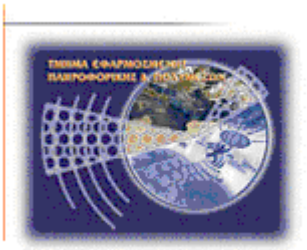
Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων



Πτυχιακή εργασία

Τίτλος: ***Ικανοποίηση περιορισμών σε χωρικές βάσεις δεδομένων μέσω διαδικτύου***

Τρουλάκη Στυλιανή (Α.Μ. 2016)

Επιβλέπων καθηγητής : Παπαδάκης Νικόλαος

Ημερομηνία παρουσίασης:25 Νοεμβρίου 2010

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον καθηγητή μου, κ. Νικόλαο Παπαδάκη για την εμπιστοσύνη που μου έδειξε αναθέτοντας μου τη συγκεκριμένη πτυχιακή εργασία και για τις πολύτιμες συμβουλές που μου έδωσε ώστε να την φέρω εις πέρας.

Τέλος, ένα πολύ μεγάλο ευχαριστώ στην οικογένεια μου που με στήριξε καθ' όλη τη διάρκεια των σπουδών μου και εξακολουθεί να με στηρίζει σε όλα μου τα βήματα.

Abstract

In this paper, we study the ramification problem in the setting of spatial databases.

We will describe the ramification problem in conventional and in spatial databases, explaining the difference between them and the reason that the solutions that are proposed for conventional databases cannot address the ramification problem in spatial databases. Next, we will describe the representation of spatial data in relational databases. Then, we will explain the solution we propose, describing the concept and the algorithms and we will give a complete example of the execution of the tool (Java program), describing what input should be given and what the results will be like. After that, we will explain the architecture of the system. In the end, we will present a proof of the correctness of the systems, the complexity of our solution and some evaluation results from the tests we made.

The key ideas of our approach are:

- To describe the problem in relational spatial databases.
- To development algorithms for discover and delete inconsistencies.
- To use dynamic rules to capture the direct effects of actions, and static rules to capture the indirect effects of actions.
- To develop a system which implement the solution in SQL for a relation databases.

In the following, we present the language SQL, which is used to write the algorithms which have the solution to the ramification problem, as well as the language PHP, which maintains databases. Finally, we make a reference to technology JSP, which also supports databases and presenting the reasons why we preferred the JSP to PHP.

Σύνοψη

Σε αυτή την εργασία, μελετάμε το πρόβλημα διακλάδωσης στον καθορισμό των χωρικών βάσεων δεδομένων.

Θα περιγράψουμε το πρόβλημα διακλάδωσης σε κλασσικές και σε χωρικές βάσεις δεδομένων, εξηγώντας τη διαφορά μεταξύ τους και τον λόγο που οι λύσεις που προτείνονται για τις κλασσικές βάσεις δεδομένων δεν μπορούν να αντιμετωπίσουν το πρόβλημα διακλάδωσης σε χωρικές βάσεις δεδομένων. Στη συνέχεια, θα περιγράψουμε την εκπροσώπηση των χωρικών δεδομένων σε σχεσιακές βάσεις δεδομένων. Έπειτα θα εξηγήσουμε τη λύση που προτείνουμε, περιγράφοντας τη γενική ιδέα και τους αλγορίθμους και θα δώσουμε ένα πλήρες παράδειγμα της εκτέλεσης του εργαλείου (Java program), περιγράφοντας τι είσοδο θα πρέπει να δώσουμε και πως θα μοιάζουν τα αποτελέσματα. Μετά από αυτό, θα εξηγήσουμε την αρχιτεκτονική του συστήματος. Στο τέλος θα παρουσιάσουμε μια απόδειξη της ορθότητας των συστημάτων, την πολυπλοκότητα της λύσης μας και τα μερικά αποτελέσματα αξιολόγησης από τις δοκιμές που κάναμε.

Οι βασικές ιδέες της προσέγγισής μας είναι οι εξής:

- Να περιγράψουμε το πρόβλημα σε σχεσιακές χωρικές βάσεις δεδομένων .
- Να αναπτύξουμε αλγορίθμους για την ανακάλυψη και την διαγραφή ασυνεπειών.
- Να χρησιμοποιήσουμε δυναμικούς κανόνες για τη σύλληψη των άμεσων επιπτώσεων των ενεργειών, και στατικούς κανόνες για τη σύλληψη των έμμεσων επιπτώσεων των ενεργειών.
- Να αναπτύξουμε ένα σύστημα που εφαρμόζει τη λύση σε SQL για μια σχεσιακή βάση δεδομένων.

Στα κεφάλαια που ακολουθούν γίνεται παρουσίαση της γλώσσας SQL , η οποία χρησιμοποιήθηκε για την εγγραφή των αλγορίθμων που έδωσαν τη λύση στο πρόβλημα της διακλάδωσης, όπως επίσης και της γλώσσας PHP, η οποία υποστηρίζει βάσεις δεδομένων.

Τέλος γίνεται αναφορά στην τεχνολογία JSP, που υποστηρίζει επίσης βάσεις δεδομένων και παρουσιάζονται οι λόγοι για τους οποίους προτιμήσαμε την JSP από την PHP.

Πίνακας Περιεχομένων

Πίνκας Εικόνων

ΚΕΦΑΛΑΙΟ 1ο

ΚΕΦΑΛΑΙΟ 2ο

ΚΕΦΑΛΑΙΟ 3ο

ΚΕΦΑΛΑΙΟ 4ο

Λίστα Πινάκων

ΚΕΦΑΛΑΙΟ 2ο

ΚΕΦΑΛΑΙΟ 3ο

ΚΕΦΑΛΑΙΟ 4ο

ΚΕΦΑΛΑΙΟ 1

Ένα εργαλείο για την αντιμετώπιση του προβλήματος διακλάδωσης σε χωρικές βάσεις δεδομένων: Μια λύση που εφαρμόζεται στην SQL

Περίληψη

Σε αυτή την εργασία, μελετάμε το πρόβλημα διακλάδωσης στον καθορισμό των χωρικών βάσεων δεδομένων. Πρότυπες λύσεις από τη βιβλιογραφία για την αιτιολογία σχετικά με τη δράση είναι ανεπαρκείς, διότι δεν μπορούν να συλλάβουν περιορισμούς ακεραιότητας σε χωρικά δεδομένα. Στην εργασία αυτή, θα δώσουμε λύση στο πρόβλημα διακλάδωσης με βάση την κατάσταση του διαφορικού λογισμού. Παρουσιάζουμε ένα εργαλείο που συνδέει τα θεωρητικά αποτελέσματα με πρακτικές εκτιμήσεις, με την προσκόμιση των κατάλληλων εντολών SQL, προκειμένου να αντιμετωπιστεί το πρόβλημα διακλάδωσης σε χωρικές βάσεις δεδομένων.

1.1 Εισαγωγή

Οι Χωρικές βάσεις δεδομένων μπορούν να εφαρμοστούν κυρίως σε Γεωγραφικά και Πληροφοριακά Συστήματα για την αναπαράσταση-μοντελοποίηση του χώρου και την αναπαράσταση των αντικειμένων σε αυτόν το χώρο (Chen & Gong, 1998; Oracle, 2003). Μπορούν επίσης να εφαρμοστούν σε πολυμέσα (επεξεργασία βίντεο) και σε δίκτυα (αναπαράσταση δικτύων, hubs).

Τα διάφορα είδη γεωμετρικών σχημάτων, ανάλογα με την πολυπλοκότητας τους, δεν καταστούν δυνατή την άμεση και ευέλικτη ανάλυση των προβλημάτων. Οι διαχειριστές, προκειμένου να είναι σε θέση να παρέχουν ορισμένες συγκρίσιμες ιδιότητες σε αυτά τα σχήματα, εισάγουν την έννοια του κουτιού οριοθέτησης. Το κουτί οριοθέτησης περικυκλώνει τα γεωμετρικά σχήματα σε ένα κουτί, του οποίου η περιοχή είναι η λιγότερο πιθανή, παρέχοντας έτσι έναν ευκολότερο τρόπο να χειριστεί κάποιος τα σχήματα. Ο ρόλος του κουτιού οριοθέτησης και η χρήση του από τους διαχειριστές σαν παράθυρο ερώτημα (window query), σημείο ερώτημα (point query), τομή ερώτημα (intersection query), περίβλημα ερώτημα (enclosure query), περιορισμός ερώτημα (containment query) και παρακείμενο ερώτημα (adjacent query) είναι πολύ σημαντικά εργαλεία για το χειρισμό των σχημάτων (Kiiveri, 1997).

Εξαιτίας της πολυπλοκότητας που μπορεί ένα σχήμα να έχει, αλλά και λόγω των πιθανών αλλαγών που ενδέχεται να προκύψουν, η χρήση των χωρικών φορέων σε μια χωρική βάση δεδομένων μπορεί να είναι μια δύσκολη διαδικασία, τόσο στην ενημέρωση όσο και στην εξεύρεση δεδομένων. Υπό αυτές τις συνθήκες, μπορούμε να κατατάξουμε τις μεθόδους φορέα σε τρεις ευρείες κατηγορίες: τις μονοδιάστατες μεθόδους πρόσβασης, τις ανά περιοχή μεθόδους πρόσβασης και τις ανά σημείο μεθόδους πρόσβασης. Οι πολυδιάστατες μέθοδοι πρόσβασης που αναφέρονται χρειάζονται εναλλακτικούς τρόπους για να έχουν πρόσβαση στα δεδομένα, ώστε να καταστεί δυνατή η σύγκριση των γεωμετρικών παραστάσεων στο διάστημα μεταξύ των k-διάστατων και μονοδιάστατων παραστάσεων.

Οι παρακάτω ενότητες θα αναφέρονται σε περιπτώσεις ασυνέπειας, οι οποίες μπορεί να συμβούν κατά την ενημέρωση των χωρικών δεδομένων. Πιο συγκεκριμένα, υποθέτουμε ότι τα χωρικά δεδομένα αποθηκεύονται ως μια ακολουθία σημείων (το ένα δίπλα στο άλλο) στο δισδιάστατο χώρο, και ότι μια αλλαγή σε ένα σχήμα παρουσιάζεται όταν αλλάζουν οι συντεταγμένες ενός σημείου . Σε περίπτωση που συμβαίνει μια τέτοια αλλαγή, υπάρχει η δυνατότητα να προκύψει επικάλυψη μεταξύ δύο γεωμετρικών σχημάτων. Αυτό είναι μια ασυνέπεια που αντιμετωπίζεται στο παρόν έγγραφο. Πιο συγκεκριμένα, προτείνουμε έναν αλγόριθμο, ο οποίος παράγει SQL κώδικα συναλλαγής. Όταν συμβαίνει μια αλλαγή στη βάση δεδομένων και προκύπτει μια αντίφαση, ο κώδικας αυτός είναι υπεύθυνος για την αλλαγή των παρακείμενων γεωμετρικών σχημάτων, έτσι ώστε η επικάλυψη μεταξύ τους να αφαιρεθεί. Η αλλαγή του πρώτου σχήματος είναι το άμεσο αποτέλεσμα της εκτέλεσης μιας συναλλαγής, ενώ η αλλαγή των υπόλοιπων σχημάτων είναι το έμμεσο αποτέλεσμα της εκτέλεσης μιας συναλλαγής και γίνεται για τη διατήρηση του περιορισμού της Μη-Επικάλυψης (Not-Overlap). Η

περιγραφή των έμμεσων επιδράσεων μιας συναλλαγής, όταν υπάρχουν περιορισμοί ακεραιότητας, είναι το πρόβλημα διακλάδωσης.

Η εγγύηση της συνοχής των δεδομένων που είναι αποθηκευμένα σε μια βάση δεδομένων είναι πολύ σημαντικό και δύσκολο πρόβλημα. Η συνοχή των δεδομένων καθορίζεται από την ικανοποίηση των περιορισμών ακεραιότητας (Andrea Rodriguez, 2004; Kalum, Li, & Wijeratne, 2006; Kiiveri, 1997; Oracle, 2003; Scott, 1994) στις διαφορετικές καταστάσεις της βάσης δεδομένων. Μία κατάσταση της βάσης δεδομένων θεωρείται έγκυρη (συνεπής) μόνο όταν πληρούνται όλοι οι περιορισμοί ακεραιότητας. Όταν εκτελείται μια συναλλαγή, το γενικό πλαίσιο της βάσης δεδομένων τροποποιείται. Στη νέα κατάσταση (που περιλαμβάνει τις άμεσες συνέπειες των πράξεων) η βάση δεδομένων μπορεί να είναι ασυνεπής επειδή ορισμένοι περιορισμοί ακεραιότητας δεν ικανοποιούνται. Ως εκ τούτου, είναι απαραίτητο να παραχθούν ορισμένα πρόσθετα αποτελέσματα (έμμεσες επιπτώσεις) για να ικανοποιηθούν οι περιορισμοί ακεραιότητας. Μπορούμε να υποθέσουμε ότι μια ατομική πράξη είναι μια ενέργεια.

Στο γενικό αυτό πλαίσιο, το πρόβλημα διακλάδωσης (McCarthy & Hayes, 1969) ασχολείται με τις έμμεσες επιπτώσεις των ενεργειών με την παρουσία των περιορισμών. Το πρόβλημα διακλάδωσης έχει μεγάλη σημασία στα συστήματα βάσεων δεδομένων. Οι χρήστες και οι σχεδιαστές βάσεων δεδομένων μπορεί να μην γνωρίζουν ακριβώς όλες τις έμμεσες επιδράσεις των συναλλαγών τους. Αυτό σημαίνει ότι οι χρήστες / σχεδιαστές μπορούν να εκτελέσουν μια συναλλαγή η οποία έχει ως αποτέλεσμα να παραβιάζει τους περιορισμούς ακεραιότητας. Η πιο προφανής λύση είναι να προσδιοριστούν χειροκίνητα όλα τα έμμεσα αποτελέσματα. Το πρόβλημα με αυτή τη λύση 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 προσδιοριστούν χειροκίνητα. Επίσης, οι χρήστες / σχεδιαστές μπορεί να μην γνωρίζουν όλες τις έμμεσες επιδράσεις των συναλλαγών τους. Έστω συστήματα βάσεων δεδομένων με πολλές εκατοντάδες συναλλαγές, πολλές εκατοντάδες περιορισμούς ακεραιότητας και περισσότερους από έναν σχεδιαστή. Σε αυτές τις βάσεις δεδομένων, με ένα μεγάλο αριθμό περιορισμών και συναλλαγών, οι έμμεσες επιπτώσεις μπορεί να είναι πάρα πολλές για να ανακαλυφθούν με το χέρι. Σημειώστε ότι η ίδια ακολουθία συναλλαγών ενδέχεται να έχει διαφορετικές έμμεσες επιπτώσεις, εάν το γενικό πλαίσιο της βάσης δεδομένων στην έναρξη της εκτέλεσης είναι διαφορετικό.

Στην παρακάτω ενότητα, θα περιγράψουμε το πρόβλημα διακλάδωσης σε κλασσικές και σε χωρικές βάσεις δεδομένων, εξηγώντας τη διαφορά μεταξύ τους και τον λόγο που οι λύσεις που προτείνονται για τις κλασσικές βάσεις δεδομένων δεν μπορούν να αντιμετωπίσουν το πρόβλημα διακλάδωσης σε χωρικές βάσεις δεδομένων. Στη συνέχεια, στο τμήμα 3, θα περιγράψουμε την εκπροσώπηση των χωρικών δεδομένων σε σχεσιακές βάσεις δεδομένων, όπως έχει προταθεί. Στη συνέχεια, στο τμήμα 4, θα εξηγήσουμε τη λύση που προτείνουμε, περιγράφοντας τη γενική ιδέα και τους αλγορίθμους. Στο τμήμα 5 θα δώσουμε ένα πλήρες παράδειγμα της εκτέλεσης του εργαλείου (Java program), περιγράφοντας τι είσοδο θα πρέπει να δώσουμε και πως θα μοιάζουν τα αποτελέσματα. Μετά από αυτό, στο τμήμα 6, θα εξηγήσουμε την αρχιτεκτονική του συστήματος. Στο τμήμα 7, θα παρουσιάσουμε μια απόδειξη της ορθότητας των συστημάτων, την πολυπλοκότητα της λύσης μας και τα μερικά αποτελέσματα αξιολόγησης από τις δοκιμές που κάναμε.

1.2 Το πρόβλημα διακλάδωσης σε κλασσικές και σε χωρικές βάσεις δεδομένων

1.2.1 Το πρόβλημα διακλάδωσης σε κλασσικές βάσεις δεδομένων

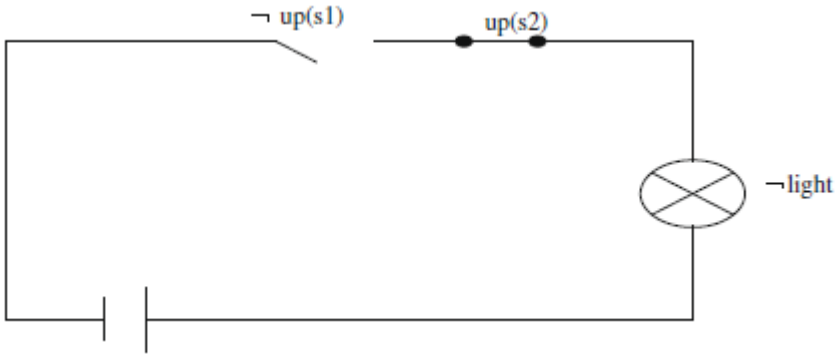
Το πρόβλημα διακλάδωσης είναι ένα πολύ δύσκολο πρόβλημα που ανακύπτει στη ρομποτική, στην τεχνολογία λογισμικού και στις βάσεις δεδομένων. Έχουμε εισαγάγει το πρόβλημα με ένα παράδειγμα. Ας υποθέσουμε ότι μας ενδιαφέρει η διατήρηση μιας βάσης δεδομένων που περιγράφει ένα απλό κύκλωμα (Εικ. 1), το οποίο έχει δύο διακόπτες και μία λάμπα. Η συμπεριφορά του κυκλώματος περιγράφεται από τους ακόλουθους περιορισμούς ακεραιότητας:

$$up(s_1) \wedge up(s_2) \equiv light, \quad (1)$$
$$\neg up(s_1) \Rightarrow \neg light, \quad (2)$$
$$\neg up(s_2) \Rightarrow \neg light. \quad (3)$$

Ο πρώτος περιορισμός συνεπάγεται ότι, όταν οι δύο διακόπτες είναι ενεργοποιημένοι, τότε η λάμπα είναι αναμμένη. Ο δεύτερος και ο τρίτος περιορισμός σημαίνουν ότι εάν ένας διακόπτης είναι απενεργοποιημένος, τότε η λάμπα δεν πρέπει να είναι αναμμένη.

Η ενέργεια toggle_switch αλλάζει την κατάσταση ενός διακόπτη, ως εξής:

$$toggle\_switch(s) \rightarrow up(s) \text{ if } \neg up(s),$$
$$toggle\_switch(s) \rightarrow \neg up(s) \text{ if } up(s).$$



Εικ. 1.1 Απλό ηλεκτρικό κύκλωμα.

Οι παραπάνω προτάσεις περιγράφουν τα άμεσα αποτελέσματα της ενέργειας toggle_switch. Μία κατάσταση είναι συνεπής όταν καλύπτει όλους τους περιορισμούς ακεραιότητας. Ας υποθέσουμε ότι το κύκλωμα είναι στην κατάσταση $S = \{\neg up(s_1), up(s_2), \neg light\}$. Η κατάσταση S είναι συνεπής, επειδή ικανοποιεί όλους τους περιορισμούς ακεραιότητας. Ας υποθέσουμε ότι έχουμε εκτελέσει την ενέργεια toggle_switch (s_1). Η ενέργεια αυτή έχει ως άμεση επίπτωση στην αλλαγή της κατάστασης του switch (s_1) από $\neg up(s_1)$ σε $up(s_1)$. Τώρα η κατάσταση του κυκλώματος είναι $S_1 = \{up(s_1), up(s_2), \neg light\}$. Η κατάσταση αυτή είναι ασυνεπής, διότι παραβιάζει τον πρώτο περιορισμό ακεραιότητας. Για να επιτευχθεί μία συνεπής κατάσταση πρέπει να αλλάξουμε την κατάσταση S_1 σε μία από τις ακόλουθες καταστάσεις:

$$S_2 = \{up(s_1), up(s_2), light\},$$
$$S_3 = \{up(s_1), \neg up(s_2), \neg light\}.$$

Και οι δύο καταστάσεις (S_2,S_3) είναι συνεπείς, διότι ικανοποιούνται όλοι οι περιορισμοί ακεραιότητας. Όπως παρατηρούμε, οι ενημερώσεις από $\neg light$ σε light στην κατάσταση S_2 ή η αλλαγή από $up(s_2)$ σε $\neg up(s_2)$ στην κατάσταση S_3 συμβαίνουν, ώστε να επιτευχθεί μία συνεπής κατάσταση και όχι ένα άμεσο αποτέλεσμα της ενέργειας toggle_switch(s_1).

Το εύλογο συμπέρασμα είναι ότι η λάμπα πρέπει να ανάβει. Για να συναχθεί το συμπέρασμα αυτό πρέπει να καθορίσουμε ποιες θα μπορούσε να είναι οι έμμεσες συνέπειες της ενέργειας. Το οποίο θα συζητήσουμε στο τμήμα 4. Παρατηρήστε ότι οι έμμεσες επιπτώσεις υφίστανται λόγω της παρουσίας των περιορισμών ακεραιότητας. Το πρόβλημα διακλάδωσης αναφέρεται στη συνοπτική περιγραφή των έμμεσων επιπτώσεων μιας ενέργειας παρουσία των περιορισμών. Διάφοροι τρόποι αντιμετώπισης του προβλήματος διακλάδωσης προτείνονται στη βιβλιογραφία. Η πλειοψηφία από αυτούς βασίζονται στην κατάσταση (McCarthy & Hayes, 1969) και στο συμβάν του διαφορικού λογισμού .

1.2.2 Βασική ορολογία στην κατάσταση του διαφορικού λογισμού

- Όλα τα κατηγορήματα και οι συναρτήσεις των οποίων η πραγματική αξία αλλάζει από μία παγκόσμια κατάσταση σε μία άλλη καλούνται fluents (κατηγορήματα).
- Μία πιθανή εξέλιξη του κόσμου είναι μια σειρά ενεργειών και αντιπροσωπεύεται από έναν όρο πρώτης τάξης, που ονομάζεται situation (κατάσταση).
- Μία ενέργεια μπορεί να αλλάξει την τιμή ορισμένων fluents.
- Μία κατάσταση είναι συνεπής όταν ικανοποιούνται όλοι οι περιορισμοί ακεραιότητας.
- Μια ενέργεια που πραγματοποιείται σε μια κατάσταση παράγει άλλη κατάσταση ως αποτέλεσμα.
- Η δυαδική συνάρτηση do ορίζεται ως εξής, με do(a,s) υποδηλώνοντας την κατάσταση που θα προκύψει από την εκτέλεση της ενέργειας a στην κατάσταση s.
- Μια ενέργεια μπορεί να εκτελεστεί, εφόσον πληρεί ορισμένες προϋποθέσεις. Καλούμε αυτές τις προϋποθέσεις, όπως έχουμε ήδη αναφέρει. Το δυαδικό κατηγόρημα Poss δηλώνει εάν κατέχει μία προϋπόθεση. Όταν το κατηγόρημα Poss(a,s) είναι αληθές τότε η ενέργεια a, μπορεί να εκτελεστεί στην κατάσταση s.

Μεταξύ των προτεινόμενων απλούστερων λύσεων είναι εκείνες με βάση την ελάχιστη αλλαγή προσέγγισης (Ginsberg & Smith, 1988; Winslett, 1988). Οι λύσεις αυτές δείχνουν ότι όταν μια ενέργεια εμφανίζεται σε μια κατάσταση S, κάποιος πρέπει να βρει τη συνεπή κατάσταση S′, η οποία έχει λιγότερες αλλαγές από την κατάσταση S. Για παράδειγμα, εξετάζουμε τη μοντελοποίηση ενός απλού κυκλώματος ως παράδειγμα. Έστω η κατάσταση $S = \{ up(s_1), up(s_2), \neg light\}$. Η ενέργεια toggle_switch ($up(s_1)$) αλλάζει την κατάσταση του κυκλώματος σε $S' = \{ up(s_1), up(s_2), \neg light \}$, το οποίο έρχεται σε αντίθεση. Υπάρχουν δύο συνεπείς καταστάσεις: $S_1 = \{ up(s_1), up(s_2), light \}$ και $S_2 = \{ up(s_1), \neg up(s_2), \neg light\}$. Είναι λογικό να ανάψει η λάμπα, ενώ ο διακόπτης s_2 δεν είναι απενεργοποιημένος. Είναι εύλογο για τη λάμπα να ανάβει ως έμμεσο αποτέλεσμα "ανεβάσματος" άλλου διακόπτη, αλλά δεν είναι λογικό να "κατεβαίνει" ένας διακόπτης ως έμμεσο αποτέλεσμα "ανεβάσματος" άλλου διακόπτη. Έτσι προτιμάμε την S_1 περισσότερο από την S_2. Η ελάχιστη αλλαγή προσέγγισης δεν μπορεί να επιλέξει ένα από αυτά, επειδή και τα δύο εξίσου είναι κοντά με την αρχική κατάσταση S. Οι λύσεις που βασίζονται στην κατηγοριοποίηση των fluents (Lifshitz, 1990, 1991, 1991) λύνουν το προηγούμενο πρόβλημα. Τα fluents κατηγοριοποιούνται σε πρωτογενή και δευτερογενή. Ένα πρωτεύον fluent μπορεί να αλλάξει μόνο ως άμεσο αποτέλεσμα μιας ενέργειας, ενώ ένα δευτερεύον μπορεί να αλλάξει μόνον ως έμμεσο αποτέλεσμα μιας ενέργειας. Μετά που μια ενέργεια λαμβάνει χώρα, θα επιλέξουμε την κατάσταση με τις λιγότερες αλλαγές στα πρωτεύοντα fluents. Στο προηγούμενο παράδειγμα, ο χωρισμός είναι $Fp = \{ up(s_1), up(s_2)\}$ και $Fs = \{ light \}$, όπου Fp και Fs είναι τα πρωτεύοντα και δευτερεύοντα fluents, αντίστοιχα. Τώρα επιλέγουμε την κατάσταση S_1, διότι δεν περιέχει οποιεσδήποτε αλλαγές στα πρωτογενή fluents. Η κατηγοριοποίηση των fluents λύνει το πρόβλημα διακλάδωσης μόνο αν όλα τα fluents μπορεί να κατηγοριοποιούνται. Αν κάποια fluents είναι πρωτεύοντα για ορισμένες ενέργειες και δευτερεύοντα για κάποιες άλλες αυτό δεν είναι μια ικανοποιητική λύση.

Μια άλλη λύση είναι η χρήση αιτιολογικών σχέσεων (Thielscher, 1997, 1988? McCain & Turner, 1978-1984). Κάθε αιτιολογική σχέση αποτελείται από δύο μέρη. Το πρώτο τμήμα, που ονομάζεται γενικό πλαίσιο, αποτελείται από μία φόρμουλα fluent η οποία, όταν είναι αληθής, θεσπίζει μια αιτιολογική σχέση μεταξύ της ενέργειας και του αποτελέσματός της. Το τελευταίο τμήμα είναι το έμμεσο αποτέλεσμα μιας ενέργειας (ονομάζεται η αιτία αυτού του αποτελέσματος).Μία αιτιολογική συσχέτιση έχει τη μορφή:

$$Є \text{ causes } \rho \text{ if } \Phi,$$

Όπου Є είναι μια ενέργεια, ρ είναι το άμεσο / έμμεσο αποτέλεσμα και Φ είναι μία φόρμουλα fluent η οποία εξαρτάται από το γενικό πλαίσιο.

Πρότυπες λύσεις για το πρόβλημα διακλάδωσης (Antonis & Rob, 1997; Clarke & Wing, 1996; Denecker & Ternovska, 2007; Dimitris & John, 1996; Elkan, 1992; Fusaoka, 1966; Kakas, Miller, &

Toni, 2001; Kowalski, 1992; Lifschitz, 1991, 1990; Lindsay, 1988; Marakakis, Kounali, & Vassilakis, 2006; McCain & Turner, 1995; Miller & Shanahan, 1999; Nikos & Dimitris, 2001, 2002a, 2002b; Papadakis & Plexousakis, 2003; Pinto, 1994; Pinto & Reiter, 1993; Reiter, 1996, 2001; Thielscher, 1997, 1988) βασίζονται πάνω στην επιμονή των fluents και στην ιδέα ότι οι ενέργειες έχουν επιπτώσεις στην επόμενη κατάσταση μόνο. Όπως θα εξηγήσουμε στο επόμενο τμήμα αυτό καθιστά αυτές τις λύσεις ανεπαρκείς για το πρόβλημα μέσα σε ένα χωροταξικό πλαίσιο.

1.2.3 Το πρόβλημα διακλάδωσης σε χωρικές βάσεις δεδομένων

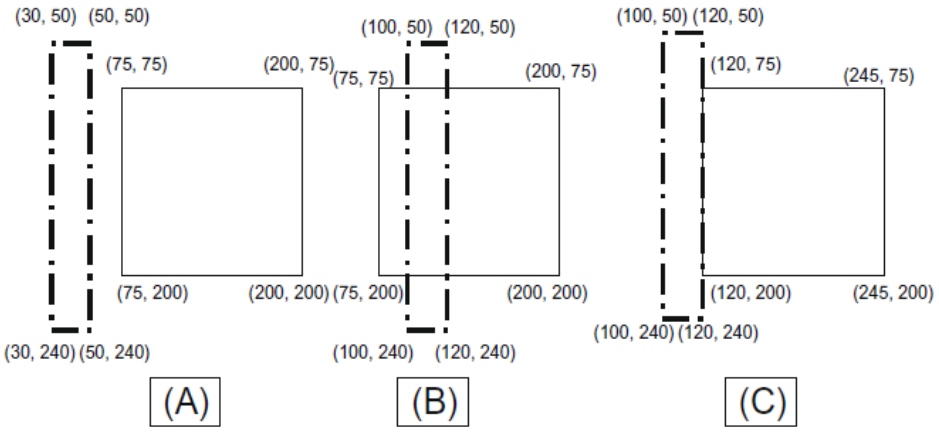
Στην ενότητα αυτή, παρουσιάζουμε το πρόβλημα διακλάδωσης σε χωρικές βάσεις δεδομένων. Το πρόβλημα διακλάδωσης αναφέρεται στην περιγραφή των έμμεσων αποτελεσμάτων μιας ενέργειας, όταν υφίστανται περιορισμοί ακεραιότητας. Οι περιορισμοί ακεραιότητας αναφέρονται στην τιμή των δεδομένων και πρέπει να ικανοποιούνται σε κάθε στιγμιότυπο της βάσης δεδομένων. Μπορούμε να θεωρήσουμε ότι μια ατομική συναλλαγή είναι μια ενέργεια. Έτσι, κάθε αλλαγή σε μια βάση δεδομένων είναι το αποτέλεσμα της εκτέλεσης ενός συνόλου ενεργειών, οι οποίες έχουν άμεσες και έμμεσες επιπτώσεις. Όταν μια ενέργεια αλλάζει τα δεδομένα της βάσης δεδομένων, μπορεί να προκύψει μία παραβίαση των περιορισμών ακεραιότητας . Στην περίπτωση αυτή, ορισμένες έμμεσες επιπτώσεις θα πρέπει να δημιουργούνται αυτόματα (πρόσθετες αλλαγές στο περιεχόμενο της βάσης δεδομένων), προκειμένου να ικανοποιήσουν το ICs.

Στη συνέχεια, παρουσιάζουμε το πρόβλημα διακλάδωσης με ένα παράδειγμα, όπως φαίνεται στην Εικ. 2. Υποθέτουμε ότι αυτή η βάση δεδομένων αποθηκεύει δύο σχήματα (το ένα με το διακεκομμένο περίγραμμα και το άλλο με το συμπαγές περίγραμμα), τα οποία δεν πρέπει να αλληλεπικαλύπτονται. Οι αριθμοί στις παρενθέσεις δείχνουν τις συντεταγμένες για κάθε σημείο ενός σχήματος (x, y). Όπως μπορείτε να δείτε, αυτή η εικόνα δείχνει τη βάση δεδομένων σε τρεις διαφορετικές καταστάσεις: (α) την αρχική κατάσταση, (β) την κατάσταση της βάσης δεδομένων μετά από τα άμεσα αποτελέσματα και, (γ) την κατάσταση της βάσης δεδομένων μετά από τα έμμεσα αποτελέσματα.

Υποθέτουμε ότι η αρχική κατάσταση της βάσης δεδομένων είναι εκείνη που υποβλήθηκε στην Εικ. 2A. Μπορούμε να δούμε ότι η βάση δεδομένων έχει δύο σχήματα αποθηκευμένα (το διακεκομμένο και το συμπαγές) και υπάρχει ένας περιορισμός ακεραιότητας ο οποίος υπαγορεύει ότι αυτά τα σχήματα δεν πρέπει να αλληλεπικαλύπτονται (όπως φαίνεται στο παράδειγμα, η αρχική κατάσταση της βάσης δεδομένων δεν έχει περιορισμούς-παραβιάσεις).

Στο επόμενο βήμα (Εικ. 2B), υποθέτουμε ότι έχουμε μια ενημέρωση στη βάση δεδομένων, όπως όταν ένας χρήστης της βάσης θέλει να "κουνήσει " το διακεκομμένο σχήμα για 70 μονάδες προς τα δεξιά (αύξηση της x συντεταγμένης). Έτσι, οι άμεσες συνέπειες είναι ότι το διακεκομμένο σχήμα έχει μεταφερθεί σε νέα τοποθεσία. Δυστυχώς, υπάρχουν ορισμένες έμμεσες επιπτώσεις, οι οποίες παραβιάζουν τους περιορισμούς ακεραιότητας της βάσης δεδομένων. Το διακεκομμένο σχήμα, μετά τη μετακίνηση, επικάλυψε το συμπαγές σχήμα, παραβιάζοντας τον Not_Overlap περιορισμό που υπήρχε μεταξύ τους.

Στο τρίτο και τελικό βήμα (Εικ. 2C), μπορούμε να δούμε πώς το πρόγραμμα διόρθωσε το πρόβλημα διακλάδωσης με την προσαρμογή της θέσης του συμπαγούς σχήματος, έτσι ώστε να μην συμπίπτει με το διακεκομμένο σχήμα. Κινώντας το συμπαγές σχήμα στη νέα αυτή θέση, το πρόγραμμα μας διαβεβαίωσε για τη διατήρηση των περιορισμών ακεραιότητας των βάσεων δεδομένων.



Εικ. 1.2. Ένα παράδειγμα του προβλήματος διακλάδωσης και ο τρόπος που επιλύεται από το πρόγραμμα.

Παρατηρήστε ότι οι έμμεσες επιπτώσεις υφίστανται λόγω της παρουσίας των περιορισμών ακεραιότητας. Το πρόβλημα διακλάδωσης αναφέρεται στη συνοπτική περιγραφή των έμμεσων επιπτώσεων της ενέργειας, παρουσία των περιορισμών.

Όπως παρατηρούμε, κάθε λύση που έχει προταθεί για το πρόβλημα διακλάδωσης των κλασσικών βάσεων δεδομένων δεν μπορεί να επιλύσει το πρόβλημα σε χωρικές βάσεις δεδομένων, επειδή δεν παρέχεται μηχανισμό ο οποίος:

1.Να ανιχνεύει ότι υπάρχει επικάλυψη μεταξύ των δύο σχημάτων.

2.Να περιγράφει πώς η αξία πολλών από τα σημεία συνολικά θα πρέπει να αλλάξει, έτσι ώστε ο Not_Overlap περιορισμός να μην παραβιάζεται.

Σε αυτή την εργασία, περιγράφουμε:

1. Ένα σύστημα που παράγει SQL κώδικα, ο οποίος είναι υπεύθυνος για την αυτόματη παραγωγή έμμεσων αποτελεσμάτων. Η είσοδος του συστήματος είναι τα ονόματα των πινάκων και των στηλών της βάσης δεδομένων και επίσης οι περιορισμοί ακεραιότητας που υφίστανται. Με αυτή την είσοδο, το σύστημα παράγει τον απαιτούμενο SQL κώδικα για τη διατήρηση των περιορισμών ακεραιότητας των δεδομένων.

2.Αλγορίθμους που ανιχνεύουν την παραβίαση των περιορισμών μιας βάσης δεδομένων και παράγουν τις απαιτούμενες έμμεσες επιπτώσεις, προκειμένου να επιτυγχάνεται πάλι μια συνεπής κατάσταση. Κάθε ένας από αυτούς τους αλγορίθμους έχει ένα συγκεκριμένο σκοπό είτε την ανίχνευση παραβίασης των περιορισμών της βάσης δεδομένων ή τον υπολογισμό των έμμεσων επιπτώσεων που θα πρέπει να παράγονται. Οι αλγόριθμοι αυτοί εξασφαλίζουν ότι, μετά την εκτέλεσή τους η βάση δεδομένων θα είναι συνεπής - είτε με την παραγωγή ορισμένων έμμεσων επιπτώσεων ή από την ανέπαφη έξοδο από την βάση δεδομένων (υπό την προϋπόθεση ότι δεν δημιουργήθηκαν αλλαγές).

1.3 Η εκπροσώπηση των χωρικών δεδομένων σε σχεσιακή βάση δεδομένων

Υποθέτουμε ότι ο πίνακας σε μια βάση δεδομένων με πληροφορίες για τα σχήματα (πολύγωνα) δημιουργείται από τα επόμενα πεδία:

Polygons table	
Polygon ID	το id του πολυγώνου στη βάση
Serial Number	το serial number αυτού του σημείου στη βάση
X	η x συντεταγμένη αυτού του σημείου
Y	η y συντεταγμένη αυτού του σημείου

Ένα παράδειγμα ενός πολυγώνου, που ονομάζεται A παρασκευάζεται από την ένωση των σημείων A, B, C και D, θα ήταν να προστεθούν στον πίνακα οι επόμενες τέσσερις σειρές:

1. A 0 Ax Ay

2. A 1 Bx By

3. A 2 Cx Cy

4. A 3 Dx Dy

Ξεκινάμε από την αναζήτηση του σημείου που έχει polygon_ID A και Serial Number 0. Ενώνουμε αυτό με το επόμενο σημείο (το σημείο με polygon_ID A και Serial Number 1). Συνεχίζουμε αυτή τη διαδικασία μέχρι να ενώσουμε το τελευταίο σημείο με το προηγούμενό του. Το τελικό βήμα είναι να ενώσουμε το τελευταίο σημείο με το πρώτο (στο παράδειγμα αυτό το σημείο D, που έχει Serial Number 3 με το σημείο A). Έτσι, το πολύγωνο A είναι το σχήμα που δημιουργείται από την ένωση του σημείου A με το σημείο B, έπειτα του σημείου B με το σημείο C, έπειτα του σημείου C με το σημείο D και, τέλος, του σημείου D με το σημείο A (το τελευταίο σημείο με το πρώτο σημείο).

Το εργαλείο (Java program) παίρνει την είσοδο από το GUI (Εικ. 4), που είναι το όνομα του πίνακα στη βάση δεδομένων που περιέχει τα πολύγωνα, το όνομα της στήλης με το κλειδί (id) για τα πολύγωνα, τον τύπο από το key field στη βάση δεδομένων, το όνομα της στήλης για το serial number του σημείου για ένα πολύγωνο, το όνομα στήλης με τη x συντεταγμένη και το όνομα της στήλης με τη y συντεταγμένη. Η τελευταία και πιο σημαντική είσοδος που παίρνει το πρόγραμμα είναι οι κανόνες που έδωσε ο χρήστης και περιγράφουν τους περιορισμούς της βάσης. Αυτοί οι κανόνες δίνονται στην μορφή:

The Not_Overlap rule: Not_Overlap(A, B);

Αυτό σημαίνει ότι το πολύγωνο με id A δεν πρέπει να αλληλεπικαλύπτεται με το πολύγωνο που έχει id B. Αν τα A και B είναι τύπου varchar , θα πρέπει να γραφτούν 'A' και 'B'.

The Connect rule: Connect(A, B);

Αυτό σημαίνει ότι το πολύγωνο με id A πρέπει να συνδέεται με το πολύγωνο που έχει id B (έτσι , θα πρέπει να μετακινηθούν μαζί). Η προηγούμενη σημείωση ισχύει και εδώ, δηλαδή, αν τα A και B είναι τύπου varchar , θα πρέπει να γραφτούν οπότε ο αλγόριθμος επιστρέφει οπότε ο αλγόριθμος επιστρέφει οπότε ο αλγόριθμος επιστρέφει 'A' και 'B'.

Στη συνέχεια, παράγονται δύο αρχεία. Ένα με όνομα code.pl, το οποίο περιέχει τον κώδικα pl/SQL, ο οποίος είναι υπεύθυνος για την επιδιόρθωση και τη διατήρηση των περιορισμών της βάσης δεδομένων όπως περιγράφτηκαν στην είσοδο. Το δεύτερο αρχείο ονομάζεται drop- Code.pl και περιέχει τον κώδικα SQL που χρειάζεται για την πτώση των πινάκων, των διαδικασιών και των λειτουργιών που δημιουργήθηκαν από την εκτέλεση του αρχείου code.pl. Τώρα, τι κάνει ο κώδικας pl/SQL που περιέχεται στο code.pl αρχείο:

- Αρχικά δηλώνει δύο πίνακες:

- Τον NotOverlapConstraints: Αυτός ο πίνακας περιέχει δύο πεδία, που ονομάζονται p1 και p2, τα οποία είναι και τα δύο του ίδιου τύπου με τη στήλη κλειδί του πίνακα πολύγωνα. Κάθε γραμμή που εγγράφεται σε αυτόν τον πίνακα, σημαίνει ότι τα πολύγωνα p1 και p2 δεν θα πρέπει να αλληλεπικαλύπτονται.

-Τον ConnectConstraints: Αυτός ο πίνακας περιέχει δύο πεδία, που ονομάζονται p1 και p2, τα οποία είναι και τα δύο του ίδιου τύπου με τη στήλη κλειδί του πίνακα πολύγωνα. Κάθε γραμμή που εγγράφεται σε αυτόν τον πίνακα, σημαίνει ότι τα πολύγωνα p1 και p2 θα πρέπει να συνδεθούν, έτσι ώστε να κινούνται μαζί.

- Εισάγει σε αυτούς τους πίνακες τις τιμές που δίνονται ως είσοδο από το χρήστη, περιγράφοντας τους περιορισμούς της βάσης δεδομένων.

- Αναγνωρίζει τις ακόλουθες λειτουργίες και διαδικασίες.

– FUNCTION boundingBoxContainsPoint (pol1 VARCHAR2, pol2 VARCHAR2, point INTEGER) RETURN INTEGER

– FUNCTION polygonContainsPoint (pol1 VARCHAR2, pol2 VARCHAR2, point INTEGER) RETURN INTEGER

– FUNCTION polygonContainsPolygon (p1 VARCHAR2, p2 VARCHAR2) RETURN INTEGER

– FUNCTION edgesIntersect (pol1 VARCHAR2, point1 integer, point2 integer, pol2 VARCHAR2, point3 integer, point4 integer) RETURN INTEGER

– FUNCTION polygonIntersectsPolygon (p1 VARCHAR2, p2 VARCHAR2) RETURN INTEGER

– FUNCTION polygonOverlapsPolygon (p1 VARCHAR2, p2 VARCHAR2) RETURN INTEGER

– PROCEDURE movePolygon (polygon VARCHAR2, movex integer, movey integer)

– FUNCTION countStepsToMove(polygon VARCHAR2, movex

integer, movey integer) RETURN INTEGER – PROCEDURE resolveOverlap (polygon VARCHAR2)

– PROCEDURE main_proc

Κάθε μία από αυτές τις λειτουργίες και τις διαδικασίες (εκτός από τη main_proc) εφαρμόζει ένα συγκεκριμένο αλγόριθμο, τον οποίο θα περιγράψουμε παρακάτω:

● Η διαδικασία main_proc είναι η μόνη υπεύθυνη για την επιδιόρθωση και τη διατήρηση των περιορισμών της βάσης δεδομένων. Θα πρέπει να κληθεί να τρέχει από την χρήστη (π.χ. με την πληκτρολόγηση της exec main_proc). Αυτό που κάνει η main_proc είναι να ψάχνει στον πίνακα NotOverlapConstraints για να βρει αν υπάρχουν not_overlap περιορισμοί. Για κάθε έναν από τους περιορισμούς που βρέθηκαν (για κάθε γραμμή του πίνακα NotOverlapConstraints), η διαδικασία καλεί τη λειτουργία polygonOverlapsPolygon, με τα δύο πολύγωνα στον πίνακα να είναι τα ορίσματα, για να βρει αν παραβιάστηκε ο περιορισμός (περίπτωση που τα δύο πολύγωνα επικαλύφτηκαν ή όχι). Αν διαπιστώσει ότι ο περιορισμός αυτός έχει παραβιαστεί, καλεί τη διαδικασία resolveOverlap, δίνοντας ως όρισμα ένα από τα δύο πολύγωνα. Η διαδικασία resolveOverlap θα προσπαθήσει να μετακινήσει το δεδομένο πολύγωνο μακριά από εκείνο που το επικάλυψε (προσπαθώντας να βρει το συντομότερο τρόπο για να πάρει μια non-obstracting θέση για αυτό). Λαμβάνει επίσης υπόψην ότι το δεδομένο πολύγωνο μπορεί να συνδέεται με άλλα πολύγωνα, έτσι τοποθετεί το πολύγωνο σε μια τέτοια θέση που ούτε το συγκεκριμένο πολύγωνο, ούτε τα πολύγωνα που συνδέονται με αυτό, να επικαλύπτονται με πολύγωνα που δεν πρέπει.

1.4 Προτεινόμενη λύση

Το μόνο πράγμα που θα πρέπει να κάνει ο χρήστης μιας βάσης δεδομένων, προκειμένου να ελέγχει και να διατηρεί τους κανόνες ακεραιότητας της βάσης είναι να εκτελέσει τη main_proc διαδικασία (π.χ. με την πληκτρολόγηση "exec main_proc" στην SQL γραμμή εντολών). Αυτή η διαδικασία θα ανακτήσει τους Not_overlap περιορισμούς από τον πίνακα που τους περιέχει και για κάθε γραμμή που θα ανακτηθεί θα καλέσει τον αλγόριθμο polygonOverlapsPolygon να ελέγξει κατά πόσο υφίσταται επικάλυψη μεταξύ αυτών των δύο σχημάτων. Αν αυτός ο αλγόριθμος επιστρέψει αληθές αποτέλεσμα, η main_proc θα καλέσει τον resolveOverlap αλγόριθμο να επιλύσει αυτή την επικαλύψει και, έπειτα, θα συνεχίσει με την επόμενη γραμμή.

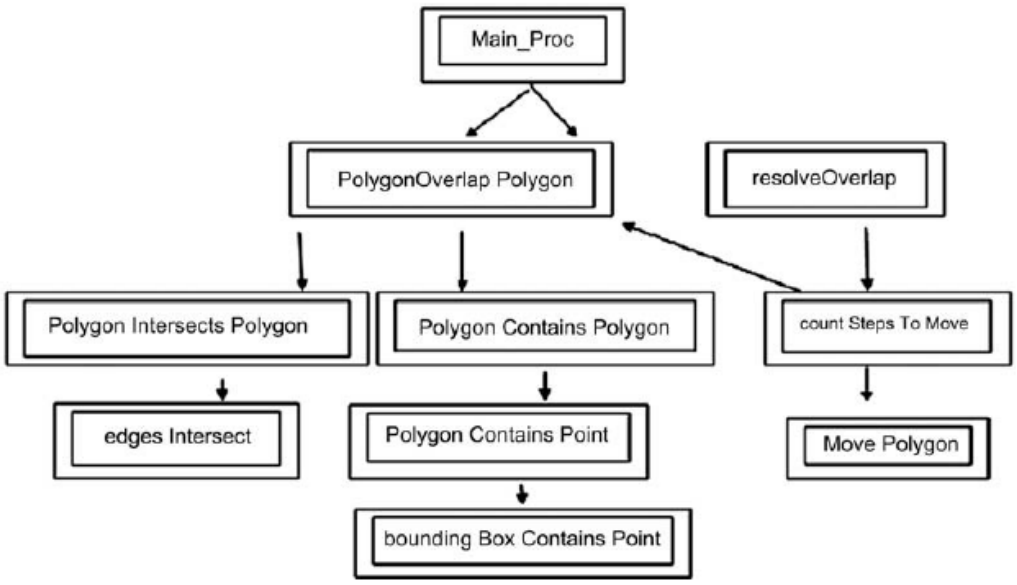
Ο αλγόριθμος PolygonOverlapsPolygon θα καλέσει πρώτα τον polygonIntersectsPolygonAlgorithm να ελέγξει αν υπάρχουν τεμνόμενες ακμές μεταξύ των δύο πολυγώνων. Αν υπάρχει μια διασταύρωση, επιστρέφει αληθές, αλλιώς θα καλέσει τον αλγόριθμο polygonContainsPolygon να ελέγξει εάν ένα από τα δύο πολύγωνα είναι εντελώς "μέσα" στο άλλο πολύγωνο. Αν αυτή είναι η περίπτωση, επιστρέφει αληθές, διαφορετικά επιστρέφει ψευδές.

Ο αλγόριθμος PolygonIntersectsPolygon θα προσπαθήσει να ελέγξει αν το πρώτο πολύγωνο έχει μία ακμή που διασταυρώνεται με μια ακμή του δεύτερου πολυγώνου. Αυτό θα γίνει με τη βοήθεια του edgesIntersect αλγορίθμου (ο αλγόριθμος αυτός θα κληθεί για κάθε ζεύγος ακμών μεταξύ του πρώτου και του δεύτερου πολυγώνου). Αν διαπιστωθεί μια διασταύρωση ο αλγόριθμος θα επιστρέψει αληθές, διαφορετικά θα επιστρέψει ψευδές.

Εάν δεν υπάρχουν ακμές που να τέμνονται, ο polygonOverlapsPolygon αλγόριθμος θα καλέσει τον αλγόριθμο PolygonContainsPolygon (όπως προαναφέρθηκε).

Αυτός ο αλγόριθμος θα ελέγχει εάν το ένα από τα δύο πολύγωνα περιέχει το άλλο. Έτσι, ο αλγόριθμος αυτός θα καλέσει τον αλγόριθμο polygonContainsPoint για αυτά τα δύο πολύγωνα. Ο αλγόριθμος polygonContainsPoint θα ελέγξει αρχικά, χρησιμοποιώντας τον αλγόριθμο boundingBoxContainsPoint, αν το πλαίσιο οριοθέτησης του πολυγώνου περιέχει το συγκεκριμένο σημείο. Αν το κάνει, θα προσπαθήσει να ελέγξει κατά πόσο το δεδομένο πολύγωνο περιέχει το συγκεκριμένο σημείο και να επιστρέψετε αληθές ή ψευδές αναλόγως.

Όπως προαναφέρθηκε, αν η main_proc βρει μια επικάλυψη (μέσω της χρήσης του αλγορίθμου polygonOverlapsPolygon), θα καλέσει τον resolveOverlap αλγόριθμο για την επίλυση αυτής της επικάλυψης. Ο αλγόριθμος αυτός θα χρησιμοποιήσει τον countStepsToMove αλγόριθμο, έτσι ώστε να βρει την καλύτερη λύση (αυτή με τα ελάχιστα απαιτούμενα βήματα) και στη συνέχεια θα καλέσει τον αλγόριθμο movePolygon να μετακινήσει το πολύγωνο στη νέα θέση. Ο countStepsToMove αλγόριθμος θα προσπαθήσει να μετακινήσει το πολύγωνο σε μια νέα θέση και να ελέγξει κατά πόσο εξακολουθεί το πολύγωνο να επικαλύπτεται ή όχι. Έτσι, θα χρησιμοποιήσει τον αλγόριθμο movePolygon και στη συνέχεια τον polygonOverlapsPolygon αλγόριθμο. Στην Εικ. 3, υπάρχει ένα διάγραμμα που εμφανίζει τους αλγόριθμους που χρησιμοποιούνται και τον τρόπο που αλληλεπιδρούν μεταξύ τους (εμφανίζει, για κάθε αλγόριθμο, ποιους άλλους αλγόριθμους χρησιμοποιεί).



Εικ. 1.3. Οι αλγόριθμοι που χρησιμοποιούνται και ο τρόπος που αλληλεπιδρούν μεταξύ τους.

1.4.1 Αλγόριθμοι

1.4.1.1 Ο BoundingBoxContainsPoint αλγόριθμος

Ο αλγόριθμος BoundingBoxContainsPoint χρησιμοποιείται για να αποφασίσουμε αν το πλαίσιο οριοθέτησης ενός πολυγώνου (το τετράγωνο που περιέχει πλήρως το δεδομένο πολύγωνο) περιέχει ένα συγκεκριμένο σημείο ενός άλλου πολυγώνου ή όχι.

Ο BoundingBoxContainsPoint αλγόριθμος:

```
Parameters IN
   pol1  : the key for the first polygon
   pol2  : the key for the second polygon
   point : the serial number of the second polygon's point.
Returns Integer:
   1: if the bounding box of the pol1 polygon contains the given
point.
   0: otherwise
1. Set minx, maxx :=  pol1's first point's x coordinate.
       miny, maxy :=  pol1's first point's y coordinate.
2. For each row of the polygon table that contains a new point of
pol1 do
3. polx :=  row.x;
   poly :=  row.y;
4. IF minx < polx THEN
   minx :=  polx;
END IF;
5. IF maxx > polx THEN
   maxx :=  polx;
END IF;
6. IF miny < poly THEN
   miny :=  poly;
END IF;
7. IF maxy > poly THEN
   maxy :=  poly;
END IF;
After checking all pol1's points do
8. IF pointx > =minx AND pointx < =maxx AND pointy > =miny
AND pointy < =maxy
THEN
   RETURN 1;
END IF;
9. RETURN 0;
```

Στα βήματα 4-7 ο αλγόριθμος προσπαθεί να βρει τις ελάχιστες και μέγιστες τιμές για τις x και y συντεταγμένες του πολυγώνου pol1. Μετά από όλους αυτούς τους ελέγχους, αλγόριθμος ελέγχει κατά πόσο το δεδομένο σημείο είναι μέσα στο πλαίσιο οριοθέτησης ή όχι. Αν το σημείο είναι μέσα στο πλαίσιο οριοθέτησης, τότε , η x συντεταγμένη του θα πρέπει να είναι μέσα στην ελάχιστη και στη μέγιστη x συντεταγμένη του πλαισίου οριοθέτησης (pointx > = minx AND pointx <= maxx). Το ίδιο ισχύει και για την y συντεταγμένη (pointy > = miny AND pointy <= maxy). Έτσι, ο αλγόριθμος επιστρέφει 1 (αληθές).

Αν το σημείο είναι έξω από το πλαίσιο οριοθέτησης, τότε, είτε η x και / είτε η y του συντεταγμένη δεν θα είναι μέσα στις ελάχιστες και στις μέγιστες x και y συντεταγμένες του πλαισίου οριοθέτησης, και γι'αυτό ο αλγόριθμος επιστρέφει 0 (ψευδές).

1.4.1.2 Ο PolygonContainsPoint αλγόριθμος

Ο αλγόριθμος PolygonContainsPoint χρησιμοποιείται για να αποφασίσει κατά πόσον ένα πολύγωνο περιέχει ένα συγκεκριμένο σημείο ενός δεύτερου πολυγώνου ή όχι. Χρησιμοποιεί τον προηγούμενο αλγόριθμο (boundingBoxContainsPoint) για να βεβαιωθεί ότι το πλαίσιο οριοθέτησης του πρώτου πολυγώνου περιέχει αυτό το σημείο. Αν αυτό δεν συμβαίνει, τότε το πολύγωνο σίγουρα δεν θα περιέχει ούτε αυτό το σημείο.

Ο PolygonContainsPoint αλγόριθμος:

```
Parameters IN
   pol1  : the key for the first polygon
   pol2  : the key for the second polygon
   point : the serial number of the second polygon's point.
Returns Integer:
   1: if polygon pol1 contains the given point of polygon  pol2.
   0: otherwise
1. IF boundingBoxContainsPoint(pol1, pol2, 0) = 0 THEN
   RETURN 0;
END IF;
2. SET
   px, py := x and y coordinates of the pol2 polygon's point to be
checked.
   hits := 0;
   lastx, lasty := x and y coordinates of pol1's last point.
3. For each row of the polygon table that contains pol1's next point
in order do
4. curx := row.x
   cury := row.y;
5. IF cury = lasty THEN
   goto continueLabel;
END IF;
6. IF curx < lastx THEN
   IF px >= lastx THEN
      goto continueLabel;
   END IF;
   leftx := curx;
ELSE
   IF px >= curx THEN
      goto continueLabel;
   END IF;
   leftx := lastx;
END IF;
7. IF cury < lasty THEN
   IF py < cury OR py >= lasty THEN
      goto continue Label;
   END IF;
   IF px < leftx THEN
      hits :=hits + 1;
      goto continue Label;
   END IF;
   test1 := px - curx;
   test2 := py - cury;
ELSE
   IF py < lasty OR py >= cury THEN
      goto continue Label;
   END IF;
   IF px < leftx THEN
      hits := hits + 1;
      goto continue Label;
   END IF;
   test1 := px - lastx;
   test2 := py - lasty;
END IF;
```

```
8. IF test1 < (test2/(lasty - cury) * (lastx - curx)) THEN
   hits := hits + 1;
END IF;
9. <<continue Label>>
   lastx := curx;
   lasty := cury;
After checking all pol1's points
10. IF bitand(hits, 1) ! = 0 THEN
    RETURN 1;
END IF;
11. RETURN 0;
```

1.4.1.3 Ο PolygonContainsPolygon αλγόριθμος

Ο αλγόριθμος PolygonContainsPolygon καθορίζει αν ένα συγκεκριμένο πολύγωνο περιέχει πλήρως (ή περιέχεται από) άλλο πολύγωνο ή όχι. Ο αλγόριθμος χρησιμοποιεί την υπόθεση ότι, αν ένα πολύγωνο περιέχει πλήρως ένα άλλο πολύγωνο, τότε θα περιέχει ΟΛΑ τα σημεία του πολυγώνου. Έτσι, ελέγχει μόνον αν περιέχει το πρώτο σημείο του δεύτερου πολυγώνου (ή εάν το δεύτερο πολύγωνο περιέχει το πρώτο σημείο του πρώτου πολυγώνου).

Ο PolygonContainsPolygon αλγόριθμος:

```
Parameters IN
   pol1: the key for the first polygon
   pol2: the key for the second polygon
Returns Integer:
   1: if polygon pol1 contains (or is contained by) polygon   pol2.
   0: otherwise
1. IF polygonContainsPoint(p1, p2, 0) = 1 THEN
    RETURN 1;
END IF;
2. IF polygonContainsPoint(p2, p1, 0) = 1 THEN
    RETURN 1;
END IF;
3. RETURN 0;
```

Στο πρώτο βήμα, ο αλγόριθμος ελέγχει αν το πρώτο πολύγωνο περιέχει ένα σημείο του δεύτερου πολυγώνου. Αν συμβαίνει αυτό, τότε ο αλγόριθμος υποθέτει ότι περιέχει πλήρως το δεύτερο πολύγωνο (επειδή θα έχει ελεγχθεί ότι δεν υπάρχουν καθόλου ακμές που να διασταυρώνουν η μία την άλλη μεταξύ των δύο πολυγώνων), οπότε ο αλγόριθμος επιστρέφει 1 (αληθές). Στο δεύτερο βήμα, ο αλγόριθμος ελέγχει αν το δεύτερο πολύγωνο περιέχει ένα σημείο του πρώτου. Και πάλι, αν αυτό συμβαίνει, ο αλγόριθμος επιστρέφει 1 (αληθές). Διαφορετικά, αυτό σημαίνει ότι ούτε το πρώτο πολύγωνο περιέχει το δεύτερο ούτε το δεύτερο το πρώτο, οπότε ο αλγόριθμος επιστρέφει 0 (ψευδές)

1.4.1.4 Ο EdgesIntersect αλγόριθμος

Ο EdgesIntersect αλγόριθμος καθορίζει εάν οι δύο ακμές που περιγράφονται από τέσσερα σημεία (edge1 = point1 - point2, edge2 = point3 - point4) τέμνονται ή όχι.

Ο EdgesIntersect αλγόριθμος:

```
Parameters IN:
   Pol1: the key for the first polygon
   Pol2: the key for the second polygon
   Point1: pol1 edge's first point's serial number
   Point2: pol1 edge's second point's serial number
   Point3: pol2 edge's first point's serial number
   Point4: pol2 edge's second point's serial number
Returns Integer:
```

```
   1: If the two edges intersect
   0: otherwise
1. x1, y1 := point1's x and y coordinates
   x2, y2 := point2's x and y coordinates
   x3, y3 := point3's x and y coordinates
   x4, y4 := point4's x and y coordinates
2. IF x1 <= x2 THEN
   left1x := x1;
   left1y := y1;
   right1x := x2;
   right1y := y2;
ELSE
   left1x := x2;
   left1y := y2;
   right1x := x1;
   right1y := y1;
END IF;
3. IF x3 <= x4 THEN
   left2x := x3;
   left2y := y3;
   right2x := x4;
   right2y := y4;
ELSE
   left2x := x4;
   left2y := y4;
   right2x := x3;
   right2y := y3;
END IF;
4. IF left1x > left2x THEN
   tmpx := left1x;
   left1x := left2x;
   left2x := tmpx;
   tmpx := right1x;
   right1x := right2x;
   right2x := tmpx;
   tmpy := left1y;
   left1y := left2y;
   left2y := tmpy;
   tmpy := right1y;
   right1y := right2y;
   right2y := tmpy;
END IF;
5. IF left1x < left2x AND right1x > left2x THEN
   IF left1y <= left2y AND right1y > left2y THEN
     RETURN 1;
   END IF;
   IF left1y <= right2y AND left1y > left2y THEN
     RETURN 1;
   END IF;
END IF;
6. RETURN 0;
```

Στο πρώτο βήμα ο αλγόριθμος αρχικοποιεί τις μεταβλητές. Στο δεύτερο και στο τρίτο βήμα, ο αλγόριθμος βρίσκει το πιο αριστερό σημείο για κάθε μία από τις ακμές και θέτει τις μεταβλητές left1, right1 και left2, right2 αναλόγως. Στο τέταρτο βήμα, ο αλγόριθμος καθορίζει ποιο από τα δύο "αριστερά" σημεία των δύο ακμών είναι το πιο αριστερό και ανταλλάσει ή διατηρεί αμετάβλητες τις

μεταβλητές. Στη συνέχεια, στο πέμπτο βήμα ο αλγόριθμος ελέγχει αν οι δύο ακμές διασταυρώνουν η μία την άλλη. Ο αλγόριθμος καθορίζει εάν οι ακμές τέμνονται ή όχι εξετάζοντας αν οι συντεταγμένες x και y της ακμής που βρίσκεται πιο αριστερά είναι μέσα στις x και y συντεταγμένες της άλλης ακμής. Αν το κάνουν, ο αλγόριθμος επιστρέφει 1 (αληθές), διαφορετικά επιστρέφει 0 (ψευδής).

1.4.1.5 Ο PolygonIntersectsPolygon αλγόριθμος

Ο αλγόριθμος PolygonIntersectsPolygon καθορίζει για δύο δοσμένα πολύγωνα, εάν υπάρχει μία ακμή από το πρώτο πολύγωνο που διασταυρώνεται με μια ακμή από το δεύτερο πολύγωνο. Χρησιμοποιεί τον προηγούμενο αλγόριθμο (edgesIntersect), προκειμένου να βρει αν οι δύο ακμές τέμνονται.

Ο PolygonIntersectsPolygon αλγόριθμος:

```
Parameters IN:
   Pol1: the key for the first polygon
   Pol2: the key for the second polygon
Returns Integer:
   1: if the two polygons have edges that intersect
   0: otherwise
1. point1 := pol1's polygon last point's serial number
2. For each row with the next pol1's point in order do
3. point2 := row.serial;
4. point3 := pol2's polygon last point's serial number
5. For each row with the next pol2's point in order do
6. point4 := row.serial;
7. IF edgesIntersect (pol1, pol2, point1, point2, point3, point4)
THEN
   RETURN 1;
   END IF;
8. point3 := point4;
9. End the loop starting at step5.
10. point1 := point2;
11. End the loop starting at step 2.
12. RETURN 0;
```

1.4.1.6 Ο PolygonOverlapsPolygon αλγόριθμος

Ο αλγόριθμος PolygonOverlapsPolygon αποφασίζει εάν δύο συγκεκριμένα πολύγωνα επικαλύπτονται. Χρησιμοποιεί τους PolygonIntersectsPolygon και PolygonContainsPolygon αλγορίθμους.

Ο PolygonOverlapsPolygon αλγόριθμος:

```
Parameters IN:
   Pol1: the key for the first polygon
   Pol2: the key for the second polygon
Returns Integer:
   1: if the two polygons overlap
   0: otherwise
1. IF polygonIntersectsPolygon(p1, p2) = 1 THEN
      RETURN 1;
END IF;
2. IF polygonContainsPolygon(p1, p2) = 1 THEN
      RETURN 1;
END IF;
3.   RETURN 0;
End PolygonOverlapsPolygon;
```

Στο πρώτο βήμα, ο αλγόριθμος καλεί τον PolygonIntersectsPolygon αλγόριθμο για να αποφασίσει αν υπάρχουν ακμές των δύο πολυγώνων που τέμνονται. Εάν υπάρχουν, ο αλγόριθμος επιστρέφει 1 (αληθές). Εάν δεν υπάρχουν ακμές που να τέμνονται, ο αλγόριθμος καλεί τον αλγόριθμο PolygonContainsPolygon να αποφασίσει εάν το ένα πολύγωνο περιέχει το άλλο. Αυτό γίνεται μέσω του ελέγχου, εφόσον ένα πολύγωνο περιέχει ένα σημείο από το άλλο πολύγωνο. Αν αυτό συμβαίνει, και δεδομένου ότι δεν υπάρχουν ακμές που να τέμνονται, αυτό σημαίνει ότι περιέχει αυτό το πολύγωνο, και ο αλγόριθμος επιστρέφει 1 (αληθές). Αν και οι δύο από αυτούς τους ελέγχους αποτύχουν (return false), τότε τα δύο πολύγωνα δεν επικαλύπτονται, οπότε ο αλγόριθμος επιστρέφει 0 (ψευδές).

1.4.1.7 Ο MovePolygon αλγόριθμος

Ο αλγόριθμος MovePolygon λαμβάνει ως ορίσματα το id ενός πολυγώνου και δύο ακέραιους αριθμούς που αντιπροσωπεύουν την αλλαγή στις x και y συντεταγμένες, και κατά ακολουθία θα πρέπει να φτιάχνει και να κινεί το συγκεκριμένο πολύγωνο στη δεδομένη κατεύθυνση (μεταβάλλει τις τιμές των x και y συντεταγμένων όλων των σημείων του πολυγώνου από το ποσό που δίνεται ως όρισμα). Ο αλγόριθμος κινεί, επίσης, όλα τα πολύγωνα που είναι συνδεδεμένα με το συγκεκριμένο πολύγωνο, με έναν Connect περιορισμό.

Ο MovePolygon αλγόριθμος:

```
Parameters IN:
   polygon varchar2; //the id of the polygon that should be moved
   movex integer;    //the amount that the polygon should move in
                     //the x axis
   movey integer;    //the amount that the polygon should move in
                     //the y-axis
Variables:           //a cursor needed to retrieve the rows from
                     //theConnectConstraints table
CURSOR cur IS
   SELECT*
   from ConnectConstraints
   where p1 = polygon OR p2 = polygon;
   row ConnectConstraints%ROWTYPE; //a row to hold the
                                   //information retrieved each
                                   //time by the cursor
   p varchar2(10);                 //a variable to temporary hold
                                   //the id of a polygon
                                   //retrieved from the
                                   //ConnectConstraints table.
1. UPDATE Polygon set X = X + movex, Y = Y + movey  where id=
polygon;
2. OPEN cur;
   LOOP
     FETCH cur into row;
     EXIT WHEN cur%NOTFOUND;
     IF row.p1 = polygon THEN
        p := row.p2;
     ELSE
        p := row.p1;
     END IF;
     UPDATE Polygon set X = X + movex, Y = Y + movey where
id = p;
  END LOOP;
END movePolygon;
```

Στο πρώτο βήμα, ο αλγόριθμος movePolygon ενημερώνει τον πίνακα με τα πολύγωνα και «κινεί» το συγκεκριμένο πολύγωνο προς την κατεύθυνση που δίνεται ως όρισμα. Πιο συγκεκριμένα,

αλλάζει όλα τα σημεία του δεδομένου πολυγώνου. Θέτει τις συντεταγμένες x και y από όλα αυτά τα σημεία ίσες με τις προηγούμενες τιμές τους καθώς και το ποσό που δίνεται ως όρισμα (στις μεταβλητές movex και movey). Σημείωση: Οι μεταβλητές movex και movey μπορεί να είναι αρνητικές, π.χ. αν η movex είναι 1, τα σημεία θα μετακινηθούν αριστερά προς τον άξονα x). Στο δεύτερο βήμα, ο αλγόριθμος ξεκινά ένα βρόχο ανακτώντας την επόμενη σύνδεση περιορισμού από τον πίνακα ConnectConstraints που περιλαμβάνει το συγκεκριμένο πολύγωνο. Σε κάθε βήμα αυτού του βρόχου, το πολύγωνο που συνδέεται με αυτό που δίνεται ως όρισμα μετακινείται επίσης, όπως ακριβώς ο αλγόριθμος μετακόμισε αυτό το πολύγωνο στο πρώτο βήμα. Όταν ο αλγόριθμος τελειώσει, όλα τα πολύγωνα (αυτό που δίνεται ως όρισμα και αυτό που συνδέεται με αυτό με έναν Connect περιορισμό) θα έχουν μεταφερθεί από την movex στον άξονα x και από την movey στον άξονα y.

1.4.1.8 Ο CountStepsToMove αλγόριθμος

Ο CountStepsToMove αλγόριθμος λαμβάνει ως όρισμα ένα πολύγωνο και δύο ακέραιους αριθμούς (movex και movey) εκπροσωπεύοντας την κατεύθυνση της κίνησης προς τις συντεταγμένες x και y, αντίστοιχα, και επιστρέφει τον αριθμό των βημάτων που θα χρειαστούν για να μετακινήσουμε αυτό το πολύγωνο και να ξεκαθαρίσουμε (ότι δεν είναι ούτε αυτό το πολύγωνο ούτε αυτό που συνδέεται με αυτό που θα επικαλυφθεί με πολύγωνα που δεν πρέπει).

Ο CountStepsToMove αλγόριθμος:

```
Parameters IN:
   polygon varchar2   //the id of the polygon
   movex integer,     //the direction in the x-axis (value
                      //between -1, 0, 1)
   movey integer,     //the direction in the y-axis (value
                      //between -1, 0, 1)
Variables used:
   TYPE cur_t IS REF CURSOR;
   cursor1 cur_t;
   cursor2 cur_t;
   steps integer;
   row1 NotOverlapConstraints%ROWTYPE;
   row2 ConnectConstraints%ROWTYPE;
   p varchar2(10);
   flag integer;
Return Integer: the number of steps needed to move the polygon to a
clear space.
1. flag := 0;
       steps := 0;
2. WHILE flag = 0
       LOOP
3. steps := steps + 1;
4. movePolygon(polygon, movex, movey);
5. OPEN cursor1 FOR
    SELECT*
    from NotOverlapConstraints
    where p1 = polygon OR p2 = polygon;
       LOOP
          FETCH cursor1 into row1;
          EXIT WHEN cursor1%NOTFOUND;
6. IF polygonOverlapsPolygon(row1.p1, row1.p2) = 1 THEN
           goto continueLabel;
    END IF;
END LOOP;
CLOSE cursor1;
7. OPEN cursor1 FOR
```

```
    SELECT*
    from ConnectConstraints
    where p1 = polygon OR p2 = polygon;
        LOOP
        FETCH cursor1 into row2;
        EXIT WHEN cursor1%NOTFOUND;
        IF polygon = row2.p1 THEN
            p := row2.p2;
        ELSE
            IF p := row2.p1;
        END IF;
8.  OPEN cursor2 FOR
    SELECT *
    from NotOverlapConstraints
    where p1 = p OR p2 = p;
        LOOP
        FETCH cursor2 into row1;
        EXIT WHEN cursor2%NOTFOUND;
9.  IF polygonOverlapsPolygon(row1.p1, row1.p2) = 1 THEN
        goto continueLabel;
    END IF;
    END LOOP;
    CLOSE cursor2;
    END LOOP;
    CLOSE cursor1;
10. flag := 1;
11. <<continueLabel>>
      NULL;
      END LOOP;
12. movePolygon(polygon, -(movex*steps), -(movey*steps));
13. RETURN steps;
END countStepsToMove;
```

Στο πρώτο βήμα, ο αλγόριθμος αρχικοποιεί τις μεταβλητές flag και steps σε 0. Στη συνέχεια, στο δεύτερο στάδιο, φτιάχνει ένα βρόχο, όσο (while) flag = 0 (πράγμα που σημαίνει ότι εξακολουθεί να υπάρχει ένας περιορισμός not_overlap που παραβιάζεται. Κάθε φορά που ο αλγόριθμος εισέρχεται στο βρόχο, αυξάνει τη μεταβλητή steps (μετράει ένα ακόμα βήμα) και μετακινεί το πολύγωνο και αυτό που συνδέεται με αυτό (χρησιμοποιώντας τον αλγόριθμο movePolygon) μία φορά και πάλι προς την movex, movey κατεύθυνση.

Στη συνέχεια, ξανά τσεκάρει αν υπάρχει not_overlap περιορισμός, που περιέχει το πολύγωνο που δίνεται ως όρισμα, που παραβιάζεται. Αν υπάρχει, ο αλγόριθμος συνεχίζει με το βρόχο (πηγαίνει στο continueLabel και από εκεί στην αρχή και πάλι, με τη μεταβλητή flag να παραμένει ίση με 0). Εάν δεν υπάρχει κανένας τέτοιος περιορισμός που να παραβιάζεται, ο αλγόριθμος προχωράει στο επόμενο βήμα, το οποίο είναι να προσπαθήσει και να βρει για κάθε πολύγωνο που συνδέεται με αυτό που δίνεται ως όρισμα με έναν Connect περιορισμό, αν υπάρχει ένας περιορισμός Not_overlap που παραβιάζεται. Για άλλη μια φορά, αν υπάρχει, συνεχίζει με το βρόχο. Σε αντίθετη περίπτωση, θέτει τη μεταβλητή flag ίση με 1 (έτσι ώστε ο βρόχος που αρχίζει με το δεύτερο στάδιο να τελειώσει). Κινεί το πολύγωνο πίσω στην αρχική του θέση (κινώντας το, step φορές στην πίσω κατεύθυνση από τις movex, movey και στη συνέχεια επιστρέφει τον αριθμό των βημάτων που έλαβε για να φτάσει σε αυτή την τοποθεσία (την τιμή που η μεταβλητή step κατέχει).

1.4.1.9 Ο ResolveOverlap αλγόριθμος

Ο ResolveOverlap αλγόριθμος λαμβάνει ως όρισμα το id του πολυγώνου που επικαλύπτεται με ένα άλλο (έτσι ένας από τους περιορισμούς στον πίνακα NotOverlapConstraints παραβιάζεται) και

προσπαθεί να μετακινήσει αυτό το πολύγωνο σε έναν τόπο όπου ούτε αυτό ούτε τα πολύγωνα που συνδέονται σε αυτό να επικαλύπτονται με πολύγωνα που δεν πρέπει (όταν υπάρχουν Connect περιορισμοί για αυτό το πολύγωνο και για τα άλλα, στα οποία υπάρχουν επίσης not_overlap περιορισμοί που ισχύουν). Αυτός ο αλγόριθμος χρησιμοποιεί τον αλγόριθμο countStepsToMove για να βρει την καλύτερη (συντομότερη) διαδρομή προς εκκαθάριση για αυτό το πολύγωνο και τον αλγόριθμο movePolygon για να μετακινήσει το πολύγωνο και εκείνο που συνδέεται με αυτό σε εκείνο το μέρος.

Ο ResolveOverlap αλγόριθμος:

```
Parameters IN:
   i integer;
   steps integer;
   minSteps integer;
   bestx integer;
   besty integer;
     CURSOR cur IS SELECT * from NotOverlapConstraints where
p1 = polygon OR p2 = polygon;
row NotOverlapConstraints%ROWTYPE;
1. minSteps := countStepsToMove(polygon, 0, -1);
    bestx := 0;
    besty := -1;
2. steps := countStepsToMove(polygon, 1, -1);
IF steps < minSteps THEN
    minSteps := steps;
    bestx := 1;
    besty : -1;
END IF;
3. steps := countStepsToMove(polygon, 1, 0);
IF steps < minSteps THEN
    minSteps := steps;
    bestx := 1;
    besty := 0;
END IF;
4. steps := countStepsToMove(polygon, 1, 1);
IF steps < minSteps THEN
    minSteps := steps;
    bestx := 1;
    besty := 1;
END IF;
5. steps := countStepsToMove(polygon, 0, 1);
IF steps < minSteps THEN
    minSteps := steps;
    bestx := 0;
    besty := 1;
END IF;
6. steps := countStepsToMove(polygon, -1, 1);
IF steps < minSteps THEN
    minSteps := steps;
    bestx := -1;
    besty :=  1;
END IF;
7. steps := countStepsToMove(polygon, -1, 0);
IF steps < minSteps THEN
    minSteps := steps;
    bestx := -1;
    besty :=  0;
END IF;
```

```
8. steps := countStepsToMove(polygon, -1, -1);
IF steps < minSteps THEN
    minSteps := steps;
    bestx := -1;
    besty := -1;
END IF;
9. movePolygon(polygon, bestx*minSteps, besty*minSteps);
END resolveOverlap;
```

Στο πρώτο βήμα, ο resolveOverlap αλγόριθμος υποθέτει ότι το καλύτερο μονοπάτι είναι να μετακινήσουμε το συγκεκριμένο πολύγωνο προς το «βορρά » (διατηρώντας την x συντεταγμένη όπως είναι και μειώνοντας την y συντεταγμένη). Έτσι, μετράει τα βήματα (καλώντας τον countStepsToMove αλγόριθμο) που παίρνει η πράξη της μετακίνησης αυτού του πολυγώνου προς το βορρά (έως ότου δεν υπάρχουν Not_overlap περιορισμοί , που αφορούν αυτό το πολύγωνο ή τα πολύγωνα που συνδέονται με αυτό , που να παραβιάζονται) και θέτει τη μεταβλητή minSteps να κρατήσει αυτόν τον αριθμό των βημάτων. Επίσης, θέτει τις μεταβλητές bestX και bestY να κρατήσουν τον τρόπο που οι συντεταγμένες x και y αλλάζουν (αυτές οι τιμές τώρα είναι 0 και 1, αντίστοιχα).

Στο δεύτερο βήμα, ο αλγόριθμος προσπαθεί να βρει αν θα ήταν γρηγορότερο να κινήσει αυτό το πολύγωνο «βορειοανατολικά» (αυξάνοντας τη x συντεταγμένη και μειώνοντας τη y συντεταγμένη). Μετρά τα βήματα αυτής της λειτουργίας (όπως και στη προηγούμενη) και ελέγχει αν είναι ταχύτερη (δηλαδή αν steps < minSteps). Αν είναι, αλλάζει την τιμή min-Steps σε steps και κάνει τις bestX και bestY μεταβλητές ίσες με 1 και 1, αντίστοιχα.

Στα επόμενα βήματα (βήματα 3-8), ο αλγόριθμος ελέγχει το υπόλοιπο των κατευθύνσεων και αλλάζει τις μεταβλητές minSteps , bestX και bestY κάθε φορά που βρίσκει μια συντομότερη διαδρομή. Τέλος, στο τελευταίο βήμα (βήμα 9), ο αλγόριθμος μετακινεί το πολύγωνο στον τόπο που διαπίστωσε ότι ήταν «καθαρός» , καλώντας τον movePolygon αλγόριθμο με minSteps * bestX και minSteps * bestY όπως τα ορίσματα του αλγορίθμου.

1.5 Ένα πλήρες παράδειγμα του εργαλείου

Ας υποθέσουμε ότι έχουμε δημιουργήσει έναν πίνακα σε μια βάση δεδομένων με τις ακόλουθες εντολές SQL:

```
create table polygon
(
id varchar2(10),
serial integer,
x number,
y number
);
```

Ο πίνακας polygon προσθέτει τον περιορισμό pk_polygon με πρωτεύον κλειδί (id,serial). Έτσι, ο πίνακας αυτός θα μπορεί να κρατήσει πληροφορίες σχετικά με τα πολύγωνα (κρατάει πληροφορίες σχετικά με το id του πολύγωνου και τα σημεία των συντεταγμένων του και για τον τρόπο που αυτά τα σημεία των συντεταγμένων είναι συνδεδεμένα μεταξύ τους). Στη συνέχεια, υποθέτουμε ότι θέλουμε να προσθέσουμε στις πληροφορίες του πίνακα περίπου τρία πολύγωνα:

Το πολύγωνο που ονομάζεται 'A', το οποίο κατασκευάζεται από την ένωση των εξής σημείων (κάθε σημείο είναι συνδεδεμένο με το προηγούμενο και το επόμενό του σημείο, τα πρώτα και τα τελευταία σημεία συνδέονται επίσης):

(a) (75, 75)
(b) (200, 75)
(c) (200, 200)
(d) (75, 200)

Τοποθετούμε αυτές τις πληροφορίες στον πίνακα με τις ακόλουθες SQL εντολές:

Insert into polygon values ('A' , 0, 75 , 75)
Insert into polygon values ('A', 1, 200 , 75)
Insert into polygon values ('A', 2 , 200, 200)
Insert into polygon values ('A', 3 , 75, 200)

Το πολύγωνο που ονομάζεται 'B', το οποίο κατασκευάζεται από την ένωση των εξής σημείων (παρόμοια με το πολύγωνο A):

(a) (100, 50)
(b) (150, 50)
(c) (150, 250)
(d) (100, 250)

Όπως και στην προηγούμενη εργασία μας, εισάγουμε τις πληροφορίες αυτές με εκτέλεση:

Insert into polygon values ('B', 0, 100, 50)
Insert into polygon values ('B', 1, 150, 50)
Insert into polygon values ('B', 2, 150, 250)
Insert into polygon values ('B', 3, 100, 250)

Το πολύγωνο που ονομάζεται 'C', το οποίο κατασκευάζεται από την ένωση των εξής σημείων (παρόμοια με το πολύγωνο A):

(a) (200, 100)
(b) (220, 100)
(c) (220, 120)
(d) (200, 120)

Όπως και στην προηγούμενη εργασία μας, εισάγουμε τις πληροφορίες αυτές με εκτέλεση:

Insert into polygon values ('C', 0, 200, 100)
Insert into polygon values ('C', 1, 220, 100)
Insert into polygon values ('C', 2, 220, 120)
Insert into polygon values ('C', 3, 200, 120)

Η τελική παραδοχή μας θα είναι οι περιορισμοί της βάσης δεδομένων. Θέλουμε να βεβαιωθούμε ότι τα πολύγωνα A και B δεν θα αλληλεπικαλυφθούν ποτέ και επίσης, έχουμε κατά νου ότι το πολύγωνο A και το πολύγωνο C θα μείνουν συνδεδεμένα (εάν θα πρέπει το πολύγωνο A να μετακινηθεί, το πολύγωνο C θα πρέπει να μετακινηθεί επίσης, υπό τη συγκεκριμένη κατεύθυνση και απόσταση με το πολύγωνο A). Τώρα, θα εκτελέσουμε το εργαλείο (Java program) με τις ακόλουθες εισόδους (όπως φαίνεται στην Εικ. 4). Όπως φαίνεται παρακάτω, εισάγουμε το απαιτούμενο σήμα εισόδου στο εργαλείο και γράφουμε τους κανόνες:

Not_Overlap ('A', 'B'); and
Connect ('A', 'C');

Στη συνέχεια, πατάμε το κουμπί " Create Program " και το εργαλείο δημιουργεί τα δύο αρχεία εξόδου (μπορούμε να τα βρούμε στο φάκελο όπου το πρόγραμμα βρισκόταν) και εμφανίζει ένα μήνυμα ενημέρωσης ότι δημιούργησε ένα πρόγραμμα που χρησιμοποιεί την είσοδο που δώσαμε, με δύο περιορισμούς, έναν Not_Overlap περιορισμό και έναν Connect περιορισμό. Το παραγόμενο αποτέλεσμα θα είναι δύο αρχεία, ένα με όνομα code.pl και ένα με όνομα dropCode.pl. Το πρώτο περιέχει τον pl/SQL κώδικα που θα διατηρήσει τους περιορισμούς της βάσης δεδομένων που δίνονται και το δεύτερο θα αφαιρέσει όλους τους πίνακες/τις διαδικασίες/τις συναρτήσεις που δημιουργήθηκαν από την εκτέλεση του pl/SQL κώδικα. Ο κώδικας pl/SQL που περιέχεται στο code.pl αρχείο σε αυτό το παράδειγμα θα είναι:

- Πρώτον, υπάρχουν οι εντολές SQL για τη δημιουργία των δύο πινάκων που θα κρατούν τις πληροφορίες σχετικά με τους περιορισμούς της βάσης δεδομένων:

  CREATE TABLE NotOverlapConstraints

  (p1 varchar2 (10), p2 varchar2 (10))

  Κάθε εγγραφή που εισάγεται στον πίνακα αυτό θα κατέχει πληροφορίες για έναν Not_overlap περιορισμό. Οι P1 και P2 θα κρατάνε τις τιμές των ids των πολυγώνων που δεν πρέπει να αλληλεπικαλύπτονται.

  CREATE TABLE ConnectConstraints

  (p1 varchar2 (10), p2 varchar2 (10))

  Κάθε εγγραφή, που εισάγεται στον πίνακα αυτό θα κρατάει πληροφορίες για έναν Connect περιορισμό. Οι P1 και P2 θα κρατάνε τις τιμές των ids των πολυγώνων που θα πρέπει να συνδέονται.

- Έπειτα, υπάρχουν οι εντολές για την προετοιμασία των δύο πινάκων που δημιουργήθηκαν προηγουμένως με τις τιμές που περιγράφουν τους περιορισμούς της βάσης δεδομένων. Σε αυτό το παράδειγμα θα υπάρχουν δύο εντολές εισαγωγής:

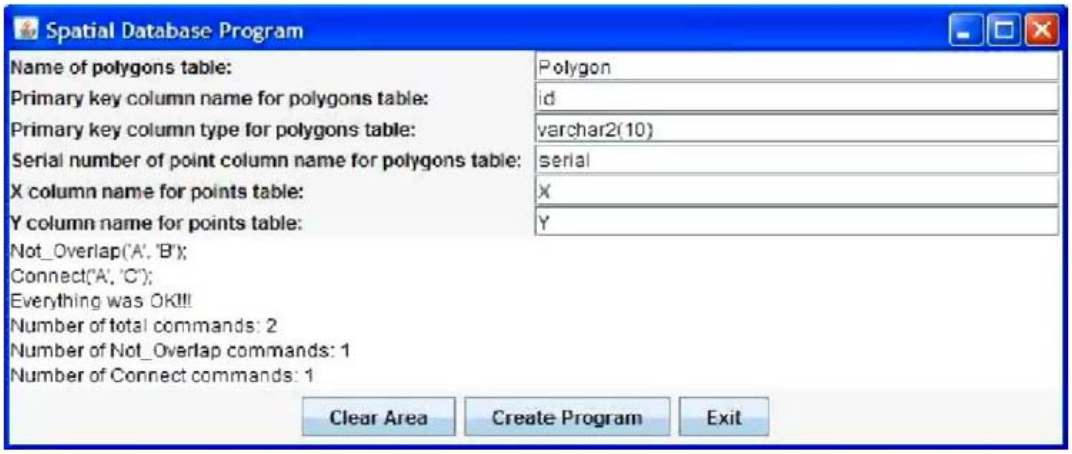
  INSERT INTO NotOverlapConstraints(p1, p2) VALUES ('A', 'B');

  Αυτή η γραμμή που προστίθεται στον πίνακα αυτόν περιγράφει τον πρώτο περιορισμό που θέλουμε, ότι τα πολύγωνα Α και Β δεν πρέπει να αλληλεπικαλύπτονται.

  INSERT INTO ConnectConstraints (p1, p2) VALUES ('A', 'C');
  Αυτή η γραμμή που προστίθεται στον πίνακα αυτόν περιγράφει το δεύτερο περιορισμό που θέλουμε, ότι τα πολύγωνα Α και Γ θα πρέπει να συνδέονται.

- Στη συνέχεια, έρχεται η δήλωση και η εφαρμογή των λειτουργιών και των διαδικασιών που περιγράφτηκαν προηγουμένως.

- Τέλος, δηλώνεται η main_proc διαδικασία.



Εικ. 1.4. GUI (Graphical User Interface).

Ο χρήστης, τώρα, πρέπει να τρέξει αυτό το αρχείο στη βάση δεδομένων και στη συνέχεια χειροκίνητα να εκτελέσει την main_proc διαδικασία (π.χ. με την πληκτρολόγηση exec main_proc στη γραμμή εντολών). Η main_proc διαδικασία θα καταστήσει έναν βρόγχο, ο οποίος κάθε φορά θα πηγαίνει στον επόμενο Not_overlap περιορισμό από τον αντίστοιχο πίνακα. Στη συνέχεια, θα ελέγξει αν επικαλύπτονται τα δύο πολύγωνα που στην εν λόγω γραμμή επικαλύπτονται (με την κλήση της λειτουργίας polygonOverlaps- Polygon). Αν το κάνουν, θα προσπαθήσει να επιλύσει αυτήν την

επικάλυψη (καλώντας τη λειτουργία resolveOverlap). Αφού τελειώσει η εκτέλεση της main_proc, οι περιορισμοί της βάση δεδομένων σίγουρα δεν θα παραβιάζονται.

Σε αυτό το παράδειγμα, η main_proc θα ανακτήσει μόνο μία γραμμή από τον NotOverlapConstraints πίνακα, ο οποίος θα έχει τις τιμές ('A', 'B'). Έτσι, θα ελέγχει εάν τα πολύγωνα A και B επικαλύπτονται, με την κλήση της λειτουργίας polygonOverlapsPolygon με ('A', 'B') ως ορίσματα.

Ο PolygonOverlapsPolygon θα εξακριβώσει αν τα δύο πολύγωνα τέμνονται, για να καλέσει τη συνάρτηση PolygonIntersectsPolygon με ('A', 'B') ως ορίσματα.
Ο πρώτος έλεγχος που ο PolygonIntersectsPolygon θα κάνει, θα είναι κατά πόσον η ακμή του πολυγώνου A παρασκευάζεται από την ένωση των δύο πρώτων σημείων της (75, 75) και (200, 75) με τα δύο πρώτα σημεία του πολυγώνου B (100, 50) και (150, 50), το οποίο προφανώς θα είναι αρνητικό (αυτές οι ακμές είναι παράλληλες μεταξύ τους και με τον άξονα x). Όμως, την επόμενη φορά που ο αλγόριθμος εισέρχεται στο βρόγχο, θα προσπαθήσει να ελέγξει τα δύο πρώτα σημεία του πολυγώνου A ξανά (75, 75) και (200, 75), με το δεύτερο και το τρίτο σημείο του πολυγώνου B (150, 50) και (150, 250). Οι δύο αυτές ακμές τέμνονται, και έτσι ο αλγόριθμος polygonIntersectsPolygon τελειώνει και επιστρέφει 1 (αληθές).

Ο polygonOverlapsPolygon παίρνει το 1 που επιστρέφεται από την συνάρτηση που καλέσαμε, έτσι ώστε να επιστρέψει και πάλι 1 (αληθές). Στη συνέχεια, η main_proc, έχοντας διαπιστώσει μία παράβαση στους περιορισμούς, καλεί την resolveOverlap με A ως όρισμα για την επίλυση αυτής της παραβίασης.

Η λειτουργία resolveOverlap θα μετρήσει τα βήματα που απαιτούνται για να κινηθεί το πολύγωνο A για την επίλυση της παραβίασης της επικάλυψης. Σε αυτό το παράδειγμα, οι υπολογισμοί που θα γίνουν είναι οι εξής:

- Steps to up-left: 101 (x: -1, y: -1)
- Steps to up: 150 (x: 0, y: -1)
- Steps to up-right: 76 (x: 1, y: -1)
- Steps to right: 75 (x: 1, y: 0)
- Steps to down-right: 76 (x: 1, y: 1)
- Steps to down: 166 (x: 0, y: 1)
- Steps to down-left: 101 (x: -1, y: 1)
- Steps to left: 101 (x: -1, y: 0)

Το παραπάνω σύστημα σημαίνει ότι αν θέλουμε να μετακινήσουμε το πολύγωνο A κάπου που οι περιορισμοί δεν θα παραβιάζονται ακολουθώντας την "up-left" κατεύθυνση, θα πρέπει να το μετακινήσουμε για το ποσό των 101 βημάτων και να μειώσουμε κατά 1την x συντεταγμένη (ως εκ τούτου, μείωση της x συντεταγμένης κατά -101) σε κάθε βήμα των σημείων του και το ίδιο ισχύει και για τη συντεταγμένη y των σημείων του. Μπορούμε εύκολα να καταλάβουμε από τα παραπάνω ότι η κατεύθυνση που θα επιλεγεί είναι η σωστή κατεύθυνση (κοστίζει το ελάχιστο ποσό των βημάτων). Έτσι, η resolveOverlap θα καλέσει τη συνάρτηση movePolygon με (75, 0) ως ορίσματα, επειδή έχουμε επιλέξει τη σωστή κατεύθυνση, η οποία είναι να αυξήσουμε το x και να αφήσουμε το y ως έχει, έτσι μπορούμε να κάνουμε (steps * 1, steps *0). Η λειτουργία movePolygon θα κινήσει αρχικά το πολύγωνο A στην νέα του θέση, έτσι ώστε οι νέες συντεταγμένες των σημείων x και y του πολυγώνου A να είναι:

- (150, 75)
- (275, 75)
- (275, 200)
- (150, 200)

Στη συνέχεια, η λειτουργία movePolygon θα ξεκινήσει μια επανάληψη, ανακτώντας σε κάθε βήμα τον επόμενο Connect περιορισμό που σχετίζεται με το πολύγωνο A. Θα ανακτήσει μία γραμμή, η

οποία θα περιέχει ('A','C') και σημαίνει ότι το C πολύγωνο ενώθηκε με το πολύγωνο A. Έτσι, η λειτουργία movePolygon θα μετακινήσει το πολύγωνο C με τον ακριβή τρόπο με τον οποίο μεταφέρθηκε το πολύγωνο A. Οι νέες x και y συντεταγμένες των σημείων του πολυγώνου C θα είναι:

- (275, 100)
- (295, 100)
- (295, 120)
- (275, 120)

Μετά από αυτό, ο έλεγχος διαβιβάζεται στη main_proc πάλι, η οποία θα τελειώσει την επανάληψη που ξεκίνησε (καμία γραμμή δεν θα ανακτηθεί, καθώς έχουμε μόνο έναν περιορισμό επικάλυψης) και στη συνέχεια θα τελειώσει την εκτέλεση. Οι νέες θέσεις για τα πολύγωνα A και C φροντίζουν ώστε όλοι οι περιορισμοί της βάση δεδομένων να συντηρούνται καλά.

1.6 Η αρχιτεκτονική των συστημάτων

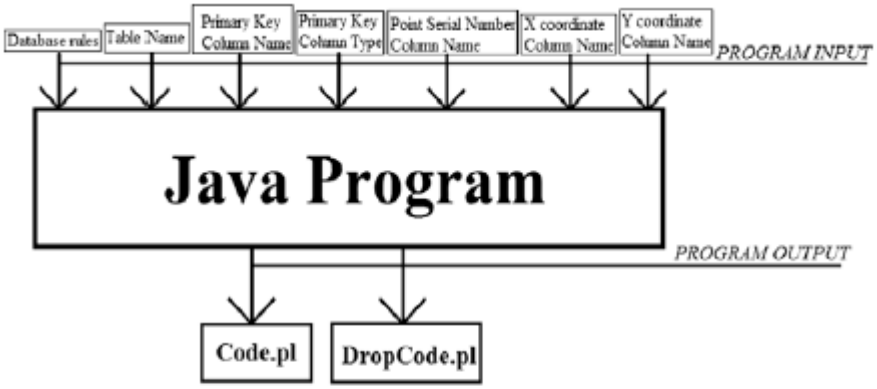
Η αρχιτεκτονική του εργαλείου που παράγει τις SQL εντολές φαίνεται στην Εικ. 5 (το GUI του συστήματος φαίνεται στην Εικ. 4). Ακολουθεί μια γενική περιγραφή του λογικού κεφαλαίου της Java. Τα εισερχόμενα δεδομένα περιέχουν:

Τους κανόνες της βάσης, που είναι οι Not-overlap και Connect περιορισμοί μεταξύ των σχημάτων που είναι αποθηκευμένα στη βάση. Η σύνταξη αυτών των δύο εντολών είναι η ακόλουθη: Not_Overlap(ID1,ID2); όπου Not Overlap είναι η λέξη κλειδί για τους Not_Overlap περιορισμούς (Το Connect χρησιμοποιείται για τους Connect περιορισμούς) και ID(1, 2) είναι οι τιμές του πεδίου πρωτεύοντος κλειδιού για το σχήμα που εμπεριέχεται στον περιορισμό.

- Το όνομα του πίνακα ,ως γραμματοσειρά.
- Το όνομα της στήλης του κλειδιού.
- Τον τύπο της στήλης του κλειδιού, όπως δίνεται όταν δηλώνεται ο πίνακας.
- Το όνομα της στήλης του σειριακού αριθμού (serial number), οποιουδήποτε τύπου.
- Το όνομα της στήλης της x συντεταγμένης, οποιουδήποτε τύπου αριθμών.
- Το όνομα της στήλης της y συντεταγμένης, οποιουδήποτε τύπου αριθμών.

Η έξοδος περιλαμβάνει:

Το Code.pl: ένα αρχείο με εντολές PL / SQL, το οποίο όταν εκτελείται στη βάση δεδομένων που ερωτείται, θα δημιουργήσει τις βασικές διαδικασίες που θα διαφυλάξουν την ακεραιότητά του. Το DropCode.pl: ένα αρχείο με τις βασικές PL / SQL εντολές, έτσι ώστε οι πίνακες και οι διαδικασίες που δημιουργήθηκαν από το πρόγραμμα να μπορούν να διαγραφούν.



Database Rules:	Syntax: Not_Overlap(id,id); or Connect(id, id);
Table Name:	String, the name of the database table.
Key column Name:	Name of the column of the table that is primary key.
Key column Type:	Type of the column of the table that is primary key (any used type can be given, exactly as when declared the table).
Serial column name:	Name of the column of the table for the serial number of a point
X column Name:	Name of the column of the table for the x coordinate of a point.
Y column Name:	Name of the column of the table for the y coordinate of a point.

Εικ. 1.5. Η αρχιτεκτονική του συστήματος.

Java λογική μονάδα (Java logical unit) : Αυτό το μέρος είναι το κύριο και πιο σημαντικό μέρος του συστήματος. Αποτελεί τον κώδικα σε γλώσσα Java, που παράγει τα τελικά αρχεία εξόδου με όλες τις εντολές PL / SQL.

Κλάσεις και πακέτα (Classes and packages) : Οι κλάσεις και τα πακέτα που χρησιμοποιήθηκαν είναι οι ακόλουθες:

1.6.1 Package lexicalAnalyzer

Το πακέτο αυτό περιέχει το λεξιλογικό αναλυτή της εφαρμογής.

1.6.1.1 Token.java

Η κλάση αυτή αντιπροσωπεύει το σύμβολο της εισόδου που αναγνωρίζεται. Τα πεδία ενός θέματος από την εν λόγω κατηγορία έχουν πεδία: String value σύμβολο εισόδου σε μορφή γραμματοσειράς int type ακέραιος για τον τύπο του συμβόλου, μπορεί να είναι ένα από τα παρακάτω:

"ID" ένας πίνακας πεδίο πρωτεύοντος κλειδιού
"NOT_OVERLAP " μια λέξη-κλειδί για έναν Not Overlap περιορισμό
"CONNECT " μια λέξη-κλειδί για έναν Connect περιορισμό
"L_PAR " η παρένθεση αριστερά συμβολική "("
"R_PAR " η δεξιά παρένθεση συμβολική ")"
"COMMA " το κόμμα συμβολική ","
"ENDMARK " το ερωτηματικό στο τέλος μιας πρότασης
"EOF" το τέλος του συμβόλου αρχείου
" UNKNOWN " ένα άγνωστο σύμβολο
"WHITE" ένας χαρακτήρας λευκού κενού

Η κλάση, πέρα από τη μέθοδο Token (int t, String s), που είναι ο κατασκευαστής της κλάσης, περιλαμβάνει επίσης την toString() μέθοδο στην οποία επιστρέφει τον τύπο του συμβόλου και την τιμή του π.χ. "ID-key ".

1.6.1.2 Scanner.java

Αυτή η κλάση πραγματικά επιτυγχάνει τη λεκτική ανάλυση της εισόδου. Άλλες μέθοδοι κλάσης, εκτός από κατασκευαστή Scanner(String t) είναι η μέθοδος getNextToken() που αναγνωρίζει την επόμενη Token από την είσοδο και την επιστρέφει, η lookNextToken() η οποία είναι παρόμοια με την getNextToken() με τη μόνη διαφορά ότι επιστρέφει την επόμενη Token χωρίς προχωρήσει στην αναζήτηση αυτής που ακολουθεί .

1.6.2 Package plsql

Το πακέτο που παράγει τον PL / SQL κώδικα και τον γράφει στα αρχεία code.pl και drop.pl.

1.6.2.1 CodeCreator.java

Η κλάση αυτή δημιουργεί κώδικα PL / SQL που χρειάζεται για την εφαρμογή των κανόνων που η εφαρμογή χρησιμοποίησε σαν είσοδο μετά την λεξικολογική ανάλυση. Οι έθοδοι της κλάσης είναι οι εξής:

- Η Public CodeCreator(String polygonTable, String polygonKey, String polygonKeyType, String serial, String x, y String) throws exception (πετάει εξαίρεση): ο κατασκευαστής του αντικειμένου, παίρνει ως είσοδο όλα τα απαιτούμενα πεδία που είναι παρμένα από το GUI.
- Η Public void createProgram(JTextArea text) throws exception (πετάει εξαίρεση): Αυτή η μέθοδος καλείται, όταν όλες οι απαιτούμενες πληροφορίες είναι έτοιμες και οι περιορισμοί της βάσης δεδομένων είναι διαθέσιμοι για να αναλύσουν το εσωτερικό της δεδομένης περιοχής κειμένου.
- Η Public void end() throws Exception (πετάει εξαίρεση): η μέθοδος αυτή χρησιμοποιείται για να οριστικοποιήσει την παραγωγή του κώδικα PL / SQL και να κλείσει τα αρχεία.

1.6.2.2 CodeWriter.java

Η κλάση αυτή γράφει τον κώδικα στα code.pl, dropCode.pl αρχεία που η CodeCreator κλάση έχει παράγει. Εκτός από την CodeWriter(), που είναι ο κατασκευαστής και δημιουργεί ένα νέο αντικείμενο CodeWriter και αρχικοποιεί τα 2 Buffered Writers, για να ανοίξουν τα "code.pl" και "dropCode.pl" αρχεία, άλλες μέθοδοι αυτής της κατηγορίας είναι: η μέθοδος close() η οποία κλείνει τους 2 writers (out and drop out writers) τερματίζοντας την καταχώριση και στα δύο αρχεία, η μέθοδος writeCode (java.lang.String) που γράφει το αποδεκτό String σαν είσοδο στο code.pl αρχείο, η writeNewLine() μέθοδος που είναι void και γράφει σε αρχεία το χαρακτήρα νέας γραμμής και η writeDrop(String, String) μέθοδος η οποία γράφει τον απαραίτητο κώδικα στο dropCode.pl αρχείο για να πετάξει το συγκεκριμένο στοιχείο με το συγκεκριμένο όνομα.

1.6.3 Package syntaxAnalyzer

Το πακέτο αυτό κάνει την συντακτική ανάλυση των κανόνων που δίνει ένας χρήστης, περιλαμβάνει την GUI (Graphic User Interface) κλάση και την main class που εκτελεί την εφαρμογή.

1.6.3.1 Parser.java

Η κλάση που φτιάχνει τη συντακτική ανάλυση της εισόδου του χρήστη βασισμένη σε συγκεκριμένους κανόνες. Οι κανόνες πρέπει να έχουν την ακόλουθη μορφή:

ID∧ID[∧ID] *→>ID;

Χρησιμοποιεί το λεξιλογικό αναλυτή που αναφέρεται παραπάνω. Οι μέθοδοι της κλάσης είναι η Parser(JTextArea) που είναι ο κατασκευαστής της κλάσης που δέχεται ως όρισμα το κείμενο που θα αναλυθεί και η μέθοδος parse() η οποία επιστρέφει ένα πίνακα από λίστες με αντικείμενα του τύπου Token (μόνο ID για λόγους ευκολίας). Αυτός ο πίνακας έχει δύο υποδοχές και χρησιμοποιείται για την αποθήκευση των Not_overlap εντολών στο δείκτη 0 και των Connect εντολών στο δείκτη 1.

1.6.3.2 Gui.java

Η κλάση GUI χρησιμοποιείται για την εμφάνιση της διεπαφής στο χρήστη και απαιτεί από αυτόν να εισάγει τις πληροφορίες που απαιτούνται για τη διατήρηση της βάσης δεδομένων.

1.7 Ορθότητα, πολυπλοκότητα και τα αποτελέσματα αξιολόγησης

1.7.1 Ορθότητα

Θεώρημα 1.7.1: Τα συστήματα αντιμετωπίζουν το πρόβλημα της διακλάδωσης των χωρικών βάσεων δεδομένων.

Απόδειξη. Το πρόγραμμα, με σκοπό την παραγωγή του κώδικα SQL, χρειάζεται σαν εισαγωγή τους περιορισμούς ακεραιότητα της βάσης δεδομένων (τους Not_overlap και τους Connect περιορισμούς). Έτσι, όταν ο κώδικας SQL εκτελείται στη βάση δεδομένων, δημιουργεί ένα βρόγχο για κάθε Not_overlap περιορισμό που δίνεται, και ελέγχει αν παραβιάζεται ή όχι, έτσι δεν υπάρχει καμία πιθανότητα το πρόγραμμα να μην βρει τέτοια παράβαση. Όταν το πρόγραμμα ελέγχει έναν Not_overlap περιορισμό, θα βρει σίγουρα την επικάλυψη (εάν υπάρχει), γιατί δοκιμάζει τα δύο σχήματα με τον ακόλουθο τρόπο:

● Πρώτον, ελέγχει εάν τα δύο σχήματα έχουν δύο ακμές που τέμνονται. Παίρνει κάθε ακμή από το πρώτο σχήμα και τη δοκιμάζει με κάθε ακμή του άλλου σχήματος.

● Για κάθε δύο ακμές, το πρόγραμμα καταρχάς βρίσκει ποια ακμή έχει το πιο αριστερό σημείο (αυτό με την μικρότερη τιμή της χ συντεταγμένης) (edge1, έτσι ώστε τα δύο σημεία να είναι τα: left1 και right1) και ποιο είναι το αριστερό σημείο της άλλης ακμής (edge2, έτσι ώστε τα δύο σημεία να είναι τα: left2 και right2).

● Αν υπάρχει μια τομή, τότε ο ακόλουθος έλεγχος θα είναι αληθής, αλλιώς ψευδής:

```
IF left1x < left2x AND right1x > left2x THEN
  IF left1y <= left2y AND right1y > left2y THEN
     RETURN 1;
  END IF;
  IF left1y <= right2y AND left1y > left2y THEN
     RETURN 1;
  END IF;
END IF;
```

● Αν το πρόγραμμα δεν βρει καθόλου ακμές που να τέμνονται, τότε θα ελέγξει αν ένα σχήμα περιέχεται πλήρως μέσα σε ένα άλλο (σίγουρα δεν γίνεται να περιέχεται εν μέρη, γιατί τότε θα υπήρχαν ακμές οι οποίες θα τέμνονταν). Προκειμένου να ελεγχθεί αυτό, το πρόγραμμα δοκιμάζει εάν υπάρχει ένα σημείο του σχήματος το οποίο βρίσκεται μέσα στο σχήμα του άλλου.

● Έτσι, δίνοντας ένα σημείο και ένα σχήμα, το πρόγραμμα πρώτα ελέγχει εάν το σημείο εντάσσεται μέσα στο πλαίσιο οριοθέτησης (το ελάχιστο πλαίσιο που περιλαμβάνει πλήρως το σχήμα) του σχήματος. Εάν όχι, τότε το σημείο (και έτσι όλο το σχήμα) δεν περιλαμβάνεται. Εάν βρίσκεται στο εσωτερικό του πλαισίου οριοθέτησης, τότε το πρόγραμμα δοκιμάζει εάν είναι επίσης μέσα στο σχήμα.

● Για να βρούμε αν το σημείο είναι μέσα στο πολύγωνο, προσπαθούμε να βρούμε αν μία γραμμή αρχίζει από σημείο που τέμνεται με περιττό αριθμό ακμής. Αν διασταυρωθεί με ζυγό αριθμό τότε είναι έξω (http, xxxx).

●. Όταν το πρόγραμμα ολοκληρώσει τον έλεγχο, αν υπάρχει παραβίαση περιορισμού αλληλοεπικάλυψης, θα την βρει σίγουρα.

Το επόμενο πράγμα είναι ότι όταν το πρόγραμμα ολοκληρώσει την εκτέλεση, δεν θα υπάρξει κανένας Not_overlap περιορισμός που θα παραβιαστεί, επειδή όλες οι αλληλεπικαλύψεις, θα έχουν επιλυθεί. Όταν το πρόγραμμα βρει μία επικάλυψη (με τον τρόπο που προαναφέρθηκε), θα

προσπαθήσουμε να το λύσουμε, με το ελάχιστο ποσό των βημάτων (ποσό των κινήσεων στη x και y συντεταγμένη). Μετράμε τα βήματα που χρειάζονται για να μετακινήσουμε το σχήμα μακριά από την επικάλυψη σε κάθε μία από αυτές τις κατευθύνσεις (up, up-right, right, down-right, down, down-left, left, up-left). Για κάθε κατεύθυνση που δοκιμάζεται, το πρόγραμμα μετακινεί το σχήμα (και όλα αυτά που είναι συνδεδεμένα με αυτό) ένα βήμα προς αυτή την κατεύθυνση και στη συνέχεια ελέγχει αν η παραβίαση του Not_overlap περιορισμού έχει επιλυθεί. Αν όχι, συνεχίζει τη μετακίνηση του σχήματος και όλων αυτών που είναι συνδεδεμένα μαζί του με έναν Connect περιορισμό. Αν ναι, τότε θα ελέγχει για κάθε Connect περιορισμό που το σχήμα αυτό έχει, αν το πολύγωνο που συνδέεται με αυτό που μετακινήθηκε έχει έναν Not_overlap περιορισμό που παραβιάζεται. Αν ένας από αυτούς τους ελέγχους είναι αληθής (παραβιάζονται περισσότεροι περιορισμοί) συνεχίζει να μετακινεί το πρώτο σχήμα (και όλα αυτά που είναι συνδεδεμένα με αυτό). Αν όχι, τότε αυτή είναι μια θέση στην οποία το σχήμα θα μπορούσε να τοποθετηθεί, για την επίλυση αυτής της επικάλυψης. Μετά τον έλεγχο κάθε κατεύθυνσης, το πρόγραμμα επιλέγει αυτή για την οποία απαιτούνται τα ελάχιστα βήματα και μετακινεί το σχήμα.

1.7.2 Η πολυπλοκότητα της λύσης μας

Στην ενότητα αυτή, παρουσιάζουμε την πολυπλοκότητα του αλγορίθμου που έχουμε παρουσιάσει στο τμήμα 4.

1.7.2.1 Ο MovePolygon αλγόριθμος

Υποθέτουμε ότι κάθε πολύγωνο έχει N αριθμό σημείων και ότι το πολύγωνο που θέλουμε να μετακινήσουμε έχει M αριθμό άλλων πολυγώνων που συνδέονται με αυτό (με Connect περιορισμούς), το καθένα με N αριθμό σημείων. Στη συνέχεια τα βήματα που αυτός ο αλγόριθμος πρέπει να εκτελέσει είναι τα εξής:

```
UPDATE Polygon set X=X+movex, Y=Y+movey where                    N
  id=polygon;
OPEN cur;                                                        1
LOOP                                                  M steps in the
                                                                loop

FETCH cur into row;                                              1
EXIT WHEN cur
IF row.p1=polygon THEN                                           1
      p := row.p2;
          ELSE
      p := row.p1;                                               1
END IF;
      UPDATE Polygon set X=X+movex, Y=Y+movey                    N
      where id=p;
END LOOP;
END movePolygon;
```

Έτσι ο αλγόριθμος αυτός χρειάζεται N+1+M * (1+1+1+1+N) = N+1+N*M+4*M = (N^2) βήματα.

1.7.2.2 Ο EdgesIntersect αλγόριθμος

Σε αυτόν τον αλγόριθμο, δεν υπάρχουν βρόγχοι, οπότε είναι σαφές ότι ο αλγόριθμος είναι O(1). Πιο συγκεκριμένα, τα βήματα που αυτός ο αλγόριθμος χρειάζεται να εκτελέσει είναι τα εξής:

```
x1, y1 := point1s x and y coordinates                            2
x2, y2 := point2s x and y coordinates                            2
x3, y3 := point3s x and y coordinates                            2
x4, y4 := rpoint4s x and y coordinates                           2
IF x1 <= x2 THEN                                                 1
```

```
    left1x := x1;                                                    1
    left1y := y1;                                                    1
    right1x := x2;                                                   1
    right1y := y2;                                                   1
ELSE
    left1x := x2;
    left1y := y2;
    right1x := x1;
    right1y := y1;
END IF;
IF x3 <= x4 THEN                                                     1
    left2x := x3;                                                    1
    left2y := y3;                                                    1
    right2x := x4;                                                   1
    right2y := y4;                                                   1
ELSE
    left2x := x4;
    left2y := y4;
    right2x := x3;
    right2y := y3;
END IF;
IF left1x > left2x THEN                                              1
    tmpx := left1x;                                                  1
    left1x := left2x;                                                1
    left2x := tmpx;                                                  1
    tmpx := right1x;                                                 1
    right1x := right2x;                                              1
    right2x := tmpx;                                                 1
    tmpy := left1y;                                                  1
    left1y := left2y;                                                1
    left2y := tmpy;                                                  1
    tmpy := right1y;                                                 1
    right1y := right2y;                                              1
    right2y := tmpy;                                                 1
END IF;
IF left1x < left2x AND right1x > left2x THEN                         1
IF left1y <= left2y AND right1y > left2y THEN                        1
       RETURN 1;                                                     1
    END IF;
    IF left1y <= right2y AND left1y > left2y THEN                    1
       RETURN 1;
    END IF;
END IF;
RETURN 0;
```

Ο αλγόριθμος αυτός λοιπόν χρειάζεται 2+2+2+2+1+1+1+1+1+1+1+1+1+ 1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+1+ 1+1+1+1+1 βήματα = 34 βήματα.

Έτσι η πολυπλοκότητα είναι 0(1).

1.7.2.3 Ο PolygonIntersectsPolygon αλγόριθμος

Υποθέτουμε ότι τα δύο πολύγωνα που θα δοκιμάσουμε για διασταύρωση έχουν N αριθμό σημείων το κάθε ένα. Μπορούμε να υποθέσουμε επίσης ότι ο edgesIntersect χρειάζεται 1 βήμα για να εκτελεστεί (όπως περιγράφτηκε πιο πριν). Έτσι, τα βήματα που κάνει αυτός ο αλγόριθμος είναι τα εξής:

```
point1 :=  pol1's polygon last points serial number              1
2. For each row with the next pol1's point in order do    N(steps
                                                        in the loop)
3. point2 := row.serial;                                         1
4. point3 := pol2's polygon last points serial number            1
5. For each row with the next pol2's point in order do    N(steps
                                                        in the loop)
6. point4 := row.serial;                                         1
7. IF edgesIntersect (pol1, pol2,                                1
      point1, point2, point3, point4)THEN
        RETURN 1;                                                1
        END IF;
8. point3 := point4;                                             1
9. End the loop starting at step5.
10. point1 := point2;                                            1
11. End the loop starting at step 2.
12. RETURN 0;
```

Ο αλγόριθμος αυτός λοιπόν χρειάζεται 1+N * (1+1+N*(1+1+1+1) +1) βήματα. Έτσι η πολυπλοκότητα είναι $O(N^2)$.

Για το υπόλοιπο των αλγορίθμων αρχικά παρουσιάζουμε την πολυπλοκότητα του κάθε αλγορίθμου, ανεξάρτητα από την πολυπλοκότητα των αλγορίθμων που χρησιμοποιεί και στη συνέχεια περιγράφουμε την αλλαγή της πολυπλοκότητας λόγω της χρήσης άλλων αλγορίθμων.

BoundingBoxContainsPoint: (1)

PolygonContainsPoint: (n)

PolygonContainsPolygon: O(1) => O(N) επειδή χρησιμοποιεί τον PolygonContainsPoint

edgesIntersect: O(1)

polygonIntersectsPolygon: (M * N) όπου M, N είναι οι αριθμοί των σημείων των δύο πολυγώνων, οπότε αν υποθέσουμε ότι M = N = υπερβολικά μεγάλο, τότε ο αλγόριθμος είναι $O(N^2)$.

polygonOverlapsPolygon: O(1) => (N) από την χρήση του PolygonContainsPolygon και (m* n) ή (N^2) από τη χρήση του PolygonIntersectsPolygon, έτσι ώστε ο αλγόριθμος να είναι (N^2).

movePolygon: O(M * N), όπου M είναι ο αριθμός των πολυγώνων που έχουν έναν Connect περιορισμό με τον οποίο μεταφέρθηκαν και N είναι ο αριθμός των σημείων των πολυγώνων. Επίσης, υποθέτουμε ότι μία ενημέρωση εντολών, που επηρεάζει τον N αριθμό γραμμών στον πίνακα, χρειάζεται βήματα (μόνο εάν υπήρχε αριθμός εντολών ενημέρωσης, επηρεάζοντας μια γραμμή στον πίνακα). Και πάλι, αν υποθέσουμε ότι M = N = πάρα πολύ μεγάλο, ο αλγόριθμος είναι $O(N^2)$.

countStepsToMove: $O(N^3)$ => $O(N^5)$επειδή ο αλγόριθμος δημιουργεί έναν βρόχο με N βήματα (μέχρις ότου να μην παρατηρούνται άλλες αλληλοεπικαλύψεις), μέσα στις οποίες δημιουργεί έναν άλλο βρόγχο με N βήματα (προκειμένου να έχουν πρόσβαση κάθε Connect περιορισμοί για το πολύγωνο στο ερώτημα) και, στη συνέχεια, μέσα στο δεύτερο βρόγχο δημιουργεί έναν άλλο βρόγχο με N βήματα (για να αποκτήσετε πρόσβαση στους Not_overlap περιορισμούς που αφορούν το πολύγωνο που ανακτήθηκε στον προηγούμενο βρόγχο). Μέσα στον τρίτο βρόγχο, ο αλγόριθμος χρησιμοποιεί τον polygonOverlapsPolygon αλγόριθμο, ο οποίος έχει μια πολυπλοκότητα $O(N^2)$. Έτσι, τελικά, ο αλγόριθμος είναι $O(N^5)$.

resolveOverlap: O(1) => $O(N^5)$. Επειδή ο αλγόριθμος χρησιμοποιεί τον countStepsToMove αλγόριθμο ο οποίος είναι (N^5). Επίσης χρησιμοποιεί τον movePolygon ο οποίος είναι (N^2), αλλά κρατάει την πολυπλοκότητα (N^5).

1.7.3 Τα αποτελέσματα των αξιολογήσεων

Στην ενότητα αυτή, παρουσιάζουμε μερικά αποτελέσματα που βρήκαμε μετά την προσομοίωση του συστήματός μας. Οι δοκιμές έγιναν σε ένα PC με επεξεργαστή Intel Celeron (2,4 GHz) CPU και 1 GB DDR2 RAM. Σε αυτές τις δοκιμές, εξετάσαμε τις διαφορετικές επιπτώσεις της κάθε παραμέτρου του συστήματος, όπως τον αριθμό των σημείων που κάθε σχήμα έχει, το εύρος των σημείων ενός σχήματος, τον αριθμό των σχημάτων που περιλαμβάνει ένας περιορισμός και την πολυπλοκότητα των περιορισμών.

1.7.3.1 Δοκιμή 1

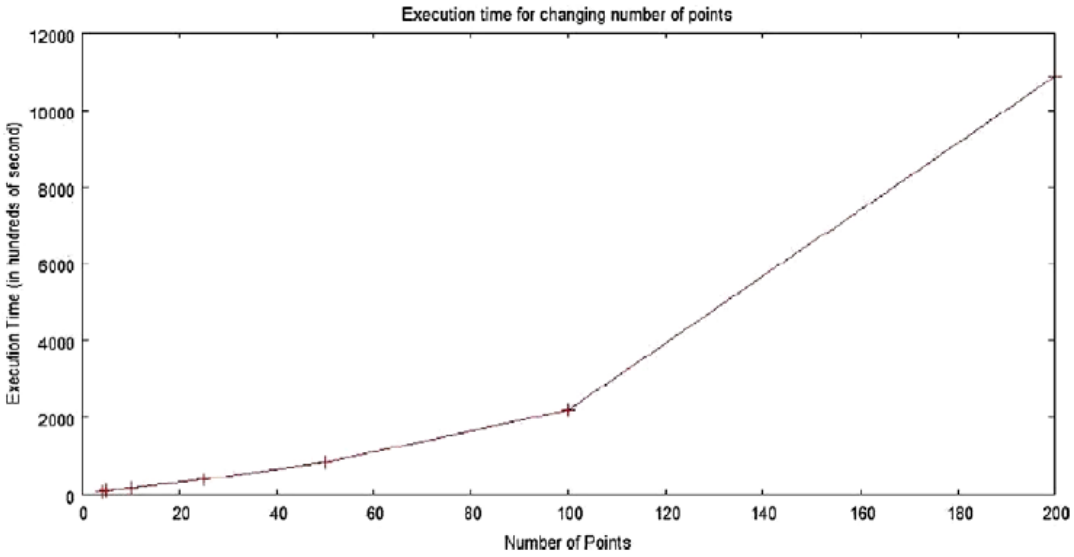
Η πρώτη δοκιμή θα δείξει ότι ο χρόνος εκτέλεσης του προγράμματος αυξάνεται με τον αριθμό των σημείων που κάθε πολύγωνο έχει. Δεδομένου ότι έχουμε δύο πολύγωνα στη βάση δεδομένων (που ονομάζονται A και B) και έναν Not_overlap περιορισμό μεταξύ τους, θα αρχίσουμε την προσθήκη σημείων σε κάθε πολύγωνο και τον υπολογισμό του χρόνου εκτέλεσης του αλγορίθμου. Τα αποτελέσματα εμφανίζονται στην Εικ. 6.

1.7.3.2 Δοκιμή 2

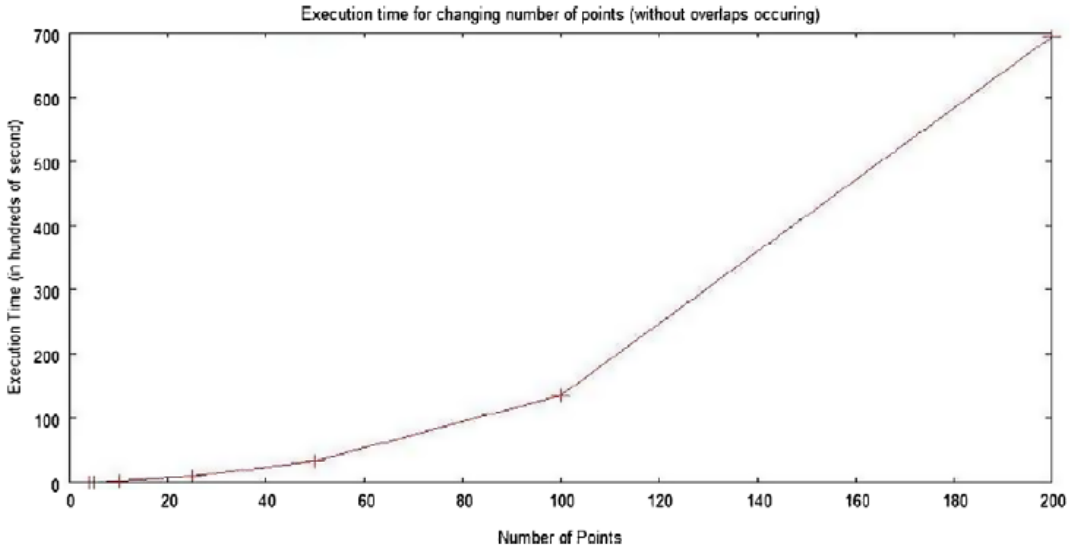
Η δεύτερη δοκιμή θα δείξει ότι ο χρόνος εκτέλεσης του προγράμματος αυξάνεται ελαφρώς με την αύξηση του αριθμού των σημείων σε κάθε πολύγωνο (όπως και στην προηγούμενη δοκιμή), όταν δεν παραβιάζονται οι Not_Overlap περιορισμοί. Το πρόγραμμα είναι πολύ πιο γρήγορο σε κάθε περίπτωση όταν δεν παραβιάζονται οι περιορισμοί της βάσης δεδομένων. Τα αποτελέσματα αυτής της δοκιμής εμφανίζονται στην Εικ. 7.

1.7.3.3 Δοκιμή 3

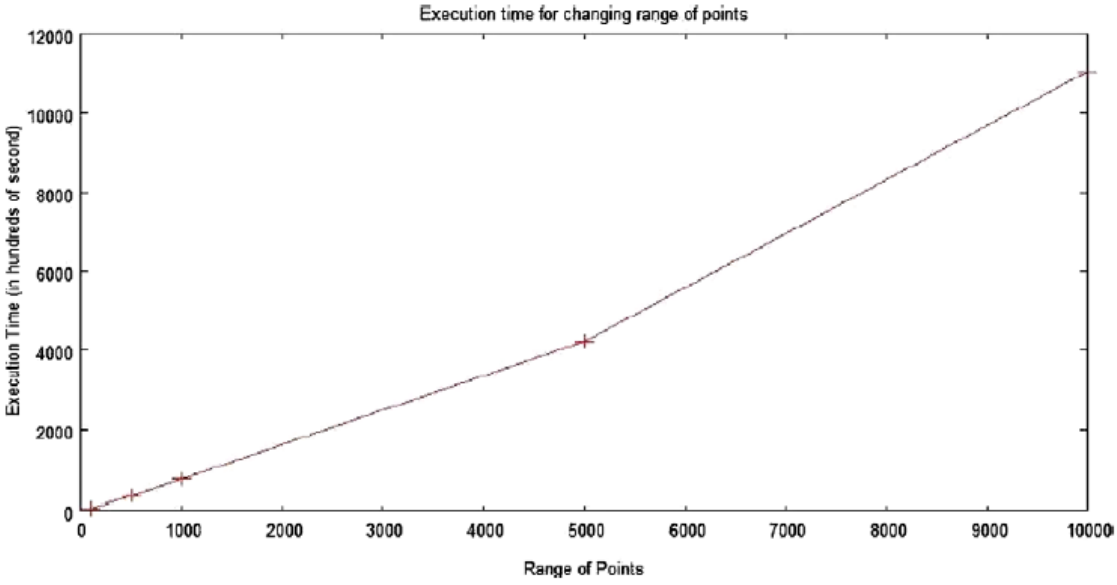
Στην τρίτη δοκιμή, θα εξετάσουμε πώς το φάσμα των συντεταγμένων από το σημείο ενός πολυγώνου επηρεάζει τον χρόνο εκτέλεσης του προγράμματος. Εάν ένα πολύγωνο έχει σημεία με μεγάλο εύρος (μακριά από το σημείο (0, 0)), ο χρόνος εκτέλεσης αυξάνεται. Λαμβάνοντας υπόψη δύο πολύγωνα (που ονομάζονται A και B) και έναν Not_overlap περιορισμό μεταξύ τους, έχουμε αυξήσει το εύρος των σημείων τους. Τα αποτελέσματα της δοκιμής αυτής εμφανίζονται στην Εικ. 8. Ο x άξονας αντιπροσωπεύει το φάσμα των σημείων ενός πολυγώνου, το οποίο είναι μια τιμή των 1000 που σημαίνει ότι τα σημεία είναι μεταξύ 0 και 1000 (π.χ. το πολύγωνο με εύρος 1000 μπορεί να είναι το ορθογώνιο με τα σημεία (0, 0) (1000,0) (1000, 1000) (1000, 0)).



Εικ. 1.6. Ο χρόνος εκτέλεσης του προγράμματος για την αλλαγή του αριθμού των σημείων.



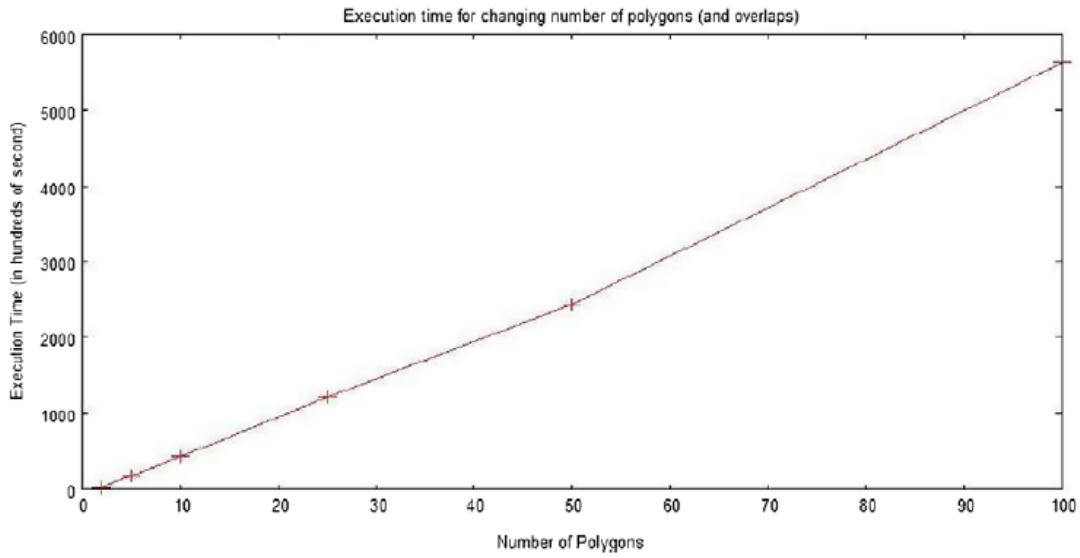
Εικ.1.7. Ο χρόνος εκτέλεσης του προγράμματος για την αλλαγή του αριθμού των σημείων (χωρίς την παρατήρηση αλληλοεπικαλύψεων).



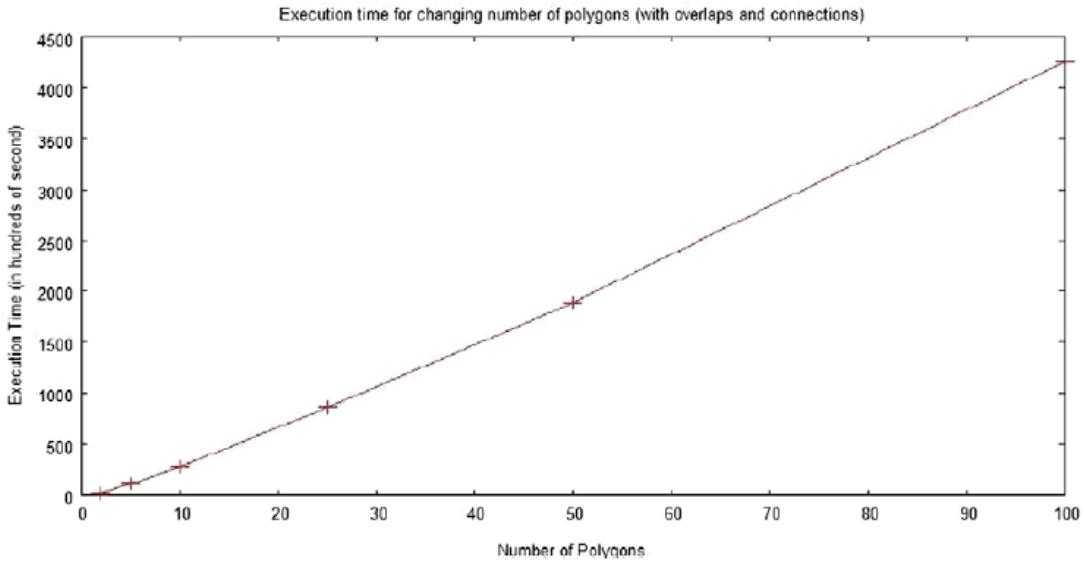
Εικ. 1.8. Ο χρόνος εκτέλεσης του προγράμματος για την αλλαγή του εύρους των σημείων.

1.7.3.4 Δοκιμή 4

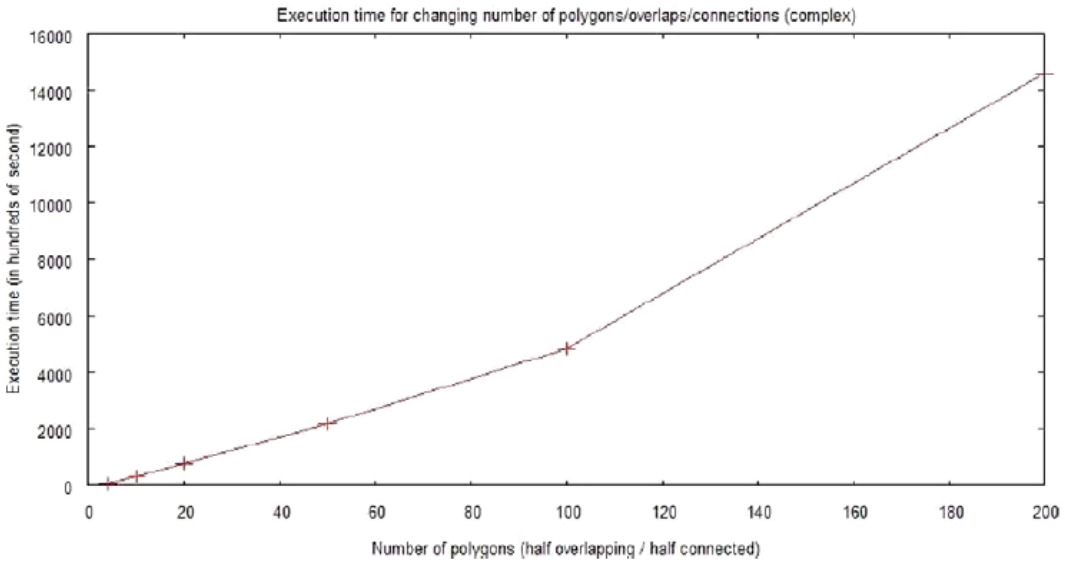
Στην τέταρτη δοκιμή, υποθέτουμε ότι έχουμε πολύγωνα με τέσσερα σημεία το καθένα, που επικαλύπτονται μεταξύ τους και υπάρχουν Not_overlap περιορισμοί. Η δοκιμή θα δείξει ότι ο χρόνος εκτέλεσης του προγράμματος αυξάνει όταν υπάρχει μια αύξηση στον αριθμό των πολυγώνων της βάσης και στον αριθμό των Not_overlap περιορισμών. Έτσι, σε κάθε στάδιο της δοκιμής αυτής, υποθέτουμε ότι έχουμε ένα X αριθμό πολυγώνων, για κάθε επικάλυψη με το επόμενο. Επίσης, ας υποθέσουμε ότι κάθε πολυγώνου δεν πρέπει να αλληλεπικαλύπτεται με το επόμενό του (Not_overlap περιορισμός). Για παράδειγμα, αν έχουμε τέσσερα πολύγωνα στη βάση δεδομένων (που ονομάζονται A, B, C και D), οι Not_overlap περιορισμοί είναι (A-B, B- C, C - D). Τα αποτελέσματα εμφανίζονται στην Εικ. 9.



Εικ. 1.9. Ο χρόνος εκτέλεσης του προγράμματος για την αλλαγή του αριθμού των πολυγώνων (και του αριθμού των Not_overlap περιορισμών.



Εικ. 1.10. Ο χρόνος εκτέλεσης του προγράμματος για την αλλαγή του αριθμού των πολυγώνων, των Not_overlap και των Connect περιορισμών.



Εικ. 1.11. Ο χρόνος εκτέλεσης του προγράμματος για την αλλαγή του αριθμού των πολυγώνων, των Not_overlap και των Connect περιορισμών (πιο σύνθετο).

1.7.3.5 Δοκιμή 5

Σε αυτή τη δοκιμή, κάνουμε το ίδιο πράγμα όπως και στην προηγούμενη δοκιμή, αλλά προσθέτουμε Connect περιορισμούς μεταξύ κάθε πολυγώνου και του πολυγώνου μετά από το επόμενό του. Για παράδειγμα, εάν έχουμε 4 πολύγωνα στη βάση δεδομένων (που ονομάζονται A, B, C και D), οι Not_overlap περιορισμοί είναι (A-B, B- C, C - D) και οι Connect περιορισμοί είναι (A- C, B- D). Τα αποτελέσματα εμφανίζονται στην Εικ. 10.

1.7.3.6 Δοκιμή 6

Σε αυτή την τελευταία δοκιμή, κάνουμε το ίδιο πράγμα όπως στη δοκιμή 5, αλλά περιπλέκουμε τους περιορισμούς. Έτσι, κάθε πολύγωνο είναι συνδεδεμένο με το επόμενο και δεν θα πρέπει να επικαλύπτεται το πολύγωνο μετά το επόμενό του (στο οποίο είναι συνδεδεμένο). Για παράδειγμα, αν έχουμε 4 πολύγωνα (που ονομάζονται A, B, C και D), οι Not_overlap περιορισμοί θα είναι (A-C) και οι Connect περιορισμοί θα είναι (A-B, C-D). Τα αποτελέσματα εμφανίζονται στην Εικ. 11.

1.8 Συμπεράσματα

Στην παρούσα εργασία, μελετήσαμε το πρόβλημα της διακλάδωσης στον καθορισμό των σχεσιακών χωρικών βάσεων δεδομένων. Πιο συγκεκριμένα, προτείναμε μία λύση και ένα μέσο που υλοποιεί τη λύση σε SQL.

Οι βασικές ιδέες της προσέγγισής μας είναι οι εξής:

- Να περιγράψουμε το πρόβλημα σε σχεσιακές χωρικές βάσεις δεδομένων .
- Να αναπτύξουμε αλγορίθμους για την ανακάλυψη και την διαγραφή ασυνεπειών.
- Να χρησιμοποιήσουμε δυναμικούς κανόνες για τη σύλληψη των άμεσων επιπτώσεων των ενεργειών, και στατικούς κανόνες για τη σύλληψη των έμμεσων επιπτώσεων των ενεργειών.
- Να αναπτύξουμε ένα σύστημα που εφαρμόζει τη λύση σε SQL για μια σχεσιακή βάση δεδομένων.

1.9 Μελλοντική εργασία και επεκτάσεις

Σε μελλοντική εργασία, σκοπεύουμε να επεκτείνουν το σύστημά μας για την περίπτωση που παρουσιάζεται ο χρόνος (χρονο-χωρικές βάσεις δεδομένων).

ΚΕΦΑΛΑΙΟ 2

Η Δομημένη Γλώσσα Ερωτημάτων SQL

ΕΙΚ. 2.1. Το Λογότυπο της SQL.

2.1 Τι Είναι η SQL

Η SQL αποτελεί μια στάνταρτ γλώσσα του ANSI για να μπορούμε να έχουμε πρόσβαση σε βάσεις δεδομένων.

Τα αρχικά SQL σημαίνουν **Structured Query Language**, δηλαδή **Δομημένη Γλώσσα Ερωτημάτων.**

Η SQL μάς δίνει τη δυνατότητα να έχουμε πρόσβαση σε μια βάση δεδομένων (database).

Η SQL αποτελεί μια στάνταρτ γλώσσα του ANSI (ANSI standard language).

Η SQL μπορεί να εκτελέσει ερωτήματα (queries) σχετικά με μια βάση δεδομένων.

Η SQL μπορεί να ανακτήσει δεδομένα από μια βάση δεδομένων.

Η SQL μπορεί να εισαγάγει νέες εγγραφές σε μια βάση δεδομένων.

Η SQL μπορεί να διαγράψει εγγραφές από μια βάση δεδομένων.

Η SQL μπορεί να ενημερώσει εγγραφές σε μια βάση δεδομένων.

Η SQL είναι πολύ εύκολη στην εκμάθηση.

Η SQL αποτελεί ένα στάνταρτ του ANSI (American National Standards Institute) για να μπορούμε να έχουμε πρόσβαση σε συστήματα βάσεων δεδομένων. Οι εντολές της SQL χρησιμοποιούνται για να ανακτήσουμε (retrieve) και να ενημερώσουμε (update) δεδομένα σε μια βάση δεδομένων (database).

Η SQL συνεργάζεται με προγράμματα βάσεων δεδομένων όπως είναι τα εξής : Access, Informix, Microsoft SQL Server, Oracle, Sybase και πολλά άλλα.

2.2 Οι Πίνακες Βάσεων Δεδομένων (Database Tables)

Οι βάσεις δεδομένων (databases) περιέχουν αντικείμενα (objects) που ονομάζονται Πίνακες (Tables). Οι Εγγραφές (Records) των δεδομένων αποθηκεύονται σ' αυτούς τους πίνακες. Οι Πίνακες αναγνωρίζονται με τα ονόματά τους, όπως "Persons", "Orders", "Suppliers" κ.ά.

Οι Πίνακες περιέχουν Στήλες (Columns) και Γραμμές (Rows) με δε-δομένα. Οι Γραμμές (Rows) περιέχουν εγγραφές (records), όπως μία εγγραφή για κάθε άτομο. Οι Στήλες (Columns) περιέχουν δεδομένα, όπως First Name, Last Name, Address και City.

Ακολουθεί ένα παράδειγμα ενός Πίνακα που ονομάζεται "Persons" :

LastName	FirstName	Address	City
Παπαδόπουλος	Δημήτριος	Τυρνόβου 15	Φλώρινα
Αντωνιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα
Γεωργιάδης	Νικόλαος	Φον Κοζάνη 10	Κοζάνη

Πίνακ. 2.1: Πίνακας SQL.

Τα LastName, FirstName, Address και City είναι οι Στήλες (Columns) του πίνακα. Οι Γραμμές (Rows) περιέχουν τρεις εγγραφές για τρία άτομα.

2.3 Τα Ερωτήματα της SQL (SQL Queries)

Με την SQL, μπορούμε να κάνουμε ένα ερώτημα (Query) σε μια βάση δεδομένων και να έχουμε ένα αποτέλεσμα (Result) σε μορφή πίνακα (tabu-lar form).

Ενα ερώτημα σαν το εξής :

SELECT LastName FROM Persons

Θα δώσει ένα αποτέλεσμα σαν το εξής :

LastName
Παπαδόπουλος
Αντωνιάδης
Γεωργιάδης

Πίνακ. 2.2: Πίνακας SQL.

Πρέπει να έχουμε υπόψη μας ότι μερικά συστήματα βάσεων δεδομένων απαιτούν το σύμβολο ; (semicolon) στο τέλος μιας εντολής SQL. Δεν θα χρησιμοποιήσουμε εδώ το σύμβολο ;.

2.4 Χειρισμός Δεδομένων της SQL (Data Manipulation)

Οπως υπονοεί και το όνομά της, η SQL είναι μια σύνταξη για την εκτέλεση ερωτημάτων (queries). Αλλά η γλώσσα της SQL περιλαμβάνει επίσης μια σύνταξη για την ενημέρωση εγγραφών, την εισαγωγή νέων εγγραφών και τη διαγραφή υπαρχόντων εγγραφών.

Αυτές οι εντολές ερωτημάτων και ενημέρωσης αποτελούν μαζί τη *Γλώσσα Χειρισμού Δεδομένων (Data Manipulation Language,* ***DML)***που αποτελεί κομμάτι της SQL :

SELECT - εξάγει δεδομένα από μια βάση δεδομένων.
UPDATE - ενημερώνει δεδομένα σε μια βάση δεδομένων.
DELETE - διαγράφει δεδομένα από μια βάση δεδομένων.
INSERT - εισάγει νέα δεδομένα σε μια βάση δεδομένων.

2.5 Ορισμός Δεδομένων της SQL (Data Definition)

Η Γλώσσα Ορισμού Δεδομένων *(Data Definition Language,* ***DDL)***, που αποτελεί μέρος της SQL, επιτρέπει τη δημιουργία και τη διαγραφή πινάκων μιας βάσης δεδομένων. Μπορούμε επίσης να

ορίσουμε indexes (keys), να καθορίσουμε συνδέσμους (links) ανάμεσα στους πίνακες και να επιβάλλουμε περιορισμούς ανάμεσα στους πίνακες μιας βάσης δεδομένων.

Οι σημαντικότερες εντολές DDL στην SQL είναι οι εξής :

CREATE TABLE - δημιουργεί έναν νέον πίνακα σε μια βάση δεδομένων.
ALTER TABLE - τροποποιεί έναν πίνακα σε μια βάση δεδομένων.
DROP TABLE - διαγράφει έναν πίνακα από μια βάση δεδομένων.
CREATE INDEX - δημιουργεί έναν index (search key).
DROP INDEX - διαγράφει έναν index.

2.6 Η SQL και οι Ενεργές Σελίδες Διακομιστή

Η SQL αποτελεί ένα σημαντικό κομμάτι της ASP, επειδή *το Ενεργό Αντικείμενο Δεδομένων (Active Data Object, **ADO**)* που χρησιμοποιείται στην ***ASP** (Active Server Pages)* για να μπορούμε να έχουμε πρόσβαση σε βάσεις δεδομένων, βασίζεται στην SQL για την πρόσβαση στα δεδομένα.

2.7 Η Εντολή Select της SQL

Η εντολή SELECT επιλέγει στήλες (columns) δεδομένων από μια βάση δεδομένων. Το αποτέλεσμα αποθηκεύεται σε μορφή πίνακα και αποκαλείται result set. Την χρησιμοποιούμε για να εμφανίζουμε (επιλέγουμε) πληροφορίες από έναν πίνακα ως εξής :

***SELECT** ονόματα_στηλών **FROM** όνομα_πίνακα*

Παράδειγμα : Επιλογή Στηλών από έναν Πίνακα

Για να επιλέξουμε τις στήλες "LastName" και "FirstName", χρησιμοποιούμε μια εντολή SELECT, ως εξής :

SELECT LastName, FirstName FROM Persons

Το αποτέλεσμα :

LastName	FirstName
Παπαδόπουλος	Δημήτριος
Αντωνιάδης	Αντώνιος
Γεωργιάδης	Νικόλαος

Πίνακ. 2.3: Πίνακας SQL.

Παράδειγμα : Επιλογή όλων των Στηλών

Για να επιλέξουμε όλες τις στήλες από τον πίνακα "Person", χρησιμοποιούμε το σύμβολο * αντί για όνομα στήλης, ως εξής :

*SELECT * FROM Persons*

Το αποτέλεσμα :

LastName	FirstName	Address	City
Παπαδόπουλος	Δημήτριος	Τυρνόβου 15	Φλώρινα
Αντωνιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα
Γεωργιάδης	Νικόλαος	Φον Κοζάνη 10	Κοζάνη

Πίνακ. 2.4: Πίνακας SQL.

2.8 Το Where Clause της SQL

Το WHERE clause χρησιμοποιείται για να καθορίσουμε ένα κριτήριο επιλογής (selection criteria). Για να μπορέσουμε να επιλέξουμε δεδομένα υπό συνθήκη από έναν πίνακα, πρέπει να προσθέσουμε ένα WHERE clause σε μια εντολή SELECT, ως εξής :

SELECT *στήλη* ***FROM*** *πίνακα* ***WHERE*** *στήλη συνθήκη τιμή*

Με το WHERE clause, μπορούμε να χρησιμοποιήσουμε αυτές τις συνθήκες :

Τελεστής (Operator)	Συνθήκη (Condition)
=	Ίσο
<>	Όχι ίσο
>	Μεγαλύτερο από
<	Μικρότερο από
>=	Μεγαλύτερο από ή ίσο με
<=	Μικρότερο από ή ίσο με
LIKE	Επεξηγείται παρακάτω

Πίνακ. 2.5: Πίνακας SQL.

Σε μερικές εκδόσεις της SQL, ο τελεστής για το όχι ίσο (<>), μπορεί να γραφεί ως εξής : ***!=***.

Παράδειγμα : Επιλογή Ατόμων από μια Πόλη

Για να επιλέξουμε μόνο τα άτομα που κατοικούν στην πόλη Φλώρινα, προσθέτουμε ένα WHERE clause στην εντολή SELECT, ως εξής :

*SELECT * FROM Persons WHERE City='Φλώρινα'*

Το αποτέλεσμα :

LastName	FirstName	Address	City	Year
Παπαδόπουλος	Δημήτριος	Τυρνόβου 15	Φλώρινα	951
Αντωνιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα	1978

Πίνακ. 2.6: Πίνακας SQL.

Έχουμε χρησιμοποιήσει μονά εισαγωγικά (single quotes) στις τιμές των συνθηκών στα παραδείγματα. Η SQL χρησιμοποιεί μονά εισαγωγικά στις αλφαριθμητικές τιμές (κείμενο). Τα περισσότερα συστήματα βάσεων δεδομένων αποδέχονται και τα διπλά εισαγωγικά (double quotes). Οι αριθμητικές τιμές δεν πρέπει να περικλείονται σε εισαγωγικά.

Για τις τιμές κειμένου (αλφαριθμητικές) :

Αυτό είναι σωστό :
*SELECT * FROM Persons WHERE FirstName='Δημήτριος'*
Αυτό δεν είναι σωστό :
SELECT * FROM Persons WHERE FirstName=Δημήτριος

Για τις αριθμητικές τιμές :

Αυτό είναι σωστό :
*SELECT * FROM Persons WHERE Year>1965*
Αυτό δεν είναι σωστό :
*SELECT * FROM Persons WHERE Year>'1965'*

2.9 Η Συνθήκη LIKE

Η συνθήκη LIKE χρησιμοποιείται για να καθορίσουμε μια αναζήτηση για ένα υπόδειγμα (pattern) σε μια στήλη.

Η σύνταξη είναι ως εξής :
***SELECT** στήλη **FROM** πίνακα **WHERE** στήλη **LIKE** υπόδειγμα*
Μπορούμε να χρησιμοποιήσουμε το σύμβολο "%" για να ορίσουμε χαρακτήρες μπαλαντέρ (wildcards) πριν και μετά από το υπόδειγμα.

Παράδειγμα : Επιλογή Ατόμων με Υπόδειγμα Ονόματος

Η επόμενη εντολή SQL θα επιστρέψει τα άτομα που το όνομά τους αρχίζει από 'Ο'.
*SELECT * FROM Persons WHERE FirstName LIKE 'O%'*
Η επόμενη εντολή SQL θα επιστρέψει τα άτομα που το όνομά τους τελειώνει σε 'α'.
*SELECT * FROM Persons WHERE FirstName LIKE '%α'*
Η επόμενη εντολή SQL θα επιστρέψει τα άτομα που το όνομά τους περιέχει το 'γα'.
*SELECT * FROM Persons WHERE FirstName LIKE '%γα%'*
Ολα τα παραπάνω παραδείγματα θα επιστρέψουν το εξής αποτέλεσμα :

LastName	FirstName	Address	City	Year
Δημητρίου	Ολγα	Ταγμ. Ναούμ 20	Φλώρινα	1951

Πίνακ. 2.7: Πίνακας SQL.

2.10 Οι Λογικοί Τελεστές And και Or

Τα AND και OR ενώνουν δύο ή περισσότερες συνθήκες σ' ένα WHE-RE clause. Ο τελεστής AND εμφανίζει μια γραμμή αν ΟΛΕΣ οι συνθήκες είναι αληθείς (true), ενώ ο τελεστής OR εμφανίζει μια γραμμή αν ΟΠΟΙΑΔΗΠΟΤΕ από τις συνθήκες είναι αληθής (true).

Ο αρχικός πίνακας είναι ο εξής :

LastName	FirstName	Address	City
Παπαδόπουλος	Δημήτριος	Τυρνόβου 15	Φλώρινα
Γεωργιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα
Γεωργιάδης	Νικόλαος	Ι. Καραβίτη 2	Φλώρινα

Πίνακ. 2.8: Πίνακας SQL.

Παράδειγμα

Χρησιμοποιούμε το AND για να εμφανίσουμε αυτούς που το όνομά τους είναι "Νικόλαος" και το επώνυμό τους είναι "Γεωργιάδης" :

*SELECT * FROM Persons*
WHERE FirstName='Νικόλαος'
AND LastName='Γεωργιάδης'

Αποτέλεσμα :

LastName	FirstName	Address	City
Γεωργιάδης	Νικόλαος	Ι. Καραβίτη 2	Φλώρινα

Πίνακ. 2.9: Πίνακας SQL.

Παράδειγμα

Χρησιμοποιούμε το OR για να εμφανίσουμε αυτούς που το όνομά τους είναι "Νικόλαος" ή το επώνυμό τους είναι "Γεωργιάδης" :

*SELECT * FROM Persons*
WHERE firstname='Νικόλαος'
OR lastname='Γεωργιάδης'

Αποτέλεσμα :

LastName	FirstName	Address	City
Γεωργιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα
Γεωργιάδης	Νικόλαος	Ι. Καραβίτη 2	Φλώρινα

Πίνακ. 2.10: Πίνακας SQL.

Παράδειγμα

Μπορούμε επίσης να συνδυάσουμε τα AND και OR, χρησιμοποιώντας παρενθέσεις για να σχηματίσουμε πολύπλοκες εκφράσεις :

*SELECT * FROM Persons WHERE*
(FirstName='Αντώνιος' OR FirstName='Νικόλαος')
AND LastName='Γεωργιάδης'
Αποτέλεσμα :

LastName	FirstName	Address	City
Γεωργιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα
Γεωργιάδης	Νικόλαος	Ι. Καραβίτη 2	Φλώρινα

Πίνακ. 2.11: Πίνακας SQL.

2.11 Ο Τελεστής Between ... And

Ο τελεστής BETWEEN ... AND επιλέγει μια περιοχή δεδομένων ανάμεσα σε δύο τιμές. Οι τιμές μπορεί να είναι αριθμοί, κείμενο ή ημερομηνίες.

SELECT *όνομα_στήλης* ***FROM*** *όνομα_πίνακα*
WHERE *όνομα_στήλης*
BETWEEN *τιμή1* ***AND*** *τιμή2*

Αρχικός πίνακας :

LastName	FirstName	Address	City
Παπαδόπουλος	Δημήτριος	Τυρνόβου 15	Φλώρινα
Γεωργιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα
Γεωργιάδης	Νικόλαος	Ι. Καραβίτη 2	Φλώρινα
Μαρκόπουλος	Νικόλαος	Φον Κάραγιαν 20	Κοζάνη

Πίνακ. 2.12: Πίνακας SQL.

Παράδειγμα

Για να εμφανίσουμε τα άτομα που βρίσκονται αλφαβητικά ανάμεσα στους "Γεωργιάδης" και "Μαρκόπουλος", αλλά και να τους περιλαμβάνουν :

*SELECT * FROM Persons WHERE LastName*
BETWEEN 'Γεωργιάδης' AND 'Μαρκόπουλος'
Αποτέλεσμα :

LastName	FirstName	Address	City
Γεωργιάδης	Αντώνιος	Π. Μελά 100	Φλώρινα
Γεωργιάδης	Νικόλαος	Ι. Καραβίτη 2	Φλώρινα
Μαρκόπουλος	Νικόλαος	Φον Κάραγιαν 20	Κοζάνη

Πίνακ. 2.13: Πίνακας SQL.

Παράδειγμα

Για να εμφανίσουμε τα άτομα που βρίσκονται έξω από την περιοχή που χρησιμοποιήσαμε στο προηγούμενο παράδειγμα, χρησιμοποιούμε τον τελεστή NOT :

*SELECT * FROM Persons WHERE LastName*
NOT BETWEEN 'Γεωργιάδης' AND 'Μαρκόπουλος'
Αποτέλεσμα :

LastName	FirstName	Address	City
Παπαδόπουλος	Δημήτριος	Τυρνόβου 15	Φλώρινα

Πίνακ. 2.14: Πίνακας SQL.

2.12 Η Λέξη Κλειδί Distinct

Η λέξη κλειδί DISTINCT χρησιμοποιείται για να επιστρέφει μόνο διακριτές (διαφορετικές) (distinct, different) τιμές. Η εντολή SELECT της SQL επιστρέφει στοιχεία από τις στήλες ενός πίνακα, αλλά τι μπορούμε να κάνουμε αν θέλουμε να επιλέξουμε μόνο διακριτά στοιχεία (distinct eleme-nts);

Στην SQL, αυτό που πρέπει να κάνουμε είναι να προσθέσουμε μια λέξη κλειδί DISTINCT στην εντολή SELECT, ως εξής :

SELECT DISTINCT *ονόματα_στηλών* ***FROM*** *όνομα_πίνακα*

Παράδειγμα

Επιλογή εταιρειών από έναν πίνακα παραγγελιών.
Ένας απλός πίνακας παραγγελιών :

Company	OrderNumber
Line Computers	3412
Sony	2312
Algorithm	4678
Sony	6798

Πίνακ. 2.15: Πίνακας SQL.

Η επόμενη εντολή SQL :
SELECT Company FROM Orders
θα δώσει αυτό το αποτέλεσμα :

Company
Line Computers
Sony
Algorithm
Sony

Πίνακ. 2.16: Πίνακας SQL.

Βλέπουμε ότι η εταιρεία Sony εμφανίζεται δύο φορές στο αποτέλεσμα. Μερικές φορές δεν το θέλουμε αυτό.

Παράδειγμα

Επιλογή ξεχωριστών εταιρειών από έναν πίνακα παραγγελιών.
Η επόμενη εντολή SQL :
SELECT DISTINCT Company FROM Orders
θα δώσει αυτό το αποτέλεσμα :

Company
Line Computers
Sony
Algorithm

Πίνακ. 2.17: Πίνακας SQL.

Τώρα η εταιρεία Sony εμφανίζεται μόνο μία φορά στο αποτέλεσμα.

2.13 Η Λέξη Κλειδί Order By

Η λέξη κλειδί ORDER BY χρησιμοποιείται για να ταξινομήσει το αποτέλεσμα. Το ORDER BY clause χρησιμοποιείται για να ταξινομήσει τις γραμμές.

Πίνακας Παραγγελίες :

Company	OrderNumber
Digital Shop	3412
ABC Shop	**5678**
Sony	2312
Sony	6798

Πίνακ. 2.18: Πίνακας SQL.

Παράδειγμα

Για να εμφανίσουμε τις εταιρείες σε αλφαβητική σειρά :
SELECT Company, OrderNumber FROM Orders
ORDER BY Company
Αποτέλεσμα :

Company	OrderNumber
ABC Shop	5678
Digital Shop	3412
Sony	6798
Sony	2312

Πίνακ. 2.19: Πίνακας SQL.

Παράδειγμα

Για να εμφανίσουμε τις εταιρείες σε αλφαβητική σειρά ΚΑΙ (AND) τις παραγγελίες σε αριθμητική σειρά :

SELECT Company, OrderNumber FROM Orders
ORDER BY Company, OrderNumber
Αποτέλεσμα :

Company	OrderNumber
ABC Shop	5678
Digital Shop	3412
Sony	2312
Sony	6798

Πίνακ. 2.20: Πίνακας SQL.

Παράδειγμα

Για να εμφανίσουμε τις εταιρείες σε αντίστροφη αλφαβητική σειρά (reverse alphabetical order):
SELECT Company, OrderNumber FROM Orders
ORDER BY Company DESC
Αποτέλεσμα :

Company	OrderNumber
Sony	2312
Sony	6798
Digital Shop	3412
ABC Shop	5678

Πίνακ. 2.21: Πίνακας SQL.

2.14 Η Εντολή INSERT INTO

Η εντολή INSERT INTO εισάγει νέες γραμμές σ' έναν πίνακα. Η σύνταξή της είναι ως εξής :

INSERT INTO *όνομα_πίνακα*
VALUES *(τιμή1, τιμή2, ...)*
Μπορούμε επίσης να καθορίσουμε τις στήλες για τις οποίες θέλουμε να εισάγουμε δεδομένα :
INSERT INTO *όνομα_πίνακα(στήλη1, στήλη2, ...)*
VALUES *(τμή1, τιμή2, ...)*

Ο επόμενος πίνακας "Persons" :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα

Πίνακ. 2.22: Πίνακας SQL.

και αυτή η εντολή SQL :
INSERT INTO Persons
VALUES ('Σιάμκουρης', 'Ιωάννης', 'Π. Μελά 90', 'Καστοριά')
δίνουν αυτό το αποτέλεσμα :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά

Πίνακ. 2.23: Πίνακας SQL.

Εισαγωγή Δεδομένων σε Συγκεκριμένες Στήλες

Ο επόμενος πίνακας "Persons" :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά

Πίνακ. 2.24: Πίνακας SQL.

και η επόμενη εντολή SQL :
INSERT INTO Persons (LastName, Address)
VALUES ('Νικολάου', 'Ταγμ. Ναούμ 30')
δίνουν αυτό το αποτέλεσμα :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά
Νικολάου		Ταγμ. Ναούμ 30	

Πίνακ. 2.25: Πίνακας SQL.

2.15 Η Εντολή Update

Η εντολή UPDATE ενημερώνει ή αλλάζει γραμμές. Η σύνταξή της είναι ως εξής :
UPDATE *όνομα_πίνακα* ***SET*** *όνομα_στήλης=νέα_τιμή*
WHERE *όνομα_στήλης=τιμή*
Αρχικός πίνακας Person :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά
Νικολάου		Ταγμ. Ναούμ 30	

Πίνακ. 2.26: Πίνακας SQL.

Ενημέρωση μίας Στήλης σε μια Γραμμή

Θέλουμε να προσθέσουμε ένα όνομα στο άτομο που έχει το επώνυμο "Νικολάου" :
UPDATE Person SET FirstName = 'Αθηνά'
WHERE LastName = 'Νικολάου'

Ενημέρωση Πολλών Στηλών σε μια Γραμμή

Για το ίδιο άτομο θέλουμε να αλλάξουμε τη διεύθυνση και να προσθέσουμε ένα όνομα για την πόλη :

UPDATE Person
SET Address = 'Μεγαρόβου 12', City ='Φλώρινα'
WHERE LastName = 'Νικολάου'
Αποτέλεσμα :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά
Νικολάου	Αθηνά	Μεγαρόβου 12	Φλώρινα

Πίνακ. 2.27: Πίνακας SQL.

2.16 Η Εντολή Delete

Η εντολή DELETE χρησιμοποιείται για να διαγράψουμε γραμμές από έναν πίνακα. Η σύνταξή της είναι ως εξής :

DELETE FROM *όνομα_πίνακα*
WHERE *όνομα_στήλης = τιμή*

Αρχικός πίνακας Person :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά
Νικολάου	Αθηνά	Μεγαρόβου 12	Φλώρινα

Πίνακ. 2.28: Πίνακας SQL.

Διαγραφή μιας Γραμμής

Θα διαγράψουμε την "Νικολάου Αθηνά" :
DELETE FROM Person WHERE LastName = 'Νικολάου'
Αποτέλεσμα :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά

Πίνακ. 2.29: Πίνακας SQL.

2.17 Οι Συναρτήσεις Count της SQL

Η SQL έχει ενσωματωμένες συναρτήσεις για τη μέτρηση (counting) των εγγραφών μιας βάσης δεδομένων.

Η σύνταξη για τις ενσωματωμένες συναρτήσεις COUNT είναι η εξής :

SELECT COUNT*(στήλη)* ***FROM*** *πίνακας*

2.17.1 Η Συνάρτηση COUNT(*)

Η συνάρτηση COUNT(*) επιστρέφει τον αριθμό των επιλεγμένων γραμμών από μια επιλογή (selection).

Μ’ αυτόν τον πίνακα "Persons" :

Name	Age
Αντωνιάδης Ηλίας	34
Μάρκου Ιωάννης	45
Ιωάννου Γεώργιος	19

Πίνακ. 2.30: Πίνακας SQL.

Το επόμενο παράδειγμα επιστρέφει τον αριθμό των γραμμών του πίνακα :
SELECT COUNT(*) FROM Persons
Αποτέλεσμα : 3

Το επόμενο παράδειγμα επιστρέφει τον αριθμό των ατόμων που είναι πάνω από 20 χρονών :
SELECT COUNT(*) FROM Persons where Age>20
Αποτέλεσμα : 2

2.17.2 Η Συνάρτηση COUNT(column)

Η συνάρτηση COUNT(column) επιστρέφει τον αριθμό των γραμμών χωρίς τιμή NULL στη συγκεκριμένη στήλη.

Μ’ αυτόν τον πίνακα "Persons" :

Name	Age
Αντωνιάδης Ηλίας	34
Μάρκου Ιωάννης	45
Ιωάννου Γεώργιος	

Πίνακ. 2.31: Πίνακας SQL.

Το επόμενο παράδειγμα βρίσκει τον αριθμό των ατόμων που έχουν τιμή στο πεδίο "Age" του πίνακα "Persons" :

SELECT COUNT(Age) FROM Persons

Αποτέλεσμα : 2

Η συνάρτηση COUNT(column) είναι χρήσιμη για να βρίσκουμε τις στήλες που δεν έχουν τιμή. Το αποτέλεσμα είναι κατά ένα λιγότερο από τον αριθμό των γραμμών του αρχικού πίνακα επειδή ένα από τα άτομα δεν έχει τιμή στο πεδίο age.

2.18 Οι Λέξεις Κλειδιά COUNT και DISTINCT

Οι λέξεις κλειδιά DISTINCT και COUNT μπορούν να χρησιμοποιηθούν μαζί για να μετρήσουμε τον αριθμό των διακριτών αποτελεσμάτων (di-stinct results).

Η σύνταξη είναι ως εξής :

SELECT DISTINCT COUNT*(στήλες)* ***FROM*** *πίνακας*

Με τον επόμενο πίνακα "Orders" :

Company	OrderNumber
Hitachi	3412
Sony	2312
ABC	4678
Sony	6798

Πίνακ. 2.32: Πίνακας SQL.

Η επόμενη εντολή SQL :

SELECT COUNT(Company) FROM Orders

θα δώσει αυτό το αποτέλεσμα : 4

Η επόμενη εντολή SQL :

SELECT DISTINCT COUNT(Company) FROM Orders

θα δώσει αυτό το αποτέλεσμα : 3

2.19 Οι Συναρτήσεις της SQL

Η SQL έχει πολλές ενσωματωμένες συναρτήσεις για να μπορούμε να κάνουμε μετρήσεις (counting) και υπολογισμούς (calculations).

Η γενική σύνταξη για τις ενσωματωμένες συναρτήσεις της SQL είναι η εξής :

SELECT *function(στήλη)* ***FROM*** *πίνακας*

Ο αρχικός πίνακας :

Name	Age
Αντωνιάδης Ηλίας	34
Μάρκου Ιωάννης	45
Ιωάννου Γεώργιος	19

Πίνακ. 2.33: Πίνακας SQL.

2.19.1 Η Συνάρτηση AVG(column)

Η συνάρτηση AVG επιστρέφει τη μέση τιμή μιας στήλης σε μια επιλογή. Οι τιμές NULL δεν περιλαμβάνονται στους υπολογισμούς.

Το επόμενο παράδειγμα επιστρέφει τον μέσο όρο ηλικίας των ατόμων που υπάρχουν στον πίνακα "Persons" :

SELECT AVG(Age) FROM Persons

Αποτέλεσμα : 32.67

Το επόμενο παράδειγμα επιστρέφει τον μέσο όρο ηλικίας των ατόμων που είναι πάνω από 20 χρονών :

SELECT AVG(Age) FROM Persons where Age>20

Αποτέλεσμα : 39.5

2.19.2 Η Συνάρτηση MAX(column)

Η συνάρτηση MAX επιστρέφει την μεγαλύτερη τιμή μιας στήλης. Οι τιμές NULL δεν περιλαμβάνονται στον υπολογισμό.

Παράδειγμα

SELECT MAX(Age) FROM Persons

Αποτέλεσμα : 45

2.19.3 Η Συνάρτηση MIN(column)

Η συνάρτηση MIN επιστρέφει την μικρότερη τιμή μιας στήλης. Οι τιμές NULL δεν περιλαμβάνονται στον υπολογισμό.

Παράδειγμα

SELECT MIN(Age) FROM Persons

Αποτέλεσμα : 19

Οι συναρτήσεις MIN και MAX μπορούν επίσης να χρησιμοποιηθούν σε στήλες που περιέχουν κείμενο (text), για να βρούμε την μεγαλύτερη ή μικρότερη τιμή σε αλφαβητική σειρά.

2.19.4 Η Συνάρτηση SUM(column)

Η συνάρτηση SUM επιστρέφει το άθροισμα μιας στήλης για μια συγκεκριμένη επιλογή (selection). Οι τιμές NULL δεν περιλαμβάνονται στον υπολογισμό.

Παράδειγμα

Το παρακάτω παράδειγμα επιστρέφει το άθροισμα όλων των ηλικιών του πίνακα "person" :
SELECT SUM(Age) FROM Persons
Αποτέλεσμα : 98

Παράδειγμα

Το παρακάτω παράδειγμα επιστρέφει το άθροισμα των ηλικιών του πίνακα "person" για τα άτομα που είναι πάνω από 20 χρονών :
SELECT SUM(Age) FROM Persons where Age>20
Αποτέλεσμα : 79

2.20 Η Λέξη Κλειδί Group By

Η λέξη κλειδί GROUP BY έχει προστεθεί στην SQL επειδή οι αθροιστικές συναρτήσεις (aggregate functions), όπως είναι η SUM, επιστρέφουν το σύνολο όλων των τιμών μιας στήλης κάθε φορά που καλούνται.

Χωρίς την λέξη κλειδί GROUP BY, το να βρούμε το άθροισμα για κά-θε ανεξάρτητη ομάδα τιμών μιας στήλης θα ήταν αδύνατο.

Η σύνταξη της GROUP BY είναι η εξής :
***SELECT** στήλη, **SUM**(στήλη) **FROM** πίνακας **GROUP BY** στήλη*

Παράδειγμα με GROUP BY

Ο παρακάτω πίνακας "Sales" :

Company	Amount
Grundig	5500
IBM	4500
Grundig	7100

Πίνακ. 2.34: Πίνακας SQL.

και αυτή η εντολή SQL :
SELECT Company, SUM(Amount) FROM Sales
θα δώσουν αυτό το αποτέλεσμα :

Company	SUM(Amount)
Grundig	17100
IBM	17100
Grundig	17100

Πίνακ. 2.35: Πίνακας SQL.

Ο παραπάνω κώδικας δεν είναι έγκυρος επειδή η στήλη που επιστρέφει δεν αποτελεί μέρος ενός αθροίσματος (aggregate). Ένα GROUP BY clause μπορεί να το διορθώσει αυτό, ως εξής :

SELECT Company, SUM(Amount) FROM Sales
GROUP BY Company
και θα δώσει αυτό το αποτέλεσμα :

Company	SUM(Amount)
Grundig	12600
IBM	4500

Πίνακ. 2.36: Πίνακας SQL.

2.21 Η Λέξη Κλειδί HAVING

Η λέξη κλειδί HAVING έχει προστεθεί στην SQL επειδή η λέξη κλειδί WHERE δεν μπορεί να χρησιμοποιηθεί σε αθροιστικές συναρτήσεις, όπως είναι η SUM.

Η σύνταξη της συνάρτησης HAVING είναι η εξής :
SELECT στήλη, SUM(στήλη) FROM πίνακας
GROUP BY στήλη
HAVING SUM(στήλη) συνθήκη

Ο παρακάτω πίνακας "Sales" :

Company	Amount
Grundig	5500
IBM	4500
Grundig	7100

Πίνακ. 2.:37 Πίνακας SQL.

και αυτή η εντολή SQL :

SELECT Company, SUM(Amount) FROM Sales
GROUP BY Company HAVING SUM(Amount)>10000
θα δώσουν αυτό το αποτέλεσμα :

Company	SUM(Amount)
Grundig	12600

Πίνακ. 2.38: Πίνακας SQL.

2.22 Τα Ψευδώνυμα (Aliases)

Στην SQL, τα ψευδώνυμα (aliases) χρησιμοποιούνται για ονόματα στηλών και πινάκων.

2.22.1 Ψευδώνυμο Στήλης (Column Name Alias)

Η σύνταξη είναι ως εξής :
SELECT *στήλη* ***AS*** *column_alias* ***FROM*** *πίνακας*

Παράδειγμα με Column Alias

Ο επόμενος πίνακας Persons :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά
Νικολάου	Αθηνά	Μεγαρόβου 12	Φλώρινα

Πίνακ. 2.39: Πίνακας SQL.

και η εξής εντολή SQL :
SELECT LastName AS Family, FirstName AS Name
FROM Persons
δίνουν αυτό το αποτέλεσμα :

Family	Name
Σουμπάση	Μαρία
Σιάμκουρης	Ιωάννης
Νικολάου	Αθηνά

Πίνακ. 2.40: Πίνακας SQL.

2.22.2 Ψευδώνυμο Πίνακα (Table Name Alias)

Η σύνταξη είναι ως εξής :
SELECT *στήλη* ***FROM*** *πίνακας* ***AS*** *table_alias*

Παράδειγμα με Table Alias

Ο επόμενος πίνακας Persons :

LastName	FirstName	Address	City
Σουμπάση	Μαρία	Καλλέργη 15	Φλώρινα
Σιάμκουρης	Ιωάννης	Π. Μελά 90	Καστοριά
Νικολάου	Αθηνά	Μεγαρόβου 12	Φλώρινα

Πίνακ. 2.41: Πίνακας SQL.

και η εξής εντολή SQL :
SELECT LastName, FirstName
FROM Persons AS Employees
δίνουν αυτό το αποτέλεσμα :

LastName	FirstName
Σουμπάση	Μαρία
Σιάμκουρης	Ιωάννης
Νικολάου	Αθηνά

Πίνακ. 2.42: Πίνακας SQL.

2.23 Ένωση Πινάκων (Join)

Μερικές φορές πρέπει να επιλέξουμε δεδομένα από δύο πίνακες για να δημιουργήσουμε ένα πιο πολύπλοκο αποτέλεσμα. Θα πρέπει να κάνουμε μια ***ένωση*** *(join).*

Οι πίνακες μιας βάσης δεδομένων μπορούν να συσχετιστούν μεταξύ τους με ***κλειδιά*** *(keys).* Ένα ***πρωτεύον κλειδί*** *(primary key)* είναι μια στήλη με μια μοναδική τιμή στην κάθε γραμμή. Ο σκοπός είναι να ενώσει τα δεδομένα μαζί από διάφορους πίνακες, χωρίς να έχουμε επανάληψη όλων των δεδομένων σε κάθε πίνακα.

Στον πίνακα "Employees" παρακάτω, η στήλη "ID" είναι το πρωτεύον κλειδί (primary key), που σημαίνει ότι δεν μπορούν να υπάρχουν δύο γραμμές που να έχουν το ίδιο ID. Το ID είναι αυτό που ξεχωρίζει δύο άτομα ακόμη κι αν έχουν το ίδιο όνομα.

Στους πίνακες παραδειγμάτων παρακάτω, προσέχουμε τα εξής :

Η στήλη "ID" είναι το πρωτεύον κλειδί του πίνακα "Employees".

Η στήλη "ID" του πίνακα "Orders" χρησιμοποιείται για να αναφερόμαστε στα άτομα του πίνακα "Employees", χωρίς να χρησιμοποιούμε τα ονόματά τους.

Πίνακας Employees :

ID	Name
01	Νικολάου Αθηνά
02	Γεωργιάδης Ηλίας
03	Παπαδόπουλος Στέφανος
04	Σουμπάσης Ιωάννης

Πίνακ. 2.43: Πίνακας SQL.

Πίνακας Orders :

ID	Product
01	Printer
03	Computer
03	Scanner

Πίνακ. 2.44: Πίνακας SQL.

Παράδειγμα

Ποιοι έχουν παραγγείλει προϊόντα και τι έχουν παραγγείλει;
SELECT Employees.Name, Orders.Product
FROM Employees, Orders
WHERE Employees.ID = Orders.ID
Αποτέλεσμα :

Name	Product
Νικολάου Αθηνά	Printer
Παπαδόπουλος Στέφανος	Computer
Παπαδόπουλος Στέφανος	Scanner

Πίνακ. 2.45: Πίνακας SQL.

Παράδειγμα

Ποιοι έχουν παραγγείλει εκτυπωτή (printer);
SELECT Employees.Name
FROM Employees, Orders
WHERE Employees.ID = Orders.ID
AND Orders.Product = 'Printer'
Αποτέλεσμα :

Name
Νικολάου Αθηνά

Πίνακ. 2.46: Πίνακας SQL.

2.24 Δημιουργία Βάσης Δεδομένων και Πίνακα

Για να δημιουργήσουμε μια βάση δεδομένων, μπορούμε να δώσουμε την εξής εντολή :
CREATE DATABASE *όνομα_βάσης_δεδομένων*

Για να δημιουργήσουμε έναν πίνακα σε μια βάση δεδομένων, μπορούμε να δώσουμε την εξής εντολή :

CREATE TABLE *όνομα_πίνακα*
(
όνομα_στήλης_1 τύπος_δεδομένων,
όνομα_στήλης_2 τύπος_δεδομένων,
…
)

Παράδειγμα

Θα δημιουργήσουμε έναν πίνακα με όνομα "Person", με τέσσερις στήλες με ονόματα "LastName", "FirstName", "Address" και "Age" :

CREATE TABLE Person
(
LastName varchar,
FirstName varchar,
Address varchar,
Age number
)

Θα ορίσουμε ένα μέγιστο μήκος για μερικές στήλες :
CREATE TABLE Person
(
LastName varchar(30),
FirstName varchar,
Address varchar,
Age number(3)
)

Ο τύπος δεδομένων (data type) καθορίζει τι είδος δεδομένων θα περιέχει η στήλη. Ο παρακάτω πίνακας περιέχει τους πιο συνηθισμένους τύπους δεδομένων της SQL :

Τύπος Δεδομένων (Data Type)	**Περιγραφή (Description)**
char(size)	Περιέχει ένα string σταθερού μήκους που μπορεί να περιέχει γράμματα, αριθμούς και ειδικούς χαρακτήρες. Το σταθερό μέγεθος καθορίζεται στις παρενθέσεις.
varchar(size)	Περιέχει ένα string μεταβλητού μήκους που μπορεί να περιέχει γράμματα, αριθμούς και ειδικούς χαρακτήρες. Το μέγιστο μέγεθος καθορίζεται στις παρενθέσεις.
number(size)	Περιέχει έναν αριθμό, όπου ο μέγιστος αριθμός των ψηφίων καθορίζεται στις παρενθέσεις.
number(size, d)	Περιέχει έναν αριθμό, όπου ο μέγιστος αριθμός των ψηφίων καθορίζεται στο size και ο μέγιστος αριθμός των ψηφίων στα δεξιά της υποδιαστολής καθορίζεται στο d.
date	Περιέχει μια ημερομηνία.

Πίνακ. 2.47: Πίνακας SQL.

2.25 Διαγραφή Βάσης Δεδομένων και Πίνακα

Για να διαγράψουμε μια βάση δεδομένων, χρησιμοποιούμε την εξής εντολή :
DROP DATABASE *όνομα_βάσης_δεδομένων*

Για να διαγράψουμε έναν πίνακα, χρησιμοποιούμε την εξής εντολή :
DROP TABLE *όνομα_πίνακα*

Για να διαγράψουμε τα δεδομένα ενός πίνακα χωρίς να διαγράψουμε τον πίνακα, χρησιμοποιούμε την εξής εντολή :
DELETE TABLE *όνομα_πίνακα*

2.26 Η Εντολή Alter Table

Η εντολή ALTER TABLE χρησιμοποιείται για να προσθέσουμε ή να διαγράψουμε στήλες από έναν υπάρχοντα πίνακα.
Η σύνταξή της για τις δύο αυτές περιπτώσεις είναι ως εξής :
ALTER TABLE *όνομα_πίνακα* ***ADD*** *όνομα_στήλης τύπος_δεδομένων*
ALTER TABLE *όνομα_πίνακα* ***DROP*** *όνομα_στήλης*

Πίνακας Person :

LastName	FirstName	Address
Σταύρου	Μαρίνα	Σαρανταπόρου 10

Πίνακ. 2.48: Πίνακας SQL.

Παράδειγμα

Για να προσθέσουμε μια στήλη με όνομα "City" στον πίνακα "Person", δίνουμε την εξής εντολή:

ALTER TABLE Person ADD City varchar(30)

Αποτέλεσμα :

LastName	FirstName	Address	City
Σταύρου	Μαρίνα	Σαρανταπόρου 10	

Πίνακ. 2.49: Πίνακας SQL.

Παράδειγμα

Για να διαγράψουμε (drop) τη στήλη "Address" του πίνακα "Person", δίνουμε την εξής εντολή :

ALTER TABLE Person DROP Address

Αποτέλεσμα :

LastName	FirstName	City
Σταύρου	Μαρίνα	

Πίνακ. 2.50: Πίνακας SQL.

ΚΕΦΑΛΑΙΟ 3

Η Γλώσσα Προγραμματισμού PHP

Εικ. 3.1. Το Λογότυπο της PHP.

3.1 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 από ειδικά tags αρχής και τέλους για να μπορούμε να εισερχόμαστε και να εξερόμαστε 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.

3.2 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. 3.1: 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.

3.3 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.

3.4 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" ?>
```

3.5 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 */
      */
?>
```

3.6 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.

3.6.1 Πίνακες

3.6.1.1 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().

3.6.2 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
```

3.6.2 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();
```

3.7 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"
```

3.7.1 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.

3.7.2 Οι Μεταβλητές της PHP

Αυτές οι μεταβλητές δημιουργούνται από την ίδια την PHP.

- ***Argv***
- ***argc***
- ***PHP_SELF***
- ***HTTP_COOKIE_VARS***
- ***HTTP_GET_VARS***
- ***HTTP_POST_VARS***

3.7.3 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.

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

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.

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.

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

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.

3.7.4 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.

3.7. 5 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.

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) :

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.

3.7.6 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.

3.7.7 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);
```

3.7.8 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().

3.8 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__***
- ***__LINE__***
- ***PHP_VERSION***
- ***PHP_OS***
- ***TRUE***
- ***FALSE***
- ***E_ERROR***
- ***E_WARNING***
- ***E_PARSE***
- ***E_NOTICE***

3.9 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.

3.10 Οι Τελεστές

3.10.1 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. 3.2:Οι Αριθμητικοί τελεστές της PHP

3.10.2 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» γι' 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.

3.10.3 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.

Παράδειγ-μα	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. 3.3:Οι Τελεστές Διαδικών Πράξεων της PHP

3.10.4 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. 3.4: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.

3.10.5 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().

3.10.6 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. 3.5:Οι Τελεστές αύξησης / μείωσης της PHP

3.10.7 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. 3.6:Οι Λογικοί τελεστές της PHP

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

3.10.8 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!"
```

3.11 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 βρόχος, 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.

3.11.1 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.

3.11.2 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";
}
```

3.11.3 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 ifαποτιμηθεί 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";
n} else {
    print "Το a είναι μικρότερο από το b";
```

```
}
```

3.11.4 Η Εντολή While

Οι βρόχοι while αποτελούν τον απλούστερο τύπο βρόχου στην PHP και συμπεριφέρονται όπως ακριβώς οι αντίστοιχοι βρόχοι 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 βρόχου. Όπως και με την εντολή if, μπορούμε να ομαδοποιήσουμε πολλές εντολές μέσα στον ίδιο βρόχο while χρησιμοποιώντας τα { και } ή την εναλλακτική σύνταξη :

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

3.11.5 Η Εντολή Do .. While

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

3.11.6 Η Εντολή For

Οι βρόχοι for είναι οι πιο πολύπλοκοι βρόχοι στην PHP. Συμπεριφέρονται όπως οι αντίστοιχοί τους στην C και η σύνταξη ενός βρόχου 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 βρόχου. Στην αρχή της κάθε επανάληψης αποτιμάται η έκφραση2 και αν αποτιμηθεί σε TRUE, ο βρόχος συνεχίζεται και εκτελούνται οι περιεχόμενες εντολές. Αν αποτιμηθεί σε FALSE, σταματάει η εκτέλεση του βρόχου. 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 βρόχος θα εκτελείται ασταμάτητα. Αυτό είναι χρήσιμο όταν θέλουμε να βγούμε από τον βρόχο χρησιμοποιώντας μια εντολή break.

3.11.7 Η Εντολή Break

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

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

3.11.8 Η Εντολή Continue

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

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

3.11.9 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.

3.12 Κλάσεις και Αντικείμενα

Μια κλάση (class) είναι μια συλλογή από μεταβλητές και από συναρτήσεις που εφαρμόζονται σ' αυτές τις μεταβλητές. Για να ορίσουμε μια κλάση 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;
        }
```

Οι κλάσεις 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 κλάση Cart. Η συνάρτηση add_item() αυτού του αντικειμένου καλείται για να προσθέσει ένα προϊόν με κωδικό αριθμό 10 στο cart. Οι κλάσεις μπορεί να είναι επεκτάσεις (extensions) άλλων τάξεων. Η προκύπτουσα κλάση έχει όλες τις μεταβλητές και τις συναρτήσεις της βασικής κλάσης 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);
```

3.13 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) εικόνων GIF. Θα χρειασθεί να μεταγλωττίσουμε την PHP με τη βιβλιοθήκη GD των συναρτήσεων εικόνας για να μπορέσει να δουλέψει αυτό.

3.14 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).

3.15 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.

3.16 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.

3.17 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.

3.18 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.

ΚΕΦΑΛΑΙΟ 4
Η Τεχνολογία JSP (JavaServer Pages)

Εικ. 4.1. Το Λογότυπο της JSP.

4.1 Οι Δυναμικές Εφαρμογές του Web

Όλοι μας θα έχουμε παρατηρήσει ότι όταν μπαίνουμε στο site ενός τουριστικού γραφείου, για παράδειγμα, και καταχωρούμε τον επιθυμητό προορισμό, εμφανίζονται διαφορετικές ιστοσελίδες, με διαφορετικό κείμενο και διαφορετικές φωτογραφίες, ανάλογα με την καταχώρηση που έχουμε κάνει. Πώς, όμως, γίνεται αυτό και λαμβάνουμε διαφορετικές πληροφορίες από το ίδιο ακριβώς Web site; Τα Web sites αυτού του είδους διαθέτουν μια εφαρμογή, που αποκαλείται εφαρμογή του Web (Web application), η οποία δέχεται τις αιτήσεις (requests) του χρήστη (επισκέπτη) του site και δημιουργεί μια απάντηση (response) που να ταιριάζει με το κάθε συγκεκριμένο ερώτημα (query) που υποβάλλεται.

Επειδή η εφαρμογή του Web δημιουργεί αυτές τις ιστοσελίδες στον αέρα (on-the-fly), λέμε ότι επιστρέφει δυναμικό περιεχόμενο (dynamic content). Αυτό σημαίνει ότι η απόκριση απ' αυτά τασites, όπως είναι η εμφάνιση πληροφοριών για τα διάφορα τουριστικά μέρη, δημιουργείται δυναμικά κάθε φορά που καταχωρείται ένα συγκεκριμένο ερώτημα.

Τι σημαίνει ο όρος Εφαρμογή του Web (Web Application);

Μια εφαρμογή του Web (Web application) είναι ένα πρόγραμμα που εκτελείται σ' έναν Web server. Η JSP (JavaServer Pages) είναι μια τεχνολογία της εταιρείας Sun Microsystems που μπορεί και διανέμει (επιστρέφει) δυναμικό περιεχόμενο (dynamic content) στους Web clients.

4.2 Τι Είναι οι JSPs (JavaServer Pages)

Οι JSPs (JavaServer Pages) είναι μια τεχνολογία που έχει δημιουργηθεί από την εταιρεία Sun Microsystems για να μπορεί να δημιουργεί δυναμικό περιεχόμενο (dynamic content) στο Web. Πρόκειται για HTML έγγραφα (ιστοσελίδες) τα οποία αναμειγνύονται με τη γλώσσα προγραμματισμού Java, η οποία και έχει τη δυνατότητα να παρέχει (δημιουργεί) αυτό το δυναμικό περιεχόμενο.

Οι JSPs είναι μια εφαρμογή στην πλευρά του server (server-side application), που σημαίνει ότι δέχονται μια αίτηση (request) και παράγουν μια απόκριση ή απάντηση (response). Σε γενικές γραμμές, οι αιτήσεις γίνονται από έναν Web client και η απόκριση είναι ένα παραγόμενο HTML έγγραφο (ιστοσελίδα) το οποίο στέλνεται πίσω στον Web client. Επειδή οι JSPs είναι μια εφαρμογή στην πλευρά του server, έχουν πρόσβαση σε πηγές (resources) στον server, όπως είναι τα Servlets, JavaBeans, EJBs, αλλά και σε βάσεις δεδομένων.

Τι σημαίνει ο όρος Web Client;

Ένας Web client είναι ένα πελάτης (client) ο οποίος αλληλεπιδρά μ' έναν Web server χρησιμοποιώντας το πρωτόκολλο HTTP. Ο πελάτης χρησιμοποιεί έναν φυλλομετρητή (browser) του Web και ζητάει έγγραφα από τον Web server δίνοντάς του ένα συγκεκριμένο URL. Υπάρχουν πολλά πλεονεκτήματα από τη χρήση των JavaServer Pages. Επειδή οι JSPs χρησιμοποιούν τη γλώσσα προγραμματισμού Java, ακολουθούν την πολιτική write-once, run-anywhere. Αυτό σημαίνει ότι μια JSP μπορεί να εκτελεσθεί σ' έναν οποιονδήποτε application server ο οποίος υποστηρίζει τις

JSPs χωρίς να χρειασθεί κάποια τροποποίηση στον κώδικα.

Οι JSPs μπορούν να γραφούν σ' έναν text editor με την επέκταση (extension) .jsp. Ένα πρόγραμμα το οποίο υποστηρίζει το γράψιμο (δημιουργία) των JSPs είναι το DreamWeaver. Ένα άλλο πλεονέκτημα των JSPs είναι η χρήση των tag libraries. Οι JSPs χρησιμοποιούν τα tags, τα οποία είναι παρόμοια μ' αυτά της HTML και της XML, για να εισάγουν δυναμικό περιεχόμενο (dynamic content). Τα tag libraries ορίζουν επιπλέον tags τα οποία μπορούν να χρησιμοποιηθούν για να αντικαταστήσουν τμήματα κώδικα.

Ένα άλλο σημαντικό πλεονέκτημα των JSPs είναι ο διαχωρισμός των ρόλων. Οι προδιαγραφές των JSPs επιτρέπουν να μοιραστεί το φορτίο σε δύο κατηγορίες : στο γραφικό περιεχόμενο της σελίδας και στο δυναμικό περιεχόμενο της σελίδας. Αυτό σημαίνει στην πράξη ότι η ομάδα που δεν γνωρίζει τη γλώσσα προγραμματισμού Java μπορεί να δημιουργήσει το γραφικό περιεχόμενο της σελίδας και ένας προγραμματιστής της Java να δημιουργήσει το δυναμικό περιεχόμενο της σελίδας.

Όταν γράφουμε ένα JSP, είναι ευκολότερο να γράψουμε πρώτα τον HTML κώδικα και μετά να εισάγουμε τον κώδικα της Java για να δημιουργήσουμε το δυναμικό περιεχόμενο.

4.3 Η Εξέλιξη του Web

Η τεχνολογία του Web έχει εξελιχθεί τρομακτικά τα τελευταία χρόνια. Πριν από λίγο μόλις καιρό το Web αποτελείτο από στατικά έγγραφα μόνο. Τώρα, τα προγράμματα μπορούν και ενσωματώνονται στην HTML και το δυναμικό περιεχόμενο μπορεί να προστεθεί στα HTML έγγραφα κατά την ώρα της εκτέλεσης (runtime).

4.3.1 Το Στατικό Web

Το Στατικό Web αποτελεί την πιο απλή δομή του Web και σύμφωνα μ' αυτό ο Web client συνδέεται στον Web server χρησιμοποιώντας το πρωτόκολλο TCP/IP και κάνει αιτήσεις (requests) χρησιμοποιώντας το πρωτόκολλο HTTP.

Ο server στέλνει μια έτοιμη σελίδα HTML στον Web client, η οποία περιέχει κείμενο, υπερσυνδέσμους (hyperlinks) και τα tags μορφοποίησης. Δεν περιέχει καθόλου δυναμικό περιεχόμενο (dynamic content) ούτε παρέχει κάποιον τρόπο για να μπορέσει ο χρήστης να αλληλεπιδράσει μαζί της (user interaction).

4.3.2 Το Δυναμικό Web

Το Δυναμικό (Dynamic) Web έφερε ακόμα περισσότερη λειτουργικότητα εισάγοντας τα προγράμματα CGI (Common Gateway Interface). Τα προγράμματα CGI βρίσκονται στον server και δέχονται αιτήσεις (requests), χρησιμοποιούν τις πηγές που βρίσκονται στην πλευρά του server (server-side resources) και δημιουργούν μια σελίδα HTML σαν απόκριση (response) στην αίτηση (request) του χρήστη.

Μπορούν να γραφούν σε πολλές γλώσσες, όπως είναι οι Java, C++, Perl, Python και Visual Basic. Παραδείγματα προγραμμάτων σε CGI είναι τα JSPs, τα Servlets και τα ASPs (Active Server Pages). Επειδή αυτά τα προγράμματα καθορίζουν (δημιουργούν) το HTML έγγραφο που στέλνεται πίσω στον client, οι χρήστες λαμβάνουν αποκρίσεις (responses) που ανταποκρίνονται στις συγκεκριμένες τους απαιτήσεις.

Ένα άλλο πλεονέκτημα της JSP είναι ότι χρησιμοποιεί την ισχυρή γλώσσα προγραμματισμού Java για να δημιουργεί το δυναμικό της περιεχόμενο. Αυτό σημαίνει ότι βρίσκονται στη διάθεσή μας εκατοντάδες τάξεις (classes) και μέθοδοι (methods).

4.4 Τα Tags και τα Attributes

Ο,τιδήποτε γράφουμε σε JSP κώδικα είναι είτε ένα απλό (μονό) tag ή ένα ζευγάρι από tags ή απλό κείμενο. Τα tags έχουν attributes (ιδιότητες ή χαρακτηριστικά) που διαμορφώνουν τη συμπεριφορά τους. Τα tags μπορεί να είναι είτε HTML tags ή JSP tags.

Ένα παράδειγμα ενός tag που μπορεί να είναι και απλό αλλά και σε ζευγάρι είναι το <jsp:useBean>. Σαν απλό tag μπορεί να χρησιμοποιηθεί για να εκκινήσει (instantiate) μια τάξη (class) της Java και να της εκχωρήσει ένα όνομα, ως εξής :

```
<jsp:useBean name="myClass" class="SimpleClass" scope="page" />
```

Σαν ζευγάρι, μπορεί να χρησιμοποιηθεί για να εκτελεσθεί κάποιος κώδικας εφόσον έχει γίνει η επιτυχής εκκίνηση της τάξης (class) της Java, ως εξής :

```
<jsp:useBean …>
        … κώδικας HTML ή/και JSP …
<jsp:useBean/>
```

Όπως τα tags της HTML, έτσι και στην JSP, τα tags μπορεί να συνοδεύονται από attributes (ιδιότητες ή χαρακτηριστικά) που διαμορφώνουν τη συμπεριφορά τους. Είδαμε προηγουμένως ότι το tag <jsp:useBean> χρησιμοποιούσε τρία attributes, τα name, class και scope, ως εξής :

```
<jsp:useBean name="myClass" class="SimpleClass" scope="page" />
```

Τα attributes αυτά δίνουν οδηγίες στο tag useBean για να εκκινήσει (instantiate) την τάξη (class) SimpleClass, να την ονομάσει myClass και να την απορρίψει όταν το JSP τελειώσει με τη δημιουργία της σελίδας.

4.5 Τα Σχόλια στην JSP

Στην JSP μπορούμε να χρησιμοποιήσουμε τα εξής τρία είδη σχολίων: HTML, JSP και Java. Τα σχόλια της HTML δημιουργούνται με τα tags <!-- και -->. Όταν εμφανισθούν αυτά τα tags σ' έναν JSP κώδικα, θα προστεθούν ανέπαφα στο response που θα δημιουργηθεί και θα παρουσιασθούν στον HTML κώδικα που θα σταλεί στον φυλλομετρητή του χρήστη.

Όμως, πρέπει να έχουμε υπόψη μας ότι αν υπάρχουν εκφράσεις της JSP μέσα σε HTML σχόλια, αυτές θα υπολογισθούν και το αποτέλεσμα θα συμπεριληφθεί στο response που θα σταλεί στον φυλλομετρητή του χρήστη.

Για παράδειγμα, αν ο JSP κώδικας περιέχει τα εξής :

```
<!-- Αυτό είναι ένα απλό σχόλιο -->
<!-- 3 + 3 = <%= 3 + 3 %> -->
```

Τότε ο HTML κώδικας που θα προκύψει και θα σταλεί στον φυλλομετρητή του χρήστη θα είναι ο εξής :

```
<!-- Αυτό είναι ένα απλό σχόλιο -->
<!-- 3 + 3 = 6 -->
```

Ο φυλλομετρητής του χρήστη θα αγνοήσει και τα δύο παραπάνω σχόλια.

Τα σχόλια της JSP δεν συμπεριλαμβάνονται στο response που στέλνεται πίσω στον φυλλομετρητή και δημιουργούνται με τα tags <%-- και --%>, ως εξής :

```
<%-- Αυτό το σχόλιο δεν θα σταλεί στον φυλλομετρητή --%>
```

Έτσι λοιπόν, τα σχόλια της JSP αγνοούνται από το JSP και δεν εμφανίζονται στον πηγαίο κώδικα της ιστοσελίδας στον φυλλομετρητή.

Τα σχόλια της Java χρησιμοποιούνται για να σχολιάσουμε τον κώδικα Java που υπάρχει ενσωματωμένος σ' ένα JSP. Ως γνωστόν, με τα // μπορούμε να έχουμε σχόλιο μίας γραμμής και με τα /* και */ σχόλιο πολλών γραμμών.

4.6 Μεταβλητές και Τύποι Δεδομένων

Οι εκφράσεις (expressions) αποτελούν έναν συνδυασμό από σταθερές, μεταβλητές και μεθόδους μαζί με τελεστές (operators). Οι εκφράσεις μπορούν να βρεθούν είτε μέσα σε JSP scriptlets ή μέσα σ' ένα tag έκφρασης (expression tag) της JSP. Αν βρίσκονται σ' ένα scriptlet, οι εκφράσεις αποτελούν συνήθως μέρος μιας εντολής εκχώρησης ή της κλήσης μιας μεθόδου. Αυτό σημαίνει ότι η έκφραση λαμβάνει τιμή (υπολογίζεται) και η τιμή αυτή εκχωρείται σε κάποια άλλη μεταβλητή ή μεταβιβάζεται σε κάποια μέθοδο.

Ενώ, όταν βρίσκονται σ' ένα tag, οι εκφράσεις υπολογίζονται, το αποτέλεσμά τους μετατρέπεται σε μια παράσταση από strings και τα strings αυτά συμπεριλαμβάνονται στον προκύπτοντα HTML κώδικα που δημιουργείται από το JSP. Η σύνταξη για να δηλώσουμε ένα tag έκφρασης της JSP είναι η εξής : `<%= έκφραση %>`

Όταν εκτελεσθεί το JSP, η έκφραση θα λάβει την τιμή της και θα γίνει τμήμα του HTML κώδικα που θα προκύψει. Αν, όμως, η μεταβλητή δεν είναι string, θα μετατραπεί αυτόματα σε string και θα συμπεριληφθεί στον HTML κώδικα που θα προκύψει. Τα scriptlets δίνουν τη δυνατότητα στους web developers να ενσωματώσουν κώδικα της Java στα JSPs που δημιουργούν και αυτό για να μπορέσουν να εκμεταλλευτούν τις μεγάλες δυνατότητες μιας ισχυρής γλώσσας προγραμματισμού και να δημιουργήσουν έτσι δυναμικό περιεχόμενο.

Τα scriptlets διαθέτουν πολλές δομές ελέγχου της ροής του προγράμματος (flow control structures), όπως είναι οι γνωστές εντολές if, switch και for. Τα directive tags χρησιμοποιούνται για να δώσουν εντολή στα JSPs να εκτελέσουν συγκεκριμένες λειτουργίες ή να έχουν κάποια συγκεκριμένη συμπεριφορά. Για παράδειγμα, το επόμενο directive tag εισάγει κάποιες τάξεις (classes) τηςJava από ένα πακέτο (packet) :

```
<%@ page import="java.sql.*" %>
```

Υπάρχουν τριών ειδών directive tags, τα page, include και taglib.

Τα tags ενέργειας (action tags) χρησιμοποιούνται για να επεκτείνουν τη λειτουργικότητα των JSPs. Υπάρχουν ενέργειες που δίνουν τη δυνατότητα στα JSPs να χρησιμοποιήσουν αντικείμενα που έχουν δημιουργηθεί στην Java, να συμπεριλάβουν (include) άλλα JSPs, να προωθήσουν (forward) requests σ' άλλα JSPs και να αλληλεπιδράσουν (interact) μ' ένα Java plugin. Υπάρχουν έξιtags ενέργειας στην JSP : useBean, setProperty, getProperty, include, forward και plugin.

4.6.1 Δήλωση Μεταβλητών στην JSP

Η δήλωση των μεταβλητών και των μεθόδων γίνεται με την τοποθέτησή τους μέσα στα tags δήλωσης της JSP, που είναι τα <%! και %>. Οι δηλώσεις αυτές ορίζουν την ύπαρξη των μεταβλητών και των μεθόδων που θα είναι διαθέσιμες στον υπόλοιπο JSP κώδικα. Οι μεταβλητές (variables) μάς δίνουν τη δυνατότητα να αναφερόμαστε στο περιεχόμενο ενός συγκεκριμένου τμήματος της μνήμης. Η δομή και το μέγεθος των δεδομένων που παριστάνει μια μεταβλητή είναι γνωστά ως τύπος δεδομένων (data type).

Στον αντικειμενοστραφή (object-oriented) προγραμματισμό, οι μεταβλητές συχνά αποκαλούνται αντικείμενα (objects) και οι τύποι δεδομένων αποκαλούνται τάξεις (classes). Ο όρος αντικείμενο χρησιμοποιείται συχνότερα όταν αναφερόμαστε σε μια μεταβλητή που έχει δηλωθεί χρησιμοποιώντας μια τάξη της Java σαν τύπο δεδομένων. Για να ορίσουμε μια μεταβλητή, θα πρέπει να δηλώσουμε τον τύπο και το όνομα της μεταβλητής μέσα στα tags <%! και %> και με την εξής γενική σύνταξη :

```
τύπος_δεδομένων όνομα_μεταβλητής [=αρχική_τιμή];
```

Όπως μπορούμε να δούμε, η απόδοση αρχικών τιμών στις μεταβλητές (αρχικοποίηση) είναι προαιρετική και θα την δούμε αργότερα. Ακολουθεί ένα παράδειγμα προγράμματος σε JSP που δηλώνει τρεις μεταβλητές.

```
<html>
<head>
<title> Παράδειγμα Δήλωσης Μεταβλητών </title>
</head>
<body>
            <%!
int myInteger;
String myString;
float myFloat;
            %>
</body>
</html>
```

4.6.2 Οι Τύποι Δεδομένων της JSP

Οι πιο κοινοί τύποι δεδομένων (data types) που θα συναντήσουμε στην JSP είναι οι εξής :

Τύπος Δεδομένων	**Περιγραφή**
int	Ακέραιοι αριθμοί
float	Αριθμοί κινητής υποδιαστολής
double	Πραγματικοί αριθμοί διπλής ακρίβειας
long	Ακέραιοι αριθμοί μεγάλης τιμής
char	Χαρακτήρας
String	Ένα Java class με πολλούς χαρακτήρες
Vector	Ένα Java utility class που είναι μια συλλογή αντικειμένων διαφορετικών τύπων
Enumeration	Ένα Java utility interface για τη σειριακή επιθεώρηση συλλογών αντικειμένων

Πίνακ. 4.1:Οι Τύποι Δεδομένων της JSP.

Όταν δηλώνουμε μια μεταβλητή, εκχωρείται κάποια μνήμη για να κρατήσει τα δεδομένα της μεταβλητής. Η μνήμη μπορεί να εκχωρηθεί εμμέσως (implicitly) ή αμέσως (explicitly), ανάλογα με τον τύπο δεδομένων που χρησιμοποιούμε. Οι βασικοί (θεμελιώδεις) τύποι δεδομένων, όπως είναι οι int, float, double, long και char, εκχωρούν μνήμη όταν δηλώνουμε τις μεταβλητές.

Οι τύποι δεδομένων που δημιουργούνται (υλοποιούνται) μέσω των Java classes πρέπει να εκχωρηθούν ρητά (explicitly) με τον τελεστή **new**. Ο τύπος δεδομένων String είναι μια τάξη (class) της

Java, αλλά χρησιμοποιείται τόσο πολύ ώστε ο μεταγλωττιστής της Java αντιμετωπίζει τα strings σαν να ήταν βασικοί (θεμελιώδεις) τύποι δεδομένων.

Ακολουθεί ένα παράδειγμα με τη χρήση του τελεστή new για να δηλώσουμε αντικείμενα (objects) στην Java.

```
<html>
<head>
<title> Παράδειγμα Δήλωσης Αντικειμένων της Java </title>
</head>
<body>
            <%!
String myString = new String();     // άμεση εκχώρηση μνήμης
String   String01;                                  // έμμεση   εκχώρηση
μνήμης
Vector myVector = new Vector();   // άμεση εκχώρηση μνήμης
           %>
</body>
</html>
```

4.7 Δήλωση Πινάκων στην JSP

Για να δηλώσουμε έναν πίνακα, γράφουμε πρώτα τον τύπο δεδομένων των τιμών που θα περιέχει ο πίνακας, μετά τα σύμβολα [ ], το όνομα της μεταβλητής πίνακα και τέλος είτε τον τελεστήnew για να δημιουργήσουμε έναν κενό πίνακα ή τα σύμβολα { και } με τις τιμές του καινούργιου πίνακα ανάμεσά τους. Ακολουθεί ένα χαρακτηριστικό παράδειγμα.

```
<html>
<head>
<title> Δήλωση Πινάκων στην JSP </title>
</head>
<body>
            <%!
int[ ] population = {15, 14, 50, 20};
String cities = {"Florina", "Grevena", "Kozani", "Kastoria"};
           %>
            <% for(int i=0; i<4; i++){ %> <hr>
                        <br> Πόλη : <%= cities[i] %>
                        <br> Πληθυσμός : <%= population[i] %>
            <% } %>
</body>
```

```
</html>
```

Οι πίνακες στην JSP έχουν δείκτες που ξεκινούν με την τιμή 0. Στο παραπάνω παράδειγμα, ο πίνακας population περιέχει 4 ακέραιες τιμές, τις 15, 14, 50 και 20, ενώ ο πίνακας cities περιέχει 4strings (ονόματα πόλεων). Κατά την εκτέλεση του κώδικα, χρησιμοποιούμε έναν βρόχο for για να έχουμε πρόσβαση στα στοιχεία των πινάκων.

4.8 Δήλωση Μεθόδων στην JSP

Η δήλωση μιας μεθόδου στην JSP γίνεται μέσα στα tags δήλωσης <%! και %> και περιλαμβάνει τον τύπο δεδομένων της τιμής επιστροφής της μεθόδου, το όνομα της μεθόδου, τη λίστα παραμέτρων και τις εντολές (σώμα) της μεθόδου. Η δήλωση μιας μεθόδου έχει την εξής σύνταξη :΄

```
τύπος_επιστροφής όνομα_μεθόδου (λίστα_παραμέτρων) {σώμα_μεθόδου}
```

Το όνομα_μεθόδου είναι ένα αναγνωριστικό (identifier) που χρησιμοποιείται για να αναφερόμαστε στη μέθοδο με το όνομά της. Η λίστα λίστα_παραμέτρων αποτελείται από μια λίστα μεταβλητών και των τύπων δεδομένων τους που είναι χωρισμένες με κόμματα και μέσα σε παρενθέσεις. Το σώμα_μεθόδου αποτελείται από εντολές της Java και μπορεί προαιρετικά να επιστρέφει μια τιμή σ' αυτόν που κάλεσε τη μέθοδο.

Ο τύπος επιστροφής, το όνομα της μεθόδου και οι παράμετροι αποτελούν την υπογραφή (signature) της μεθόδου. Ακολουθεί ένα απλό παράδειγμα δήλωσης μιας μεθόδου η οποία ελέγχει αν ένα login password έχει ένα ελάχιστο μήκος.

```
<%! boolean verifyPasswordLength(String password) {
                        if(password.length()   <   MIN_PASSWORD_LEN)
return false;
                    return true;
          }
%>
```

4.9 Δημιουργία Δικών μας Τύπων Δεδομένων

Μπορούμε να δημιουργήσουμε δικούς μας τύπους δεδομένων, δημιουργώντας μια κατάλληλη τάξη (class). Για παράδειγμα, η επόμενη τάξη δηλώνει έναν καινούργιο τύπο δεδομένων που αποκαλείται Friend :

```
public class Friend {
          protected String lastName, firstName;
}
```

Μπορούμε τώρα να χρησιμοποιήσουμε τις μεθόδους get και set σ' έναν JSP κώδικα για να έχουμε πρόσβαση στα δεδομένα, ως εξής :

```
<html>
<head>
<title>Μια Σελίδα JSP με την τάξη Friend </title>
</head>
<body>
<%!     Friend mary = new Friend("Papadopoulou", "Mary"); %>
```

```
<p> Επώνυμο Φίλου : <%= mary.getLastName() %>
<p> Όνομα Φίλου : <%= mary.getFirstName() %>
</body>
</html>
```

4.10 Αναφορά σε Μεταβλητές Μέσα από Εκφράσεις

Ο τελεστής + συνενώνει strings σ' ένα μόνο string και προσθέτει αριθμούς, ενώ όταν έχουμε συνδυασμό από strings και αριθμούς, πρώτα μετατρέπονται οι αριθμοί σε strings και μετά ενώνονται σ' ένα ενιαίο string.

4.11 Κλήση Μεθόδων Μέσα από Εκφράσεις

Οι εκφράσεις της JSP μπορεί να περιέχουν μεταβλητές αλλά και κλήσεις σε μεθόδους. Οι μέθοδοι μπορεί να δηλωθούν είτε τοπικά μέσα στον JSP κώδικα ή σ' ένα Java class. Τοπικά οι μέθοδοι δηλώνονται με τα tags δήλωσης της JSP, τα γνωστά <%! και %>.

Οι μέθοδοι επιστρέφουν τιμές διαφορετικών τύπων δεδομένων και μετατρέπονται σε strings από τα tags έκφρασης της JSP έτσι ώστε να μπορούν να ενσωματωθούν στον HTML κώδικα που θα προκύψει.

4.12 Τα Scriptlets

Τα scriptlets αποτελούνται από ενσωματωμένο κώδικα της Java ανάμεσα στα tags <% και %>. Ο κώδικας της Java αποτελείται από ένα σύνολο δηλώσεων, εκφράσεων και εντολών που ελέγχουν προγραμματιστικά το περιεχόμενο του JSP.

Τα scriptlets ορίζουν την κύρια πορεία της εκτέλεσης που ελέγχει τη δυναμική συμπεριφορά μιας ιστοσελίδας. Όταν φορτώνεται ένα JSP, η εκτέλεση ξεκινά με την πρώτη εμφάνιση ενόςscriptlet, το οποίο μπορεί μετά να χρησιμοποιήσει τις δηλωμένες μεταβλητές και μεθόδους.

Η Java παρέχει τις εξής δύο δομές ελέγχου της ροής ενός προγράμματος :

Εντολές επιλογής : **if** και **switch.**

Εντολές επανάληψης : **for**, **while** και **do while**.

4.13 Οι Αιτήσεις του Χρήστη (User Requests)

Θα δούμε τώρα πώς μπορούμε να συγκεντρώσουμε και να επεξεργαστούμε τις αιτήσεις του χρήστη (user requests). Οι αιτήσεις αυτές αποτελούνται από μηνύματα που περιέχουν πληροφορίες που αφορούν τους χρήστες και την αλληλεπίδρασή τους με τον φυλλομετρητή. Οι αιτήσεις δημιουργούνται από τον φυλλομετρητή και στέλνονται σ' έναν JSP κώδικα για επεξεργασία σαν αποτέλεσμα κάποιων ενεργειών που μπορούν να κάνουν οι χρήστες καθώς αλληλεπιδρούν με τον φυλλομετρητή, όπως είναι το κλικ σ' έναν υπερσύνδεσμο ή η υποβολή μιας φόρμας.

Οι πληροφορίες ενός request αφορούν τον χρήστη, το απομακρυσμένο μηχάνημα (server), το τοπικό μηχάνημα, τα HTTP headers, το URL με το οποίο έγινε η κλήση, μερικές παραμέτρους της ιστοσελίδας και τα πεδία φόρμας (form fields) της HTML. Όλα αυτά τα στοιχεία είναι προσβάσιμα από ένα JSP μέσω του αντικειμένου request όταν καλείται ο κώδικας JSP. Το JSP χρησιμοποιεί το αντικείμενο request για να αποκτήσει πληροφορίες από τον φυλλομετρητή, τα cookies, τις επικεφαλίδες (headers) και τη σύνοδο του χρήστη (user session).

4.13.1 Δημιουργία και Επεξεργασία των User Requests

Μια αίτηση (request) περιέχει πληροφορίες οι οποίες μεταβιβάζονται (περνιούνται) από έναν χρήστη (user) σ' ένα JSP. Ένα μέρος από τη διαδικασία δημιουργίας ενός request είναι να δηλωθεί ή δημιουργηθεί ένα URL και θα δούμε πρώτα τα διάφορα μέρη ενός URL και τη σύνταξή του.

Ένα URL έχει την εξής γενική σύνταξη :

```
πρωτόκολλο://host:port/διαδρομή?queryString
```

πρωτόκολλο – δηλώνει τους υποκείμενους μηχανισμούς για τη μεταφορά πληροφοριών ανάμεσα σε απομακρυσμένα μηχανήματα. Έγκυρα πρωτόκολλα είναι τα http, https, ftp, rmi και corba. Εδώ θα χρησιμοποιήσουμε το πρωτόκολλο http.

host – το όνομα του απομακρυσμένου μηχανήματος ή η IP διεύθυνση του μηχανήματος προς το οποίο θα προωθηθεί η αίτηση (request).

port – δηλώνει τον αριθμό θύρας του μηχανήματος στην οποία το μηχάνημα του host server θα περιμένει να ακούσει για αιτήσεις (requests). Για το πρωτόκολλο http η θύρα (port) είναι συνήθως η 80 και αποτελεί και την προκαθορισμένη (default) τιμή.

διαδρομή – είναι μια σειρά από καταλόγους (φακέλους) χωρισμένους με το σύμβολο / για να μπορέσουμε να βρούμε το JSP αρχείο που θα εκτελεσθεί (κληθεί).

queryString – μια λίστα από ζευγάρια ονόματος/τιμής που μεταβιβάζονται στο JSP σαν όρισμα (argument).

Ακολουθεί ένα παράδειγμα ενός URL το οποίο καλεί ένα JSP μ' ένα query string :

```
http://www.florina.gr:80/myJSP/Lesson1/request.jsp?p1=val1&p2=val2
```

Η διαδρομή (virtual path) του παραπάνω URL είναι ό,τι βρίσκεται ανάμεσα στη δήλωση της θύρας (port) και πριν από το query string, δηλαδή το: `myJSP/Lesson1/request.jsp.`

Το query string είναι ό,τι βρίσκεται μετά από το σύμβολο ?, δηλαδή το: `p1=val1&p2=val2.` Το query string περιέχει μια λίστα από ζευγάρια με ονόματα και τιμές, τα οποία διαχωρίζονται με τον χαρακτήρα &. Η χρήση του query string θυμίζει το πέρασμα παραμέτρων στις κλήσεις των συναρτήσεων.

Ένας φυλλομετρητής δημιουργεί requests όταν ο χρήστης κάνει κλικ σ' έναν υπερσύνδεσμο (hyperlink) ή υποβάλει μια HTML φόρμα. Ο παρακάτω HTML κώδικας δημιουργεί έναν υπερσύνδεσμο (hyperlink).

```
<html>
<head>
<title> Request JSP </title>
</head>
<body>
            Κάντε κλικ
<a href="request.jsp?p1=v1&p2=v2">εδώ</a>
για να στείλετε ένα request σ' έναν JSP κώδικα
</body>
</html>
```

Όταν ο χρήστης κάνει κλικ στον υπερσύνδεσμο, ο φυλλομετρητής δημιουργεί ένα request το οποίο στέλνεται στην ιστοσελίδα request.jsp. Ο JSP κώδικας μπορεί να έχει πρόσβαση στα στοιχεία (πληροφορίες) του request μέσω του αντικειμένου request. Ο JSP κώδικας μπορεί να χρησιμοποιήσει το αντικείμενο request για να πάρει πληροφορίες για το URL που κάνει την κλήση, για το μηχάνημα του server και τις θύρες (ports).

Τα requests που δημιουργούνται από υπερσυνδέσμους (hyperlinks) είναι στατικά requests, δηλ. γράφονται όπως ακριβώς είναι κατά τη φάση της σχεδίασης της ιστοσελίδας σ' ένα στατικό URLκαι περιέχουν πάντα τις ίδιες τιμές στο query string ανεξάρτητα από τον χρήστη που έκανε κλικ στον υπερσύνδεσμο. Αντίθετα, τα requests που δημιουργούνται από τις φόρμες της HTML είναι δυναμικά requests με την έννοια ότι ο χρήστης έχει τη δυνατότητα να συμπληρώσει κάποια στοιχεία στα πεδία της φόρμας.

Έτσι, όταν υποβάλλεται η φόρμα, δημιουργείται το URL με τις τιμές των πεδίων που έχει δώσει ο χρήστης, δηλ. το URL δημιουργείται δυναμικά κατά τη φάση της εκτέλεσης της εφαρμογής (runtime) όταν υποβάλλεται η φόρμα, και φυσικά θα περιέχει διαφορετικές τιμές κάθε φορά που γίνεται η

υποβολή της φόρμας.

4.14 Οι Παράμετροι του Query String

Τα requests του χρήστη μπορούν να περιέχουν μια λίστα από ζευγάρια ονόματος/τιμής τα οποία μεταβιβάζονται ως ορίσματα (παράμετροι, arguments) στον JSP κώδικα που καλείται. Το καθένα από τα ορίσματα αυτά στη λίστα των ορισμάτων αναφέρεται ως παράμετρος (parameter).

Οι παράμετροι μεταβιβάζονται σαν μέρος του URL το οποίο καλεί το JSP. Αυτό γίνεται δηλώνοντας ένα query string και προσθέτοντας το σύμβολο ? και μετά τη λίστα των παραμέτρων με διαχωριστικό το σύμβολο &. Η γενική σύνταξη ενός URL για να κληθεί ένα JSP με μια λίστα παραμέτρων είναι η εξής : *URL?queryString* , όπου το queryString έχει την εξής σύνταξη :

```
name1=value1[,value2, … valueN1][&name2=value1, [,value2,…valueN2]]
```

Παρατηρούμε ότι μια παράμετρος μπορεί να έχει και περισσότερες από μία τιμές. Επίσης, το κάθε ζευγάρι ονόματος/τιμής ξεχωρίζει από το σύμβολο & και ακόμη το σύμβολο ? δηλώνει την αρχή της λίστας των παραμέτρων.

Ακολουθεί ένα παράδειγμα ενός JSP κώδικα το οποίο μεταβιβάζεται με δύο παραμέτρους, τις lastName και firstName, με τις αντίστοιχες τιμές Papas και John.

```
http://serverflo:7001/myJSP/name.jsp?lastName=Papas&firstName=John
```

Οι παράμετροι με πολλές τιμές δηλώνονται παρέχοντας πολλές τιμές χωρισμένες με κόμματα για μια δεδομένη παράμετρο. Το επόμενο παράδειγμα εκχωρεί δύο τιμές στην παράμετρο colors .

```
http://localhost:7001/myJSP/myFlag.jsp?colors=blue&colors=white&count
ry=Greece
```

Η λίστα των παραμέτρων στέλνεται στο JSP μέσω του αντικειμένου HttpServletRequest και γίνεται διαθέσιμη σ' ένα JSP μέσω του στάνταρτ α-ντικειμένου request του JSP. Το αντικείμενοrequest παρέχει τρεις μεθόδους για να έχουμε πρόσβαση στις παραμέτρους που του μεταβιβάζονται, οι οποί-ες είναι οι εξής :

String getParameter(String paramName), επιστρέφει την τιμή μιας παραμέτρου, όπου το paramName είναι το όνομα της παραμέτρου.

Enumeration getParameterNames(), επιστρέφει όλα τα ονόματα των παραμέτρων σαν μια αρίθμηση (enumeration) από αντικείμενα String. Μπορούμε μετά να πλοηγηθούμε στην αρίθμηση εξάγοντας το κάθε αντικείμενο String, το οποίο μπορεί μετά να χρησιμοποιηθεί σαν όρισμα στη μέθοδο getParameter(String) για να πάρουμε την τιμή της παραμέτρου.

String() getParameterValues(String paramName), επιστρέφει όλες τις τιμές μιας παραμέτρου με πολλές τιμές και τις καταχωρεί σ' έναν πίνακα.

4.15 Οι Φόρμες της HTML

Τα requests που δημιουργούνται από υπερσυνδέσμους (hyperlinks) έχουν έτοιμο (συγκεκριμένο) query string χρησιμοποιώντας το tag <a>. Οι φόρμες της HTML μάς δίνουν τη δυνατότητα να δημιουργήσουμε ένα URL μ' ένα query string που δημιουργείται δυναμικά όταν ο χρήστης υποβάλει τη φόρμα. Τα ονόματα και οι τιμές του query string εξάγονται από τη φόρμα και δημιουργείται και προστίθεται στο URL η λίστα των ζευγαριών ονόματος/ τιμής.

Οι φόρμες της HTML παρέχουν πολλά είδη συστατικών GUI (graphical user interface) που λειτουργούν ως πεδία (fields) που μπορούμε να χρησιμοποιήσουμε για να δημιουργήσουμε φόρμες ώστε ο χρήστης να καταχωρίσει τιμές και να τις υποβάλει σε μια ιστοσελίδα με JSP κώδικα για περαιτέρω επεξεργασία.

Οι φόρμες δηλώνονται με το tag <form> ως εξής :

```
<form action="action" method=method>

                    … στοιχεία GUI για φόρμες …

</form>
```

Η παράμετρος action του tag <form> δηλώνει την ενέργεια που θα λάβει χώρα όταν θα υποβληθεί η φόρμα. Για τα JSPs, χρησιμοποιούμε το URL του JSP που θα καλέσουμε. Ένα query stringπροστίθεται στο URL. Τα ονόματα και οι τιμές των ζευγαριών ονόματος/τιμής στο query string δημιουργούνται δυναμικά από τα ονόματα των στοιχείων GUI και τις καταχωρίσεις του χρήστη στη φόρμα.

Η παράμετρος method του tag <form> δηλώνει την HTTP μέθοδο που θα χρησιμοποιηθεί για να σταλεί (post) το request στο JSP. Η τιμή της παραμέτρου αυτής μπορεί να είναι μια από τις GET,POST ή PUT. Για τα JSPs, χρησιμοποιούμε τη μέθοδο POST.

Μέσα στα tags της φόρμας, μπορούμε να δηλώσουμε όσα στοιχεία φόρμας θέλουμε χρησιμοποιώντας τα HTML tags <input>, <textarea> και <select>. Η απλή σύνταξη του tag <input> είναι η εξής:

```
<input type=type name=name value=defaultValue>
```

Η παράμετρος type μάς δίνει τη δυνατότητα να επιλέξουμε ένα από πολλά είδη στοιχείων, όπως text, checkbox, radio, image, reset και submit.

Η σύνταξη του tag <textarea> είναι η εξής :

```
<textarea name=name rows=rows cols=columns value=defaultText>
```

Το tag <textarea> δηλώνει μια περιοχή κειμένου όπου ο χρήστης μπορεί να καταχωρίσει κείμενο σε πολλές γραμμές. Η τιμή αυτού του tag είναι το κείμενο που έχει καταχωρίσει ο χρήστης.

Το tag <select> δηλώνει μια λίστα επιλογών απ' όπου ο χρήστης μπορεί να επιλέξει είτε μία ή πολλές. Η τιμή αυτού του tag είναι η επιλογή που έχει κάνει ο χρήστης. Η σύνταξη του tag <select> είναι η εξής :

```
<select name=name [size=size multiple]>
<option [selected]> επιλογή </option>
…
<option [selected]> επιλογή </option>
</select>
```

Βλέπουμε ότι όλα τα στοιχεία της φόρμας έχουν μια παράμετρο που καλείται name. Επίσης, ότι όλα τα στοιχεία της φόρμας έχουν μια τιμή που συσχετίζεται με το name και που ορίζεται είτε με καταχώριση από τον χρήστη ή από τον προγραμματιστή με μια προκαθορισμένη (default) τιμή που ορίζεται στην παράμετρο value. Τα ονόματα και οι τιμές των πεδίων της φόρμας χρησιμοποιούνται για να δημιουργηθεί το query string.

Όλες οι φόρμες θα πρέπει να διαθέτουν ένα πλήκτρο Submit (υποβολής) που να ενημερώνει τον φυλλομετρητή ότι η φόρμα έχει συμπληρωθεί και ότι το JSP θα πρέπει να κληθεί με το URL που καθορίζεται στην παράμετρο action του tag <form>.

4.16 Αποθήκευση και Ανάκτηση των Cookies

Τα cookies είναι μικρά αρχεία κειμένου τα οποία αποθηκεύονται στον υπολογιστή μας κατά την πλοήγησή μας στο διαδίκτυο. Συνήθως περιγράφουν στοιχεία μας όπως όνομα χρήστη (user name) και συνθηματικό πρόσβασης (password) με σκοπό κατά την επίσκεψή μας στον ίδιο ιστότοπο αργότερα, να μας "θυμάται" και να κάνει login χωρίς να γράψουμε εμείς τίποτα. Τα cookies μπορεί να προέρχονται από τον ιστότοπο τον οποίο έχουμε επισκεφθεί ή από κάποιον άλλον. Συνήθως είναι άκακα, έχει όμως αποδειχθεί ότι μπορούν να στείλουν πληροφορίες για τη συμπεριφορά μας στο διαδίκτυο. Υπάρχουν προγράμματα που καθαρίζουν τα κακόβουλα cookies, ενώ αν ο χρήστης επιθυμεί να τα διαγράψει δίνεται αυτή η δυνατότητα μέσα από το φυλλομετρητή ιστοσελίδων.

Τα cookies δημιουργούνται από την εφαρμογή στην πλευρά του server (server-side application), δηλ. το JSP, και αποθηκεύονται στο μηχάνημα του client εξ ονόματος του JSP. Τα JSP εμπορούν να προσθέσουν cookies στο μηχάνημα του client χρησιμοποιώντας τη μέθοδο (συνάρτηση) addCookie() του αντικειμένου response. Η μέθοδος addCookie() λαμβάνει ένα

στιγμιότυπο (instance) της τάξης (class) Cookie, η οποία μπορεί να περιέχει πληροφορίες σχετικά με τον χρήστη.

Το μηχάνημα του client αποθηκεύει τα cookies στον τοπικό σκληρό δίσκο σ' έναν φάκελο που ορίζεται από τον φυλλομετρητή που εκτελείται στο μηχάνημα του client. Τα cookies ανακτώνται από την JSP από το μηχάνημα του client όταν ο πελάτης (client) στείλει μια αίτηση (request) στο JSP. Τα JSPs έχουν πρόσβαση στα cookies χρησιμοποιώντας τη μέθοδο getCookies() του αντικειμένου request. Η μέθοδος getCookies() επιστρέφει έναν πίνακα (array) από αντικείμενα Cookie. Τα cookies παριστάνονται από την τάξη Cookie σαν αντικείμενα τάξης (class objects) στην πλευρά του server.

4.17 Δημιουργία Δυναμικής Απάντησης στον Χρήστη

Ένα από τα μεγάλα πλεονεκτήματα των JSPs είναι η ικανότητά τους να δημιουργούν δυναμικό περιεχόμενο (dynamic content). Τα JSPs δημιουργούν δυναμικές ιστοσελίδες χρησιμοποιώντας τις δομές ελέγχου της Java, όπως είναι οι βρόχοι for και οι εντολές if. Για παράδειγμα, ένα JSP μπορεί να εμφανίσει ένα συγκεκριμένο μήνυμα κειμένου πολλές φορές χρησιμοποιώντας έναν βρόχο for.

Παρόμοια, τα JSPs μπορούν να δημιουργήσουν πίνακες με δυναμικό αριθμό γραμμών και στηλών επαναλαμβάνοντας τα κατάλληλα tags μέσα σ' έναν βρόχο. Και οι φόρμες μπορούν να δημιουργηθούν δυναμικά ακολουθώντας κάποια συγκεκριμένη λογική διάταξη (layout). Η δημιουργία δυναμικού περιεχομένου (dynamic content) στις εφαρμογές του Web είναι σημαντική όταν το περιεχόμενο πρέπει να εκφράζει (παριστάνει) τα πιο καινούργια δεδομένα και προσωπικές πληροφορίες. Ένα παράδειγμα μιας τέτοιας εφαρμογής είναι ένα online χαρτοφυλάκιο για το χρηματιστήριο.

Ένα χαρτοφυλάκιο για το χρηματιστήριο δίνει τη δυνατότητα στους χρήστες να παρακολουθούν τις πληροφορίες online, όπως είναι οι τρέχουσες τιμές των μετοχών και οι μέγιστες και ελάχιστες καθημερινές τιμές. Μια τυπική ιστοσελίδα μιας τέτοιας εφαρμογής θα πρέπει να περιέχει έναν πίνακα που να εμφανίζει μια ποικιλία πληροφοριών του χρηματιστηρίου σε ξεχωριστές γραμμές, οι οποίες θα δημιουργούνται από έναν JSP κώδικα.

Η κάθε μετοχή θα έχει μια δική της γραμμή και κάθε φορά που ο χρήστης θα επισκέπτεται τη σελίδα, ο πίνακας θα διαφέρει, επειδή η JSP έχει πρόσβαση στις πιο πρόσφατες πληροφορίες και δημιουργεί τον πίνακα στον αέρα (on-the-fly). Η σελίδα θα μπορεί επίσης να περιέχει συνδέσμους (links) και πλήκτρα εντολών (buttons) για να μπορούμε να κάνουμε τροποποιήσεις στον πίνακα, όπως προσθήκη ή αφαίρεση μιας γραμμής ή ταξινόμηση του πίνακα σύμφωνα με κάποια στήλη.

Άλλα παραδείγματα εφαρμογών που χρησιμοποιούν δυναμικό περιεχόμενο είναι σελίδες ειδήσεων, πληροφορίες για τον καιρό και εξατομικευμένοι (προσωπικοί) δικτυακοί τόποι.

4.18 Για ποιους λόγους προτιμήσαμε την JSP από την PHP

Η PHP είναι μια open-source γλώσσα συγγραφής σεναρίων (scripting language) που ενσωματώνεται μέσα στον κώδικα της HTML και εκτελείται στην πλευρά του server (server-side scripting)και είναι πολύ παρόμοια με τη JSP και την ASP. Ορίζει τη δική της scripting γλώσσα, η οποία μοιάζει πολύ με την Perl. Σε αντίθεση η JSP χρησιμοποιεί τη Java ως scripting γλώσσα.

Η PHP είναι πολύ δημοφιλής, χρησιμοποιείται σε πάνω από ένα εκατομμύριο ιστοσελίδες, αλλά το κύριο πλεονέκτημά της φαίνεται να είναι το ότι είναι περισσότερο "scripty" και το ότι μοιάζει πολύ με την Perl, πράγμα που την κάνει λιγότερο εκφοβιστική για τη μεγάλη μάζα των HTML hacker. Σε μακροπρόθεσμη βάση, η JSP και η Java θα παρέχουν ένα πιο ισχυρό σύστημα.

Εδώ είναι μια λίστα των λόγων για τους οποίους η JSP είναι καλύτερη από την PHP:

Οτιδήποτε μπορείτε να κάνετε με την PHP, μπορείτε να κάνετε και με την JSP. Το αντίθετο δεν ισχύει.

Η JSP είναι πολύ πιο ισχυρή, καθώς έχει πρόσβαση σε όλες τις βιβλιοθήκες της Java. Η PHP έχει πρόσβαση μόνο σε PHP βιβλιοθήκες.

Η JSP είναι αντικειμενοστραφής, έτσι οδηγεί σε καθαρότερο κώδικα που είναι πιο εύκολο να αποσφαλματωθεί, να διατηρηθεί και να βελτιωθεί. (Η PHP επιτρέπει επίσης αντικείμενα, αλλά το μοντέλο αντικειμένου είναι πιο πρωτόγονο, και οι περισσότερες scripted σελίδες αγνοούν τα αντικείμενα PHP και χρησιμοποιήσουν τις κανονικές μεταβλητές.)

Η ισοδύναμη σύνταξη σε JSP είναι τόσο απλή στην εκμάθηση, ώστε να μπορεί κάποιος να την

τρέξει εξίσου γρήγορα, δηλαδή, δεν υπάρχει επιπλέον κόστος εκκίνησης κατά τη χρήση της Java, τουλάχιστον όχι ενός σημαντικού.

Java προγραμματιστές (σε αντίθεση με τους δεκαπεντάχρονους hackers ή HTML monkeys) εκτιμούν τη σημασία της καθαρής γλώσσας με πολύπλοκες δομές δεδομένων και ισχυρή πληκτρολόγηση.

Με την JSP, εάν ο κώδικας μέσα σε μια σελίδα είναι πάρα πολύ μεγάλος, ή αν θέλουμε να τον χρησιμοποιήσουμε κάπου αλλού, μπορούμε να τον αποκόψουμε, να είναι σε μια Java class, και να τον επικαλούμαστε από οπουδήποτε στην εφαρμογή μας ενώ με την PHP, κολλάμε μέσα στο πλαίσιο HTML.

Η αντίληψη της JSP για τη στατική διαχείριση και την επιμονή είναι πιο σαφής και πιο ισχυρή από ό, τι της PHP. Με τη JSP, μπορούμε να καθορίσουμε αν μια μεταβλητή επιμένει ή αν είναι μόνο τοπική. Ο κινητήρας JSP κάνει αυτόματα το σωστό με τα cookies, έτσι ώστε να έχουμε πρόσβαση στην μεταβλητή για αργότερα αιτήματα. Με την PHP, το μόνο που έχουμε είναι "global" και "not global" μεταβλητές και πρέπει να δουλέψουμε με τα cookies ή τις κρυφές μεταβλητές χειροκίνητα.

ΚΕΦΑΛΑΙΟ 5

Βιβλιογραφία-Πηγές

5.1 Βιβλιογραφία

1. Ryan Stephens "Μάθετε την SQL σε 24 ώρες" Εκδόσεις Μ.Γκιούρδας.
2. Σταυρακούδης Αθανάσιος "Βάσεις Δεδομένων και SQL. Μια πρακτική προσέγγιση" Εκδόσεις Κλειδάριθμος.
3. Ramakrishnan Gehrke "Συστήματα διαχείρισης Βάσεων Δεδομένων" Εκδόσεις Τζιόλα.
4. Larry Ullman "Εισαγωγή στην PHP για τον παγκόσμιο ιστό" Εκδόσεις Κλειδάριθμος.

5.2 Πηγές

1. http://portal.acm.org/citation.cfm?id=1645821
2. http://en.wikipedia.org
3. http://dide.flo.sch.gr/Plinet/plinet.html#a12
4. http://en.wikipedia.org/wiki/SQL
5. http://www.sqlcourse.com/intro.html
6. http://www.w3schools.com/sql/default.asp
7. http://www.1keydata.com/
8. http://www.sql.org/
9. http://www.php.net/
10. http://www.techteam.gr/wiki/PHP
11. http://www.phpfreaks.com
12. http://www.w3schools.com/php/default.asp
13. http://el.wikipedia.org/wiki/PHP
14. http://translate.google.gr/translate?hl=el&sl=el&tl=en&u=http://www.jguru.com/faq/view.jsp%3FEID%3D10596
15. http://www.jsptut.com/
16. http://java.sun.com/products/jsp/