριστερά, μπορούμε να γράψουμε επίσης $b = $a = 5. Ένα άλλο καλό παράδειγμα είναι οι pre- και post- increment and decrement. Οι χρήστες της PHP/FI 2 και πολλών άλλων γλωσσών ίσως να είναι εξοικειωμένοι με την γραφή variable++ και variable--. Πρόκειται για τους τελεστές αύξησης (increment) και μείωσης (decrement).

Στην PHP/FI 2, η εντολή $a++ δεν έχει καμία τιμή καθώς δεν αποτελεί έκφραση και έτσι δεν μπορούμε να την εκχωρήσουμε ή να την χρησιμοποιήσουμε κατά κάποιον τρόπο. Η PHPβελτιώνει τις δυνατότητες για αύξηση/μείωση (increment/decrement) περιλαμβάνοντας κι αυτές τις εκφράσεις επίσης, όπως στην C. Το pre-increment, που γράφεται ως ++$variable είναι ίσο με την τιμή της μεταβλητής αυξημένης κατά ένα, δηλ. Η PHP αυξάνει την τιμή της μεταβλητής πριν χρησιμοποιήσει την τιμή της. Το post-increment, που γράφεται ως $variable++ είναι ίσο με την αρχική τιμή της $variable, πριν αυτή αυξηθεί, δηλ. Η PHP αυξάνει την μεταβλητή αφού έχει χρησιμοποιήσει την τιμή της. Ένας πολύ κοινός τύπος εκφράσεων είναι οι εκφράσεις σύγκρισης (comparison expressions), οι

οποίες αποτιμούνται σε 0 ή 1, που σημαίνουν FALSE ή TRUE αντίστοιχα. Η PHP υποστηρίζει τους τελεστές σύγκρισης >, >=, ==, !=, < και <=. Αυτές οι εκφράσεις χρησιμοποιούνται συνήθως μέσα σε εκτελέσεις υπό συνθήκη (conditional execution), όπως είναι οι εντολές if.

Το τελευταίο παράδειγμα εκφράσεων που θα δούμε είναι οι συνδυασμένες εκφράσεις τελεστή και εκχώρησης. Γνωρίζουμε ήδη ότι αν θέλουμε να αυξήσουμε το $a κατά 1, μπορούμε να γράψουμε απλά $a++ ή ++$a. Αλλά, αν θέλουμε να προσθέσουμε περισσότερο από το 1, όπως για παράδειγμα το 3; Θα μπορούσαμε να γράψουμε το $a++ πολλές φορές, αλλά αυτό δεν είναι προφανώς κάτι αποδοτικό. Η πρόσθεση του 3 στην τρέχουσα τιμή του $a μπορεί να γραφεί σαν $a += 3, που σημαίνει «πάρε την τιμή της $a, πρόσθεσε το 3 σ' αυτήν και εκχώρησε το αποτέλεσμα πίσω ξανά στην $a». Υπάρχουν και οι τελεστές $a -= 5, δηλ. αφαίρεση του 5 από την τιμή της $a, $b *= 7, δηλ. πολλαπλασιασμός της τιμής της $b με το 7 κ.ά. Υπάρχει μια ακόμα περίεργη έκφραση, ο τριαδικός τελεστής υπό συνθήκη (ternary conditional operator) :

```
$first ? $second : $third
```

Αν η τιμή της πρώτης υποέκφρασης είναι true, δηλ. όχι μηδενική, τότε αποτιμάται η δεύτερη υποέκφραση και αυτό είναι και το αποτέλεσμα της έκφρασης υπό συνθήκη (conditionalexpression). Αλλιώς, αποτιμάται η τρίτη υποέκφραση και αυτή είναι η τιμή.

Το ακόλουθο παράδειγμα αναφέρεται στα παραπάνω :

```
function double($i) {
    return $i*2;
}
$b = $a = 5;/* εκχωρείται η τιμή 5 στις μεταβλητές $a και $b */
$c = $a++;/* post-increment, εκχωρείται η αρχική τιμή της $a
(5) στην $c */
$e = $d = ++$b;/* pre-increment, εκχωρείται η αυξημένη τιμή της
$b (6) στις $d και $e και σ' αυτό το σημείο, οι $d και $e είναι
ίσες με 6 */
$f = double($d++); /* εκχωρείται η διπλή τιμή της $d πριν από
την αύξησή της, δηλ. 2*6 = 12 στην $f */
$g = double(++$e); /* εκχωρείται η διπλή τιμή της $e μετά από
την αύξησή της, δηλ. 2*7 = 14 στην $g */
$h = $g += 10; /* πρώτα, η $g αυξάνεται κατά 10 και τελειώνει
με την τιμή 24, η τιμή της εκχώρησης (24) εκχωρείται μετά στην
$h και η $h τελειώνει με την τιμή 24 επίσης */
```

Η PHP δεν διαθέτει έναν λογικό τύπο δεδομένων (boolean) και η αληθής τιμή των εκφράσεων στην PHP υπολογίζεται με παρόμοιο τρόπο με την Perl. Αυτό σημαίνει ότι μια μη μηδενική αριθμητική τιμή είναι TRUE και το 0 είναι FALSE. Οι αρνητικές τιμές δεν είναι ίσες με μηδέν και έτσι θεωρούνται TRUE. Το άδειο string και το string "0" είναι FALSE και όλα τα άλλα strings είναι TRUE. Με τις μη scalar τιμές (πίνακες και αντικείμενα), αν μια τιμή δεν περιέχει κάποιο στοιχείο θεωρείται FALSE, αλλιώς θεωρείται TRUE.

2.15 Οι Τελεστές

2.15.1 Οι Αριθμητικοί Τελεστές

Οι αριθμητικοί τελεστές (arithmetic operators) της PHP είναι οι εξής :

Παράδειγμα	Όνομα	Αποτέλεσμα

$a + $b	Πρόσθεση	Άθροισμα των $a και $b
$a - $b	Αφαίρεση	Διαφορά των $a και $b
$a * $b	Πολλαπλασιασμός	Γινόμενο των $a και $b
$a / $b	Διαίρεση	Πηλίκο των $a και $b
$a % $b	Ακέραιο υπόλοιπο (modulus)	Ακέραιο υπόλοιπο του $a διαιρούμενο με το $b

***Πίνακ.3**: Αριθμητικοί τελεστές της PHP*

2.15.2 Οι Τελεστές Εκχώρησης

Ο βασικός τελεστής εκχώρησης (assignment operator) είναι το =. Σημαίνει ότι ο αριστερός τελεστής γίνεται ίσος με την τιμή της έκφρασης που υπάρχει στα δεξιά.

Η τιμή μιας έκφρασης εκχώρησης είναι η τιμή που εκχωρείται, δηλ. η τιμή της έκφρασης $a = 3 είναι το 3. Αυτό μας δίνει τη δυνατότητα να κάνουμε μερικά έξυπνα κόλπα :

```
$a = ($b = 4) + 5; // το $a γίνεται ίσο με 9 και το $b με 4
```

Εκτός από τον βασικό τελεστή εκχώρησης, υπάρχουν «συνδυασμένοι τελεστές» για όλους τους δυαδικούς αριθμητικούς και αλφαριθμητικούς τελεστές οι οποίοι μας δίνουν τη δυνατότητα να χρησιμοποιήσουμε μια τιμή σε μια έκφραση και μετά να ορίσουμε την τιμή της με το αποτέλεσμα αυτής της έκφρασης. Για παράδειγμα :

```
$a = 3;
$a += 5;            // κάνει το $a ίσο με 8 σαν $a = $a + 5;
$b = "Hello ";
$b .= "There!";    // κάνει το $b ίσο με "Hello There!" σαν $b =
$b . "There!";
```

Η παραπάνω εκχώρηση αντιγράφει την αρχική μεταβλητή στην καινούργια, που αποκαλείται εκχώρηση με τιμή (assignment by value) και έτσι οι αλλαγές που θα συμβούν στη μια απ' αυτές δεν θα επηρεάσουν και την άλλη. Η PHP4 υποστηρίζει την εκχώρηση με αναφορά (assignment by reference), χρησιμοποιώντας την σύνταξη $var = &$othervar;, κάτι που δεν ισχύει στην PHP3. Η εκχώρηση με αναφορά σημαίνει ότι και οι δύο μεταβλητές δείχνουν στα ίδια δεδομένα και τίποτα δεν αντιγράφεται.

2.15.3 Οι Τελεστές Δυαδικών Πράξεων

Οι τελεστές δυαδικών πράξεων (bitwise operators) μάς δίνουν τη δυνατότητα να αλλάξουμε την τιμή συγκεκριμένων δυαδικών ψηφίων (bits) μέσα σ' έναν ακέραιο.

Παράδειγμα	**Όνομα**	**Αποτέλεσμα**
$a & $b	And	Επιστρέφει 1 αν τα αντίστοιχα bits είναι 1 και στην $a και στην $b
$a | $b	Or	Επιστρέφει 1 αν ένα από τα αντίστοιχα bits είναι ίσα με 1 στην $a ή στην $b
$a Ù $b	Xor	Επιστρέφει 1 αν ένα από τα αντίστοιχα bits είναι ίσα με 1 στην $a ή στην $b αλλά όχι και στις δύο ταυτόχρονα
~$a	Not	Επιστρέφει 1 αν το αντίστοιχο bit του $a είναι 0, αλλιώς επιστρέφει 0
$a << $b	Shift left	Μετακινεί τα bits του $a κατά $b βήματα προς τα αριστερά, όπου το κάθε βήμα σημαίνει πολλαπλασιασμός επί 2
$a >> $b	Shift right	Μετακινεί τα bits του $a κατά $b βήματα προς τα δεξιά, όπου το κάθε βήμα σημαίνει διαίρεση με το 2

***Πίνακ.4**: Τελεστές Διαδικών Πράξεων της PHP*

2.15.4 Οι Τελεστές Σύγκρισης

Οι τελεστές σύγκρισης (comparison operators) μάς δίνουν τη δυνατότητα να συγκρίνουμε δύο τιμές.

Παράδειγμα	Όνομα	Αποτέλεσμα
$a = = $b	Ίσο	True αν το $a είναι ίσο με το $b
$a = = = $b	Ακριβώς ίδιο	True αν τα $a είναι ίσο με $b και είναι του ίδιου τύπου (μόνο στην PHP4)
$a != $b	Όχι ίσο	True αν το $a δεν είναι ίσο με το $b
$a < $b	Μικρότερο από	True αν το $a είναι μικρότερο από το $b
$a > $b	Μεγαλύτερο από	True αν το $a είναι μεγαλύτερο από το $b
$a <= $b	Μικρότερο από ή ίσο με	True αν το $a είναι μικρότερο ή ίσο από το $b
$a >= $b	Μεγαλύτερο από ή ίσο με	True αν το $a είναι μεγαλύτερο ή ίσο από το $b

***Πίνακ.5**: Τελεστές Σύγκρισης της PHP*

Ένας άλλος τελεστής υπό συνθήκη (conditionaloperator) είναι ο ?: ή τριαδικός (trinary) τελεστής, ο οποίος λειτουργεί όπως στην C και σ' άλλες γλώσσες, ως εξής :

```
(expr1) ? (expr2) : (expr3);
```

Η παραπάνω έκφραση αποτιμάται στην expr2 αν η expr1 έχει αποτιμηθεί σε true και στην expr3 αν η expr1 έχει αποτιμηθεί σε false.

2.15.5 Οι Τελεστές Εκτέλεσης

Η PHP υποστηρίζει έναν τελεστή εκτέλεσης (execution operator), τον backticks (``). Η PHP θα προσπαθήσει να εκτελέσει τα περιεχόμενα των backticks σαν μια εντολ shell. Η έξοδος μπορεί να ανατεθεί σε μια μεταβλητή.

```
$output = `ls -al`;
echo "<pre>$output</pre>";
```

Μπορούμε να δούμε επίσης και τις συναρτήσεις system(), passthru(), exec(), popen() και escapeshellcmd().

2.15.6 Οι Τελεστές Αύξησης/Μείωσης

Η PHP υποστηρίζει τους τελεστές αύξησης και μείωσης που θυμίζουν την C (C-style pre- and post-increment and decrement operators).

Παράδειγμα	Όνομα	Αποτέλεσμα
++$a	Pre-increment	Αυξάνει το $a κατά ένα και μετά το επιστρέφει
$a++	Post-increment	Επιστρέφει το $a και μετά το αυξάνει κατά ένα
--$a	Pre-decrement	Μειώνει το $a κατά ένα και μετά το επιστρέφει
$a--	Post-decrement	Επιστρέφει το $a και μετά το μειώνει κατά ένα

***Πίνακ.6**: Τελεστές αύξησης / μείωσης της PHP*

Ακολουθεί ένα απλό παράδειγμα script :

```php
<?php
	echo "<h3> Postincrement </h3>";
	$a = 5;
	echo "Πρέπει να είναι 5 : " . $a++ . "<br>\n";
	echo "Πρέπει να είναι 6 : " . $a . "<br>\n";
	echo "<h3> Preincrement </h3>";
	$a = 5;
	echo "Πρέπει να είναι 6 : " . ++$a . "<br>\n";
	echo "Πρέπει να είναι 6 : " . $a . "<br>\n";

	echo "<h3> Postdecrement </h3>";
	$a = 5;
	echo "Πρέπει να είναι 5 : " . $a-- . "<br>\n";
	echo "Πρέπει να είναι 4 : " . $a . "<br>\n";

	echo "<h3> Predecrement </h3>";
	$a = 5;
	echo "Πρέπει να είναι 4 : " . --$a . "<br>\n";
	echo "Πρέπει να είναι 4 : " . $a . "<br>\n";
?>
```

2.15.7 Οι Λογικοί Τελεστές

Οι λογικοί τελεστές (logical operators) της PHP είναι οι εξής :

Παράδειγμα	Όνομα	Αποτέλεσμα
$a and $b	And	True αν και το $a και το $b είναι true
$a or $b	Or	True αν ένα από τα $a ή $b είναι true
$a xor $b	Xor	True αν ένα από τα $a ή $b είναι true αλλά όχι και τα δύο
!$a	Not	True αν το $a δεν είναι true
$a && $b	And	True αν και το $a και το $b είναι true
$a || $b	Or	True αν ένα από τα $a ή $b είναι true

***Πίνακ.7**: Λογικοί τελεστές της PHP*

Ο λόγος που υπάρχουν δύο διαφορετικές παραλλαγές των τελεστών and και or είναι ότι λειτουργούν με διαφορετικές προτεραιότητες.

2.15.8 Οι Τελεστές των Αλφαριθμητικών (Strings)

Υπάρχουν δύο τελεστές για τα αλφαριθμητικά (strings). Ο πρώτος είναι ο τελεστής συνένωσης (concatenation operator), '.', ο οποίος επιστρέφει την ένωση του δεξιού και του αριστερού του ορίσματος. Ο δεύτερος είναι ο τελεστής εκχώρησης συνένωσης (concatenating assignment operator), '.='.

```
$a = "Hello ";
$b = $a . "World!";  // το $b περιέχει το "Hello World!"
$a = "Hello ";
$a .= "World!";      // το $a περιέχει το "Hello World!"
```

2.16 Οι Δομές Ελέγχου (Control Structures)

Ένα script της PHP αποτελείται από μια σειρά εντολών, όπου μια εντολή μπορεί να είναι μια εκχώρηση, μια κλήση συνάρτησης, ένας βρόγχος, μια εντολή υπό συνθήκη ή ακόμη και μια εντολή που δεν κάνει τίποτα (μια κενή, empty, εντολή). Οι εντολές τελειώνουν συνήθως με τον χαρακτήρα ; (semicolon). Επιπλέον, οι εντολές μπορούν να ομαδοποιηθούν σε μια εντολή-ομάδα (statement-group) αν περικλείσουμε μια ομάδα εντολών με άγκιστρα { και }. Μια εντολή-ομάδα αποτελεί και η ίδια μια εντολή.

2.16.1 Η Εντολή If

Η σύνταξη της εντολής if στην PHP είναι παρόμοια μ' αυτήν της C :

```
if (έκφραση)
    … εντολή …
```

Αν η έκφραση αποτιμηθεί σε TRUE, η PHP θα εκτελέσει την εντολή, ενώ αν αποτιμηθεί σε FALSE, θα την αγνοήσει. Το επόμενο παράδειγμα θα εμφανίσει ένα μήνυμα αν το $a είναι μεγαλύτερο από το $b :

```
if ($a > $b)
    print "Το a είναι μεγαλύτερο από το b";
```

Μπορούμε να ομαδοποιήσουμε περισσότερες από μία εντολές, ως εξής :

```
if ($a > $b) {
    print "Το a είναι μεγαλύτερο από το b";
    $b = $a;
}
```

2.16.2 Η Εντολή Else

Η εντολή else επεκτείνει μια εντολή if για να εκτελέσει μια εντολή στην περίπτωση που η έκφραση στην εντολή if αποτιμηθεί σε FALSE. Για παράδειγμα, ο ακόλουθος κώδικας εμφανίζει ένα ανάλογο μήνυμα :

```
if ($a > $b) {
    print "Το a είναι μεγαλύτερο από το b";
```

```
} else {
        print "Το a ΔΕΝ είναι μεγαλύτερο από το b";
}
```

2.16.3 Η Εντολή Elseif

Η εντολή elseif είναι ένας συνδυασμός των εντολών if και else. Επεκτείνει μια εντολή if για να εκτελέσει μια διαφορετική εντολή στην περίπτωση που η έκφραση της εντολής if αποτιμηθεί σε FALSE, αλλά θα εκτελέσει αυτήν την εναλλακτική έκφραση μόνο αν η συνθήκη έκφρασης της elseif αποτιμηθεί σε TRUE. Για παράδειγμα, ο επόμενος κώδικας ελέγχει τρεις περιπτώσεις και θα εμφανίσει ένα ανάλογο μήνυμα αν το a είναι μεγαλύτερο, ίσο ή μικρότερο από το b :

```
if ($a > $b) {
        print "Το a είναι μεγαλύτερο από το b";
} elseif ($a == $b) {
        print "Το a είναι ίσο με το b";
} else {
        print "Το a είναι μικρότερο από το b";
}
```

2.16.4 Ένας Άλλος Τρόπος Σύνταξης των Δομών Ελέγχου

Η PHP προσφέρει μια εναλλακτική σύνταξη για μερικές από τις δομές ελέγχου της (control structures) και πιο συγκεκριμένα τις if, while, for και switch. Η βασική μορφή της εναλλακτικής σύνταξης είναι να αλλάξει την αγκύλη ανοίγματος στον χαρακτήρα : και την αγκύλη κλεισίματος στο endif;, endwhile;, endfor; ή endswitch;αντίστοιχα.

```
<?php if ($a==5): ?>
        Το a είναι ίσο με 5
<?php endif; ?>
```

Στο παραπάνω παράδειγμα, ένα μπλοκ της HTML βρίσκεται μέσα σε μια εντολή if που είναι γραμμένη με την εναλλακτική σύνταξη. Το μπλοκ της HTML θα εμφανισθεί μόνο αν το $aείναι ίσο με 5. Η εναλλακτική σύνταξη ισχύει και για τις εντολές else και elseif επίσης.

Ακολουθεί μια δομή if με elseif και else στην εναλλακτική σύνταξη :

```
if ($a == 5):
        print "Το a είναι ίσο με 5";
        print "...";
elseif ($a == 6):
        print "Το a είναι ίσο με 6";
        print "!!!";
else:
        print "Το a δεν είναι ούτε 5 ούτε 6";
endif;
```

2.16.5 Η Εντολή While

Οι βρόγχοι while αποτελούν τον απλούστερο τύπο βρόγχου στην PHP και συμπεριφέρονται όπως ακριβώς οι αντίστοιχοι βρόγχοι στην C. Η βασική μορφή μιας εντολής while είναι η εξής :

```
while (έκφραση) … εντολή …
```

Η εντολή while λέει στην PHP να εκτελεί συνέχεια την ή τις εντολές για όσο διάστημα η έκφραση της while αποτιμάται σε TRUE. Η τιμή της έκφρασης ελέγχεται κάθε φορά στην αρχή του βρόγχου. Όπως και με την εντολή if, μπορούμε να ομαδοποιήσουμε πολλές εντολές μέσα στον ίδιο βρόγχο while χρησιμοποιώντας τα { και } ή την εναλλακτική σύνταξη :

```
while (έκφραση): … εντολή ... endwhile;
```

Τα δύο επόμενα παραδείγματα είναι ολόιδια και εμφανίζουν και τα δύο τους αριθμούς 1 έως 10 :

```
$i = 1;
while ($i <= 10) {
      print $i++;
}
```

```
$i = 1;
while ($i <= 10):
      print $i;
      $i++;
endwhile;
```

2.16.6 Η Εντολή Do .. While

Οι βρόγχοι do .. while είναι πολύ παρόμοιοι με τους βρόγχους while, εκτός από το ότι η έκφραση ελέγχεται στο τέλος κάθε επανάληψης και όχι στην αρχή. Η βασική διαφορά τους από τους βρόγχους while είναι ότι η πρώτη επανάληψη ενός βρόγχου do .. while θα εκτελεσθεί σίγουρα τουλάχιστον μία φορά.

Υπάρχει μία μόνο σύνταξη για τον βρόγχο do .. while :

```
$i = 0;
do {
      print $i;
} while ($i>0);
```

Ο παραπάνω βρόγχος θα εκτελεσθεί μία φορά ακριβώς, εφόσον μετά από την πρώτη επανάληψη, όταν ελέγχεται η έκφραση, αποτιμάται σε FALSE και έτσι τερματίζεται η εκτέλεση του βρόγχου. Μπορούμε να χρησιμοποιήσουμε την εντολή break για να σταματήσουμε την εκτέλεση ενός βρόγχου στη μέση του κώδικα :

```
do {
      if ($i < 5) {
            print "Το i δεν είναι αρκετά μεγάλο";
            break;
```

```
        }
        $i *= $factor;
        if ($i < $minimum_limit) {
              break;
        }
        print "To i είναι OK";
        ... process I ...
} while(0);
```

2.16.7 Η Εντολή For

Οι βρόγχοι for είναι οι πιο πολύπλοκοι βρόγχοι στην PHP. Συμπεριφέρονται όπως οι αντίστοιχοί τους στην C και η σύνταξη ενός βρόγχου for είναι η εξής :

```
for (έκφραση1; έκφραση2; έκφραση3) … εντολή …
```

Η πρώτη έκφραση (έκφραση1) αποτιμάται (εκτελείται) μία φορά, χωρίς να υπάρχει κάποια συνθήκη, στην αρχή του βρόγχου. Στην αρχή της κάθε επανάληψης αποτιμάται η έκφραση2 και αν αποτιμηθεί σε TRUE, ο βρόγχος συνεχίζεται και εκτελούνται οι περιεχόμενες εντολές. Αν αποτιμηθεί σε FALSE, σταματάει η εκτέλεση του βρόγχου. Στο τέλος της κάθε επανάληψης αποτιμάται (εκτελείται) η έκφραση3. Και οι τρεις εκφράσεις μπορούν να είναι κενές (empty). Αν είναι κενή η έκφραση2, αυτό σημαίνει ότι ο βρόγχος θα εκτελείται ασταμάτητα. Αυτό είναι χρήσιμο όταν θέλουμε να βγούμε από τον βρόγχο χρησιμοποιώντας μια εντολή break. Και τα τέσσερα επόμενα παραδείγματα εμφανίζουν τους αριθμούς από 1 έως 10 :

```
/* παράδειγμα 1 */
for ($i = 1; $i <= 10; $i++) {
      print $i;
}

/* παράδειγμα 2 */
for ($i = 1;;$i++) {
      if ($i > 10) {
            break;
      }
      print $i;
}

/* παράδειγμα 3 */
$i = 1;
for (;;) {
      if ($i > 10) {
            break;
```

```
        }
        print $i;
        $i++;
}
/* παράδειγμα 4 */
for ($i = 1; $i <= 10; print $i, $i++) ;
```

Η PHP υποστηρίζει επίσης και την εναλλακτική σύνταξη για τους βρόγχους for :

```
for (έκφραση1; έκφραση2; έκφραση3): εντολή; ...; endfor;
```

Ενώ άλλες γλώσσες χρησιμοποιούν την εντολή foreach για να διασχίσουν έναν πίνακα (array) ή ένα hash, η PHP χρησιμοποιεί την εντολή while και τις συναρτήσεις list() και each() γι' αυτόν τον σκοπό.

2.16.8 Η Εντολή Break

Με την εντολή break μπορούμε να εξέλθουμε από μια δομή ελέγχου χωρίς να περιμένουμε να ικανοποιηθεί η συνθήκη εξόδου του βρόγχου.

```
$i = 0;
while ($i < 10) {
        if ($arr[$i] == "stop") {
                break;
        }
        $i++;
}
```

2.16.9 Η Εντολή Continue

Η εντολή continue χρησιμοποιείται σε δομές βρόγχου για να συνεχίσει την εκτέλεση του προγράμματος από την αρχή του βρόγχου και να αγνοήσει έτσι τις υπόλοιπες εντολές μέχρι το τέλος του βρόγχου.

```
while (list($key, $value) = each($arr)) {
        if ($key % 2) {    // αγνοεί τους άρτιους αριθμούς
                continue;
        }
        do_something_odd ($value);
}
```

2.16.10 Η Εντολή Switch

Η εντολή switch είναι παρόμοια με μια σειρά εντολών if στην ίδια έκφραση. Υπάρχουν πολλές περιπτώσεις όπου θέλουμε να συγκρίνουμε την ίδια μεταβλητή ή έκφραση με πολλές

διαφορετικές τιμές και να εκτελέσουμε ένα διαφορετικό κομμάτι κώδικα ανάλογα με την τιμή της μεταβλητής. Τα δύο επόμενα παραδείγματα είναι δύο διαφορετικοί τρόποι για να γράψουμε το ίδιο πράγμα, όπου το ένα χρησιμοποιεί μια σειρά από εντολές if και το άλλο χρησιμοποιεί την εντολήswitch :

```
if ($i == 0) {
      print "Το i είναι ίσο με 0";
}
if ($i == 1) {
      print "Το i είναι ίσο με 1";
}
if ($i == 2) {
      print "Το i είναι ίσο με 2";
}

switch ($i) {
      case 0:
            print "Το i είναι ίσο με 0";
            break;
      case 1:
            print "Το i είναι ίσο με 1";
            break;
      case 2:
            print "Το i είναι ίσο με 2";
            break;
}
```

Μόλις η εντολή switch βρει μια εντολή case με μια τιμή που να ταιριάζει με την τιμή της έκφρασης της switch, η PHP αρχίζει να εκτελεί τις εντολές. Η PHP συνεχίζει να εκτελεί τις εντολές μέχρι το τέλος του μπλοκ της switch ή μόλις συναντήσει μια εντολή break. Αν δεν συμπεριλάβουμε μια εντολή break στο τέλος μιας εντολής case, η PHP θα συνεχίσει να εκτελεί τις εντολές και από τις επόμενες εντολές case, όπως για παράδειγμα :

```
switch ($i) {
      case 0:
            print "Το i είναι ίσο με 0";
      case 1:
            print "Το i είναι ίσο με 1";
      case 2:
            print "Το i είναι ίσο με 2";
}
```

Εδώ, αν το $i είναι ίσο με 0, η PHP θα εκτελέσει όλες τις εντολές print, αν το $i είναι ίσο με

1, η PHP θα εκτελέσει τις δύο τελευταίες εντολές print και μόνο αν το $i είναι ίσο 2, θα έχουμε την αναμενόμενη συμπεριφορά και θα εκτελεσθεί μόνο η τελευταία εντολή print.

Μια εντολή case μπορεί να είναι κενή και έτσι ο έλεγχος να μεταβιβασθεί στην επόμενη εντολή case.

```
switch ($i) {
	case 0:
	case 1:
	case 2:
		print "Το i είναι μικρότερο από 3 αλλά όχι αρνητικό";
		break;
	case 3:
		print "Το i είναι ίσο με 3";
}
```

Μια ειδική περίπτωση case αποτελεί το default case, το οποίο ταιριάζει μ' ο,τιδήποτε δεν ταίριαξε από τις άλλες cases. Για παράδειγμα :

```
switch ($i) {
	case 0:
		print "Το i είναι ίσο με 0";
		break;
	case 1:
		print "Το i είναι ίσο με 1";
		break;
	case 2:
		print "Το i είναι ίσο με 2";
		break;
	default:
		print "Το i δεν είναι ίσο με 0, 1 ή 2";
}
```

Η εναλλακτική σύνταξη των δομών ελέγχου υποστηρίζεται και στην εντολή switch.

```
switch ($i):
	case 0:
		print "Το i είναι ίσο με 0";
		break;
	case 1:
		print "Το i είναι ίσο με 1";
		break;
	case 2:
```

```
            print "Το i είναι ίσο με 2";
            break;
        default:
            print "Το i δεν είναι ίσο με 0, 1 ή 2";
endswitch;
```

2.16.11 Η Συνάρτηση require()

Η εντολή require() αντικαθιστά τον εαυτό της μ' ένα συγκεκριμένο αρχείο, όπως ακριβώς δουλεύει δηλαδή η εντολή #include στην C. Κάτι σημαντικό που πρέπει να έχουμε υπόψη μας είναι ότι όταν ένα αρχείο χρησιμοποιείται σε μια από τις συναρτήσεις include() ή require(), η ανάλυση (parsing) ξεφεύγει από τον έλεγχο της PHP, πηγαίνει στον έλεγχο της HTML στην αρχή του αρχείου και επανέρχεται στον έλεγχο της PHP ξανά στο τέλος. Γι' αυτόν τον λόγο, ο κώδικας που υπάρχει μέσα στο αρχείο και ο οποίος πρέπει να εκτελεσθεί σαν κώδικας της PHP πρέπει να περικλείεται με έγκυρα tags αρχής και τέλους της PHP.

Η require() δεν είναι ουσιαστικά μια συνάρτηση της PHP και αποτελεί περισσότερο μια δομή της γλώσσας. Υπόκειται σε μερικούς διαφορετικούς ρόλους απ' ό,τι οι συναρτήσεις. Για παράδειγμα, η require() δεν υπόκειται σε δομές ελέγχου (control structures) και ακόμη, δεν επιστρέφει κάποια τιμή. Σ' αντίθεση με την include(), η require() θα διαβασθεί πάντα στο αρχείο (target file), ακόμη κι αν η γραμμή στην οποία βρίσκεται δεν εκτελείται ποτέ. Αν θέλουμε να συμπεριλάβουμε ένα αρχείο υπό συνθήκη, πρέπει να χρησιμοποιήσουμε τη συνάρτηση include(). Η εντολή υπό συνθήκη δεν θα επηρεάσει την require(). Όμως, αν η γραμμή στην οποία βρίσκεται η require() δεν εκτελεσθεί, δεν θα εκτελεσθεί ούτε ο κώδικας που υπάρχει στο αρχείο (target file). Παρόμοια, οι δομές βρόγχου δεν επηρεάζουν τη συμπεριφορά της require(). Αν και ο κώδικας που περιέχεται στο αρχείο (target file) υπόκειται ακόμα στον βρόγχο, η ίδια η require() εκτελείται μία μόνο φορά. Αυτό σημαίνει ότι δεν μπορούμε να τοποθετήσουμε μια εντολή require() μέσα σε μια δομή βρόγχου και να αναμένουμε να συμπεριλάβει τα περιεχόμενα ενός διαφορετικού αρχείου σε κάθε επανάληψη. Για να το κάνουμε αυτό, χρησιμοποιούμε μια εντολή include().

```
require( 'header.inc' );
```

Πρέπει να έχουμε υπόψη μας ότι και η include() και η require() στην ουσία τραβούν τα περιεχόμενα του αρχείου (target file) στο ίδιο το καλών αρχείο script (calling script file) και δεν καλούν το target μέσω HTTP ή με κάτι παρόμοιο. Έτσι, όποια μεταβλητή έχει ορισθεί στην εμβέλεια στην οποία λαμβάνει χώρα η συμπερίληψη θα είναι διαθέσιμη αυτόματα μέσα στο συμπεριλαμβανόμενο αρχείο, εφόσον έχει γίνει ουσιαστικά ένα μέρος του καλόντος αρχείου.

```
require( "file.inc?varone=1&vartwo=2" ); /* Δεν θα δουλέψει */
$varone = 1;
$vartwo = 2;
require( "file.inc" );
/* Οι $varone και $vartwo θα είναι διαθέσιμες στο file.inc */
```

2.16.12 Η Συνάρτηση include()

Η εντολή include() περιλαμβάνει και αποτιμά το καθορισμένο αρχείο. Κάτι σημαντικό που πρέπει να γνωρίζουμε για το πώς λειτουργεί αυτό, είναι ότι όταν ένα αρχείο γίνεται include() ή require(), η ανάλυση ξεφεύγει από τον έλεγχο της PHP και πηγαίνει στον έλεγχο της HTML στην αρχή του αρχείου (target file) και επανέρχεται ξανά στο τέλος. Γι' αυτόν τον λόγο, ο κώδικας που υπάρχει μέσα στο αρχείο target και ο οποίος πρέπει να εκτελεσθεί σαν κώδικας της PHP πρέπει να

περικλείεται με έγκυρα tags αρχής και τέλους τηςPHP. Αυτό συμβαίνει κάθε φορά που συναντάται η εντολή include(), έτσι μπορούμε να χρησιμοποιήσουμε μια εντολή include() μέσα σε μια δομή βρόγχου για να συμπεριλάβουμε έναν αριθμό από διαφορετικά αρχεία.

```
$files = array ('first.inc', 'second.inc', 'third.inc');
for ($i = 0; $i < count($files); $i++) {
      include $files[$i];
}
```

Η include() διαφέρει από την require() στο ότι η εντολή include αποτιμάται κάθε φορά που συναντάται (και μόνο όταν εκτελείται), ενώ η εντολή require() αντικαθίσταται από το αρχείο required όταν συναντάται για πρώτη φορά, αν τα περιεχόμενα του αρχείου θα αποτιμηθούν ή όχι. Για παράδειγμα, αν βρίσκεται μέσα σε μια εντολή if η συνθήκη της οποίας έχει αποτιμηθεί σε false. Επειδή η include() αποτελεί μια ειδική δομή γλώσσας, πρέπει να την περικλείσουμε σ' ένα μπλοκ εντολών αν βρίσκεται μέσα σ' ένα μπλοκ υπό συνθήκη (conditional block).

```
/* Αυτό είναι ΛΑΘΟΣ και δεν θα δουλέψει όπως αναμένεται */
if ($condition)
      include($file);
else
      include($other);
 /* Αυτό είναι ΣΩΣΤΟ */
if ($condition) {
      include($file);
} else {
      include($other);
}
```

Και στην PHP3 και στην PHP4, είναι δυνατό να εκτελέσουμε μια εντολή return μέσα σ' ένα αρχείο που έχει γίνει include(), ώστε να τερματίσουμε την επεξεργασία σ' αυτό το αρχείο και να επιστρέψουμε στο script που το κάλεσε. Υπάρχουν, όμως, μερικές διαφορές στον τρόπο που δουλεύει αυτό. Η πρώτη είναι ότι στην PHP3, το return δεν μπορεί να εμφανισθεί μέσα σ' ένα μπλοκ εκτός κι αν είναι ένα function block, όπου το return εφαρμόζεται σ' εκείνη τη συνάρτηση και όχι σ' ολόκληρο το αρχείο. Στην PHP4, όμως, δεν υπάρχει αυτός ο περιορισμός. Επίσης, η PHP4 μάς δίνει τη δυνατότητα να επιστρέψουμε τιμές από τα αρχεία που έχουν γίνει include(). Μπορούμε να αντιμετωπίσουμε την τιμή μιας κλήσης στην include() όπως και με μια κανονική συνάρτηση. Αυτό δημιουργεί ένα parse error στην PHP3. Ακολουθεί ένα παράδειγμα με την include() στην PHP3 και την PHP4. Υποθέτουμε ότι το αρχείο test.inc βρίσκεται στον ίδιο κατάλογο με το κύριο αρχείο :

```
<?php
      echo "Before the return <br>\n";
      if ( 1 ) {
            return 27;
      }
      echo "After the return <br>\n";
?>
```

Υποθέτουμε ότι το κύριο αρχείο (main.html) περιέχει τα εξής :

```
<?php
      $retval = include( 'test.inc' );
      echo "File returned : '$retval'<br>\n";
?>
```

Όταν κληθεί η main.html στην PHP3, θα δημιουργηθεί ένα λάθος ανάλυσης (parse error) στη γραμμή 2 και αυτό γιατί δεν μπορούμε να πάρουμε την τιμή μιας συνάρτησης include() στηνPHP3. Στην PHP4, όμως, το αποτέλεσμα θα είναι :

```
Before the return
File returned : '27'
```

Τώρα, ας υποθέσουμε ότι το main.html έχει τροποποιηθεί ώστε να περιέχει τα εξής :

```
<?php
      include( 'test.inc' );
      echo "Back in main.html<br>\n";
?>
```

Στην PHP4, η έξοδος θα είναι :

```
Before the return
Back in main.html
```

Όμως, η PHP3 θα δώσει την εξής έξοδο :

```
Before the return
27Back in main.html
Parse error: parse error in
/home/torben/public_html/phptest/main.html on line 5
```

Το παραπάνω parse error είναι ένα αποτέλεσμα του γεγονότος ότι η εντολή return περικλείεται σ' ένα non-function block μέσα στο αρχείο test.inc. Αν μετακινηθεί η εντολή return έξω από το μπλοκ, η έξοδος θα είναι η εξής :

```
Before the return
27Back in main.html
```

2.16.13 Συναρτήσεις Οριζόμενες από τον Χρήστη

Μπορούμε να ορίσουμε μια συνάρτηση (function) χρησιμοποιώντας μια σύνταξη σαν την εξής :

```
function foo ($arg_1, $arg_2, ..., $arg_n) {
      echo "Example function.\n";
      return $retval;
}
```

Μέσα σε μια συνάρτηση μπορεί να υπάρχει ένας έγκυρος κώδικας της PHP, ακόμη κι άλλες συναρτήσεις και ορισμοί τάξεων (class definitions). Στην PHP3, οι συναρτήσεις πρέπει να ορισθούν πριν γίνει αναφορά σ' αυτές, ενώ δεν απαιτείται κάτι τέτοιο στην PHP4. Η PHP δεν υποστηρίζει την υπέρβαση των συναρτήσεων (function overloading), ούτε είναι δυνατό να καταργήσουμε τον ορισμό (undefine) ή να ορίσουμε ξανά (redefine) ήδη δηλωμένες συναρτήσεις. Η PHP3 δεν υποστηρίζει μεταβλητό αριθμό ορισμάτων στις συναρτήσεις αν και υποστηρίζει τα προκαθορισμένα ορίσματα (default arguments). Η PHP4 υποστηρίζει και τα δύο.

2.16.13.1 Τα Ορίσματα των Συναρτήσεων

Μπορούμε να περάσουμε πληροφορίες σε συναρτήσεις μέσω της λίστας ορισμάτων (argument list), που είναι μια λίστα μεταβλητών ή και σταθερών χωρισμένων με κόμματα.

Η PHP υποστηρίζει τη μεταβίβαση (πέρασμα) των ορισμάτων (arguments) με τιμή (by value), που είναι το προεπιλεγμένο, τη μεταβίβαση με αναφορά (by reference) καθώς και τις προκαθορισμένες τιμές ορισμάτων (default argument values). Οι λίστες ορισμάτων μεταβλητού μήκους (variable-length argument lists) υποστηρίζονται μόνο στην PHP4. Μπορούμε να πετύχουμε ένα παρόμοιο αποτέλεσμα στην PHP3 μεταβιβάζοντας έναν πίνακα ορισμάτων σε μια συνάρτηση :

```
function takes_array($input) {
      echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
```

2.16.13.2 Μεταβίβαση Ορισμάτων με Αναφορά

Εξ ορισμού, τα ορίσματα των συναρτήσεων μεταβιβάζονται με τιμή (by value), που σημαίνει ότι αν αλλάξουμε την τιμή του ορίσματος μέσα στη συνάρτηση, δεν αλλάζει και εκτός της συνάρτησης. Αν θέλουμε να μπορεί μια συνάρτηση να τροποποιεί τα ορίσματά της, πρέπει να τα μεταβιβάσουμε με αναφορά (by reference). Αν θέλουμε ένα όρισμα σε μια συνάρτηση να μεταβιβάζεται πάντα με αναφορά, πρέπει να προσθέσουμε το σύμβολο & (ampersand) πριν από το όνομα του ορίσματος στον ορισμό της συνάρτησης :

```
function add_some_extra(&$string) {
      $string .= 'and something extra.';
}
$str = 'This is a string, ';
add_some_extra($str);
echo $str;      // εμφανίζει 'This is a string, and something
extra.'
```

Αν θέλουμε να περάσουμε μια μεταβλητή με αναφορά σε μια συνάρτηση που δεν το κάνει αυτό εξ ορισμού, θα πρέπει να προσθέσουμε τον χαρακτήρα & (ampersand) στο όνομα του ορίσματος κατά την κλήση της συνάρτησης :

```
function foo ($bar) {
      $bar .= ' and something extra.';
}
$str = 'This is a string, ';
foo ($str);
echo $str;    // εμφανίζει 'This is a string, '
foo (&$str);
echo $str;   // εμφανίζει 'This is a string, and something
extra.'
```

2.16.13.3 Προκαθορισμένες Τιμές Ορισμάτων

Μια συνάρτηση μπορεί να ορίσει προκαθορισμένες τιμές του στυλ της C++ για ορίσματα scalar, ως εξής :

```
function makecoffee ($type = "cappuccino") {
      return "Making a cup of $type.\n";
}
echo makecoffee ();
echo makecoffee ("espresso");
```

Η έξοδος από τον παραπάνω κώδικα είναι η εξής :

```
Making a cup of cappuccino.
Making a cup of espresso.
```

Η προκαθορισμένη τιμή πρέπει να είναι μια σταθερή έκφραση και όχι, για παράδειγμα, μια μεταβλητή ή ένα μέλος μιας τάξης (class member). Πρέπει να έχουμε υπόψη μας ότι όταν χρησιμοποιούμε προκαθορισμένα ορίσματα σε συναρτήσεις, αυτά θα πρέπει να βρίσκονται στα δεξιά των μη προκαθορισμένων ορισμάτων, αλλιώς δεν θα έχουμε τα αναμενόμενα αποτελέσματα. Ακολουθεί ένα παράδειγμα :

```
function makeyogurt ($type = "acidophilus", $flavour) {
      return "Making a bowl of $type $flavour.\n";
}
echo makeyogurt ("raspberry");  // δεν δουλεύει όπως περιμέναμε
```

Η έξοδος του παραπάνω παραδείγματος είναι η εξής :

```
Warning: Missing argument 2 in call to makeyogurt() in
/usr/local/etc/httpd/htdocs/php3test/functest.html on line 41
Making a bowl of raspberry.
```

Τώρα, συγκρίνετε το παραπάνω με το εξής :

```
function makeyogurt ($flavour, $type = "acidophilus") {
      return "Making a bowl of $type $flavour.\n";
}
echo makeyogurt ("raspberry"); // δουλεύει όπως περιμέναμε
```

Η έξοδος αυτού του παραδείγματος είναι η εξής :

```
Making a bowl of acidophilus raspberry.
```

2.16.13.4 Επιστρεφόμενες Τιμές Συναρτήσεων

Οι τιμές των συναρτήσεων επιστρέφονται με την προαιρετική εντολή return. Όλοι οι τύποι δεδομένων μπορούν να επιστραφούν, ανάμεσά τους οι λίστες (lists) και τα αντικείμενα (objects).

```
function square ($num) {
      return $num * $num;
```

```
}
echo square (4);  // εμφανίζει '16'
```

Δεν μπορούμε να επιστρέψουμε πολλαπλές τιμές από μια συνάρτηση, αλλά μπορούμε να πετύχουμε ένα παρόμοιο αποτέλεσμα αν επιστρέψουμε μια λίστα (list).

```
function small_numbers() {
      return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
```

2.16.13.5 Οι Μεταβλητές Συναρτήσεις

Η PHP υποστηρίζει την έννοια των μεταβλητών συναρτήσεων (variable functions). Αυτό σημαίνει ότι αν ένα όνομα μεταβλητής έχει παρενθέσεις, η PHP θα αναζητήσει μια συνάρτηση που έχει το ίδιο όνομα με την τιμή της μεταβλητής και θα προσπαθήσει να την εκτελέσει.

```
<?php
      function foo() {
            echo "In foo()<br>\n";
      }

      function bar( $arg = '' ) {
            echo "In bar(); argument was '$arg'.<br>\n";
      }
      $func = 'foo';
      $func();
      $func = 'bar';
      $func( 'test' );
?>
```

2.16.14 Τάξεις και Αντικείμενα

Μια τάξη (class) είναι μια συλλογή από μεταβλητές και από συναρτήσεις που εφαρμόζονται σ' αυτές τις μεταβλητές. Για να ορίσουμε μια τάξη χρησιμοποιούμε την εξής σύνταξη :

```
<?php
      class Cart {
            var $items;  // Τα items στο shopping cart
            // Προσθέτουμε $num προϊόντα του $artnr στο cart
            function add_item ($artnr, $num) {
                  $this->items[$artnr] += $num;
```

```
        }
        // Αφαιρούμε $num προϊόντα του $artnr από το cart
        function remove_item ($artnr, $num) {
            if ($this->items[$artnr] > $num) {
                $this->items[$artnr] -= $num;
                return true;
            } else {
                return false;
            }
        }
    }
?>
```

Αυτό ορίζει μια τάξη με όνομα Cart η οποία αποτελείται από έναν πίνακα προϊόντων στο cart (καλάθι αγορών) και δύο συναρτήσεις για να μπορούμε να προσθέσουμε και να αφαιρέσουμε προϊόντα από το cart. Οι τάξεις είναι τύποι και μπορούμε να δημιουργήσουμε μια μεταβλητή ενός συγκεκριμένου τύπου με τον τελεστή new, ως εξής :

```
$cart = new Cart;
$cart->add_item("10", 1);
```

Ο παραπάνω κώδικας δημιουργεί ένα αντικείμενο με όνομα $cart από την τάξη Cart. Η συνάρτηση add_item() αυτού του αντικειμένου καλείται για να προσθέσει ένα προϊόν με κωδικό αριθμό 10 στο cart. Οι τάξεις μπορεί να είναι επεκτάσεις (extensions) άλλων τάξεων. Η προκύπτουσα τάξη έχει όλες τις μεταβλητές και τις συναρτήσεις της βασικής τάξης και ό,τι προσθέσουμε εμείς. Αυτό μπορούμε να το κάνουμε με τη λέξη κλειδί extends. Δεν υποστηρίζεται η πολλαπλή κληρονομικότητα (multiple inheritance).

```
class Named_Cart extends Cart {
    var $owner;
    nction set_owner ($name) {
        $this->owner = $name;
    }
}
```

Ο παραπάνω κώδικας ορίζει μια τάξη με όνομα Named_Cart που έχει όλες τις μεταβλητές και τις συναρτήσεις της τάξης Cart συν μια επιπλέον μεταβλητή με όνομα $owner και μια επιπλέον συνάρτηση με όνομα set_owner(). Μπορούμε τώρα να ορίσουμε και να μάθουμε τον ιδιοκτήτη (owner) ενός cart.

```
$ncart = new Named_Cart;       // Δημιουργία ενός named cart
$ncart->set_owner ("kris");    // Όνομα του ιδιοκτήτη του cart
print $ncart->owner;
// εκτύπωση του ονόματος του ιδιοκτήτη του cart
$ncart->add_item ("10", 1);
// μια συνάρτηση που την έχει κληρονομήσει από το cart
```

Μέσα στις συναρτήσεις μιας τάξης, η μεταβλητή $this σημαίνει το ίδιο το αντικείμενο.

Μπορούμε να χρησιμοποιήσουμε τη σύνταξη $this->something για να έχουμε πρόσβαση σε μια οποιαδήποτε μεταβλητή ή συνάρτηση με όνομα something του τρέχοντος αντικειμένου.

Οι δημιουργοί (constructors) είναι συναρτήσεις σε μια τάξη που καλούνται αυτόματα όταν δημιουργούμε ένα νέο στιγμιότυπο (instance) μιας τάξης. Μια συνάρτηση γίνεται δημιουργός (constructor) όταν έχει το ίδιο όνομα με την τάξη.

```
class Auto_Cart extends Cart {
      function Auto_Cart () {
            $this->add_item ("10", 1);
      }
}
```

Ο παραπάνω κώδικας δημιουργεί μια τάξη με όνομα Auto_Cart που είναι μια επέκταση της τάξης Cart συν έναν δημιουργό (constructor) ο οποίος αρχικοποιεί το cart μ' ένα στοιχείο του προϊόντος που έχει κωδικό αριθμό 10 κάθε φορά που δημιουργείται ένα νέο Auto_Cart με τον τελεστή new. Οι δημιουργοί μπορούν επίσης να λάβουν ορίσματα και αυτά τα ορίσματα μπορεί να είναι προαιρετικά.

```
class Constructor_Cart extends Cart {
      function Constructor_Cart ($item = "10", $num = 1) {
            $this->add_item ($item, $num);
      }
}
// Shop the same old boring stuff.
$default_cart   = new Constructor_Cart;
// Shop for real ...
$different_cart = new Constructor_Cart ("20", 17);
```

2.16.15 Δημιουργία Εικόνων Gif

Η PHP δεν περιορίζεται στο να δημιουργεί μόνο μια έξοδο κώδικα της HTML. Μπορεί επίσης να χρησιμοποιηθεί για να δημιουργήσει αρχεία εικόνων GIF ή και ροές (streams) εικόνωνGIF. Θα χρειασθεί να μεταγλωττίσουμε την PHP με τη βιβλιοθήκη GD των συναρτήσεων εικόνας για να μπορέσει να δουλέψει αυτό.

```
<?php
      Header("Content-type: image/gif");
      $string=implode($argv," ");
      $im = imagecreatefromgif("images/button1.gif");
      $orange = ImageColorAllocate($im, 220, 210, 60);
      $px = (imagesx($im)-7.5*strlen($string))/2;
      ImageString($im, 3, $px, 9, $string, $orange);
      ImageGif($im);
      ImageDestroy($im);
```

```
?>
```

Αυτό το παράδειγμα θα κληθεί από μια σελίδα μ' ένα tag σαν το εξής :

```
<img src="button.php3?text">
```

Το παραπάνω script button.php3 λαμβάνει μετά αυτό το string "text" και το τοποθετεί στην κορυφή μιας εικόνας βάσης που είναι η "images/ button1.gif" και εξάγει την εικόνα που δημιουργείται.

Αυτό αποτελεί έναν πολύ βολικό τρόπο για να αποφύγουμε τη σχεδίαση νέων εικόνων για πλήκτρα εντολής κάθε φορά που θέλουμε να αλλάξουμε το κείμενο ενός πλήκτρου. Με τη μέθοδο αυτή δημιουργούνται δυναμικά.

2.16.16 Επικύρωση (Authentication) του HTTP με την PHP

Η επικύρωση (authentication) του HTTP στην PHP είναι διαθέσιμη μόνο όταν εκτελείται σαν ένα Apache module και δεν είναι συνεπώς διαθέσιμη στο CGI. Σ' ένα script της PHP σ' έναApache module, μπορούμε να χρησιμοποιήσουμε τη συνάρτηση Header() για να στείλουμε ένα μήνυμα "Authentication Required" στον φυλλομετρητή του χρήστη (πελάτη) για να εμφανίσει (popup) ένα παράθυρο εισόδου Username/Password. Αφού ο χρήστης έχει καταχωρήσει ένα username και ένα password, θα κληθεί ξανά το URL που περιέχει το PHP script με τις μεταβλητές $PHP_AUTH_USER, $PHP_AUTH_PW και $PHP_AUTH_TYPE που περιέχουν αντίστοιχα το user name, το password και τον τύπο επικύρωσης (authentication type). Ένα παράδειγμα script το οποίο θα επέβαλε την επικύρωση του πελάτη (client authentication) σε μια σελίδα, είναι το εξής :

```
<?php
    if(!isset($PHP_AUTH_USER)) {
        Header("WWW-Authenticate : Basic realm=\"My Realm\"");
        Header("HTTP/1.0 401 Unauthorized");
        echo "Κείμενο που θα σταλεί αν πατηθεί το Cancel\n";
        exit;
    } else {
        echo "Hello $PHP_AUTH_USER <P>";
        echo "Καταχωρήσατε το $PHP_AUTH_PW για password <P>";
    }
?>
```

Αντί να εκτυπώσουμε απλά τα $PHP_AUTH_USER και $PHP_AUTH_ PW, θα θέλαμε πιθανώς να ελέγξουμε αν είναι έγκυρα τα username και password. Αυτό μπορεί να γίνει στέλνοντας ένα ερώτημα (query) σε μια βάση δεδομένων ή αναζητώντας τον χρήστη σ' ένα αρχείο dbm. Για να μπορέσουμε να εμποδίσουμε κάποιον από το να γράψει ένα script που να αποκαλύπτει το password μιας σελίδας που έχει επικυρωθεί (authenticated) μέσω ενός παραδοσιακού εξωτερικού μηχανισμού, οι μεταβλητές PHP_AUTH δεν θα ορισθούν αν είναι ενεργοποιημένη η εξωτερική επικύρωση (external authentication) γι' αυτή τη συγκεκριμένη σελίδα. Σ' αυτήν την περίπτωση, η μεταβλητή $REMOTE_USER μπορεί να χρησιμοποιηθεί για να αναγνωρισθεί ο εξωτερικά επικυρωμένος χρήστης. Παρ' όλα αυτά, όμως, τα παραπάνω δεν αποτρέπουν κάποιον που ελέγχει ένα μη επικυρωμένο (non-authenticated) URL από το να κλέβει passwords από επικυρωμένα URLs στον ίδιοserver. Ακολουθεί ένα παράδειγμα επικύρωσης (authentication) HTTP που επιβάλει ένα νέο name/password.

```
<?php
      function  authenticate()  {
Header( "WWW-authenticate:  basic  realm='Test  Authentication
            System'");
            Header( "HTTP/1.0  401  Unauthorized");
            echo  "Πρέπει να καταχωρήσετε ένα έγκυρο login  ID και
                  password για  να  έχετε  πρόσβαση  σ'  αυτά  τα
            στοιχεία\n";
            exit;
      }
      if(!isset($PHP_AUTH_USER)  ||  ($SeenBefore ==  1  &&
            !strcmp($OldAuth,  $PHP_AUTH_USER))  )  {
            authenticate();
      }
      else  {
            echo  "Welcome:  $PHP_AUTH_USER<BR>";
            echo  "Old:  $OldAuth";
            echo  "<FORM  ACTION=\"$PHP_SELF\"
                  METHOD=POST>\n";
            echo  "<INPUT  TYPE=HIDDEN  NAME=\"SeenBefore\"
                  VALUE=\"1\">\n";
            echo  "<INPUT  TYPE=HIDDEN  NAME=\"OldAuth\"
                  VALUE=\"$PHP_AUTH_USER\">\n";
            echo  "<INPUT  TYPE=Submit
                  VALUE=\"Re  Authenticate\">\n";
            echo  "</FORM>\n";
      }
?>
```

2.16.17 Τα Cookies

Η PHP υποστηρίζει καθαρά τα HTTP cookies, τα οποία είναι ένας μηχανισμός αποθήκευσης δεδομένων στον απομακρυσμένο φυλλομετρητή για να μπορούμε έτσι να παρακολουθούμε ή να αναγνωρίζουμε τους χρήστες. Μπορούμε να ορίσουμε cookies με τη συνάρτηση setcookie(). Τα cookies αποτελούν μέρος της επικεφαλίδας (header) του HTTP και έτσι η συνάρτηση setcookie() πρέπει να κληθεί πριν σταλεί κάποια έξοδος στον φυλλομετρητή. Αυτός είναι ο ίδιος περιορισμός που έχει και η συνάρτηση header(). Τα cookies που στέλνονται σε μας από τον πελάτη (client) μετατρέπονται αυτόματα σε μια μεταβλητή της PHP όπως ακριβώς συμβαίνει με τις μεθόδους GET και POST. Αν θελήσουμε να εκχωρήσουμε πολλαπλές τιμές σ' ένα μόνο cookie, προσθέτουμε τα [ ] στο όνομα του cookie.

2.16.18 Uploads με τη Μέθοδο POST

Η PHP μπορεί να λάβει uploads αρχείων από έναν συμβατό φυλλομετρητή με το RFC-1867, όπως είναι ο Netscape Navigator και ο Microsoft Internet Explorer. Με τη δυνατότητα αυτή μπορεί κάποιος να κάνει upload και κείμενο (text) και δυαδικά αρχεία (binary files). Με τις συναρτήσεις για επικύρωση της PHP και χειρισμό αρχείων, έχουμε πλήρη έλεγχο για το ποιος έχει το δικαίωμα να κάνει upload και το τι πρέπει να γίνει με το αρχείο αφού έχει γίνει upload. Μπορούμε να δημιουργήσουμε μια φόρμα ειδικά για upload αρχείων, ως εξής :

```
<FORM ENCTYPE="multipart/form-data" ACTION="_URL_"
        METHOD=POST>
    <INPUT TYPE="hidden" name="MAX_FILE_SIZE"
        value="1000">
    Send this file : <INPUT NAME="userfile" TYPE="file">
    <INPUT TYPE="submit" VALUE="Send File">
</FORM>
```

Το _URL_ πρέπει να δείχνει σ' ένα αρχείο της PHP, ενώ το κρυμμένο πεδίο MAX_FILE_SIZE πρέπει να προηγείται από το πεδίο file input και η τιμή του είναι το μέγιστο αποδεκτό μέγεθος αρχείου (maximum filesize) σε bytes.

Σ' αυτό το αρχείο προορισμού, ορίζονται οι ακόλουθες μεταβλητές για ένα επιτυχημένο upload :

- *$userfile*, είναι το προσωρινό όνομα αρχείου με το οποίο αποθηκεύθηκε το αρχείο που έγινε upload στο μηχάνημα του server.
- *$userfile_name*, είναι το αρχικό όνομα του αρχείου στο σύστημα του αποστολέα.
- *$userfile_size*, είναι το μέγεθος του αρχείου που γίνεται upload σε bytes.
- *$userfile_type*, είναι ο τύπος mime του αρχείου, όπως image/gif.

Πρέπει να έχουμε υπόψη μας ότι το τμήμα $userfile των παραπάνω μεταβλητών είναι το ίδιο με το όνομα του πεδίου INPUT που έχει τύπο TYPE=file στη φόρμα upload, όπως userfilesto παραπάνω παράδειγμα. Τα αρχεία εξ ορισμού αποθηκεύονται στον προκαθορισμένο προσωρινό κατάλογο του server (temporary directory). Αυτό μπορεί να αλλάξει ορίζοντας τη μεταβλητή περιβάλλοντοςTMPDIR στο περιβάλλον στο οποίο εκτελείται η PHP. Αν την ορίσουμε χρησιμοποιώντας τη συνάρτηση putenv() μέσα από ένα script της PHP δεν θα δουλέψει. Το script της PHP το οποίο λαμβάνει το αρχείο που γίνεται upload θα εφαρμόσει όποια λογική είναι απαραίτητη για να καθορίσει το τι πρέπει να γίνει με το αρχείο που γίνεται upload. Μπορούμε, για παράδειγμα, να χρησιμοποιήσουμε τη μεταβλητή $file_size για να απορρίψουμε όλα τα αρχεία που είναι είτε πάρα πολύ μικρά ή πάρα πολύ μεγάλα. Μπορούμε να χρησιμοποιήσουμε τη μεταβλητή $file_type για να απορρίψουμε όλα τα αρχεία που δεν ταιριάζουν με κάποια συγκεκριμένα κριτήρια. Θα πρέπει είτε να διαγράψουμε το αρχείο από τον προσωρινό κατάλογο ή να το μετακινήσουμε κάπου αλλού ή να το μετονομάσουμε.

2.16.19 Uploading Πολλών Αρχείων

Μπορούμε να κάνουμε upload πολλά αρχεία ταυτόχρονα και να έχουμε τις πληροφορίες αυτόματα οργανωμένες σε πίνακες (arrays). Για να γίνει αυτό, πρέπει να χρησιμοποιήσουμε την ίδια σύνταξη υποβολής πίνακα στη φόρμα της HTML όπως κάνουμε με τις λίστες επιλογής και τα πλαίσια ελέγχου :

```
<form action="file-upload.html" method="post"
```

```
        enctype="multipart/form-data">
        Send these files : <br>
        <input name="userfile[]" type="file"><br>
        <input name="userfile[]" type="file"><br>
        <input type="submit" value="Send files">
</form>
```

Όταν υποβάλλεται η παραπάνω φόρμα, θα σχηματισθούν οι πίνακες userfile, $userfile_name και $userfile_size. Ο καθένας απ' αυτούς θα είναι ένας πίνακας με αριθμητικούς δείκτες. Για παράδειγμα, ας υποθέσουμε ότι υποβάλλονται τα αρχεία /home/test/review.html και /home/test/xwp.out. Σ' αυτήν την περίπτωση, το $userfile_name[0] θα περιέχει την τιμήreview.html και το $userfile_name[1] θα περιέχει την τιμή xwp.out. Παρόμοια, το $userfile_size[0] θα περιέχει το μέγεθος αρχείου του review.html κοκ.

2.16. 20 Υποστήριξη για τη Μέθοδο PUT

Η PHP παρέχει υποστήριξη για τη μέθοδο PUT του HTTP που χρησιμοποιείται από πελάτες (clients) όπως ο Netscape Composer και ο W3C Amaya. Οι αιτήσεις (requests) της PUT είναι πολύ απλούστερες από το upload ενός αρχείου και μοιάζουν ως εξής :

```
PUT /path/filename.html HTTP/1.1
```

Αυτό σημαίνει συνήθως ότι ο απομακρυσμένος πελάτης θα ήθελε να αποθηκεύσει τα περιεχόμενα ως εξής : /path/filename.html. Δεν αποτελεί προφανώς μια καλή ιδέα για το Apache ή την PHP να επιτρέπουν στον οποιονδήποτε να επικαλύπτει αρχεία στο δικό μας web tree.

Έτσι, για να χειριστούμε μια τέτοια αίτηση (request) πρέπει πρώτα να πούμε στον web server ότι θέλουμε ένα συγκεκριμένο script της PHP να χειριστεί την αίτηση. Στο Apache το κάνουμε αυτό με την οδηγία (directive) Script, που μπορεί να τοποθετηθεί οπουδήποτε στο αρχείο σύνθεσης (configuration file) του Apache :

```
Script PUT /put.php3
```

Το παραπάνω λέει στο Apache να στείλει όλες τις αιτήσεις PUT για τα URIs που ταιριάζουν στο περιεχόμενο με το script put.php3. Αυτό προϋποθέτει, φυσικά, ότι έχουμε ενεργοποιήσει την PHP για την επέκταση .php3 και ότι είναι ενεργή η PHP.

Μέσα στο αρχείο put.php3, θα γράψουμε κάτι σαν το εξής :

```
<? copy($PHP_UPLOADED_FILE_NAME,
        $DOCUMENT_ROOT.$REQUEST_URI); ?>
```

Το παραπάνω θα αντιγράψει το αρχείο στην τοποθεσία που ζητείται από τον απομακρυσμένο πελάτη. Θα θέλουμε πιθανώς να κάνουμε κάποιους ελέγχους ή/και να επικυρώσουμε τον χρήστη πριν να λάβει χώρα αυτή η αντιγραφή του αρχείου. Το μόνο τρυκ εδώ είναι ότι όταν η PHP βλέπει μια αίτηση με την μέθοδο PUT, αποθηκεύει το αρχείο που γίνεται upload σ' ένα προσωρινό αρχείο όπως ακριβώς εκείνα που χειρίζονται από την μέθοδο POST. Όταν τελειώσει η αίτηση (request), διαγράφεται αυτό το προσωρινό αρχείο. Έτσι, το script της PHP που θα γράψουμε για την μέθοδο PUT πρέπει να αντιγράψει κάπου αλλού αυτό το αρχείο.

Το όνομα αυτού του προσωρινού αρχείου βρίσκεται στην μεταβλητή $PHP_PUT_FILENAME και μπορούμε να βρούμε το προτεινόμενο όνομα για το αρχείο προορισμού στην$REQUEST_URI, που είναι αυτό που καθόρισε ο απομακρυσμένος χρήστης, αλλά δεν είμαστε υποχρεωμένοι να το χρησιμοποιήσουμε. Θα μπορούσαμε, για παράδειγμα, να αντιγράψουμε όλα τα αρχεία που γίνονται upload σ' έναν ειδικό κατάλογο uploads.

2.16.21 Χρήση Απομακρυσμένων Αρχείων

Μπορούμε να ανοίξουμε ένα αρχείο σ' έναν απομακρυσμένο web server, αν αναλύσουμε (parse) την έξοδο για τα δεδομένα που θέλουμε και μετά να χρησιμοποιήσουμε αυτά τα δεδομένα σ' ένα ερώτημα μιας βάσης δεδομένων (database query) ή να τα εξάγουμε στο δικό μας website. Το επόμενο παράδειγμα εμφανίζει τον τίτλο (title) μιας απομακρυσμένης σελίδας.

```
<?php
      $file = fopen("http://www.php.net/", "r");
      if (!$file) {
            echo "<p>Unable to open remote file.\n";
            exit;
      }
      while (!feof($file)) {
            $line = fgets($file, 1024);
            /* This only works if the title and its tags are on one
      line. */
            if (eregi("<title>(.*)</title>", $line, $out)) {
                  $title = $out[1];
                  break;
            }
      }
      fclose($file);
?>
```

Το επόμενο παράδειγμα αποθηκεύει δεδομένα σ' έναν απομακρυσμένο server.

```
<?php
      $file = fopen("ftp://ftp.php.net/incoming/outputfile", "w");
      if (!$file) {
            echo "<p>Unable to open remote file for writing.\n";
            exit;
      }
      /* Εδώ καταχωρούμε τα δεδομένα */
      fputs($file, "$HTTP_USER_AGENT\n");
      fclose($file);
?>
```

ΚΕΦΑΛΑΙΟ 3ο

MySQL

Εικ. 3.1 : Λογότυπο της MySQL

3.1 Τι Είναι οι Βάσεις Δεδομένων (Databases)

Μια βάση δεδομένων (database) αποτελείται από έναν ή περισσότερους πίνακες (tables), ο καθένας από τους οποίους περιέχει μια λίστα από κάποια πράγματα. Για μια βάση δεδομένων πελατών (clients), είναι φυσικό να ξεκινήσουμε μ' έναν πίνακα με όνομα clients που θα περιέχει μια λίστα από στοιχεία πελατών.

Ο κάθε πίνακας σε μια βάση δεδομένων περιέχει μια ή περισσότερες στήλες (columns) ή πεδία (fields), όπου η κάθε στήλη περιέχει μια συγκεκριμένη πληροφορία για τον κάθε πελάτη που υπάρχει στην βάση δεδομένων (database). Ο πίνακας clients μπορεί να περιέχει στήλες για τον κωδικό ενός πελάτη (ID), για το όνομά του (Name) καθώς και για την ημερομηνία γέννησής του (Date). Ο κάθε πελάτης που αποθηκεύουμε σ' αυτόν τον πίνακα λέμε ότι αποτελεί μια γραμμή (row)ή μια εγγραφή (record) του πίνακα. Για παράδειγμα, ας δούμε τον παρακάτω πίνακα :

ID	Name	Date
1	Αντωνιάδης	1970-04-01
2	Παπαδόπουλος	1968-02-22

***Πίνακ.8**: Παράδειγμα 1 - Βάση Δεδομένων*

Εκτός από τις στήλες για το όνομα του πελάτη (Name) και την ημερομηνία γέννησής του (Date), υπάρχει και μια στήλη με όνομα ID, ο σκοπός της οποίας είναι να εκχωρήσει έναν μοναδικό αριθμό στον κάθε πελάτη έτσι ώστε να έχουμε έναν εύκολο τρόπο αναφοράς σ' αυτόν και να μπορούμε να τον ξεχωρίσουμε από τους άλλους πελάτες. Σαν επισκόπηση, το παραπάνω είναι ένας πίνακας τριών στηλών που περιέχει δύο γραμμές ή καταχωρήσεις. Η κάθε γραμμή του πίνακα περιέχει έναν κωδικό (ID) αναγνώρισης του πελάτη, το όνομά του (text) καθώς και την ημερομηνία γέννησής του (date). Με βάση αυτήν την βασική ορολογία, είμαστε έτοιμοι να αρχίσουμε να χρησιμοποιούμε την MySQL.

3.2 Εκκίνηση (Logging onto) της MySQL

Το standard interface για να δουλέψουμε με τις βάσεις δεδομένων της MySQL είναι να συνδεθούμε με το λογισμικό του MySQL server και να δίνουμε μία εντολή την φορά. Για να κάνουμε αυτήν την σύνδεση με τον server, θα χρειασθούμε τοπρόγραμμα πελάτη (client program) της MySQL. Στο Linux, το πρόγραμμα αποκαλείται mysql και βρίσκεται εξ ορισμού στον κατάλογο /usr/local/mysql/bin, ενώ στα Windows, το πρόγραμμα αποκαλείται mysql.exe και βρίσκεται εξ ορισμού στον κατάλογο C:\mysql\bin.

Υπάρχουν δύο τρόποι για να μπορέσουμε να συνδεθούμε με τον MySQL server. Ο πρώτος είναι να χρησιμοποιήσουμε το telnet για να συνδεθούμε (log into) στον server του Web host που μας φιλοξενεί και να δώσουμε την εντολή mysql από εκεί. Ο δεύτερος είναι να φορτώσουμε (download) και να εγκαταστήσουμε το λογισμικό πελάτη (client software) της MySQL από το site http://www.mysql.com/ στον δικό μας υπολογιστή και να το χρησιμοποιήσουμε για να συνδεθούμε με τον MySQL server. Οποια μέθοδο κι αν επιλέξουμε και όποιο λειτουργικό σύστημα χρησιμοποιούμε, θα καταλήξουμε σε μια γραμμή εντολών (command line), έτοιμοι να εκτελέσουμε το πρόγραμμα πελάτη της MySQL για να συνδεθούμε στον MySQL server. Πρέπει να γράψουμε τα εξής :

```
mysql -h <hostname> -u <username> -p
```

Θα πρέπει να αντικαταστήσουμε το <hostname> με το όνομα του host ή την IP διεύθυνση του υπολογιστή στον οποίο εκτελείται ο MySQL server. Αν εκτελούμε το πρόγραμμα πελάτη στον ίδιο υπολογιστή με τον server, μπορούμε να παραλείψουμε το τμήμα -h <hostname> της εντολής αντί να γράψουμε -h localhost, για παράδειγμα. Το <username> πρέπει να είναι το δικό μας όνομα χρήστη στην MySQL. Αν εγκαταστήσαμε εμείς οι ίδιοι τον MySQL server, αυτό θα είναι το root, ενώ αν χρησιμοποιούμε τον MySQL server του Web host που μας φιλοξενεί, αυτό θα πρέπει να είναι το όνομα χρήστη της MySQL που μας έχει δοθεί. Το όρισμα -p λέει στο πρόγραμμα να ζητήσει από μας τον κωδικό εισόδου (password), το οποίο θα συμβεί μόλις δώσουμε την παραπάνω εντολή. Αν έχουμε εγκαταστήσει εμείς οι ίδιοι τον MySQL, αυτό το password θα είναι το root password που επιλέξαμε εμείς, ενώ αν χρησιμοποιούμε τον MySQL server του Web host που μας φιλοξενεί, αυτό θα πρέπει να είναι το password της MySQL που μας έχει δοθεί. Αν τα γράψαμε όλα σωστά, το πρόγραμμα πελάτη της MySQL θα παρουσιάσει τον εαυτό του και θα εμφανίσει την γραμμή εντολών της MySQL, ως εξής :

```
mysql>
```

Τώρα, ο MySQL server είναι σε θέση να παρακολουθεί περισσότερες από μία βάσεις δεδομένων, που αυτό σημαίνει ότι Web host μπορεί να στήσει έναν μόνο MySQL server για να χρησιμοποιηθεί από πολλούς από τους συνδρομητές του. Ετσι, το επόμενο βήμα μας θα πρέπει να είναι να επιλέξουμε την βάση δεδομένων με την οποία θα δουλέψουμε. Πρώτα απ' όλα, θα δούμε μια λίστα των βάσεων δεδομένων που υπάρχουν στον τρέχοντα server. Δίνουμε την επόμενη εντολή και μετά ENTER.

```
mysql> SHOW DATABASES;
```

Η MySQL θα εμφανίσει μια λίστα με τις βάσεις δεδομένων που υπάρχουν στον server, ως εξής :

```
Database

mysql

       test
```

```
2 rows in set (0.11 sec)
```

Ο MySQL server χρησιμοποιεί την πρώτη βάση δεδομένων, με όνομα mysql, για να μπορεί να παρακολουθεί τους χρήστες, τα συνθηματικά τους (passwords) καθώς και το τι επιτρέπεται να κάνουν. Θα αφήσουμε για λίγο αυτή την βάση δεδομένων. Η δεύτερη βάση δεδομένων, με όνομα test αποτελεί ένα δείγμα βάσης δεδομένων. Η διαδικασία της διαγραφής στην MySQL αποκαλείται dropping (απόρριψη) και η εντολή για να διαγράψουμε μια βάση δεδομένων είναι η εξής :

```
mysql> DROP DATABASE test;
```

Αν δώσουμε αυτήν την εντολή και πατήσουμε Enter, η MySQL θα διαγράψει την βάση δεδομένων και θα εμφανίσει το μήνυμαQuery OK σαν επιβεβαίωση. Επειδή αυτή η εντολή δεν εμφανίζει κάποιο μήνυμα προειδοποίησης, πρέπει να είμαστε πολύ προσεκτικοί όταν την δίνουμε. Θα δούμε τώρα λίγα πράγματα για την γραμμή εντολών (command line) της MySQL. Όλες οι εντολές στην MySQL τελειώνουν με τον χαρακτήρα ; (semicolon). Έτσι, αν έχουμε ξεχάσει να κλείσουμε μια εντολή με τον χαρακτήρα ;, η MySQL θα νομίζει ότι δεν έχουμε τελειώσει με την εντολή αυτή και θα περιμένει να συνεχίσουμε να γράφουμε και στην επόμενη γραμμή :

```
mysql> SHOW

-> DATABASES;
```

Η MySQL δείχνει ότι περιμένει από μας να ολοκληρώσουμε την εντολή, αλλάζοντας την προτροπή (prompt) από mysql> σε ->. Αυτό είναι βολικό όταν έχουμε να γράψουμε μακροσκελείς εντολές, καθώς μπορούμε να επεκτείνουμε τις εντολές μας σε πολλές γραμμές. Για να ακυρώσουμε την τρέχουσα εντολή και να αρχίσουμε να την γράφουμε από την αρχή, γράφουμε τους χαρακτήρες \c και πατάμε ENTER, ως εξής :

```
mysql> DROP DATABASE\c

mysql>
```

Η MySQL θα αγνοήσει την εντολή που είχαμε ξεκινήσει και θα περιμένει να δώσουμε μια άλλη εντολή. Τέλος, αν θέλουμε να εξέλθουμε από το πρόγραμμα πελάτη της MySQL, μπορούμε απλά να γράψουμε quit ή exit. Είναι οι μόνες εντολές που δεν χρειάζονται τον χαρακτήρα ; (semicolon).

```
mysql> quit

Bye
```

3.3 Τι Είναι η SQL

Το σύνολο των εντολών που θα χρησιμοποιούμε από δω και πέρα για να λέμε στην MySQL τι να κάνει, αποτελεί μέρος ενός standard που αποκαλείται Δομημένη Γλώσσα Ερωτημάτων (Structured Query Language) ή SQL. Οι εντολές της SQL αποκαλούνται επίσης και ερωτήματα (queries). Η SQL αποτελεί την standard γλώσσα για αλληλεπίδραση με τις περισσότερες βάσεις δεδομένων, έτσι ακόμα κι αν αλλάξουμε στο μέλλον από την MySQL σε μια βάση δεδομένων όπως την Microsoft SQL Server, θα διαπιστώσουμε ότι οι περισσότερες από τις εντολές είναι ολόιδιες. Δεν πρέπει να συγχέουμε την SQL με την MySQL. Η MySQL είναι το λογισμικό του διακομιστή βάσεων δεδομένων (database server software) που χρησιμοποιούμε, ενώ η SQL είναι η γλώσσα που χρησιμοποιούμε για

να αλληλεπιδράσουμε με την βάση δεδομένων.

3.4 Δημιουργία μιας Βάσης Δεδομένων (DataBase)

Η δημιουργία μιας βάσης δεδομένων είναι πολύ εύκολη υπόθεση :

```
mysql> CREATE DATABASE clients;
```

Τώρα που έχουμε μια βάση δεδομένων, πρέπει να ενημερώσουμε την MySQL ότι θέλουμε να την χρησιμοποιήσουμε. Η εντολή αυτή είναι η εξής :

```
mysql> USE clients;
```

Μπορούμε τώρα να αρχίσουμε να χρησιμοποιούμε την βάση δεδομένων.

3.5 Δημιουργία ενός Πίνακα (Table)

Η σύνταξη της εντολής SQL για την δημιουργία ενός πίνακα (table) είναι η εξής :

```
mysql> CREATE TABLE <table name> (
-> <column 1 name> <col. 1 type> <col. 1 details>,
-> <column 2 name> <col. 2 type> <col. 2 details>,
-> ...
-> );
```

Οσον αφορά το παράδειγμα που είχαμε δει νωρίτερα με τον πίνακα Jokes, είχε τις εξής τρεις στήλες (columns) : ID (αριθμός, number), Name (το όνομα του πελάτη) και Date (την ημερομηνία γέννησης του πελάτη).

Η εντολή για να δημιουργήσουμε αυτόν τον πίνακα είναι η εξής :

```
mysql> CREATE TABLE Clients (
-> ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
-> Name TEXT,
-> Date DATE NOT NULL
-> );
```

Ας το αναλύσουμε τώρα : Η πρώτη γραμμή είναι αρκετά απλή : λέει ότι θέλουμε να δημιουργήσουμε έναν νέο πίνακα με όνομα Clients. Η δεύτερη γραμμή λέει ότι θέλουμε μια στήλη (column) με όνομα ID που θα περιέχει μια ακέραια τιμή (integer, INT). Ακόμη, αυτή η στήλη δεν μπορεί να είναι κενή (NOT NULL). Επίσης, αν δεν καθορίσουμε κάποια συγκεκριμένη τιμή, όταν κάνουμε μια νέα καταχώρηση, η MySQL θα επιλέξει η ίδια μια τιμή που θα είναι κατά ένα μεγαλύτερη από την μεγαλύτερη τιμή του πίνακα μέχρι τώρα (AUTO_INCREMENT). Τέλος, αυτή η στήλη θα ενεργεί σαν ένα μοναδικό αναγνωριστικό (unique identifier) για τις καταχωρήσεις του πίνακα, έτσι όλες οι τιμές αυτής της στήλης θα πρέπει να είναι μοναδικές (PRIMARY KEY). Η τρίτη γραμμή είναι πολύ απλή : λέει ότι θέλουμε μια στήλη με όνομα Name που θα περιέχει κείμενο

(TEXT). Η τέταρτη γραμμή ορίζει την τελευταία στήλη, με όνομα Date, η οποία θα περιέχει δεδομένα του τύπου DATE και η οποία δεν θα μπορεί να είναι κενή (NOT NULL). Πρέπει να έχουμε υπόψη μας ότι, ενώ μπορούμε να γράψουμε τις εντολές της SQL με πεζά ή με κεφαλαία γράμματα, ένας MySQL server που εκτελείται σ' ένα σύστημα που είναι βασισμένο στο Unix, θα ξεχωρίζει τα πεζά από τα κεφαλαία γράμματα (case sensitive) όσον αφορά τα ονόματα των βάσεων δεδομένων και των πινάκων, εφόσον αυτά αντιστοιχούν σε καταλόγους (directories) και αρχεία (files) στον κατάλογο δεδομένων (data directory) της MySQL.

Αλλιώς, η MySQL δεν ξεχωρίζει καθόλου τα πεζά από τα κεφαλαία γράμματα (case insensitive) αλλά με μια εξαίρεση : τα ονόματα των πινάκων και των στηλών καθώς και τα άλλα ονόματα πρέπει να γράφονται ακριβώς το ίδιο όταν χρησιμοποιούνται περισσότερες από μία φορές στην ίδια εντολή. Επίσης, εκχωρήσαμε έναν συγκεκριμένο τύπο δεδομένων σε κάθε στήλη που δημιουργήσαμε. Η ID θα περιέχει ακεραίους (integers), η Name θα περιέχει κείμενο (text) και η Date θα περιέχει ημερομηνίες (dates). Η MySQL απιατεί να καθορίζουμε τον τύπο δεδομένων (data type) για κάθε στήλη από την αρχή. Αν γράψουμε σωστά την παραπάνω εντολή, η MySQL θα εμφανίσει το μήνυμα Query OK και θα έχουμε έτσι δημιουργήσει τον πρώτο μας πίνακα (table). Για να βεβαιωθούμε ότι πράγματι δημιουργήθηκε ο πίνακας, δίνουμε την εξής εντολή :

```
mysql> SHOW TABLES;
```

Η απάντηση θα είναι ως εξής :

```
Tables in clients

Clients

1 row in set
```

Αυτή είναι η λίστα όλων των πινάκων που υπάρχουν στην βάση δεδομένων clinents και περιέχει έναν μόνο πίνακα : τον Clients. Ας ρίξουμε τώρα μια πιο αναλυτική ματιά στον πίνακα Clients :

```
mysql> DESCRIBE Clients;

Field | Type | Null | Key | Default

ID | int(11) | | PRI | 0 | ...

Name | text | YES | | NULL

Date | date | | | 0000-00-00

3 rows in set
```

Η παραπάνω εντολή εμφανίζει μια λίστα των στηλών (πεδίων) που υπάρχουν στον πίνακα. Όπως μπορούμε να δούμε, υπάρχουν τρεις στήλες σ' αυτόν τον πίνακα που εμφανίζονται σαν τρεις γραμμές στο αποτέλεσμα. Για να διαγράψουμε έναν πίνακα στην MySQL, η εντολή είναι σχεδόν ολόιδια με την εντολή για την διαγραφή μιας βάσης δεδομένων :

```
mysql> DROP TABLE <tableName>;
```

3.6 Εισαγωγή Δεδομένων σε Πίνακα (Table)

Η εντολή για να εισάγουμε δεδομένα σε μια βάση δεδομένων αποκαλείται INSERT και

υπάρχουν οι εξής δύο βασικές μορφές αυτής της εντολής :

```
mysql> INSERT INTO <table name> SET

-> columnName1 = value1,

-> columnName2 = value2,

-> ...

-> ;

mysql> INSERT INTO <table name>

-> (columnName1, columnName2, ...)

-> VALUES (value1, value2, ...);
```

Ετσι, για να προσθέσουμε έναν πελάτη στον πίνακα, μπορούμε να επιλέξουμε μια από τις εξής εντολές :

```
mysql> INSERT INTO Clients SET

-> Name = "Αβραμόπουλος",

-> Date = "1950-03-21";

mysql> INSERT INTO Clients

-> (Name, Date) VALUES (

-> "Δημητρίου",

-> "1960-07-14"

-> );
```

Πρέπει να έχουμε υπόψη μας ότι στην δεύτερη μορφή της εντολής INSERT, η σειρά με την οποία γράφουμε τις στήλες πρέπει να ταιριάζει με την σειρά με την οποία γράφουμε τις αντίστοιχες τιμές.

3.7 Εμφάνιση των Αποθηκευμένων Δεδομένων

Η εντολή για να δούμε τα δεδομένα που είναι αποθηκευμένα στους πίνακες μιας βάσης δεδομένων είναι η SELECT και αποτελεί την πιο πολύ χρησιμοποιούμενη και πιο πολύπλοκη εντολή της SQL. Η επόμενη εντολή θα εμφανίσει ό,τι είναι αποθηκευμένο στον πίνακα Clients :

```
mysql> SELECT * FROM Clients;
```

Αν θέλουμε να εμφανισθούν οι τιμές ορισμένων μόνο στηλών, δίνουμε την εξής εντολή :

```
mysql> SELECT ID, Date FROM Clients;
```

Το αποτέλεσμα θα είναι ως εξής :

```
ID Date

1     1950-03-21

2     1960-07-14

2 rows in set (0.00 sec)
```

Εκτός από το να εμφανίσουμε τις στήλες που θέλουμε με την εντολή Select, μπορούμε να τροποποιήσουμε την εμφάνισή τους με συναρτήσεις (functions). Μια συνάρτηση, η LEFT(), εμφανίζει έναν μέγιστο καθορισμένο αριθμό χαρακτήρων για μια στήλη. Για παράδειγμα, αν θέλουμε να δούμε μόνο τους 10 πρώτους χαρακτήρες της στήλης Name, μπορούμε να δώσουμε την εξής εντολή :

```
mysql> SELECT ID, LEFT(Name, 10), Date FROM Clients;

ID LEFT(Name, 20) Date

1 Αβραμόπουλ 1950-03-21

2 Δημητρίου 1960-07-14

2 rows in set (0.05 sec)
```

Μια άλλη χρήσιμη συνάρτηση είναι η COUNT(), η οποία απλά μετράει τον αριθμό των επιστρεφόμενων αποτελεσμάτων. Έτσι, για παράδειγμα, αν θέλουμε να βρούμε πόσοι πελάτες υπάρχουν στον πίνακα, μπορούμε να χρησιμοποιήσουμε την εξής εντολή :

```
mysql> SELECT COUNT(*) FROM Clients;

COUNT(*)

2

2 rows in set (0.06 sec)
```

3.8 Η Δήλωση WHERE

Χρησιμοποιώντας την δήλωση (clause) WHERE σε μια εντολή SELECT, μπορούμε να περιορίσουμε τα επιστρεφόμενα αποτελέσματα χρησιμοποιώντας κάποια συνθήκη, ως εξής :

```
mysql> SELECT COUNT(*) FROM Clients

-> WHERE Date >= "1950-01-01";
```

Το παραπάνω ερώτημα (query) θα μετρήσει τον αριθμό των πελατών που έχουν ημερομηνίες γέννησης μεγαλύτερες από ή ίσες από την 1η Ιανουαρίου 1950. Μια άλλη παραλλαγή της δήλωσης WHERE που μας δίνει την δυνατότητα να αναζητήσουμε καταχωρήσεις που περιέχουν ένα συγκεκριμένο κομμάτι κειμένου, είναι η εξής :

```
mysql> SELECT Name FROM Clients
```

```
-> WHERE Name LIKE "%ίδη%";
```

Αυτό το ερώτημα (query) εμφανίζει το κείμενο όλων των ονομάτων των πελατών που περιέχουν το κείμενο ίδη στην στήλη Name. Η λέξη κλειδί (keyword) LIKE λέει στην MySQL ότι η συγκεκριμένη στήλη πρέπει να ταιριάζει με το δεδομένο υπόδειγμα, που σ' αυτήν την περίπτωση είναι το "%ίδη%". Τα σύμβολα % εδώ σημαίνουν ότι το κείμενο ίδη μπορεί να έχει πριν και μετά ένα οποιοδήποτε κείμενο. Κάτι δηλαδή σαν τον γνωστό μας χαρακτήρα μπαλαντέρ (wildcard) * από άλλα προγράμματα.

Μπορούμε επίσης να συμπεριλάβουμε και συνθήκες στην δήλωση WHERE για να περιορίσουμε κι άλλο τα αποτελέσματα. Για παράδειγμα, για να εμφανίσουμε τους πελάτες που περιέχουν το κείμενο ίδη στο όνομά τους και που γεννήθηκαν μόνο τονΑπρίλιο του 1961, μπορούμε να χρησιμοποιήσουμε το εξής ερώτημα :

```
mysql> SELECT Name FROM Clients WHERE

-> Name LIKE "%ίδη%" AND

-> Date >= "1961-04-01" AND

-> Date < "1961-05-01";
```

3.9 Τροποποίηση των Αποθηκευμένων Δεδομένων

Για να κάνουμε αλλαγές στις τιμές ενός πίνακα μιας βάσης δεδομένων, χρησιμοποιούμε την εντολή UPDATE, η σύνταξη της οποίας είναι ως εξής :

```
mysql> UPDATE <tableName> SET

-> <col_name>=<new_value>, ...

-> WHERE <where clause>;
```

Ετσι, για παράδειγμα, αν θελήσουμε να αλλάξουμε την ημερομηνία γέννησης ενός πελάτη, θα πρέπει να δώσουμε την εξής εντολή :

```
mysql> UPDATE Clients SET Date="1960-04-01" WHERE ID=1;
```

Η στήλη ID είναι βολική σ' αυτήν την περίπτωση καθώς μας δίνει την δυνατότητα να διαχωρίσουμε εύκολα και σίγουρα τον πελάτη που θέλουμε να τροποποιήσουμε. Μπορούμε να χρησιμοποιήσουμε και την δήλωση WHERE, όπως στην επόμενη εντολή η οποία αλλάζει την ημερομηνία όλων των καταχωρήσεων πελατών που περιέχουν το κείμενο ήδη :

```
mysql> UPDATE Clients SET Date="1970-04-01"

-> WHERE Name LIKE "%ίδη%";
```

3.10 Διαγραφή Αποθηκευμένων Δεδομένων

Για να διαγράψουμε γραμμές (εγγραφές) στην SQL, η σύνταξη είναι η εξής :

```
mysql> DELETE FROM <tableName> WHERE <where clause>;
```

Ετσι, για παράδειγμα, για να διαγράψουμε όλους τους πελάτες που περιέχουν το κείνενο ίδη από τον πίνακα Clients, πρέπει να δώσουμε το επόμενο ερώτημα :

```
mysql> DELETE FROM Clients WHERE Name LIKE

"%ίδη%";
```

Η επόμενη εντολή θα αδειάσει τον πίνακα Clients με μιας :

```
mysql> DELETE FROM Clients;
```

ΚΕΦΑΛΑΙΟ 4ο

Εργαλεία ανάπτυξης

***Εικ. 4.1**: Εργαλεία ανάπτυξης*

4.1 XAMPP

***Εικ. 4.2**: Λογότυπο του XAMPP*

4.1.1 εισαγωγη

Το όνομα του Xampp είναι ένα ακρωνύμιο των:

- **X**(σημαίνει cross-platform = που λειτουργεί σε πολλές πλατφόρμες)
- **A**pache HTTP Server
- **M**ySQL
- **P**HP
- **P**erl

Το Xampp είναι ένα ολοκληρωμένο πακέτο Server που περιλαμβάνει apache, php, mysql, filezilla ftp, phpMyAdmin, perl, mercury email Server, υποστήριξη SSL και όλα αυτά με αυτοματοποιημένη εγκατάσταση και ρύθμιση. Το μόνο που έχει να κάνει ο διαχειριστής είναι να ορίσει τα subdomains στο αρχείο conf του apache.

Και τα τρία βασικά συστατικά που χρειαζόμαστε (Apache,PHP,MySQL) είναι εργαλεία OpenSource τα οποία μπορούμε να τα βρούμε δωρεάν στο διαδίκτυο. Η διαδικασία όμως να τα κατεβάζουμε ένα και να τα κάνουμε ξεχωριστά εγκατάσταση είναι δύσκολη και χρονοβόρα, για αυτό το λόγο διαλέξαμε την λύση του XAMPP το οποίο είναι πολύ εύκολο να εγκαταστήσουμε και να το χρησιμοποιήσουμε.

4.1.2 APACHE

***Εικ. 4.3**: Λογότυπο Apache*

Αναφέρεται συνήθως ως Apache και είναι ένας web server, δηλαδή ένας εξυπηρετητής του παγκόσμιου ιστού. Κάθε φορά που επισκεπτόμαστε κάποιον ιστοχώρο, ο πλοηγός μας επικοινωνεί με έναν διακομιστή HTTP. Είναι ένας από τους δημοφιλέστερους, διότι λειτουργεί σε διάφορες πλατφόρμες λειτουργικών συστημάτων, συμπεριλαμβανομένων των Unix, Microsoft Windows, Mac OS X, GNU, FreeBSD, TPF, Solaris και eComStation Το 2009, έγινε ο πρώτος web server που ξεπέρασε τα 100 εκατομμύρια ιστοσελίδων. Ήταν η πρώτη βιώσιμη εναλλακτική επιλογή που παρουσιάστηκε έναντι στον web server της Netscape Communications Corporation (γνωστός ως Sun Java System Web Server). Έκτοτε έχει γίνει ανταγωνιστικός ως προς άλλους web servers βασισμένους στο Unix, όσον αφορά τη λειτουργικότητα και την απόδοση.

Αναπτύσσεται και συντηρείται από μια ανοιχτή κοινότητα προγραμματιστών, υπό την αιγίδα του Apache Software Foundation.

4.1.3 MySQL

***Εικ. 4.4** : Το λογότυπο MySQL*

Η MySQL είναι ένα περιβάλλον διαχείρισης (manager) σχεσιακών βάσεων δεδομένων. Εκεί μπορούμε να προσθέσουμε, να ανακτήσουμε και να διαχειριστούμε πληροφορίες που είναι αποθηκευμένες σε μια βάση δεδομένων. . Σε γενικές γραμμές είναι ένα πακέτο λογισμικού (software package) που είναι πολύ καλό στην οργάνωση και τη διαχείριση μεγάλων ποσοτήτων πληροφοριών.

Με τη χρήση της MySQL είναι εύκολη η πρόσβαση σ' αυτές τις πληροφορίες χρησιμοποιώντας μια γλώσσα συγγραφής σεναρίων στην πλευρά του διακομιστή (server-side scripting languages), όπως είναι η PHP. Η σχεσιακή MySQL σημαίνει ότι μια πληροφορία αποθηκεύεται σε χωριστούς πίνακες και όχι σε έναν μεγάλο πίνακα. Μπορούν να καθιερωθούν σχέσεις μεταξύ πινάκων και να ανακτούμε πληροφορίες χρησιμοποιώντας δομημένη γλώσσα διατύπωσης ερωτήσεων (SQL).

ΠΛΕΟΝΕΚΤΗΜΑΤΑ MYSQL

- Είναι ένα πολύ γρήγορο και δυνατό σύστημα διαχείρισης βάσεων δεδομένων
- Ο MySQL διακομιστής ελέγχει την πρόσβαση στα δεδομένα, για να μπορούν να δουλεύουν πολλοί χρήστες ταυτόχρονα
- Μπορούν να υπάρχουν ταυτόχρονα περισσότερες από μια συνδέσεις με τη βάση χωρίς να υπάρχουν πολλαπλά αντίγραφά της, όπως συμβαίνει με άλλα συστήματα βάσεων δεδομένων
- Η απόδοσή της είναι καλύτερη σε μεγαλύτερο όγκο βάσεων δεδομένων
- Είναι ιδιαίτερα βελτιωμένη σε ταχύτητα για την ανάκτηση δεδομένων
- Είναι συμβατή και μεταφέρσιμη σε διάφορες πλατφόρμες και για διάφορα εργαλεία ανάπτυξης
- Είναι οικονομική
- Η MySQL είναι λογισμικό ανοιχτού κώδικα.

Γραφικά περιβάλλοντα
Προτείνουμε την εγκατάσταση των δύο γραφικών περιβαλλόντων της MySQL (MySQL GUI Tools). Αυτά περιλαμβάνουν τα

- MySQL Query Browser
- MySQL Administrator
- MySQL Migration Toolkit

Από αυτά μόνο το πρώτο είναι άμεσα χρήσιμο για την εργασία

4.1.4 PhpMyAdmin

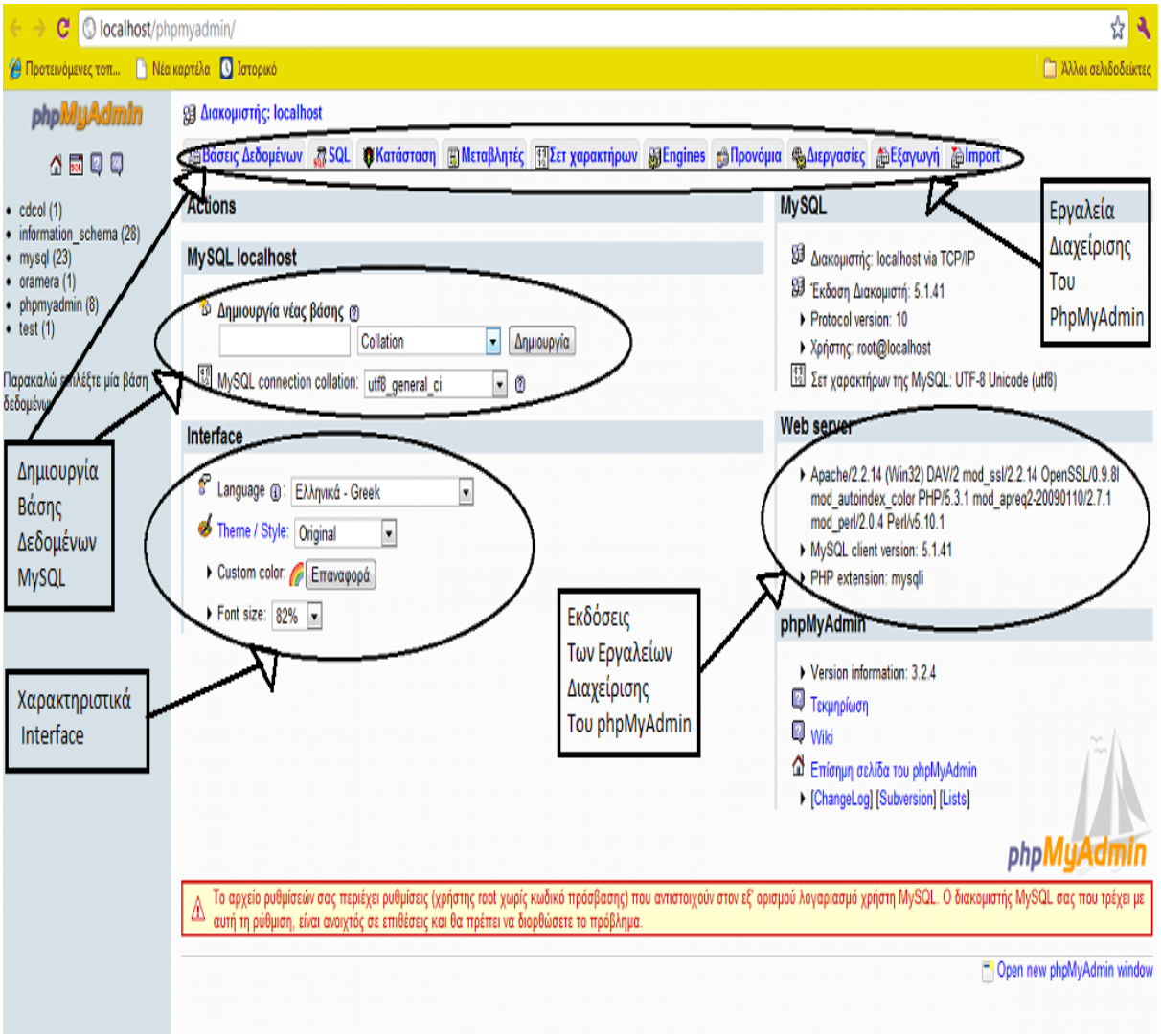
***Εικ. 4.5**: Το λογότυπο phpMyAdmin*

Το PhpMyAdmin είναι ένα εργαλείο γραμμένο σε php με το οποίο διαχειριζόμαστε τις βάσεις δεδομένων που έχουμε μέσω web.Το phpMyAdmin μπορεί να χειρίζεται πλήρως βάσεις δεδομένων, πίνακες, πεδία πινάκων αλλά και ένα ολόκληρο MySQL Server. Υποστηρίζει 54 γλώσσες, μεταξύ των οποίων και τα ελληνικά και είναι λογισμικό ανοιχτού κώδικα.

ΔΥΝΑΤΟΤΗΤΕΣ ΤΟΥ PHPMYADMIN :

Το PhpMyAdmin μπορεί να :

- Δημιουργεί και να διαγράφει βάσεις δεδομένων
- Δημιουργεί, τροποποιεί, διαγράφει, αντιγράφει και μετονομάζει πίνακες
- Κάνει συντήρηση της βάσης
- Προσθέτει, διαγράφει και τροποποιεί πεδία πινάκων
- Εκτελεί ερωτήματα SQL ακόμα και ομαδικά (batch)
- Διαχειρίζεται κλειδιά σε πεδία
- Φορτώνει αρχεία κειμένου σε πίνακες
- Δημιουργεί και διαβάζει πίνακες(που προέρχονται από dump βάσης)
- Εξάγει δεδομένα σε μορφή CVS, XML και LATEX
- Διαχειρίζεται πολλούς διακομιστές
- Διαχειρίζεται τους χρήστες MySQL και τα δικαιώματα τους
- Ελέγχει την αναφορική δραστηριότητα των δεδομένων των MyISAM πινάκων
- Δημιουργεί PDF γραφικών του layout της βάσης δεδομένων
- Εκτελεί αναζητήσεις σε όλη τη βάση δεδομένων ή μέρος αυτής
- Υποστηρίζει πίνακες InnoDB και ξένα κλειδιά
- Υποστηρίζει MySQLi, μια βελτιωμένη επέκταση του MySQL



***Εικ. 4.6**: Στιγμιότυπο εργασίας και Επεξήγηση περιβάλλοντος εργασίας*

4.1.5 PERL

***Εικ. 4.7**: Το λογότυπο Perl*

Η perl είναι μια γλώσσα scripting προγραμματισμού για ηλεκτρονικούς υπολογιστές. Είναι σχεδιασμένη για να εκτελείται από μία πληθώρα λειτουργικών συστημάτων και αρχιτεκτονικών και

διατίθεται κάτω από την άδεια ανοικτού λογισμικού GPL. Δημιουργήθηκε από τον Larry Wall το 1987 (έκδοση 1.0). Αυτή την στιγμή (2007) βρίσκεται στην έκδοση 5.8.8.

ΙΣΤΟΡΙΑ ΤΗΣ PERL

Η perl δημιουργήθηκε το 1987 απο τον Larry Wall και η πρώτη έκδοση της (1.0) ανακοινώθηκε στο alt.comp.sources του usenet στις στις 18 Δεκεμβρίου. Μετά την ταχύτατη διάδοση της γλώσσας ακολούθησαν οι εκδόσεις 2 (1988) και 3 (1989). Η τέταρτη έκδοση το 1991 δεν είχε σημαντικές διαφορές από την 3 αλλά δημιουργήθηκε ως έκδοση αναφοράς για το πρώτο βιβλίο με θέμα την γλώσσα. Η πέμπτη έκδοση ξεκίνησε να συγγράφεται το 1993 με την σταθερή έκδοση της να εκδίδεται τον Οκτώβριο του 1994. Η έκδοση αυτή έφερε σημαντικές αλλαγές και προσέθεσε σημαντικές δυνατότητες ενώ ταυτόχρονα αύξησε τον αριθμό των υποστηριζόμενων λειτουργικών συστημάτων. Η έκδοση 5 συνεχίζει να αναπτύσσεται ακόμα βρισκόμενη στην έκδοση 5.8.8. Η επόμενη έκδοση, η έκτη έχει ανακοινωθεί από το 2000 αλλά δεν έχει εμφανιστεί ακόμα.

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ

Η perl από την σχεδίαση της συνδυάζει χαρακτηριστικά άρα και ομοιότητες στον προγραμματισμό από πολλές γλώσσες όπως η C, η awk, η sed, το sh, και η Basic . Ένα από τα δυνατά της σημεία είναι η δυνατότητα σύνδεσης με μεγάλο αριθμό βάσεων δεδομένων όπως Oracle, MySql, Postgres, Microsoft SQL server, Sybase κ.α. Επιπλέον η perl έχει μια εξαιρετική μηχανή υποστήριξης κανονικών εκφράσεων(regular expressions) η οποία είναι ένας από τους λόγους που η γλώσσα είναι τόσο δημοφιλής. Αντίθετα με τις περισσότερες γνωστές γλώσσες προγραμματισμού η Perl δεν δημιουργεί ένα δυαδικό αρχείο εκτέλεσης αλλά εκτελεί (interpret) τις εντολές του προγράμματος κάθε φορά.

ΧΡΗΣΗ

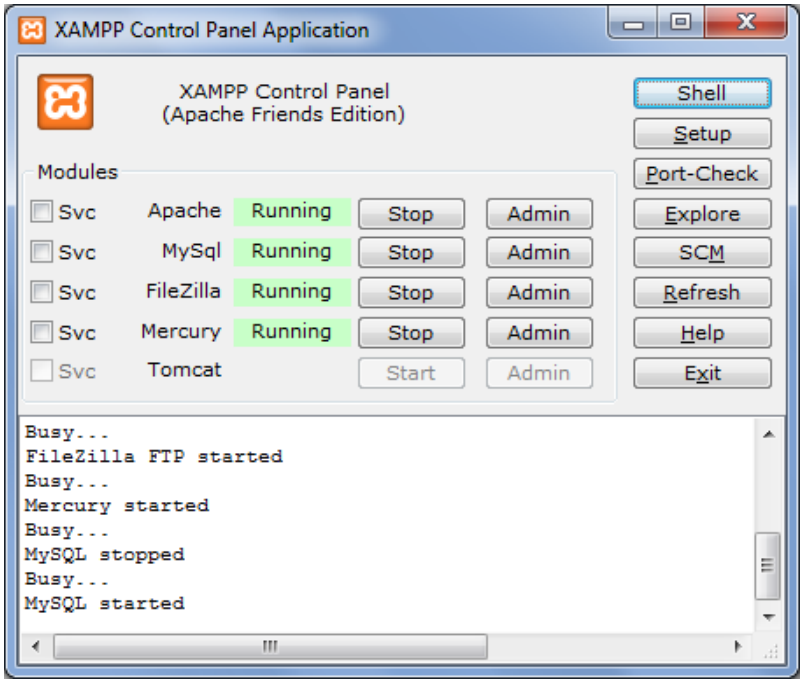
Η Perl χρησιμοποιείται από ένα ευρύ φάσμα χρηστών το οποίο περιλαμβάνει επαγγελματίες προγραμματιστές, διαχειριστές συστημάτων, web developers αλλά και απλούς χρήστες. Η πλέον διαδεδομένη χρήστη της είναι για την υλοποίηση CGI εφαρμογών οι οποίες μπορούν να εκτελεστούν από κάποιο web server όπως ο Apache και να δημιουργήσουν δυναμικές ιστοσελίδες. Επίσης χρησιμοποιείται από για να δημιουργηθούν scripts για την διαχείριση λειτουργικών συστημάτων αφού παρέχει πολύ περισσότερες δυνατότητες από το μέσο shell.

ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

Η perl έρχεται προεγκατεστημένη στην συντριπτική πλειονότητα των unix-like λειτουργικών όπως το Linux αλλά υπάρχουν εκδόσεις τις σχεδόν για κάθε λειτουργικό σύστημα. Ανάμεσα σε πολλά άλλα η Perl μπορεί να χρησιμοποιηθεί από:

- Linux
- Sun Solaris
- Microsoft Windows (όλες οι εκδόσεις συμπεριλαμβανομένου του Pocket PC)
- Mac OS & Mac OS X
- BSD
- Amiga
- Symbian

Επίσης χρησιμοποιήθηκαν στο xampp τα εξής adds on:

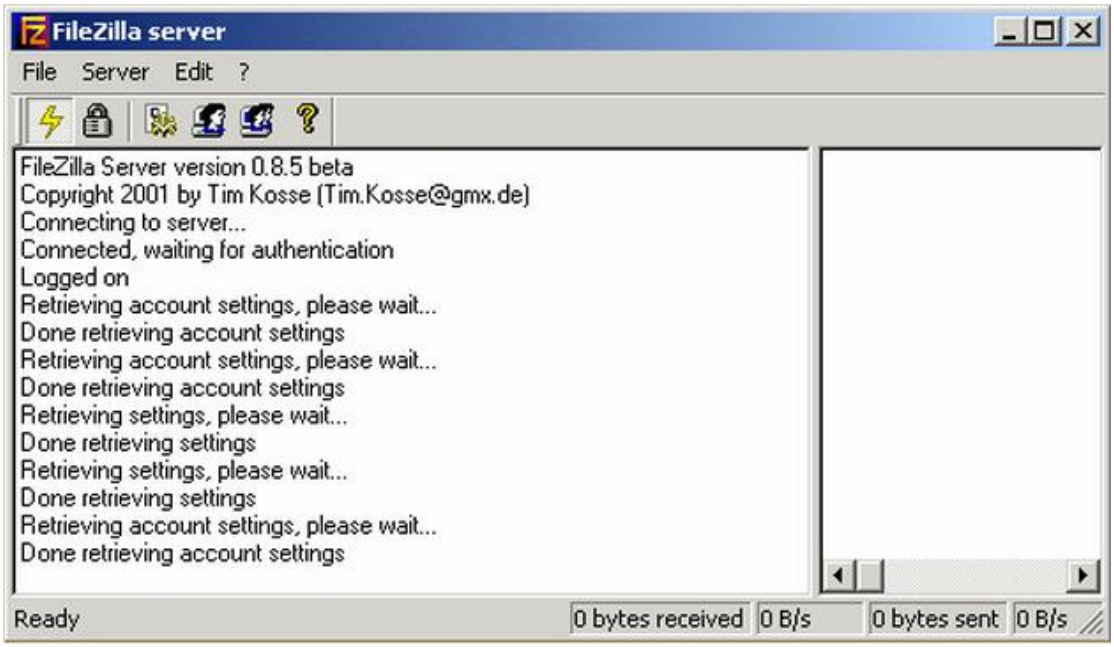


***Εικ. 4.8**: Περιβάλλον Εργασίας του XAMPP*

4.1.6 FileZilla

***Εικ. 4.9** : Λογότυπο FileZilla*

Ο FileZilla είναι ένα δωρεάν, ανοιχτού κώδικα, cross-platform FTP λογισμικό που αποτελείται από FileZilla client και FileZilla server. Δυαδικά αρχεία είναι διαθέσιμα για Windows, Linux και Mac OS X. Υποστηρίζει FTP, SFTP και FTPS. Ξεκίνησε ως ένα class project της επιστήμης υπολογιστών τη δεύτερη εβδομάδα του Ιανουαρίου του 2001 από τον Tim Kosse και δύο συμμαθητές του. Χρησιμοποιείται για ανέβασμα, κατέβασμα, διαγραφή ή μετονομασία αρχείων.



***Εικ. 4.10**: Στιγμιότυπο από FileZilla*

4.1.7 Mercury

***Εικ. 4.11**: Το λογότυπο Mercury*

Ο Mercury είναι ένας πλήρως ανεξάρτητος mail server και μπορεί να παρέχει υπηρεσίες ηλεκτρονικού ταχυδρομείου σε όλα τα συμβατά προγράμματα αλληλογραφίας, όπως το Eudora και το Microsoft Outlook. Και οι δύο εκδόσεις του Mercury επιτρέπουν υποστήριξη για διαφορετικές ομάδες πρωτοκόλλων του διαδικτύου ώστε να εγκατασταθούν όπως απαιτείται. Ο Mercury μπορεί επίσης να εγκατασταθεί ολοκληρωμένα και με το Pegasus Mail. Ο συνδυασμός Pegasus / Mercury είναι παρόμοιος με τα στοιχεία αλληλογραφίας του Microsoft Outlook / Microsoft Exchange server.

4.1.8 Tomcat

Εικ. 4.12 *: Το λογότυπο Tomcat*

Χρησιμοποιήθηκε για την παραφωγή των servlets. Ένα servlet είναι μία κλάση της java η οποία χρησιμοποιείται για την επέκταση των δυνατοτήτων ενός εξυπηρετητή ο οποίος φιλοξενεί εφαρμογές που βασίζονται στο μοντέλο αιτήματος – ανταπόκρισης. Η πιο συχνή τους εφαρμογή είναι στην επέκταση των δυνατοτήτων εξυπηρετητών διαδικτύου, δηλαδή εξυπηρετητών που βασίζονται στο πρωτόκολλο HTTP. Τα servlets επικοινωνούν με ένα εξυπηρετητή μέσω ενός container. Π.χ. Apache Tomcat για τον Apache Web Server.

4.2 Dreamweaver

Εικ. 4.13*: Το λογότυπο του Dreamweaver*

Για την δημιουργία του γραφικού περιβάλλοντος χρησιμοποιήθηκε η γλώσσα php. Ο κώδικας αποθηκεύτηκε σε ειδικό φάκελο του εξυπηρετητή που βρίσκεται στο C:/xampp/htdocs/. Για τη δημιουργία του κώδικα χρησιμοποιήσαμε το πρόγραμμα Dreamweaver. Όταν στον web browser τοποθετήσουμε τη διεύθυνση *https://127.0.0.1/kodikas.php* (kodikas.php είναι το όνομα του αρχείου php που δημιουργήσαμε στο Dreamweaver και που περιέχει τον κώδικά μας) θα τρέξουμε τη σελίδα μας και θα εκτελεστούν ό,τι εντολές περιέχουν τα queries μας. Έπειτα, αν κάνουμε ανανέωση στην περιήγηση της βάσης δεδομένων μας θα δούμε αν προστέθηκαν ή όχι εισαγωγές στη βάση μας.

Το πρόγραμμα Dreamweaver της εταιρείας Macromedia είναι ένα κορυφαίο πρόγραμμα δημιουργίας και επεξεργασίας ιστοσελίδων, δηλαδή κώδικα HTML, που είναι ιδιαίτερα εύκολο και φιλικό στη χρήση του. Το όνομα DreamWeaver προέρχεται από ένα παλιό ρομαντικό τραγούδι. Το Dreamweaver είναι εξαίρετο για να μπορούμε να δημιουργήσουμε στα γρήγορα φόρμες (forms), πλαίσια (frames), πίνακες (tables) και άλλα αντικείμενα της HTML. Είναι, όμως, ιδιαίτερα καλό όταν θέλουμε να δώσουμε σε μια ιστοσελίδα τη δυνατότητα να κάνει κάτι. Πρέπει να έχουμε υπόψη μας ότι το Dreamweaver μπορεί να χρησιμοποιηθεί και για τη δημιουργία εφαρμογών πολυμέσων.

Το Dreamweaver έχει δυνατότητες για δημιουργία δυναμικής HTML (DHMTL) και επιτρέπει κίνησης γραμμής χρόνου, απόλυτη τοποθέτηση περιεχομένων, δημιουργία επιπέδων (layers) και συγγραφή σεναρίων (scripts). Το Dreamweaver περιέχει δικές του συμπεριφορές (behaviors), που

είναι έτοιμα scripts τα οποία μπορούμε να προσθέσουμε πολύ εύκολα σ' ένα αντικείμενο.

Το Dreamweaver μάς παρέχει την ελευθερία να σχεδιάσουμε οπτικά την εμφάνιση μιας ιστοσελίδας και τη δύναμη να την κάνουμε να λειτουργεί όπως ακριβώς θέλουμε. Μπορούμε να δημιουργήσουμε τη δική μας προσωπική ιστοσελίδα (personal web page) ή μια ολόκληρη περιοχή (web site) σ' ένα εταιρικό δίκτυο (intranet).

ΚΕΦΑΛΑΙΟ 5ο

Κώδικας προγράμματος

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"
/>
<title>Untitled Document</title>
</head>

<body>
<?php
$handle = fopen("txt.txt", "rb");// Με το fopen ανοίγουμε το αρχείο
μας. rb = read
$contents = '';
if (!$handle) { // Αν το handle δεν επιστραφεί σημαίνει ότι το
πρόγραμμα δεν μπορεί να ανοίξει το αρχείο
    echo 'Δεν μπορεί να ανοίξει το αρχείο: txt.txt';
}
$temp="";

$i=0;
$ktemp=0;
while (false !== ($char = fgetc($handle))) { // Όσο διαβάζει
χαρακτήρες στο αρχείο, η fgetc παίρνει ένα-ένα χαρακτήρα από το το
κείμενο που υπάρχει

     if($char=="^"){ // Αν συναντήσει στο κείμενο τον χαρακτήρα "^"
                // παράδειγμα: 7^8^11 -> 22
                // 5^8 -> 34
                // array[0][0] -> 7
                // [0][1] -> 8
                // [0][2] -> 11
                // [0][thirdvar] -> 22
                // array[1][0] -> 5
                // [1][1] -> 8
               // [1][thirdvar] -> 34
```

```
                $array[$i][$ktemp]=$temp;
                $ktemp=$ktemp+1;
                $temp=""; // Κάνουμε πάλι το temp κενό
                continue;

        }
        if($char=="-"){ // Αν συναντήσει στο κείμενο τον χαρακτήρα "-"

                $array[$i][$ktemp]=$temp;
                $ktemp=$ktemp+1;
                $temp="";
                continue;

        }
        if($char==">"){ // Αν συναντήσει στο κείμενο τον χαρακτήρα ">"

                $temp="";
                continue;

        }
        if($char=="\n"){ // Αν συναντήσει στο κείμενο τον χαρακτήρα
"\n" -> αλλαγή γραμμής
                 $array[$i]["thirdvar"]=$temp;

                $ktemp=0;
                $temp="";
                $i=$i+1;
                continue;

        }
        $temp=$temp.$char;

        // Όταν τελειώνει η while έχουμε τον πίνακα με τα δεδομένα
}

                $array[$i]["thirdvar"]=$temp;
```

```
                $ktemp=$ktemp+1;

     $conn=mysql_connect("localhost","root","")    or die
(mysql_error("error")); // Συνδέεται με την βάση. root είναι ο
χρήστης
     mysql_select_db("oramera",$conn); // oramera είναι το όνομα της
βάσης μας

foreach ($array as $i => $value) {  // Βάζει στον πίνακα
array_to_compare όλα τα time start και τα time end του κάθε
στοιχείου

$min=0;
$max=0;

   for($counteri=0; $counteri<sizeof($array[$i])-1; $counteri++){

            $query="select * from stoixeia where
id=".$array[$i][$counteri].""; // Θα πάρουμε το timestart και το
timeend του πρώτου
   $result=mysql_query($query,$conn) ;
   if(mysql_num_rows($result)<1){
      echo "ERROR: Δεν υπάρχει η εγγραφή:
".$array[$i][$counteri]."<br>";
   }
   if(mysql_num_rows($result)>0){

$rowcounter=0;
     while($row=mysql_fetch_array($result)){ // Για κάθε γραμμή του
αρχείου " ".txt γεμίζουμε τον πίνακα και καλούμε την
compare_fieldsnew


     $array_to_compare[$array[$i][$counteri]][$rowcounter]["timestar
t"]=$row['timestart']; // Τρισδιάστατος πίνακας. Παράδειγμα:
array_to_compare [7][0][timestart]-[7][0][timeend]-> Γεμίζουμε τον
πίνακα με τις τιμές για να συνεχίσω με τους υπολογισμούς

     $array_to_compare[$array[$i][$counteri]][$rowcounter]["timeend"
]=$row['timeend'];

            $rowcounter=$rowcounter+1;
```

```
      }

    }

    }

$x=compare_fieldsnew($array_to_compare,$array[$i],$array[$i]["thirdv
ar"],$conn); // Η σημαντική συνάρτηση του προγράμματος. Της
φορτώνουμε για κάθε γραμμή τα δεδομένα.
      // Για κάθε γραμμή η συνάρτηση καλείται και παίρνει 4 ορίσματα:
      // 1ο όρισμα: Περιέχει όλα τα timestart και timeend κάθε
μεταβλητής
      // 2ο όρισμα: Περιέχει όλα τα IDS π.χ.:5,8,11
      // 3ο όρισμα: Περιέχει όλα τα αποτελέσματα π.χ.: 22
      // 4ο όρισμα: Χρησιμοποιείται για να γίνει η σύνδεση με την
βάση

 print_r($x); // Το x είναι ένας πίνακας που περιέχει όλα τα min και
τα max τα οποία θα μπουν στην βάση μέσω της instert_to_db
 echo "<br><br>";
 if(sizeof($x)>0){
 foreach ($x as $key => $value) {


insert_to_db($value["max"],$value["min"],$array[$i]["thirdvar"],$con
n);


}
 }


echo "<br><br><br>";


$array_to_compare=null; // Το κάνουμε null για να πάρει και τα
υπόλοιπα στοιχεία. Περνάμε σε άλλη γραμμή για τη εισαγωγή των
στοιχείων


}


function compare_array_fields($arr,$arr2,$thirdvar,$conn){ //
Συνάρτηση για να συγκρίνουμε τα δεδομένα του πίνακα
```

```
	for($counteri=0; $counteri<=sizeof($arr2)-1; $counteri++){

	for($counteri2=0; $counteri2<=sizeof($arr[$arr2[$counteri]])-1;
$counteri2++){
		if($arr2[$counteri+1]){

			$min=0;
$max=0;

			for($counter3=0;
$counter3<=sizeof($arr[$arr2[$counteri+1]])-1; $counter3++){
				echo "Σύγκρινε το timestart του
".$arr2[$counteri]." στη θέση".$counteri2." (
".$arr[$arr2[$counteri]][$counteri2]["timestart"]." ) με timestart
του ".$arr2[$counteri+1]." στη θέση".$counter3. "(
".$arr[$arr2[$counteri+1]][$counter3]["timestart"]." )<br>timeend
του ".$arr2[$counteri]." στη θέση".$counteri2." (
".$arr[$arr2[$counteri]][$counteri2]["timeend"]." ) με timeend του
".$arr2[$counteri+1]." στη θέση".$counter3. "(
".$arr[$arr2[$counteri+1]][$counter3]["timeend"]." )<br>";

		$max=$arr[$arr2[$counteri]][$counteri2]["timestart"];
		$min=$arr[$arr2[$counteri]][$counteri2]["timeend"];

	$timestartTouprwtouPinaka[$counteri]=$arr[$arr2[$counteri+1]][$
counter3]["timestart"];

	$timeendTouprwtouPinaka[$counteri]=$arr[$arr2[$counteri+1]][$co
unter3]["timeend"];

		if($timestartTouprwtouPinaka[$counteri]==0){
			echo "Δημιουργήθηκε κάποιο σφάλμα. Μη επιτρεπτή τιμή
ημερομηνίας: ".$timestartTouprwtouPinaka[$counteri]."<br>";

		}
		if($timestartTouprwtouPinaka[$counteri]==0){
			echo "Δημιουργήθηκε κάποιο σφάλμα. Μη επιτρεπτή τιμή
ημερομηνίας: ".$timestartTouprwtouPinaka[$counteri]."<br>";

		}

		if($min==0){
```

```
                $min=$timeendTouprwtouPinaka[$counteri];
            }


            if($timestartTouprwtouPinaka[$counteri]>$max){
                $max=$timestartTouprwtouPinaka[$counteri];

            }

            if($timeendTouprwtouPinaka[$counteri]<$min){
                $min=$timeendTouprwtouPinaka[$counteri];

            }

            echo "MIN ".$min;
            echo "MAX ".$max;

                }
            }

        }

        }
}

function compare_fieldsnew($arr,$arr2,$thirdvar,$conn){ // Παίρνουμε
το timestart και το timeend των 2 πρώτων στοιχείων π.χ.: 7 και 8->78
        //temp aerray -> Εικονικός πίνακας για καταχώρηση υπολογισμών


$tempcounter=0;
$tempcounter2=0;
$checkwithtemp=0;
        for($counteri=0; $counteri<=sizeof($arr2)-1; $counteri++){
        if(sizeof($arr2)==3 and $counteri==1){

            return $temparray;
```

```
	}
	if($counteri==sizeof($arr2)-1){
		return $temparray2;
	}
		if($counteri>0){
			$checkwithtemp=1;
		}

		if($checkwithtemp!=1){

	for($counteri2=0; $counteri2<=sizeof($arr[$arr2[$counteri]])-1;
$counteri2++){

		if($arr2[$counteri+1]){

		if($checkwithtemp!=1){

			$min=0;
			$max=0;

			for($counter3=0;
$counter3<=sizeof($arr[$arr2[$counteri+1]])-1; $counter3++){
			$haserror=0;

		$max=$arr[$arr2[$counteri]][$counteri2]["timestart"];
		$min=$arr[$arr2[$counteri]][$counteri2]["timeend"];


	$timestartTouprwtouPinaka[$counteri]=$arr[$arr2[$counteri+1]][$
counter3]["timestart"];

	$timeendTouprwtouPinaka[$counteri]=$arr[$arr2[$counteri+1]][$co
unter3]["timeend"];

		if($timestartTouprwtouPinaka[$counteri]==0){
			echo "Δημιουργήθηκε κάποιο σφάλμα. Μη επιτρεπτή τιμή
ημερομηνίας στο timestart:
".$timestartTouprwtouPinaka[$counteri]."<br>";

			$haserror=1;
```

```
            }
            if($timeendTouprwtouPinaka[$counteri]==0){
                  echo "Δημιουργήθηκε κάποιο σφάλμα. Μη επιτρεπτή τιμή
ημερομηνίας στο timeend:
".$timeendTouprwtouPinaka[$counteri]."<br>";
                  $haserror=1;
            }

            if($min==0){

                  $min=$timeendTouprwtouPinaka[$counteri];
            }

            if($timestartTouprwtouPinaka[$counteri]>$max){
                  $max=$timestartTouprwtouPinaka[$counteri];

            }

            if($timeendTouprwtouPinaka[$counteri]<$min){
                  $min=$timeendTouprwtouPinaka[$counteri];

            }

if($haserror!=1){

$temparray[$tempcounter]["min"]=$min;
$temparray[$tempcounter]["max"]=$max;
$temparray[$tempcounter]["name"]=$arr2[$counteri].$arr2[$counteri+1]
;

$tempcounter=$tempcounter+1;
}
                  }

                  }

            }
```

```
                }
        }
                else{
        if(sizeof($temparray2)>0){
        $temparray=$temparray2;
        }
                $min=0;
                $max=0;

                for($counter3=0; $counter3<=sizeof($temparray)-1;
$counter3++){
                        for($counter4=0;
$counter4<=sizeof($arr[$arr2[$counteri+1]])-1; $counter4++){
                                $haserror=0;

            $max=$temparray[$counter3]["max"];
            $min=$temparray[$counter3]["min"];


        $timestartTouprwtouPinaka[$counteri]=$arr[$arr2[$counteri+1]][$
counter4]["timestart"];

        $timeendTouprwtouPinaka[$counteri]=$arr[$arr2[$counteri+1]][$co
unter4]["timeend"];

            if($timestartTouprwtouPinaka[$counteri]==0){

                $haserror=1;
            }
            if($timestartTouprwtouPinaka[$counteri]==0){

                $haserror=1;
            }

            if($min==0){

                $min=$timeendTouprwtouPinaka[$counteri];
            }
```

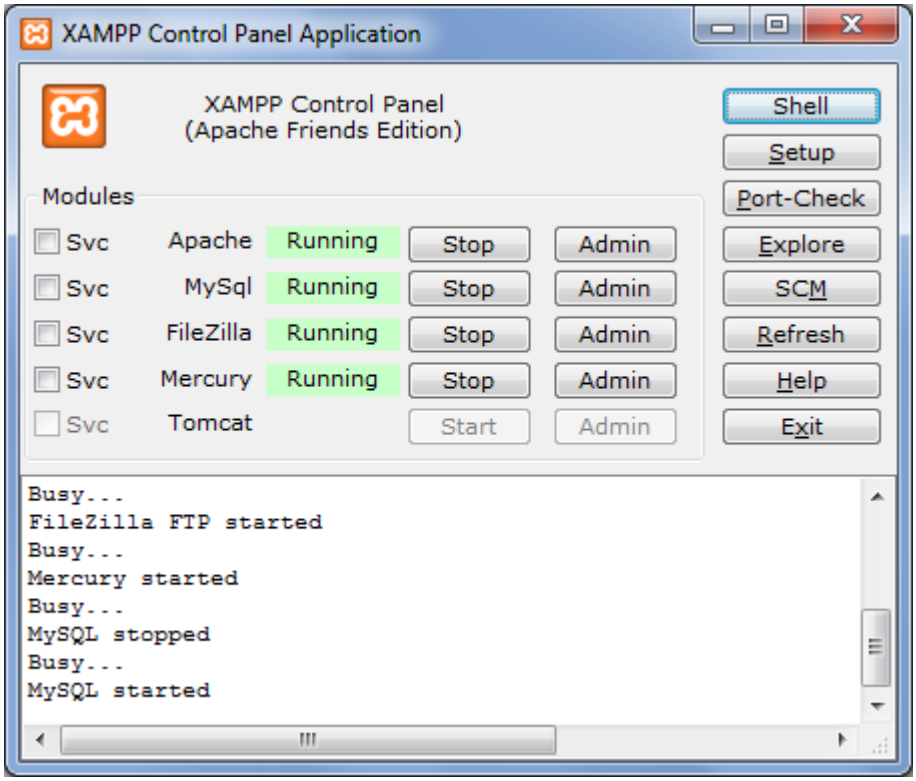
```
            if($timestartTouprwtouPinaka[$counteri]>$max){
                  $max=$timestartTouprwtouPinaka[$counteri];

            }

            if($timeendTouprwtouPinaka[$counteri]<$min){
                  $min=$timeendTouprwtouPinaka[$counteri];

            }
if($haserror!=1){

$temparray2[$tempcounter2]["min"]=$min;
$temparray2[$tempcounter2]["max"]=$max;
$temparray2[$tempcounter2]["name"]=$temparray[0]["name"].$arr2[$coun
teri+1];

$tempcounter2=$tempcounter2+1;
}
                        }

                  }

                  }
      }
}

// Βάζουμε μία επαφή στον πίνακα
function insert_to_db($max,$min,$thirdvar,$conn){ // Παίρνουμε
$max,$min,$thirdvar,$conn

if($max>$min){ // timestart > timeend
      echo "Error : Δεν πρέπει η ελάχιστη τιμή να είναι μεγαλύτερη
από την μέγιστη τιμή. Δεν ολοκληρώθηκε η Εισαγωγή στον
".$thirdvar."<br>";
   }else{
         if($min!=0 and $max!=0){
            $query3="insert into stoixeia(id,timestart,timeend)
values ('".$thirdvar."','".$max."','".$min."')";
```

```
            echo "<br>Query:".$query3."<br>";
            $result3=mysql_query($query3,$conn) ;
            echo "Ολοκληρώθηκε η εισαγωγή στον ".$thirdvar."<br>";

        }
        else{

            echo "Δεν ολοκληρώθηκε η Εισαγωγή στον
".$thirdvar."<br>";

        }
  }
   echo "<br><br>";
}

?>


</body>
</html>
```

ΚΕΦΑΛΑΙΟ 6ο

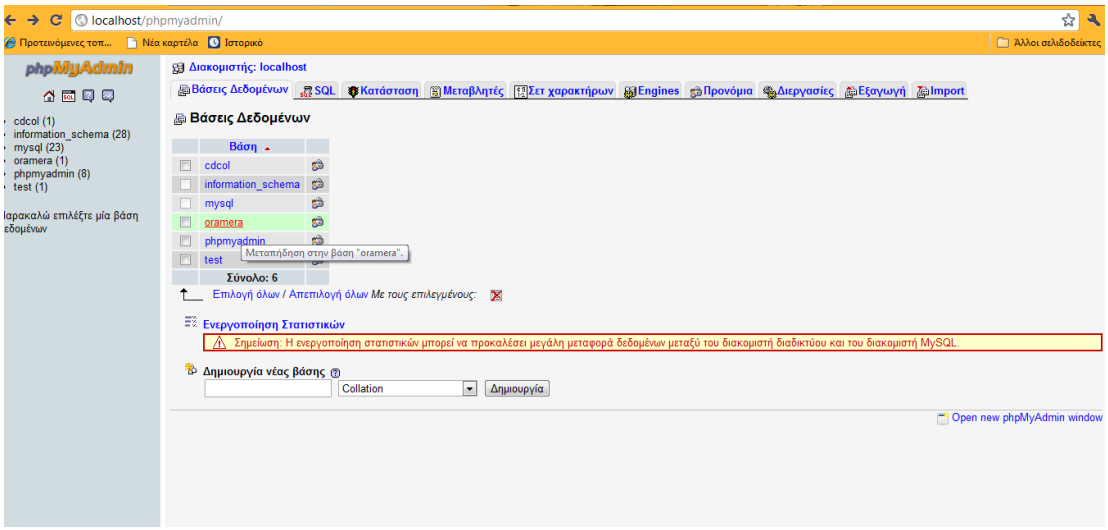
Υλοποίηση του προγράμματος.

Ανοίγουμε το XAMPP και ενεργοποιούμε έναν – έναν όλους τους servers. (Apache, MySql, FileZilla και Mercury) (Εικ.6.1)



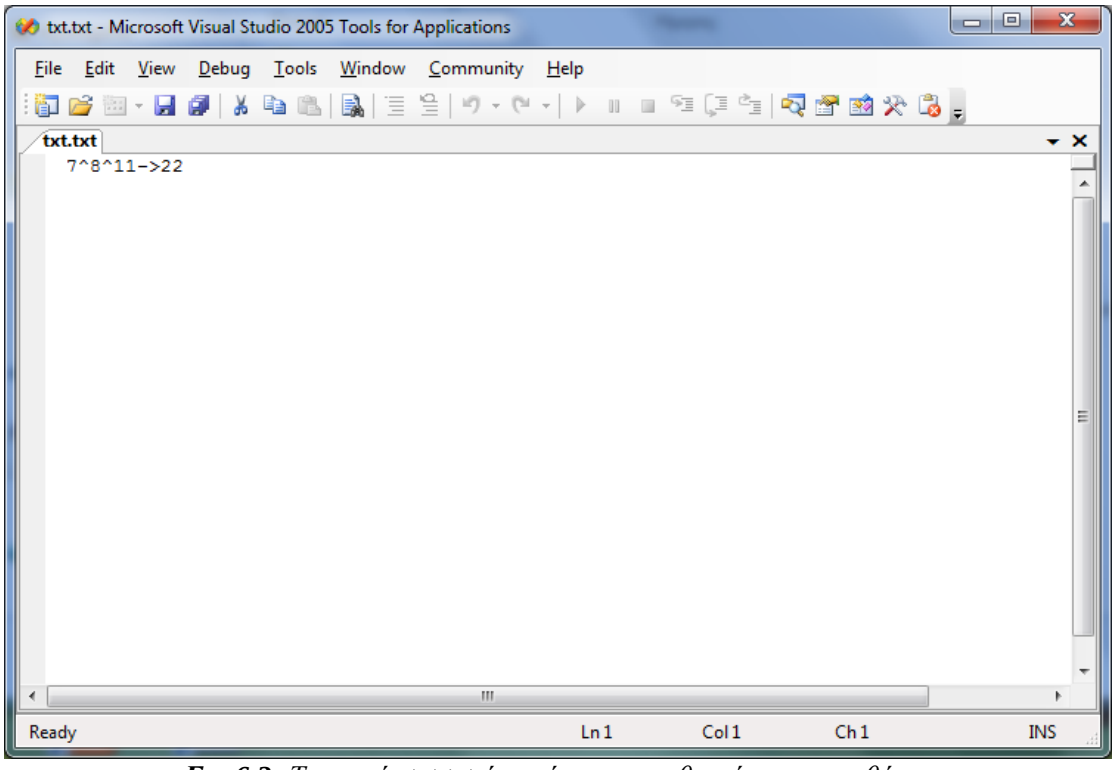
Εικ 6.1.*: Ενεργοποίηση όλων των servers*

Αφού ενεργοποιηθούν οι servers ανοίγουμε μία καινούργια σελίδα σε ένα φυλλομετρητή ιστού. Στην συνέχεια, πληκτρολογούμε ως διεύθυνση την : localhost/phpmyadmin/. Έτσι μπορούμε να διαχειριστούμε την βάση μας μέσω της PhpMyAdmin. Δημιουργούμε την βάση μας, oramera, και πατώντας πάνω της μεταβαίνουμε σε αυτήν. (Εικ. 6.2)



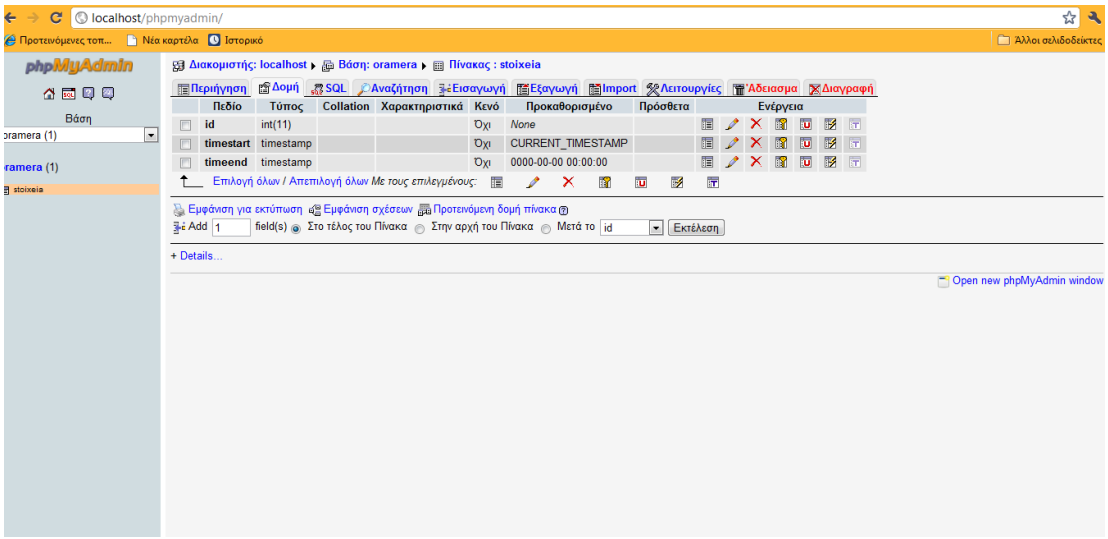
***Εικ.6.2**: Επιλογή της βάσης*

Σε ένα αρχείο .txt γράφουμε τις συνθήκες μας για τις εγγραφές και το αποθηκεύουμε στον φάκελο του xampp (htdocs).(Εικ. 6.3)



***Εικ.6.3**: Το αρχείο txt.txt όπου έχουμε αποθηκεύσει τις συνθήκες*

Δημιουργούμε τα πεδία id, timestart και timeend και ορίζουμε τους τύπους τους. Το πεδίο id είναι τύπου int ενώ τα πεδία timestart και timeend είναι τύπου timestamp. Ορίζουμε αρχικές τιμές στο timestart την τωρινή ημερομηνία και στο timeend μηδενική ημερομηνία. (Εικ. 6.4) Μπορούμε να τις επεξεργαστούμε αργότερα για να ελέγξουμε τους περιορισμούς μας.



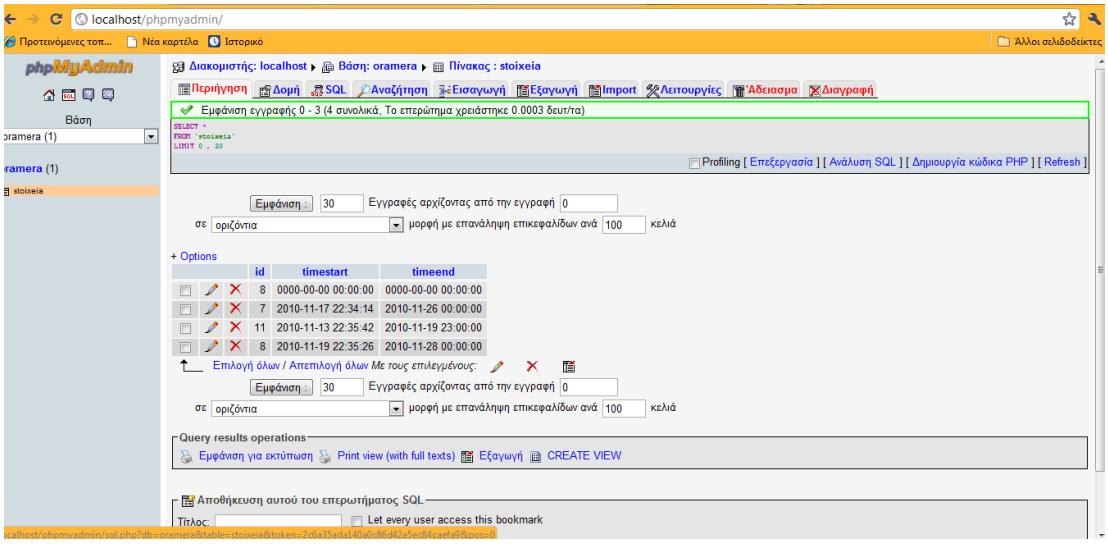
***Εικ.6.4**: Επεξεργασία των πεδίων της βάσης*

Παράδειγμα 1

Στο παρακάτω παράδειγμα έχουμε μία εγγραφή με μηδενικά χρονικά όρια. Με αυτό το παράδειγμα επιθυμούμε να ελέγξουμε την ορθότητα του κώδικα.

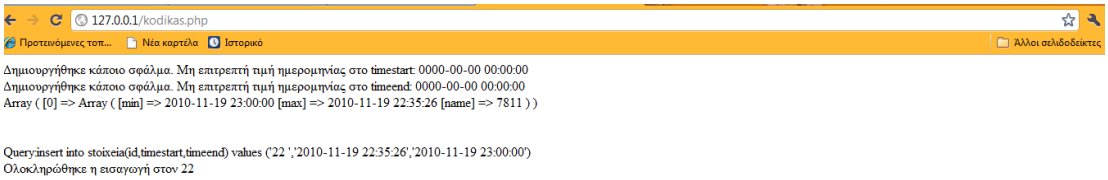
Κώδικας:

```
...

        if($timestartTouprwtouPinaka[$counteri]==0){
              echo "Δημιουργήθηκε κάποιο σφάλμα. Μη επιτρεπτή τιμή
              ημερομηνίας στο timestart:
              ".$timestartTouprwtouPinaka[$counteri]."<br>";

              $haserror=1;
        }
        if($timeendTouprwtouPinaka[$counteri]==0){
              echo "Δημιουργήθηκε κάποιο σφάλμα. Μη επιτρεπτή τιμή
              ημερομηνίας στο timeend:
              ".$timeendTouprwtouPinaka[$counteri]."<br>";
              $haserror=1;
        }

...
```



***Εικ.6.5**: Παράδειγμα για στοιχεία με μηδενικές ημερομηνίες*

Τα αποτελέσματα του κώδικα τυπώνονται στον φυλλομετρητή ιστού. Όπως βλέπουμε το πρόγραμμα μας ενημερώνει για το σφάλμα που παρουσιάστηκε με την μηδενική ημερομηνία.



***Εικ.6.6**: Απάντηση προγράμματος*

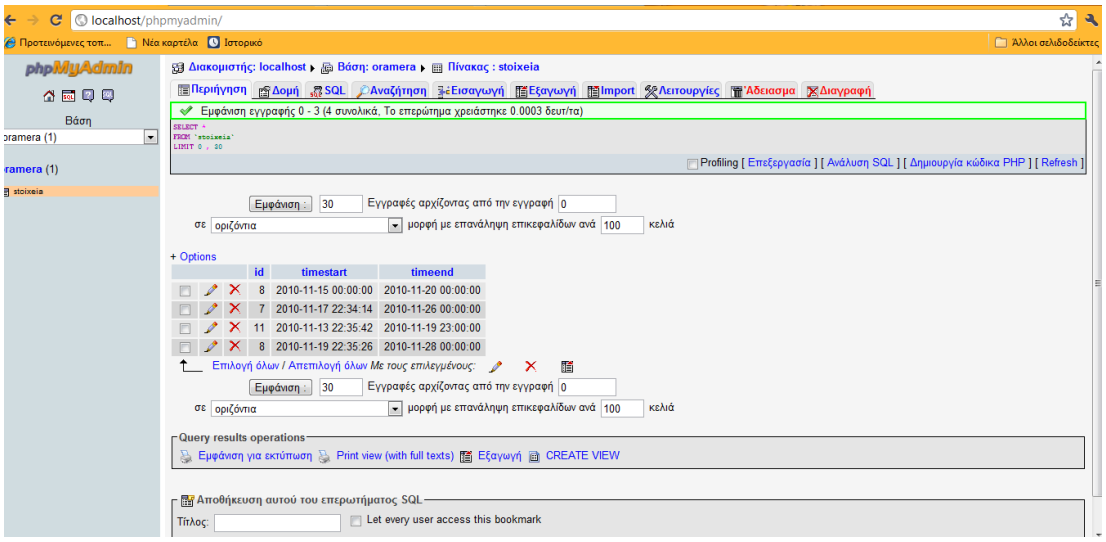
Παράδειγμα 2

Στο δεύτερο παράδειγμα υπάρχει τομή για όλες τις εγγραφές που έχουμε εισάγει (Εικ. 6.7). Έτσι η εισαγωγή στην εγγραφή 22 εκτελείται κανονικά και εμφανίζονται οι τιμές της. (Εικ. 6.8)

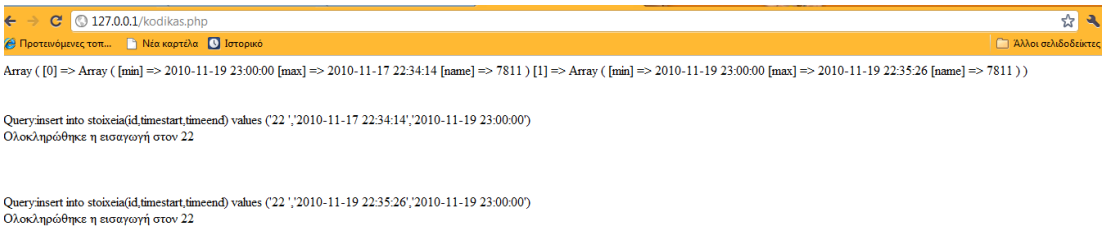
Κώδικας: (παραθέτουμε ένα τμήμα του κώδικα)

```
...
    if($min!=0 and $max!=0){
      $query3="insert into stoixeia(id,timestart,timeend)
values ('".$thirdvar."','".$max."','".$min."')";
      echo "<br>Query:".$query3."<br>";
      $result3=mysql_query($query3,$conn) ;
      echo "Ολοκληρώθηκε η εισαγωγή στον ".$thirdvar."<br>";
    }
   else{
      echo "Δεν ολοκληρώθηκε η Εισαγωγή στον".$thirdvar."<br>";
   }
```

. . .



***Εικ.6.7**: Παράδειγμα όπου υπάρχει τομή για όλες τις εγγραφές*



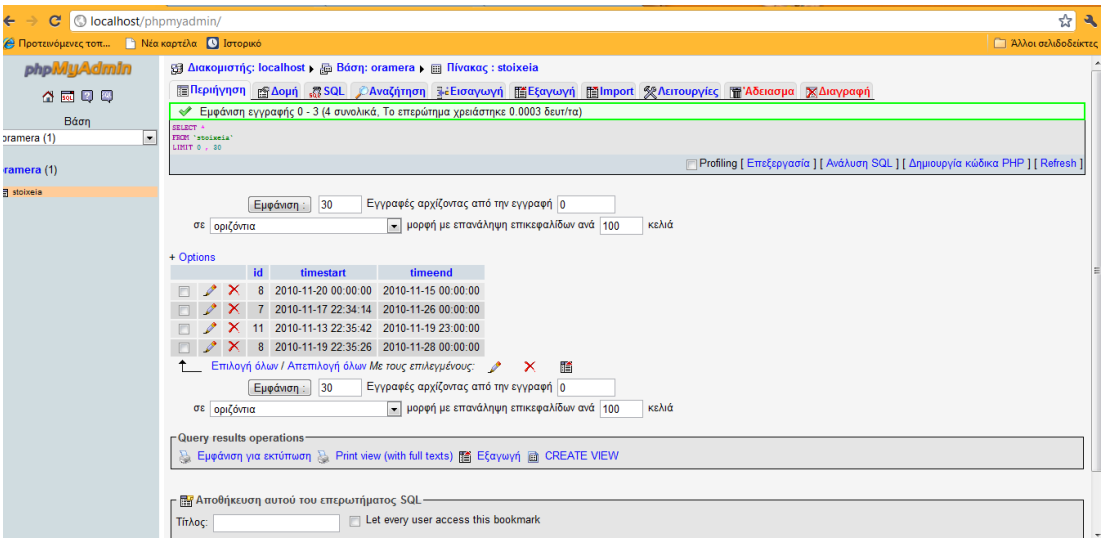
***Εικ.6.8**: Απάντηση προγράμματος*

Παράδειγμα 3

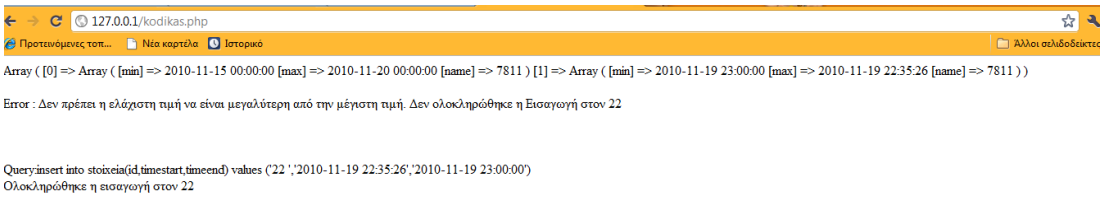
Στις εικόνες που ακολουθούν ελέγχουμε αν το timestart είναι μεγαλύτερο από το timeend. Σε αυτή την περίπτωση δεν επιτρέπουμε να ολοκληρωθεί η εγγραφή.

Κώδικας:

```
. . .

        if($max>$min){ // timestart > timeend
              echo "Error : Δεν πρέπει η ελάχιστη τιμή να είναι
              μεγαλύτερη από την μέγιστη τιμή. Δεν ολοκληρώθηκε η
              Εισαγωγή στον ".$thirdvar."<br>";
         }

. . .
```



***Εικ.6.9**: Παράδειγμα όπου timestart > timeend*



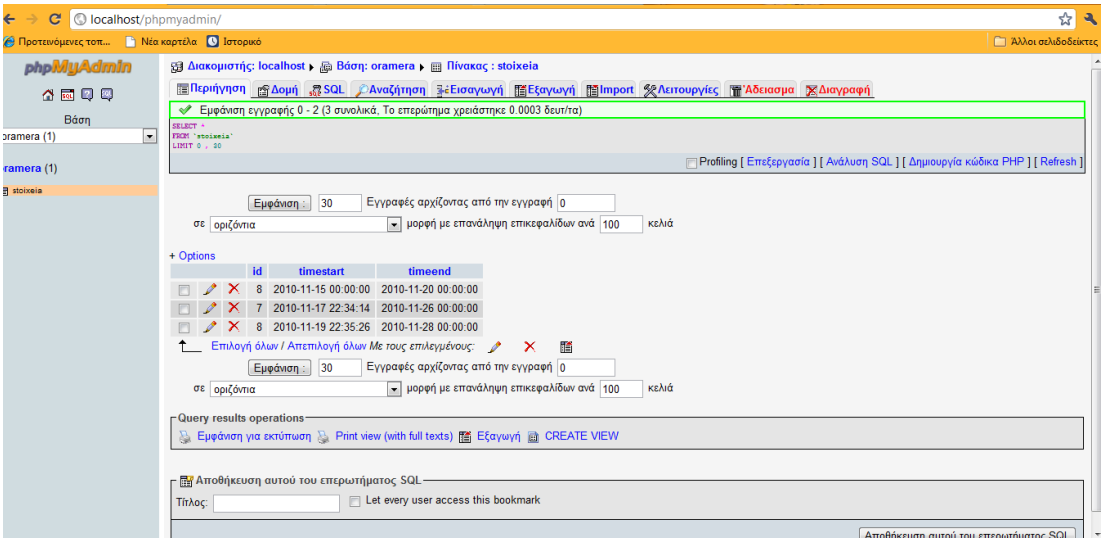
***Εικ.6.10**: Απάντηση προγράμματος*

Παράδειγμα 4

Στην τελευταία περίπτωση εξετάζουμε το ενδεχόμενο να μην υπάρχει κάποια από τις εγγραφές που έχουμε δηλώσει στο txt αρχείο που αναφέρουμε παραπάνω. Τα αποτελέσματα εμφανίζονται στην οθόνη μας (Εικ. 6.11, Εικ. 6.12) ειδοποιώντας τον χρήστη για το σφάλμα.

Κώδικας:

```
...

if(mysql_num_rows($result)<1){
        echo "ERROR: Δεν υπάρχει η εγγραφή:
        ".$array[$i][$counteri]."<br>";
}

...
```



***Εικ.6.11**: Παράδειγμα όπου δεν υπάρχει μία εγγραφή (εδώ λείπει η εγγραφη 11)*

***Εικ.6.12**: Απάντηση προγράμματος*

ΚΕΦΑΛΑΙΟ 7ο

Βιβλιογραφία – Πηγές

http://portal.acm.org/
http://en.wikipedia.org/
http://www.php.net/
http://www.mysql.com/
http://www.apachefriends.org/en/xampp.html
http://dide.flo.sch.gr/Plinet/plinet.html
http://www.apache.org/
http://www.flak.gr/linux/linux-general/phpmyadmin.html
http://www.phpmyadmin.net/home_page/index.php
http://filezilla-project.org/
http://www.perl.org/
http://tomcat.apache.org/
http://www.adobe.com/products/dreamweaver/

Μάθετε PHP, MySQL και APACHE Όλα σε Ένα, Εκδόσεις Μ. Γκιούρδας

PHP Οδηγός Προγραμματισμού, Εκδόσεις Μ. Γκιούρδας