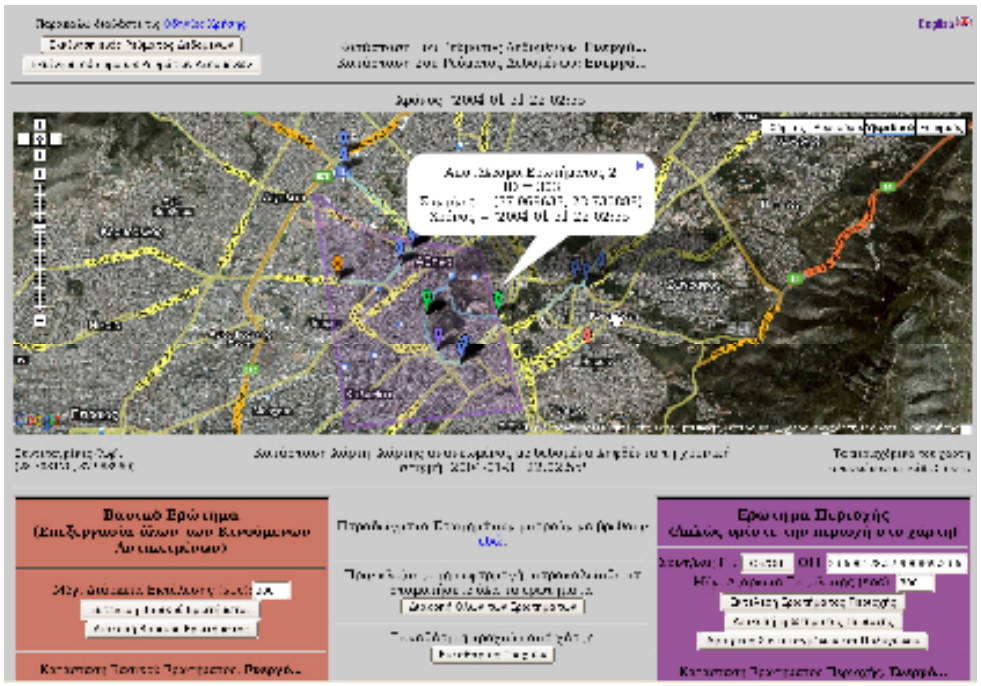
ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
Σχολή Αγρονόμων & Τοπογράφων Μηχανικών
ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΓΕΩΠΛΗΡΟΦΟΡΙΚΗ»

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ ΜΕ ΘΕΜΑ:

ΔΙΑΧΕΙΡΙΣΗ ΡΕΥΜΑΤΩΝ ΚΙΝΗΣΗΣ ΑΝΤΙΚΕΙΜΕΝΩΝ ΜΕ ΑΠΕΙΚΟΝΙΣΗ ΣΕ GoogleMaps



Συντάκτης: **Κωνσταντίνος Θεοφιλογιαννάκος**
Μηχανικός Μεταλλείων – Μεταλλουργός Ε.Μ.Π.

Επιβλέπων: **Τιμολέων Σελλής**
Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2009

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
Σχολή Αγρονόμων & Τοπογράφων Μηχανικών
ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΓΕΩΠΛΗΡΟΦΟΡΙΚΗ»

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ ΜΕ ΘΕΜΑ:

ΔΙΑΧΕΙΡΙΣΗ ΡΕΥΜΑΤΩΝ ΚΙΝΗΣΗΣ ΑΝΤΙΚΕΙΜΕΝΩΝ ΜΕ ΑΠΕΙΚΟΝΙΣΗ ΣΕ GoogleMaps

Συντάκτης: Κωνσταντίνος Θεοφιλογιαννάκος
Μηχανικός Μεταλλείων – Μεταλλουργός Ε.Μ.Π.

Επιβλέπων: **Τιμολέων Σελλής**
Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2009

Στην οικογένειά μου.

Επικοινωνία: k.theofilogiannakos@gmail.com

Πρόλογος

Η παρούσα εργασία ξεκίνησε δειλά δειλά τα Χριστούγεννα του 2007. Λόγω αυξημένων υποχρεώσεων στο χώρο εργασίας μου, πέρασαν μερικοί μήνες χωρίς να προλαβαίνω να ασχοληθώ σχεδόν καθόλου με αυτήν. Η βιβλιογραφική επισκόπηση με απασχόλησε περίπου μέχρι τον Οκτώβρη του 2008. Από το τέλος Οκτωβρίου και λόγω πίεσης χρόνου σταμάτησα την εργασία μου ως Μηχανικός Μεταλλείων μέχρι τον Φεβρουάριο του 2009.

Διάλεξα αυτό το θέμα για διαφόρους λόγους.

- Σημαντικότατος λόγος ήταν το πρωτότυπο και άκρως ενδιαφέρον κατ' εμέ θέμα της, "Διαχείριση ρευμάτων κίνησης αντικειμένων με απεικόνιση σε GoogleMaps" και μάλιστα σε πραγματικό χρόνο από οπουδήποτε στον κόσμο απλά με χρήση οποιουδήποτε υπολογιστή είναι συνδεδεμένος στο διαδίκτυο.
- Για την υλοποίηση της εφαρμογής διαφαινόταν ότι θα έπρεπε να μάθω τουλάχιστον μία γλώσσα προγραμματισμού και τη γνωστή HTML.

Πριν ασχοληθώ με την παρούσα εργασία δεν ήξερα ουσιαστικά τίποτα σχετικά με γλώσσες προγραμματισμού. Είχα ασχοληθεί μόνο με την παλαιά γλώσσα προσομοίωσης GPSS στην πτυχιακή εργασία μου στη Σχολή των Μηχανικών Μεταλλείων – Μεταλλουργών του ΕΜΠ. Ο κυριότερος λόγος λοιπόν που διάλεξα αυτήν την εργασία ήταν η επιθυμία μου να μάθω γλώσσες προγραμματισμού αλλά και γλώσσες δημιουργίας ιστοσελίδων κάτι που πέτυχα με το παραπάνω.

Για την υλοποίηση της εργασίας χρειάστηκε να μάθω να χρησιμοποιώ το λειτουργικό Linux και το σύστημα διαχείρισης ρευμάτων δεδομένων TelegraphCQ τα οποία μου ήταν παντελώς άγνωστα. Στην πορεία διαπιστώθηκε ότι για την εμφάνιση χαρτών της Google στην ιστοσελίδα θα έπρεπε να μάθω την γλώσσα προγραμματισμού διαδικτυακών εφαρμογών JavaScript, την οποία γνώριζα μόνο σαν όνομα επειδή εμφανίζεται μερικές φορές στη γραμμή κατάστασης των ιστοσελίδων στον browser. Επίσης, για την διασύνδεση της ιστοσελίδας με το TelegraphCQ έπρεπε να χρησιμοποιηθεί η PHP της οποίας την ύπαρξη δεν γνώριζα πριν την ενασχόληση με την παρούσα εργασία. Σχετικά με τη γλώσσα XML είχαμε διδαχθεί αρκετά πράγματα στο μάθημα των χωρικών βάσεων δεδομένων, αλλά σε κάθε περίπτωση έπρεπε να γίνει τουλάχιστον μία καλή επανάληψη. Με τη γλώσσα HTML, αν και τελικά είναι σχετικά απλή, δεν είχα ασχοληθεί ποτέ στο παρελθόν, εκτός από μια σύντομη αναφορά που είχε γίνει στο παραπάνω μάθημα. Με την PostgreSQL αυτή καθεαυτή δεν είχα ασχοληθεί, αλλά στο ίδιο μάθημα που διεξήγαγε ο καθηγητής κ. Τίμος Σελλής, είχαμε μάθει όλα τα απαραίτητα στοιχεία για τη διαχείριση Χωρικών Βάσεων Δεδομένων και τη χρήση των συστημάτων που τα υποστηρίζουν.

Οι χάρτες Google και η χρήση αυτών αποτέλεσε κάτι πολύ ενδιαφέρον και ελκυστικό για ενασχόληση παρόλο που δεν είχα ασχοληθεί ποτέ στο παρελθόν με αυτούς παρά μόνο ως χρήστης κυρίως του Google Earth και λιγότερο των Google Maps.

Η γλώσσα C++ αποτέλεσε για πολλά χρόνια πόλο έλξης για μένα αλλά δεν είχα ποτέ την ευκαιρία να την μάθω. Η παρούσα εφαρμογή διαθέτει συνολικά 8 προγράμματα γραμμένα σε C++ τα οποία βοηθούν σημαντικά στη σωστή λειτουργία της εφαρμογής.

Τέλος η γλώσσα προγραμματισμού Perl της οποίας την ύπαρξη δεν γνώριζα καν πριν την ενασχόλησή μου με την παρούσα εργασία χρησιμοποιήθηκε για την ανάγνωση των δεδομένων από εξωτερικό αρχείο και την αποστολή αυτών με συγκεκριμένη ποσότητα και ρυθμό (ροή) προς το ρεύμα δεδομένων του TelegraphCQ.

Μέσα από αυτήν την εργασία λοιπόν, είχα την ευκαιρία να μάθω πραγματικά πολλά και άκρως ενδιαφέροντα πράγματα που ήθελα και δεν είχα το χρόνο και άλλα που δεν γνώριζα καν την ύπαρξή τους και θα αξιοποιήσω στο άμεσο μέλλον.

Θα ήθελα λοιπόν να ευχαριστήσω θερμά όσους με βοήθησαν και με στήριξαν στην προσπάθειά μου αυτή και κυρίως τον υποψήφιο διδάκτορα Κώστα Πατρούμπα που

μου έδειξε τα πρώτα βήματα για την υλοποίηση της εφαρμογής και στάθηκε δίπλα μου σε κάθε δυσκολία, αλλά και τον επιβλέπων καθηγητή ΕΜΠ κ. Τίμο Σελλή για την επιλογή του να μου αναθέσει αυτήν την εργασία και για την εμπιστοσύνη που μου έδειξε παρόλο που δεν ήξερα γλώσσες προγραμματισμού. Θα ήθελα επίσης να ευχαριστήσω τους γονείς μου στους οποίους και αφιερώνω την παρούσα εργασία για την πραγματικά πολύτιμη καθημερινή βοήθειά τους επί τρεις μήνες που νυχθημερόν υλοποιούσα την εφαρμογή και έγραφα την εργασία. Θα ήθελα επίσης να ευχαριστήσω την συνάδελφο από το Μεταπτυχιακό Αντωνία Καραγιάννη για τις πολύτιμες συμβουλές της που συνέβαλαν στην βελτίωση της εφαρμογής από πολλές απόψεις. Ο επίσης συνάδελφος από το μεταπτυχιακό Μιχάλης Παιγνιγιάννης μου έδωσε πολύτιμες συμβουλές και αρχικές γνώσεις και τον ευχαριστώ ιδιαίτερα. Θα ήθελα επίσης να ευχαριστήσω τη Μαρία Φώτη για τη συνεργασία που είχαμε κατά το στάδιο εκπόνησης των εργασιών μας και ειδικά για την πολύτιμη ανταλλαγή απόψεων. Ακόμη, θα ήθελα να ευχαριστήσω την αδερφή μου που όποτε ζήτησα τη γνώμη της ήταν πρόθυμη να μου την δώσει και η εργασία έγινε ακόμη καλύτερη ακολουθώντας και τις συμβουλές της. Θα ήθελα τέλος να ευχαριστήσω τους Καθηγητές κ. Μ. Κάβουρα και κ. Λ. Τσούλο που μαζί με τον επιβλέποντα κ. Τ. Σελλή αποτέλεσαν την τριμελή εξεταστική επιτροπή και οι οποίοι παρακολούθησαν με ιδιαίτερο ενδιαφέρον την παρουσίαση αυτής και έκαναν καταλυτικά σχόλια και προτάσεις για την περαιτέρω βελτίωση του τελικού κειμένου της εργασίας.

Στα πλαίσια του Μεταπτυχιακού προγράμματος “Γεωπληροφορική”, το οποίο ολοκληρώνω με την παρούσα εργασία, θα ήθελα να ευχαριστήσω όλους τους Καθηγητές που μας δίδαξαν τα αντίστοιχα μαθήματα και την προθυμότατη να μας βοηθήσει με κάθε τρόπο στα διάφορα θέματα που μας απασχολούσαν κ. Ε. Παλιάτσου Γραμματέα του Μεταπτυχιακού. Θα ήθελα επίσης να ευχαριστήσω ειδικά τον καθηγητή ΕΜΠ κ. Γεώργιο Παναγιώτου για την εμπιστοσύνη που μου έδειξε κατά την επιλογή μου ως σπουδαστή στο παρόν Μεταπτυχιακό.

Αθήνα, Μάρτιος 2009
Κωνσταντίνος Θεοφιλογιαννάκος

Σύνοψη

Ως γνωστόν, στη σημερινή εποχή η τεχνολογία έχει κάνει άλματα και έχει παρεισφρήσει σε όλους τους τομείς της ανθρώπινης δραστηριότητας. Πλέον, μπορεί ο καθένας να δημιουργήσει δικές του ιστοσελίδες και να τις αναρτήσει στο διαδίκτυο με ποικίλους σκοπούς και στόχους (διαφήμιση, κέρδος, ψυχαγωγία, ενημέρωση κλπ). Οι δυνατότητες των υπολογιστών και των σχετικών με αυτά συστημάτων αυξάνονται συνεχώς και αντίστοιχα αυξάνονται οι δυνατότητες των προγραμμάτων και των εφαρμογών.

Βάσεις δεδομένων έχουν δημιουργηθεί από τα πρώτα χρόνια ανάπτυξης των υπολογιστών. Οι χωρικές βάσεις δεδομένων εμφανίστηκαν λίγο αργότερα και πλέον αξιοποιούνται σε πληθώρα εφαρμογών (Γεωγραφικά Συστήματα Πληροφοριών). Τα ρεύματα δεδομένων ακούγονται καινούργια ως έννοια αλλά δεν είναι. Εδώ και μία δεκαετία υπάρχει εκτενής έρευνα γύρω από αυτά, αλλά και πολλές εφαρμογές.

Η παρούσα εργασία αξιοποιεί πολλά διαφορετικά μεταξύ τους εργαλεία για την επίτευξη του στόχου της. Το λειτουργικό σύστημα Linux, το σύστημα διαχείρισης ρευμάτων δεδομένων TelegraphCQ, το σύστημα διαχείρισης βάσεων δεδομένων PostgreSQL, οι γλώσσες HTML, PHP, XML, JavaScript, C++ και Perl και οι χάρτες της Google συνδυάζονται για τη δημιουργία μιας διαδικτυακής εφαρμογής που στόχο έχει την παρακολούθηση και διαχείριση στόλου κινούμενων αντικειμένων σε πραγματικό χρόνο από οποιονδήποτε υπολογιστή ανά την υφήλιο χωρίς την ανάγκη εγκατάστασης συγκεκριμένου λογισμικού (χρειάζεται μόνο ένας φυλλομετρητής διαδικτύου – web browser).

Ο στόχος επετεύχθη. Κάθε εργαλείο έπαιξε σημαντικό ρόλο στη σωστή λειτουργία της εφαρμογής.

Η παρούσα εφαρμογή περιγράφεται ως εξής:

Υπάρχει ιστοσελίδα (front-end) για επικοινωνία με το χρήστη, υποβολή ερωτημάτων και εμφάνιση αποτελεσμάτων σε χάρτες Google αλλά και σε πίνακες.

Στην κεντρική αυτή ιστοσελίδα ο χρήστης μπορεί να εκτελέσει μέχρι πέντε ερωτήματα ταυτόχρονα και να δει όλα τα αποτελέσματα να οπτικοποιούνται στους ενσωματωμένους χάρτες. Υπάρχει το ένα βασικό ερώτημα το οποίο επιστρέφει επί του χάρτη όλα τα υπό διαχείριση αντικείμενα. Υπάρχει ένα ερώτημα περιοχής το οποίο μας δίδει τη δυνατότητα να σχεδιάσουμε επάνω στο χάρτη την περιοχή στην οποία θέλουμε να δούμε ποια κινούμενα αντικείμενα υπάρχουν. Υπάρχουν τέλος τρία προσαρμοσμένα ερωτήματα στα οποία ο χρήστης μπορεί να εισάγει δικά του ερωτήματα προς εκτέλεση.

Η PHP αναλαμβάνει να εκτελέσει τα υποβληθέντα από την ιστοσελίδα ερωτήματα στην PostgreSQL. Η τελευταία τα στέλνει προς επεξεργασία στο TelegraphCQ. Αυτό επιστρέφει τα αποτελέσματα στην PostgreSQL η οποία τα παραλαμβάνει και τα στέλνει στην PHP. Η PHP αποθηκεύει τα αποτελέσματα σε αρχεία της γλώσσας XML. Καθώς τα ερωτήματα απευθύνονται σε ρεύματα δεδομένων που εισρέουν συνεχώς στο σύστημα, είναι συνεχή. Έτσι, η παραπάνω διαδικασία επαναλαμβάνεται ανά τακτά χρονικά διαστήματα, μέχρι να διακοπεί η εκτέλεση των ερωτημάτων.

Τα επαναληπτικώς δημιουργούμενα αρχεία XML αναλαμβάνει να διαβάσει η JavaScript και κατόπιν αναγκαίας επεξεργασίας, τα οπτικοποιεί στους χάρτες Google της κεντρικής ιστοσελίδας της εφαρμογής και τα εμφανίζει και σε πίνακες στο αντίστοιχο τμήμα της ιστοσελίδας. Επίσης με τη συνδρομή της JavaScript, γίνεται έλεγχος των διαφόρων πεδίων της ιστοσελίδας ώστε να αποφεύγονται λάθη εισαγωγής δεδομένων και σφάλματα κατά την επεξεργασία. Ακόμη, ο χρήστης μπορεί να διαχειρίζεται εύκολα την εφαρμογή βλέποντας ποια ρεύματα δεδομένων και ποια ερωτήματα είναι ανά πάσα στιγμή ενεργά και ποια όχι.

Η εφαρμογή διαθέτει πληθώρα λειτουργιών που την καθιστούν φιλική προς το χρήστη και κατάλληλη για το σκοπό για τον οποίο δημιουργήθηκε. Έχει δοκιμαστεί εκτεταμένα σε διαφορετικές συνθήκες και με πλήθος διαφορετικών ερωτημάτων και η συμπεριφορά της ήταν η επιθυμητή. Μπορεί τέλος να χρησιμοποιηθεί σε πολλές εφαρμογές με σκοπό την παρακολούθηση και διαχείριση οχημάτων και άλλων αντικειμένων σε πραγματικό χρόνο.

Λέξεις κλειδιά: Χωρικές βάσεις δεδομένων, ρεύματα δεδομένων, συνεχή ερωτήματα, κινούμενα αντικείμενα, TelegraphCQ, GoogleMaps, PostgreSQL, JavaScript, PHP, C++, Linux, HTML, XML, Perl.

Abstract

It is well known that nowadays technology evolves rapidly and has entered in every portion of human business and activities in general. Now, everyone can create his own web pages and publish them on the internet to accomplish his own aims (promotion, profit, entertainment, inform-news etc). The capabilities of computers and systems related to them are constantly increasing and at the same time the capabilities of computer programs and applications follow that increase.

Databases have been created since the time of existence of the first computers worldwide. Spatial databases made their appearance a few years later and nowadays are used in a huge variety of applications (Geographical Information Systems etc). Data streams may appear like they are a new technology, but this isn't the case. The last decade there have been a thorough research about them and many relevant applications have been created.

The present study takes advantage of many different tools in order to achieve its purpose. Linux operating system, TelegraphCQ database management system, PostgreSQL database system, HTML, PHP, XML, JavaScript, C++ and Perl languages, and Google Maps are being combined in order to create a web application which aims to monitor and manage a set of moving objects in real time from any computer system worldwide without the need to install any special program (only a web browser is needed).

The goal was achieved. Every tool has an important reason to be a part of this application.

The present application is described below:

The front-end of the application is a web page. The user, via this web page, may submit queries to the system and watch the results on Google Maps (in the same page) and on certain arrays.

In this central web page the user may submit up to five queries simultaneously and watch all the results being visualized in the embedded maps.

There is a basic query that returns on the map all moving objects managed from this application. There is an area query which gives the user the opportunity to draw on the map the desired area where he wants to see all moving objects if there are any. At last, there are three more custom queries in which the user may insert his queries for execution.

PHP undertakes the dispatch of the submitted queries posed from the web page to PostgreSQL. PostgreSQL sends the queries to TelegraphCQ in order to be processed. TelegraphCQ returns the results to PostgreSQL which receives and dispatches them to PHP. PHP stores the results in XML files. Since queries are addressed to data streams which flow in the system continuously, they are also continuous. As a result, the abovementioned procedure is repeated in constant time intervals until the execution of the queries is stopped.

The iteratively created XML files are being read from JavaScript which, after processing them, visualizes them on GoogleMaps in the front-end and also displays them in arrays in the relevant section of the central web page. Also JavaScript, via certain functions, checks every field of the web page upon submission of any query so that to avoid errors in the data inserted and in the process of the queries. The user may also manage the application more easily as there is the possibility to watch which data streams are running and which queries are at any time active.

The application offers a variety of operations which make it user friendly and suitable for the purpose it is created. It has been tested extensively in various conditions and with many different queries and its behavior and results were the

desired ones. At last, it may be used in many cases where monitoring and management of vehicles and other moving objects in real time, is of importance and need.

Keywords: Spatial Databases, data streams, continuous queries, moving objects, TelegraphCQ, GoogleMaps, PostgreSQL, JavaScript, PHP, C++, Linux, HTML, XML, Perl.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Γενικά

Τις τελευταίες δεκαετίες και ειδικά τα τελευταία χρόνια είμαστε όλοι μάρτυρες μιας αλματώδους προόδου της τεχνολογίας, η οποία επηρεάζει όλους τους τομείς της ανθρώπινης δραστηριότητας.

Η τεχνολογία δημιουργείται από τον άνθρωπο με σκοπό να εξυπηρετήσεις τις ίδιες τις ανάγκες του ανθρώπου που μπορεί να προκύψουν κατά τη διάρκεια της ζωής του. Η τεχνολογία λοιπόν προσπαθώντας να βοηθήσει τον άνθρωπο σε οτιδήποτε κι αν αυτός κάνει και να του δώσει τη δυνατότητα να κάνει πράγματα που χωρίς αυτήν δε θα μπορούσε, έχει παρεισφρήσει σε όλους σχεδόν τους τομείς της ανθρώπινης δραστηριότητας.

Η εξέλιξη της τεχνολογίας είναι άρρηκτα συνδεδεμένη με την εξέλιξη των Ηλεκτρονικών Υπολογιστών (Η/Υ). Πολλά πράγματα που στο παρελθόν κανείς δεν φανταζόταν καν ότι θα μπορούσαν να γίνουν, σήμερα αποτελούν καθημερινή ενασχόληση των περισσοτέρων ανθρώπων. Χαρακτηριστικό παράδειγμα είναι το ηλεκτρονικό ταχυδρομείο και το διαδίκτυο κατ' επέκταση με τα οποία, σε χρόνο δευτερολέπτων, μπορεί κανείς να επικοινωνήσει με ποικίλους τρόπους με οποιονδήποτε συνάνθρωπό του ανά την υφήλιο.

1.2 Συστήματα Διαχείρισης Βάσεων Δεδομένων

Από τα πρώτα χρόνια δημιουργίας των Η/Υ αναπτύχθηκαν συστήματα καταχώρησης και επεξεργασίας πλήθους πληροφοριών κάνοντας τη ζωή όσων χρειάζονταν τέτοια συστήματα πολύ πιο εύκολη. Κλασσικό παράδειγμα χρήσης τέτοιων συστημάτων είναι σε βιβλιοθήκες και γενικότερα σε περιπτώσεις ύπαρξης πολλών αντικειμένων τα οποία χρειάζεται να παρακολουθούνται και να διαχειρίζονται με σχετική ευχέρεια. Τα συστήματα αποθήκευσης πληροφοριών με σκοπό την ανάκτησή τους, επεξεργασία τους και γενικότερη διαχείρισή τους είναι τα γνωστά Συστήματα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ). Οι πληροφορίες αποθηκεύονται σε βάσεις δεδομένων και υπάρχει η δυνατότητα θέσεως συγκεκριμένων ερωτημάτων και λήψεως των απαντήσεων από τους Η/Υ. Οι βάσεις δεδομένων ακολουθούν μια δομή προκειμένου, εκτός των άλλων, να είναι εύκολη η καταχώρηση των πληροφοριών αλλά και η λήψη απαντήσεων επί διαφόρων ερωτημάτων. Δημιουργείται ένας ή περισσότεροι πίνακες οι οποίοι έχουν ένα ή περισσότερα χαρακτηριστικά ο καθένας και στους οποίους αποθηκεύονται οι επιθυμητές πληροφορίες. Οι δυνατότητες των ΣΔΒΔ καλύπτουν ουσιαστικά κάθε πιθανή ανάγκη διαχείρισης των πληροφοριών – δεδομένων που αποθηκεύουν. Έτσι, στην περίπτωση μιας βιβλιοθήκης θα μπορούσε κάλλιστα να υπάρχει ένας πίνακας με όλα τα βιβλία και πολλές πληροφορίες γύρω από αυτά (τίτλος βιβλίου, έτος έκδοσης, ημ/νία εισαγωγής στη βιβλιοθήκη, συγγραφέας, εκδοτικός οίκος, πλήθος αντιτύπων κ.α.). Θα μπορούσαν ταυτόχρονα να υπάρχουν και άλλοι πίνακες όπως π.χ. πίνακας συγγραφέων με διάφορες πληροφορίες για αυτούς, πίνακας εκδοτικών οίκων με αντίστοιχες πληροφορίες, αλλά και πίνακας δανειστών όσων βιβλίων διατίθενται προς δανεισμό με τις αντίστοιχες πληροφορίες.

1.3 Συστήματα Διαχείρισης Ρευμάτων Δεδομένων

Τα ΣΔΒΔ, στην πλειονότητα των περιπτώσεων, έχουν σχεδιαστεί με γνώμονα την αρχική καταχώρηση μεγάλου πλήθους πληροφοριών, την μετέπειτα μικρή σε βαθμό προσθήκη, διαγραφή, τροποποίηση των πληροφοριών αυτών και την ιδιαίτερη υποστήριξη διαχείρισης αυτών μέσω θέσεως ερωτημάτων και λήψεως αποτελεσμάτων. Τα τελευταία χρόνια διαφάνηκε η πιθανή αξιοποίηση των βάσεων δεδομένων και για περιπτώσεις πολύ πιο δυναμικών πληροφοριών, περιπτώσεις κατά τις οποίες σε χρονικό διάστημα μερικών λεπτών ή και μερικών δευτερολέπτων έχουν ανανεωθεί τα στοιχεία που αυτές διαχειρίζονται. Τέτοιες περιπτώσεις είναι η παρακολούθηση των τιμών των διαφόρων μετοχών σε ένα χρηματιστήριο και η παρακολούθηση και διαχείριση των δεδομένων που διακινούνται μεταξύ διάφορων υπολογιστών. Στις περιπτώσεις αυτές είναι τόσο συχνή η αλλαγή των πληροφοριών που είναι πρακτικά αδύνατη αλλά και ασύμφορη η πληκτρολόγηση και καταχώρηση αυτών με τις κλασσικές μεθόδους των βάσεων δεδομένων. Στις περιπτώσεις αυτές τα δεδομένα εισέρχονται στο σύστημα υπό μορφή ρευμάτων.

Για την καταχώρηση και διαχείριση των παραπάνω δυναμικής φύσεως δεδομένων, αναπτύχθηκαν τα λεγόμενα Συστήματα Διαχείρισης Ρευμάτων Δεδομένων. Όταν οι πληροφορίες αλλάζουν συνεχώς, η απάντηση σε ένα κλασικό ερώτημα, θα έχει νόημα μόνο για τη συγκεκριμένη χρονική στιγμή που αυτό εκτελείται. Την επόμενη χρονική στιγμή, τα δεδομένα θα έχουν αλλάξει και συνεπώς η απάντηση στο κλασικό ερώτημα που τέθηκε δε θα ανταποκρίνεται στην πραγματικότητα αλλά σε παλαιότερα δεδομένα. Συνεπώς, δημιουργήθηκε η ανάγκη ένα ερώτημα να μην εκτελείται μόνο μια φορά αλλά να εκτελείται συνεχώς – να παραμένει διαρκώς ενεργό και να φέρνει συνεχώς τις απαντήσεις στο χρήστη.

1.4 Συστήματα Εντοπισμού Θέσης

Έκτος, από την εξέλιξη των Η/Υ και τα συστήματα διαχείρισης βάσεων και ρευμάτων δεδομένων που έχουν αναπτυχθεί, η τεχνολογική πρόοδος μας έχει δώσει πάμπολλες άλλες δυνατότητες.

Έτσι, όπως όλοι γνωρίζουμε, με την πρόοδο της τεχνολογίας, έχουν αναπτυχθεί, εκτός των άλλων, συστήματα εντοπισμού θέσης (Global Positioning Systems). Παρέχεται πλέον η δυνατότητα σε οποιονδήποτε έχει μια σχετική συσκευή (GPS) να γνωρίζει ανά πάσα στιγμή τη θέση του οπουδήποτε κι αν βρίσκεται. Η πληροφορία αυτή παρέχεται με τη βοήθεια δορυφόρων, οι οποίοι με τη σειρά τους γνωρίζουν ανά πάσα στιγμή την ακριβή τους θέση σε σχέση με ένα παγκόσμιο σύστημα αναφοράς. Όλες οι συσκευές εντοπισμού θέσης, μπορούν να μας δείξουν τις συντεταγμένες της θέσης μας ανά πάσα στιγμή.

Με την πρόοδο της τεχνολογίας υπάρχει πλέον η δυνατότητα ένα κινούμενο αντικείμενο, που διαθέτει συσκευή εντοπισμού θέσης, να στέλνει την πληροφορία των συντεταγμένων του, καθώς και άλλα στοιχεία όπως την ταυτότητά του, τη χρονική στιγμή λήψης της θέσης κλπ, σε συγκεκριμένο δέκτη πληροφοριών. Όλα τα επιβατικά πλοία ανά τον κόσμο είναι υποχρεωμένα να διαθέτουν, εκτός από συσκευή GPS και ειδική συσκευή αποστολής της θέσης τους, της ταυτότητάς τους και άλλων πληροφοριών μέσω ηλεκτρομαγνητικών κυμάτων. Επίσης, υπάρχουν πλέον κινητά τηλέφωνα που διαθέτουν ενσωματωμένο σύστημα εντοπισμού θέσης αλλά και σύνδεση στο διαδίκτυο. Είναι λοιπόν εφικτή η λήψη πληροφοριών σε πραγματικό χρόνο από ομάδα κινούμενων αντικειμένων.

1.5 Δυνατότητα εμφάνισης χάρτη σε ιστοσελίδες και απεικόνισης πληροφοριών επ' αυτού

Όπως προαναφέρθηκε, η τεχνολογική πρόοδος έχει προσφέρει πληθώρα νέων δυνατοτήτων στον καθένα μας. Υπάρχει πλέον η δυνατότητα εμφάνισης διαφόρων ειδών χαρτών σε ιστοσελίδες και εμφάνισης αντικειμένων επ' αυτών. Έχοντας κανείς γνώσεις σχετικά με τη δημιουργία ιστοσελίδων μπορεί να δημιουργήσει τον δικό του ιστοτόπο και να τον αναρτήσει σε σχετικό υπολογιστή-εξυπηρετητή διαδικτύου ώστε να μπορεί ο καθένας να τον προσπελάσει ανά πάσα στιγμή, από οποιονδήποτε υπολογιστή.

Τουλάχιστον δύο μεγάλες εταιρείες εφαρμογών διαδικτύου, παρέχουν στους προγραμματιστές-δημιουργούς ιστοσελίδων τη δυνατότητα ενσωμάτωσης διαφόρων ειδών χαρτών στις εφαρμογές τους. Έτσι, μπορεί κανείς (με την ανάλογη ενασχόληση) να ενσωματώσει σε εφαρμογές του οδικούς χάρτες, δορυφορικές εικόνες υψηλής ανάλυσης, υβριδικούς χάρτες που προκύπτουν από τον συνδυασμό των δύο, χάρτη αναγλύφου επιφανείας της γης αλλά και δικούς του χάρτες με την κατάλληλη γεωαναφορά.

Παρέχεται επίσης η δυνατότητα απεικόνισης επί του κάθε χάρτη διαφόρων πληροφοριών που επιθυμεί ο δημιουργός της ιστοσελίδας. Πιο συγκεκριμένα μπορεί κανείς να εμφανίσει σημεία-σημάδια, τεθλασμένες γραμμές και πολύγωνα.

1.6 Αντικείμενο εργασίας – Απεικόνιση κινούμενων αντικειμένων επί χάρτη

Συνδυάζοντας τις διάφορες τεχνολογίες και υπηρεσίες που παρέχονται, έχει κανείς τη δυνατότητα να δημιουργήσει δικές του εφαρμογές που με τη σειρά τους δύνανται να συμβάλλουν στην εξέλιξη των δυνατοτήτων που παρέχει η σημερινή τεχνολογία είτε μέσω των εφαρμογών αυτών καθεαυτών είτε μέσω των δυσκολιών που προέκυψαν και των πιθανών αιτίων αδυναμίας δημιουργίας των εφαρμογών.

Στην παρούσα εργασία επιχειρήθηκε η διαχείριση ρεύματος πληροφοριών κινούμενων αντικειμένων και η απεικόνιση των θέσεων και των τροχιών αυτών σε χάρτη ενσωματωμένο σε ιστοσελίδα. Επιχειρήθηκε η διαχείριση να γίνεται μέσα από την ιστοσελίδα δίνοντας τη δυνατότητα στο χρήστη να θέσει μέχρι και τρία δικά του ερωτήματα και να βλέπει σε πραγματικό χρόνο τα αποτελέσματα $1^{ου}$ σε πίνακα δίπλα στο κάθε ερώτημα και $2^{ου}$ επί του χάρτη.

Για την δημιουργία της εφαρμογής χρειάστηκε η συνδυασμένη χρήση του λειτουργικού Linux, των γλωσσών προγραμματισμού PHP, HTML και JavaScript, του συστήματος διαχείρισης ρευμάτων δεδομένων TelegraphCQ, του συστήματος διαχείρισης βάσεων δεδομένων PostgreSQL, του μορφοτύπου πληροφοριών XML και των χαρτών της Google.

1.7 Σχετικές εργασίες

Σίγουρα υπάρχουν αρκετές εργασίες στις οποίες έχει επιχειρηθεί μέχρι σήμερα η απεικόνιση διαφόρων πληροφοριών σε χάρτη ενσωματωμένο σε αντίστοιχη ιστοσελίδα. Στο διαδίκτυο υπάρχει αφθονία σχετικών πληροφοριών και εργασιών. Ενδεικτικές πηγές πληροφοριών υπάρχουν στις αναφορές στο τέλος της παρούσας εργασίας.

Υπάρχει εργασία σχετική με την οπτικοποίηση ρευμάτων τροχιάς κινούμενων αντικειμένων όπου η οπτικοποίηση δεν γίνεται σε χάρτη ενσωματωμένο σε ιστοσελίδα αλλά σε εξειδικευμένο πρόγραμμα που δημιουργήθηκε γι' αυτόν τον σκοπό. Οι διαφορές με την παρούσα εργασία είναι μεγάλες καθώς σε αυτήν την περίπτωση δημιουργήθηκε πλήρως δυναμική διαδικτυακή εφαρμογή με τη βοήθεια διαφορετικών γλωσσών προγραμματισμού, και η απεικόνιση γίνεται πάνω σε ενσωματωμένο χάρτη στην ίδια ιστοσελίδα.

1.8 Επισκόπηση λοιπών κεφαλαίων

Στο κεφάλαιο που ακολουθεί αναλύονται οι χωρικές βάσεις δεδομένων. Γίνεται αναφορά σε συγκεκριμένα συστήματα που υποστηρίζουν χαρακτηριστικά εγγενώς χωρική πληροφορία και παρατίθενται οι δυνατότητες που αυτά παρέχουν στο χρήστη. Γίνεται επίσης αναφορά στο Σύστημα Διαχείρισης Βάσεων Δεδομένων PostgreSQL.

Στο τρίτο κεφάλαιο, γίνεται μία πιο εκτενής αναφορά στα ρεύματα δεδομένων και στα συνεχή ερωτήματα που χρησιμοποιούνται στα ΣΔΡΔ. Παρουσιάζονται τα συστήματα διαχείρισης που έχουν αναπτυχθεί μέχρι σήμερα και αναλύεται εκτενώς το σύστημα TelegraphCQ που χρησιμοποιήθηκε.

Το τέταρτο κεφάλαιο αναφέρεται στα εργαλεία που επιλέχθηκαν για την ανάπτυξη της εφαρμογής. Αναλύονται οι δυνατότητες των προγραμμάτων και των γλωσσών προγραμματισμού που χρησιμοποιήθηκαν και αιτιολογείται η χρήση του καθενός έναντι άλλων.

Συνεχίζοντας στο πέμπτο κεφάλαιο, περιγράφεται η υλοποίηση της εφαρμογής και οι δυνατότητές που αυτή παρέχει στο χρήστη. Πιο συγκεκριμένα, γίνεται αναλυτική περιγραφή κάθε διαδικασίας που εκτελείται από την εφαρμογή.

Στο έκτο κεφάλαιο παρουσιάζεται η εφαρμογή σε λειτουργία με ενδεικτικά αποτελέσματα αυτής και αντίστοιχες επεξηγήσεις.

Στο τελευταίο κεφάλαιο, παρατίθενται τα συμπεράσματα από την υλοποίηση της εφαρμογής. Γίνεται αναφορά στις δυσκολίες που ανέκυψαν στα διάφορα στάδια υλοποίησής της αλλά και σε διάφορους περιορισμούς που αυτή έχει ως συνέπεια περιορισμών των διαφόρων προγραμμάτων και της διασύνδεσης αυτών. Τέλος, αναφέρονται πιθανές μελλοντικές επεκτάσεις της εφαρμογής που θα μπορούσαν να την κάνουν να ανταποκρίνεται σε περισσότερες απαιτήσεις από την πλευρά του χρήστη.

Στα Παραρτήματα της παρούσας εργασίας παρατίθενται οι κώδικες του συνόλου των προγραμμάτων, scripts και αρχείων που αναπτύχθηκαν για την παρούσα εφαρμογή καθώς και σχετικές πληροφορίες για τα πιο βασικά από αυτά.

Κεφάλαιο 2

Επισκόπηση χωρικών βάσεων δεδομένων - PostgreSQL

2.1 Επισκόπηση χωρικών βάσεων δεδομένων

2.1.1 Γενικά

Σε διάφορα επιστημονικά πεδία, υπάρχει η ανάγκη διαχείρισης γεωμετρικών, γεωγραφικών ή χωρικών πληροφοριών, δηλαδή δεδομένων που συσχετίζονται με το χώρο. Ο χώρος ενδιαφέροντος μπορεί να είναι, για παράδειγμα, η δισδιάστατη απεικόνιση τμημάτων της επιφάνειας της γης, ο σχεδιασμός μικροεπεξεργαστών και κυκλωμάτων, ένας τρισδιάστατος χώρος που επιχειρεί να αναπαραστήσει τον ανθρώπινο εγκέφαλο ή ένας άλλος τρισδιάστατος χώρος στον οποίο αναπαρίσταται η διάρθρωση μορίων πρωτεϊνών. Προσπάθειες διαχείρισης τέτοιων δεδομένων έχουν ξεκινήσει από την εμφάνιση των πρώτων σχεσιακών βάσεων δεδομένων.

Ένα σύστημα διαχείρισης χωρικών βάσεων δεδομένων, περιγράφεται ως εξής:

Αποτελεί και αυτό ένα Σύστημα Διαχείρισης Βάσεων Δεδομένων (με περισσότερες δυνατότητες)

Διαθέτει χωρικούς τύπους δεδομένων (Spatial Data Types - SDTs) στο μοντέλο δεδομένων και στη γλώσσα ερωταποκρίσεων (query language). Οι συνήθεις χωρικοί τύποι δεδομένων είναι το σημείο, η τεθλασμένη γραμμή και η περιοχή-πολύγωνο.

Υποστηρίζει χωρικούς τύπους δεδομένων στην υλοποίησή του, παρέχοντας τουλάχιστον χωρικά ευρετήρια και αποτελεσματικούς αλγόριθμους για εκτέλεση χωρικής σύνδεσης.

Παραδείγματα οντοτήτων των οποίων προκύπτει η ανάγκη για αναπαράσταση είναι για την περίπτωση πολυγώνων οι ιδιοκτησίες επάνω στην επιφάνεια της γης, τα όρια οικισμών και τα όρια περιοχών ενός χωροταξικού σχεδιασμού. Παραδείγματα τεθλασμένων γραμμών είναι το οδικό δίκτυο μιας περιοχής, το υδρογραφικό δίκτυο, το δίκτυο καλωδίων μεταφοράς ηλεκτρικού ρεύματος και τηλεφώνου, δίκτυο δρομολογίων θαλασσίων συγκοινωνιών και πολλά άλλα. Παραδείγματα σημείων που θα ενδιέφερε να αναπαρασταθούν σε μία χωρική βάση είναι οι θέσεις διαφόρων σημείων ενδιαφέροντος όπως θέσεις πόλεων σε χάρτες πολύ μικρής κλίμακας, οι θέσεις άστρων σε αναπαράσταση του ουράνιου στερεώματος, οι θέσεις διαφόρων ειδών αισθητήρων, οι θέσεις κινούμενων αντικειμένων και πολλές άλλες θέσεις.

2.1.2 Χωρικοί τύποι δεδομένων

Οι βασικοί χωρικοί τύποι δεδομένων τους οποίους πρέπει να υποστηρίζει ένα σύστημα διαχείρισης χωρικών βάσεων δεδομένων είναι τα σημεία, οι γραμμές (δηλαδή τεθλασμένες γραμμές) και οι περιοχές (δηλαδή πολύγωνα).

Οι χωρικοί αυτοί τύποι θα πρέπει να υποστηρίζουν τέσσερις κατηγορίες λειτουργιών, ως εξής:

1. Χωρικά κατηγορήματα που εκφράζουν τοπολογικές σχέσεις. Παραδείγματα είναι: α) Μία γεωμετρία οποιουδήποτε είδους, εάν εμπεριέχεται σε μία περιοχή-πολύγωνο. β) Μία γραμμή ή περιοχή, εάν διασταυρώνεται ή συναντά άλλη γραμμή ή περιοχή. γ) Μία περιοχή εάν είναι γειτονική ή εάν περικλείει μία άλλη περιοχή. δ) Ένα σημείο εάν συμπίπτει με άλλο σημείο ή εάν βρίσκεται πάνω σε μία γραμμή.
2. Λειτουργίες που επιστρέφουν ως αποτελέσματα χωρικούς τύπους δεδομένων. Παραδείγματα είναι: α) Εάν δύο γραμμές τέμνονται, τότε η τομή τους θα πρέπει να επιστρέφει σημείο. β) Εάν δύο περιοχές τέμνονται τότε, η τομή τους θα πρέπει να επιστρέφει άλλη περιοχή. γ) Εάν δύο τυχαίοι χωρικοί τύποι της ίδιας κατηγορίας ενωθούν, θα πρέπει να επιστέψουν ένα χωρικό τύπο πάλι της ίδιας κατηγορίας. δ) Εάν ζητηθεί το περίγραμμα περιοχής, θα πρέπει να επιστραφεί μία γραμμή.
3. Χωρικές λειτουργίες που επιστρέφουν αριθμούς. Παραδείγματα είναι: α) Απόσταση γεωμετρίας οποιασδήποτε κατηγορίας από γεωμετρία οποιασδήποτε κατηγορίας. β) Λειτουργίες επάνω σε περιοχές όπως ο υπολογισμός του εμβαδού και της περιμέτρου. γ) Λειτουργίες επάνω σε γραμμές όπως είναι ο υπολογισμός του μήκους αυτών.
4. Χωρικές λειτουργίες σε ομάδες αντικειμένων. Παραδείγματα είναι: α) Συνάθροιση ομάδας χωρικών αντικειμένων ίδιας κατηγορίας σε ένα χωρικό αντικείμενο της ίδιας κατηγορίας. β) Εύρεση από σύνολο αντικειμένων, αυτών που πληρούν συγκεκριμένες προϋποθέσεις όπως απόσταση (πιο κοντά, πιο μακριά κλπ) γ) Εύρεση από σύνολο αντικειμένων, αυτού που είναι πιο κοντά από όλα τα άλλα.

2.1.3 Χωρικές σχέσεις

Οι χωρικές σχέσεις είναι οι πιο σημαντικές ανάμεσα στις λειτουργίες που προσφέρει η χωρική άλγεβρα. Καθιστούν εφικτή για παράδειγμα την αναζήτηση όλων των αντικειμένων που πληρούν μια ή περισσότερες χωρικές σχέσεις σχετικά με ένα η περισσότερα χωρικά αντικείμενα.

Οι σχέσεις μπορούν να διαχωριστούν σε κατηγορίες ως εξής:

- Τοπολογικές σχέσεις όπως γειτονικός, εσωτερικός, μη επικαλυπτόμενος είναι ανεξάρτητες τοπολογικών μετασχηματισμών, όπως η αλλαγή κλίμακας, η περιστροφή ή ο μετασχηματισμός συντεταγμένων.
- Σχέσεις κατεύθυνσης όπως από πάνω, από κάτω, αριστερά, δεξιά, Βόρεια, Νότια, Ανατολικά, Δυτικά.
- Μετρικές σχέσεις όπως η απόσταση να είναι μικρότερη από τόσο.

Ανάμεσα σε αυτές, οι τοπολογικές σχέσεις είναι οι πιο θεμελιώδεις και σημαντικές και έχουν μελετηθεί σε μεγαλύτερο βάθος.

Παρακάτω ακολουθεί ενδεικτική ανάλυση ελέγχου σχέσεων μεταξύ δύο πολυγώνων.

Έστω A_1 και A_2 δύο πολύγωνα δύο διαστάσεων. Έστω ότι ∂A_1 και ∂A_2 είναι τα περιγράμματα των A_1 και A_2 αντίστοιχα. Έστω επίσης ότι A_1^0 και A_2^0 είναι τα εσωτερικά των A_1 και A_2 αντίστοιχα. Οι τοπολογικές σχέσεις μεταξύ των δύο πολυγώνων μπορούν να εξαχθούν από τον έλεγχο τομής-επικάλυψης των περιγραμμάτων και των εσωτερικών των πολυγώνων.

Η τομή του περιγράμματος του A1 (∂A_1) με το περίγραμμα του A2 (∂A_2) συμβολίζεται ως εξής: $\partial A_1 \cap \partial A_2$.

Η τομή του περιγράμματος του A1 (∂A_1) με το εσωτερικό του A2 (A_2^0) συμβολίζεται ως εξής: $\partial A_1 \cap A_2^0$.

Η τομή του εσωτερικού του A1 (A_1^0) με το περίγραμμα του A2 (∂A_2) συμβολίζεται ως εξής: $A_1^0 \cap \partial A_2$.

Η τομή του εσωτερικού του A1 (A_1^0) με το εσωτερικό του A2 (A_2^0) συμβολίζεται ως εξής: $A_1^0 \cap A_2^0$.

Το σύμβολο Ø δηλώνει το κενό, ενώ ο συμβολισμός ≠Ø δηλώνει αποτέλεσμα διάφορο του κενού.

Ακολουθεί λοιπόν, σχετικός πίνακας σχέσεων μεταξύ του πολυγώνου A_1 και του A_2.

A/A σχέσης	$\partial A_1 \cap \partial A_2$	$\partial A_1 \cap A_2^0$	$A_1^0 \cap \partial A_2$	$A_1^0 \cap A_2^0$	Περιγραφή σχέσης
1	Ø	Ø	Ø	Ø	**Το A_1 ανεξάρτητο – μη επικαλυπτόμενο με το A_2**
2	Ø	Ø	Ø	≠Ø	Μη εφικτό ενδεχόμενο
3	Ø	Ø	≠Ø	Ø	Μη εφικτό ενδεχόμενο
4	Ø	Ø	≠Ø	≠Ø	**Το A_2 είναι μέσα στο A_1**
5	Ø	≠Ø	Ø	Ø	Μη εφικτό ενδεχόμενο
6	Ø	≠Ø	Ø	≠Ø	**Το A_1 είναι μέσα στο A_2**
7	Ø	≠Ø	≠Ø	Ø	Μη εφικτό ενδεχόμενο
8	Ø	≠Ø	≠Ø	≠Ø	Μη εφικτό ενδεχόμενο
9	≠Ø	Ø	Ø	Ø	**Το A_1 ακουμπά στο A_2**
10	≠Ø	Ø	Ø	≠Ø	**Το A_1 ισούται με το A_2**
11	≠Ø	Ø	≠Ø	Ø	Μη εφικτό ενδεχόμενο
12	≠Ø	Ø	≠Ø	≠Ø	**Το A_1 υπερκαλύπτει το A_2**
13	≠Ø	≠Ø	Ø	Ø	Μη εφικτό ενδεχόμενο
14	≠Ø	≠Ø	Ø	≠Ø	**Το A_2 υπερκαλύπτει το A_1**
15	≠Ø	≠Ø	≠Ø	Ø	Μη εφικτό ενδεχόμενο
16	≠Ø	≠Ø	≠Ø	≠Ø	**Τα A_1 και A_2 έχουν επικαλυπτόμενες περιοχές**

Πίνακας υπολογισμού σχέσεων μεταξύ δύο πολυγώνων.

Αντίστοιχοι τρόποι εύρεσης σχέσεων μπορούν να αναπτυχθούν μεταξύ σημείου με πολύγωνο, γραμμής με πολύγωνο, σημείου με γραμμή, σημείου με σημείο και γραμμής με γραμμή.

2.1.4 Ερωτήματα σε χωρικές βάσεις δεδομένων

Θα μπορούσε να πει κανείς ότι το πρόβλημα των ερωταποκρίσεων στις χωρικές βάσεις δεδομένων είναι η σύνδεση των λειτουργιών της χωρικής άλγεβρας με τις ευκολίες που παρέχει η γλώσσα ερωταποκρίσεων ενός ΣΔΒΔ. Υπάρχουν όμως και άλλα θέματα που αφορούν κυρίως στο γεγονός ότι τα χωρικά δεδομένα χρήζουν γραφικής αναπαράστασης και γραφικού τρόπου εκτέλεσης ερωτημάτων ή τουλάχιστον χρήσης στα ερωτήματα τιμών χωρικών τύπων δεδομένων.

Οι θεμελιώδεις λειτουργίες της χωρικής άλγεβρας μπορούν να κατηγοριοποιηθούν ως εξής: χωρική επιλογή, χωρική σύνδεση, εφαρμογές χωρικών συναρτήσεων και άλλες ομάδες λειτουργιών. Με τον όρο χωρική επιλογή εννοείται η επιλογή αποτελεσμάτων που πληρούν τις επιθυμητές χωρικές προϋποθέσεις – χωρικά κατηγορήματα, όπως για παράδειγμα η επιλογή των αντικειμένων ή γραμμών ή πολυγώνων που βρίσκονται μέσα σε ένα μεγαλύτερο πολύγωνο (η λέξη-κλειδί "μέσα" θα πρέπει να υποστηρίζεται από την χωρική άλγεβρα). Παρομοίως, η χωρική σύνδεση είναι μία σύνδεση που συγκρίνει δύο οποιαδήποτε αντικείμενα μέσω μιας προϋπόθεσης – κατηγορήματος στις τιμές των χωρικών ιδιοτήτων αυτών, όπως η περίπτωση εύρεσης για κάθε γραμμή, των πολυγώνων ή των σημείων που βρίσκονται σε απόσταση μικρότερη από μία καθορισμένη. Οι εφαρμογές χωρικών συναρτήσεων χρησιμοποιούνται για την εκτέλεση ερωτημάτων με χωρικά κατηγορήματα – προϋποθέσεις και πιο συγκεκριμένα για τον υπολογισμό νέων τιμών χωρικών τύπων δεδομένων. Παράδειγμα αποτελεί η επιλογή γραμμών που πληρούν συγκεκριμένες προϋποθέσεις, π.χ. διέρχονται μέσα από κάποιο πολύγωνο, ενώ στις απαντήσεις ζητάμε το όνομα της γραμμής, το τμήμα που διέρχεται μέσα από το πολύγωνο και το μήκος αυτού. Τέλος, στις άλλες ομάδες λειτουργιών περιλαμβάνονται λειτουργίες που χειρίζονται ολόκληρες ομάδες χωρικών αντικειμένων με συγκεκριμένο τρόπο. Βασίζονται στη διασύνδεση μεταξύ χωρικής άλγεβρας και της άλγεβρας του ΣΔΒΔ. Η επικάλυψη χωρικών αντικειμένων είναι μία περίπτωση τέτοιας λειτουργίας και υπολογίζει και επιστρέφει την περιοχή που προκύπτει από την επικάλυψη δύο άλλων.

Τα ΣΔΧΒΔ είναι πολύ σημαντικό, τουλάχιστον όταν γίνεται χρήση αλληλεπίδρασης με το χρήστη, να υποστηρίζουν τη γραφική εισαγωγή πληροφοριών προς εκτέλεση στα ερωτήματα αλλά και τη γραφική απεικόνιση των αποτελεσμάτων εάν το επιθυμεί ο χρήστης. Οι απαιτήσεις ενός ΣΔΧΒΔ προκειμένου να υποστηρίζει κατά τον καλύτερο δυνατό τρόπο χωρικά ερωτήματα με γραφικό τρόπο, είναι εν συντομία οι παρακάτω:

- Υποστήριξη χωρικών τύπων δεδομένων (Spatial Data Types – SDTs).
- Γραφική απεικόνιση αποτελεσμάτων ερωτημάτων.
- Γραφικός συνδυασμός (επικάλυψη) πολλών αποτελεσμάτων ερωτημάτων.
- Στη γραφική εμφάνιση αποτελεσμάτων, είναι σημαντική η εμφάνιση συναφών πληροφοριών και στοιχείων στο παρασκήνιο (background) που βοηθούν το χρήστη στην καλύτερη εποπτεία και κατανόηση των αποτελεσμάτων των ερωτημάτων.
- Ύπαρξη διευκολύνσεων στον έλεγχο των περιεχομένων που συγκροτούν το γραφικό αποτέλεσμα εκτέλεσης ενός ή περισσοτέρων ερωτημάτων.
- Δυνατότητα χρήσης συσκευών κατάδειξης για επιλογή αντικειμένων σε ένα γραφικό αποτέλεσμα ή σε μία εικόνα.

- Υποστήριξη διαφορετικών γραφικών αναπαραστάσεων. Πρέπει να είναι εφικτή η αντιστοίχιση διαφορετικών γραφικών αναπαραστάσεων των αποτελεσμάτων ανάλογα με τις πληροφορίες κάθε απάντησης-αποτελέσματος κάθε ερωτήματος. Π.χ. χρήση διαφορετικών χρωμάτων ανάλογα με την απόσταση αντικειμένων από άλλο αντικείμενο κλπ.
- Υποστήριξη υπομνήματος. Σχετικό υπόμνημα θα πρέπει να εξηγεί την αντιστοίχιση των γραφικών αναπαραστάσεων με τα αντικείμενα.
- Υποστήριξη τοποθέτησης ετικετών. Θα πρέπει να είναι εφικτή η εμφάνιση τιμών διαφόρων ιδιοτήτων ως ετικέτες στα γραφικά αποτελέσματα των ερωτημάτων.
- Υποστήριξη επιλογής κλίμακας. Θα πρέπει να είναι εφικτή η επιλογή διαφόρων κλιμάκων και μάλιστα καλό θα ήταν να αλλάζει το σύμβολο των αποτελεσμάτων σε διάφορα ερωτήματα ανάλογα με την κλίμακα. Επίσης καλό θα ήταν να σε μικρές κλίμακες να εξαφανίζονται από τη γραφική αναπαράσταση διάφορα αποτελέσματα (χαρτογραφική γενίκευση).
- Υποστήριξη υποπεριοχών για εκτέλεση ερωτημάτων. Καλό θα ήταν να είναι εφικτός ο περιορισμός της περιοχής στην οποία θα εκτελεστεί μία ομάδα από επόμενα χωρικά ερωτήματα χωρίς να χρειάζεται κάθε φορά ο καθορισμός του (να μπορεί να εισάγεται αυτόματα ως επιπρόσθετος περιορισμός μέσα στα διάφορα επόμενα ερωτήματα μέχρι φυσικά την αφαίρεση αυτού του περιορισμού).

Οι απαιτήσεις αυτές μπορούν σε γενικές γραμμές να εκπληρωθούν με την προσθήκη εντολών κειμένου στη γλώσσα ερωταποκρίσεων ή μέσα στο σχεδιασμό της γραφικής διασύνδεσης με το χρήστη (Graphical User Interface – GUI).

Όσον αφορά στη χρήση στα ερωτήματα, τιμών χωρικών τύπων δεδομένων, θα πρέπει αυτοί οι τύποι δεδομένων να ενσωματωθούν στη γλώσσα ερωταποκρίσεων. Θα πρέπει πιο συγκεκριμένα να υπάρχει υποστήριξη για τα παρακάτω:

Α. Δήλωση τιμών χωρικών τύπων δεδομένων ως σταθερές σε ένα ερώτημα και δυνατότητα γραφικής εισαγωγής τέτοιων σταθερών τιμών δεδομένων.

Β. Χρήση-έκφραση των τεσσάρων κατηγοριών βασικών λειτουργιών των χωρικών τύπων δεδομένων (βλ. παρ. 2.1.2).

Γ. Περιγραφή της εμφάνισης των αποτελεσμάτων. (Πιθανώς η περιγραφή αυτή να μπορεί να ανατεθεί σε ξεχωριστό εργαλείο εμφάνισης αποτελεσμάτων.)

2.1.5 Εργαλεία για την υλοποίηση ενός χωρικού ΣΔΒΔ

Το βασικό πρόβλημα που απαιτεί λύση κατά την υλοποίηση ενός χωρικού ΣΔΒΔ είναι η υλοποίηση χωρικής άλγεβρας με τέτοιον τρόπο ώστε να μπορεί να ενσωματωθεί στην επεξεργασία ερωτημάτων του ΣΔΒΔ. Αυτό σημαίνει ότι κατ' αρχήν θα πρέπει να παρέχουμε αναπαραστάσεις για τους τύπους της άλγεβρας καθώς και αλγόριθμους/διαδικασίες για τις λειτουργίες αυτών. Εντούτοις, δεν αρκεί απλά η αποτελεσματική υλοποίηση ατομικών λειτουργιών, όπως ο έλεγχος εάν δύο περιοχές τέμνονται (intersect). Είναι αναγκαίο επίσης να μελετηθεί-αναλυθεί η χρήση τέτοιων κατηγορημάτων (predicates) στην επεξεργασία ερωτημάτων προσανατολισμένων σε ομάδες αντικειμένων, στις περιπτώσεις δηλαδή που υπάρχει χωρική επιλογή ή χωρική σύνδεση. Σε αυτό το σημείο εισέρχονται στη διαδικασία οι μέθοδοι χωρικής πρόσβασης (spatial access methods) και οι αλγόριθμοι χωρικής σύνδεσης (spatial join algorithms). Άλλες ομάδες λειτουργιών μιας χωρικής άλγεβρας απαιτούν τις δικές τους υλοποιήσεις. Απαραίτητη τέλος είναι η ανάπτυξη χωρικών ευρετηρίων για τη χωρική επιλογή.

2.1.6 Υλοποιημένα ΣΔΧΒΔ και εργαλεία οπτικοποίησης αποτελεσμάτων

Τα πιο γνωστά εμπορικά Συστήματα Διαχείρισης Βάσεων Δεδομένων διαθέτουν ειδικές εκδόσεις που υποστηρίζουν χωρικές βάσεις δεδομένων. Παραδείγματα αποτελούν τα IBM DB2 Spatial Extender, Oracle Spatial και MS-SQL Server. Αντίστοιχα στο χώρο του ελεύθερου λογισμικού, τα πιο διαδεδομένα συστήματα υποστηρίζουν πλέον διαχείριση χωρικών πληροφοριών στις κανονικές τους εκδόσεις. Αυτά είναι τα MySQL και PostgreSQL (μέσω του PostGIS).

Η οπτικοποίηση των αποτελεσμάτων γίνεται στις περισσότερες περιπτώσεις με ξεχωριστά εργαλεία τα οποία έχουν σχεδιαστεί για αυτόν το σκοπό και έχουν τη δυνατότητα διασύνδεσης-επικοινωνίας με υποσύνολο των παραπάνω συστημάτων. Παραδείγματα προγραμμάτων οπτικοποίησης αποτελούν τα Oracle MapViewer, uDIG, ESRI ArcExplorer, MapServer, GeoServer και Spatial Visualizer. Το Oracle MapViewer έχει σχεδιαστεί από την Oracle και επικοινωνεί με της Χωρικές ΒΔ αυτής. Το uDIG είναι εργαλείο οπτικοποίησης ανοιχτού κώδικα και έχει δοκιμαστεί με επιτυχία η απεικόνιση ΧΒΔ της Oracle Spatial και της IBM DB2 Spatial Extender. Το ESRI ArcExplorer διατίθεται δωρεάν από την ESRI και οπτικοποιεί με επιτυχία εκτός των άλλων, χωρικές βάσεις δεδομένων κατασκευασμένες στο σύστημα IBM DB2 Spatial Extender. Το MapServer είναι και αυτό ανοιχτού κώδικα και έχουν οπτικοποιηθεί με επιτυχία ΧΒΔ κατασκευασμένες σε PostGIS και MySQL. Το GeoServer είναι και αυτό ανοιχτού κώδικα και έχουν οπτικοποιηθεί με επιτυχία ΧΒΔ κατασκευασμένες σε PostGIS. Τέλος, το Spatial Visualizer οπτικοποιεί ΧΒΔ κατασκευασμένες στο σύστημα MS-SQL Server 2008.

Η οπτικοποίηση των αποτελεσμάτων, όπως θα δούμε σε επόμενο κεφάλαιο μπορεί να γίνει και σε περιβάλλον Google Earth αλλά και σε περιβάλλον GoogleMaps σε ιστοσελίδες.

2.2 Η PostgreSQL

Η PostgreSQL είναι ένα Αντικειμενο-Σχεσιακό Σύστημα Διαχείρισης Βάσεων Δεδομένων βασισμένο στην POSTGRES, έκδοση 4.2 που δημιουργήθηκε στο University of California at Berkeley Computer Science Department. Η POSTGRES πρωτοπόρησε σε πολλούς τομείς σχεδιασμού και διαχείρισης ΒΔ που μετά από αρκετό καιρό έγιναν διαθέσιμα σε ορισμένα εμπορικά ΣΔΒΔ.

Η PostgreSQL είναι ένας ανοιχτού κώδικα απόγονος του πρωτότυπου αυτού συστήματος που αναπτύχθηκε στο Berkeley. Υποστηρίζει μεγάλα τμήματα των SQL standards και προσφέρει πολλά νέα-μοντέρνα χαρακτηριστικά, όπως σύνθετα ερωτήματα, ξένα κλειδιά, σκανδαλιστές (triggers), όψεις (views), ακεραιότητα δοσοληψιών (transactional integrity), έλεγχος στην ταυτόχρονη χρήση πολλαπλών εκδόσεων (multiversion concurrency control), χωρικούς τύπους δεδομένων, χωρικά ευρετήρια, εκτέλεση χωρικών ερωτημάτων κ.α. Λόγω της ελεύθερης άδειας χρήσης για οποιονδήποτε σκοπό, μπορεί να χρησιμοποιηθεί, τροποποιηθεί, διανεμηθεί από οποιονδήποτε σε οποιονδήποτε δωρεάν καθώς και να χρησιμοποιηθεί ελεύθερα για ιδιωτικούς, εμπορικούς ή ακαδημαϊκούς σκοπούς.

Επίσης, η PostgreSQL μπορεί να επεκταθεί από τον χρήστη με πολλούς τρόπους, προσθέτοντας για παράδειγμα νέους τύπους δεδομένων, νέες συναρτήσεις, νέες λειτουργίες, συναθροιστικές συναρτήσεις, μεθόδους δημιουργίας ευρετηρίων και διαδικαστικές γλώσσες (procedural languages).

Καθώς η PostgreSQL διατίθεται ελεύθερα για την ανάπτυξη και άλλων λογισμικών, χρησιμοποιήθηκε για την ανάπτυξη του περιβάλλοντος διασύνδεσης με το χρήστη του Συστήματος Διαχείρισης Ρευμάτων Δεδομένων TelegraphCQ στο οποίο θα αναφερθούμε στο επόμενο κεφάλαιο.

Κεφάλαιο 3

Επισκόπηση ρευμάτων δεδομένων και συνεχών ερωτημάτων – TelegraphCQ

Επισκόπηση ρευμάτων δεδομένων και συνεχών ερωτημάτων

3.1.1 Βάσεις Δεδομένων και ΣΔΒΔ

Βάση δεδομένων (database) είναι μία συλλογή από σχετιζόμενα δεδομένα. Με τον όρο δεδομένα εννοούμε γνωστά γεγονότα-πληροφορίες που μπορούν να καταγραφούν και που έχουν κάποια υπονοούμενη σημασία.

Τα δεδομένα που επεξεργάζεται αυτή και αποθηκεύονται για μία εφαρμογή αφορούν ένα τμήμα του πραγματικού κόσμου, τον μικρόκοσμο. Στον μικρόκοσμο εντοπίζονται οι οντότητες (entities) ή αντικείμενα (objects) και τα χαρακτηριστικά τους γνωρίσματα (attributes).

Σύστημα Διαχείρισης Βάσεων Δεδομένων (Σ.Δ.Β.Δ.) (DataBase Management System - DBMS) είναι μια συλλογή από προγράμματα που επιτρέπουν στους χρήστες να δημιουργήσουν και να συντηρήσουν μία ή περισσότερες βάσεις δεδομένων. (Πηγή http://kallithea.hua.gr/tm_geo/courses/content/efarm_plirof/db_intro/).

Οι Βάσεις Δεδομένων χρησιμοποιούνται από τον πρώτο καιρό εμφάνισης των αρχικών υπολογιστών που χρησιμοποίησε ο άνθρωπος. Τα αρχικά ΣΔΒΔ υπήρχαν μόνο σε μεγάλους οργανισμούς που χρησιμοποιούσαν τους Η/Υ για την υποστήριξη του μεγάλου όγκου πληροφοριών που διέθεταν. Πλέον βάσεις δεδομένων υπάρχουν σε όλες σχεδόν τις επιχειρήσεις παγκοσμίως.

Τα πιο γνωστά εμπορικά συστήματα διαχείρισης βάσεων δεδομένων είναι τα Oracle, Microsoft Access, IBD BD2 και FileMaker. Τα πιο γνωστά Σ.Δ.Β.Δ. ανοιχτού λογισμικού (open source) είναι τα PostgreSQL και MySQL και λιγότερο γνωστά είναι τα Firebird, Ingres και MaxDB.

3.1.2 Ρεύματα Δεδομένων, Συνεχή Ερωτήματα και ΣΔΡΔ

Εδώ και αρκετά χρόνια έχει προκύψει η ανάγκη διαχείρισης εκτός των απλών στατικών σχεδόν πληροφοριών, και συνεχώς μεταβαλλόμενων σε συνάρτηση με το χρόνο δεδομένων (ρεύματα δεδομένων ή ροές δεδομένων). Τα κλασικά ΣΔΒΔ δεν έχουν σχεδιαστεί για διαχείριση τέτοιου είδους πληροφοριών και για την κάλυψη αυτής της ανάγκης αναπτύχθηκαν τα επονομαζόμενα Συστήματα Διαχείρισης Ρευμάτων Δεδομένων (Σ.Δ.Ρ.Δ.) καθώς τα δεδομένα εισέρχονται στο σύστημα υπό μορφή ρευμάτων. Τα ρεύματα δεδομένων μπορεί να ποικίλλουν σε συχνότητα αποστολής των πληροφοριών από 0 πλειάδες-καταχωρήσεις (tuples) το δευτερόλεπτο έως και αρκετές χιλιάδες ή και εκατομμύρια το δευτερόλεπτο (ανάλογα πλέον με τις επεξεργαστικές δυνατότητες του υπολογιστικού συστήματος στο οποίο ανατίθεται η διαχείριση των ρευμάτων δεδομένων και το είδος της πληροφορίας που καλούνται να διαχειριστούν).

Στις περιπτώσεις όπου οι πληροφορίες εισέρχονται στο σύστημα υπό μορφή ρευμάτων, δεν ενδιαφέρει τόσο η απάντηση σε ένα ερώτημα που θα γίνει κάποια συγκεκριμένη χρονική στιγμή, αλλά τα αποτελέσματα της συνεχούς εκτέλεσής του. Το ερώτημα ενδιαφέρει να είναι ενημερωμένο κάθε φορά με τις νέες πληροφορίες που εισέρχονται στο σύστημα και να επιστρέφει τις πιο πρόσφατες απαντήσεις στο χρήστη. Τα ερωτήματα στις περιπτώσεις των ρευμάτων δεδομένων είναι λοιπόν συνεχή και έτσι ονομάζονται.

Ένα ρεύμα δεδομένων συνίσταται από πληροφορίες υπό μορφή πλειάδων-καταχωρήσεων οι οποίες δημιουργούνται διαρκώς και εισρέουν σε ένα σύστημα. Οι πληροφορίες αυτές συνήθως χρήζουν άμεσης διαχείρισης και επεξεργασίας, ώστε να επιτυγχάνεται σε πραγματικό χρόνο η εξαγωγή της ζητούμενης κάθε φορά πληροφορίας.

Η μεγάλη εξάπλωση των δικτύων δημιούργησε και την ανάγκη ύπαρξης εφαρμογών παρακολούθησης των διακινούμενων πληροφοριών οι οποίες έχουν τα χαρακτηριστικά ρευμάτων δεδομένων.

Οι πιο συνήθεις εφαρμογές παρακολούθησης είναι οι εξής:

- Ασφάλεια στο διαδίκτυο. Πρόκειται για εφαρμογές που ελέγχουν τις διακινούμενες πληροφορίες.
- Σε χρηματιστήρια. Είναι αναγκαία η παρακολούθηση των τιμών των διαφόρων δεικτών σε πραγματικό χρόνο.
- Λειτουργία αισθητήρων. Οι αισθητήρες αποστέλλουν τις πληροφορίες που καταγράφουν με τη μορφή ρευμάτων δεδομένων σε συγκεκριμένο σύστημα που θα τις επεξεργαστεί. Πρόκειται ουσιαστικά για άλλη μία περίπτωση παρακολούθησης.
- Συναλλαγές με πιστωτικές κάρτες. Είναι αναγκαία η παρακολούθησή τους η οποία γίνεται όπως και στην περίπτωση των αισθητήρων με τη μορφή ρευμάτων δεδομένων που αποστέλλουν τα διαφόρων ειδών συστήματα προς συγκεκριμένα τερματικά.
- Στρατιωτικές εφαρμογές με συστήματα παρακολούθησης πολλών διαφορετικών κατηγοριών πληροφοριών.

Στην περίπτωση των ρευμάτων δεδομένων έναντι των βάσεων δεδομένων, δεν είναι γνωστός ο συνολικός όγκος της πληροφορίας ο οποίος μπορεί να θεωρηθεί άπειρος καθώς το αντίστοιχο σύστημα δύναται να λειτουργεί συνεχώς και να εισρέουν διαρκώς και επ' αόριστον πληροφορίες σε αυτό. Επίσης στα Συστήματα Διαχείρισης Ρευμάτων Δεδομένων συνήθως δεν ενδιαφέρουν τόσο οι πληροφορίες που εισήλθαν στο παρελθόν όσο οι πιο πρόσφατες. Οι παραπάνω διαφορές με τα ΣΔΒΔ οδήγησαν στην περίπτωση των ΣΔΡΔ στον ορισμό της έννοιας των παραθύρων. Τα παράθυρα περιορίζουν στο επιθυμητό το πλήθος των πλειάδων-καταχωρήσεων (tuples) τα οποία θα επεξεργαστεί ένα ερώτημα για την εξαγωγή των απαντήσεων.

Τα παράθυρα μπορούν να διαχωριστούν σε τρεις κατηγορίες ανάλογα με τον τρόπο καθορισμού των πλειάδων που θα τα αποτελούν σε κάθε ερώτημα.

Πρώτη και πιο συνήθης κατηγορία είναι τα χρονικά παράθυρα. Πρέπει να αναφερθεί ότι στην περίπτωση των ρευμάτων δεδομένων είναι απαραίτητη η ύπαρξη της παραμέτρου του χρόνου με κάποια μορφή μέσα στις εισερχόμενες πλειάδες-καταχωρήσεις ώστε να διαχωρίζονται οι πιο πρόσφατες από τις παλαιότερες. Εάν δεν υπάρχει η παράμετρος του χρόνου, το ΣΔΡΔ μπορεί να την προσθέσει ως τη στιγμή άφιξης της κάθε πλειάδας. Στην περίπτωση λοιπόν των χρονικών παραθύρων, οι πλειάδες τις οποίες θα επεξεργαστεί ανά πάσα στιγμή το ΣΔΡΔ για την απάντηση του συγκεκριμένου ερωτήματος, ορίζονται με βάση το χρονικό ορόσημο κάθε πλειάδας. Για παράδειγμα, ο χρήστης δύναται να ζητήσει σε ένα ερώτημα την επεξεργασία των πλειάδων που έχουν ληφθεί τα τελευταία 30 δευτερόλεπτα ανά πάσα στιγμή. Αυτές φυσικά δύνανται να είναι μεταβαλλόμενου πλήθους καθώς ο ρυθμός αύξησης στο σύστημα των διαφόρων πλειάδων μπορεί κάλλιστα να διαφέρει από τη μία στιγμή στην άλλη. Η εκτέλεση λοιπόν του ερωτήματος γίνεται ανά πάσα στιγμή στις πλειάδες που τηρούν τις προϋποθέσεις που θέτει το αντίστοιχο παράθυρο.

Δεύτερη κατηγορία είναι τα παράθυρα συγκεκριμένου πλήθους πλειάδων. Στην περίπτωση αυτή όπως είναι φανερό, το ερώτημα επεξεργάζεται συγκεκριμένο πλήθος πλειάδων-καταχωρήσεων ανεξαρτήτως της χρονικής στιγμής που αυτές εισήλθαν στο σύστημα. Αυτή είναι και η διαφορά των παραθύρων αυτών με τα χρονικά παράθυρα. Ένα παράδειγμα συγκεκριμένου πλήθους πλειάδων θα ήταν το εξής: Επιλογή των τελευταίων 100 πλειάδων για επεξεργασία στο ερώτημα.

Η τρίτη κατηγορία είναι συνδυασμός των δύο προηγουμένων. Αυτό σημαίνει ότι τίθεται περιορισμός και όσον αφορά τη χρονική στιγμή άφιξης της πληροφορίας και όσον αφορά το πλήθος των υπό επεξεργασία πλειάδων. Ένα παράδειγμα αυτής της κατηγορίας θα έλεγε το εξής: Επιλογή για επεξεργασία των πλειάδων που εισήλθαν τα τελευταία 5 λεπτά και δεν ξεπερνούν σε πλήθος τις 200 καταχωρήσεις. Οποιοσδήποτε από τους δύο περιορισμούς εμφανιστεί πρώτος, θα σταματήσει και την επιλογή των πλειάδων.

Τα χρονικά παράθυρα μπορούν να διαχωριστούν σε δύο κατηγορίες, ανάλογα με τη χρονική στιγμή έναυσης αποδοχής των πλειάδων προς επεξεργασία. Η συνήθης περίπτωση είναι αυτή των κυλιόμενων παραθύρων, όπου η χρονική στιγμή είναι μεταβαλλόμενη και ακολουθεί την χρονική στιγμή επεξεργασίας μειωμένη κατά μία συγκεκριμένη σταθερή ποσότητα χρόνου (δηλαδή τόσα δευτερόλεπτα ή τόσα λεπτά πριν μέχρι το τώρα). Η δεύτερη περίπτωση είναι αυτή του χρονικού ορόσημου. Στην περίπτωση αυτή, η χρονική στιγμή έναρξης αποδοχής των πλειάδων παραμένει σταθερή ενώ εκτελείται το ερώτημα και προφανώς όσο περνάει ο χρόνος και εισέρχονται νέα δεδομένα προς επεξεργασία, αυξάνονται οι πλειάδες που θα επεξεργάζεται το ερώτημα αυτό.

Τα συνεχή ερωτήματα μπορούν να διαχωριστούν σε τέσσερις κατηγορίες ανάλογα με το είδος του χρονικού παραθύρου που χρησιμοποιούν ως εξής:

- Ερωτήματα κυλιόμενου παραθύρου (sliding window queries). Πρόκειται για την πιο συνήθη περίπτωση συνεχών ερωτημάτων. Τα παράθυρα στην περίπτωση αυτή όπως προαναφέρθηκε, έχουν μεταβαλλόμενη αρχή και τέλος. Δηλαδή, τα όριά επιλογής των πλειάδων προς επεξεργασία ακολουθούν την παρούσα χρονική στιγμή εκτέλεσης του ερωτήματος έχοντας συγκεκριμένο χρονικό εύρος. Η ανανέωση των ορίων του παραθύρου μπορεί να είναι ομαλή ή να μεταπηδά σε επόμενη θέση μετά από συγκεκριμένα χρονικά διαστήματα. Όταν η μεταπήδηση σε επόμενη θέση γίνεται με χρονική διαφορά όσο το εύρος του παραθύρου τότε έχουμε ένα παράθυρο τύπου Jumping (Άλτης). Π.χ. Εύρος παραθύρου 50 sec και αλλαγή θέσης κάθε 50 sec (slide by 50sec). Όταν η μεταπήδηση σε επόμενη θέση γίνεται με χρονική διαφορά μεγαλύτερη από τη διακριτή μονάδα χρόνου

αλλά μικρότερη από το εύρος του παραθύρου τότε έχουμε ένα παράθυρο τύπου Tumbling (Ακροβάτης). Π.χ. Εύρος παραθύρου 50 sec και αλλαγή θέσης κάθε 50 sec (slide by 15sec). Δίδεται επίσης η δυνατότητα το χρονικό παράθυρο αντί να ακολουθεί τη φορά του χρόνου, να κινείται αντίθετα πηγαίνοντας ολοένα και πιο πίσω στο χρόνο με το συγκεκριμένο εύρος που έχει οριστεί.

- Ερωτήματα στιγμιοτύπου (one-time ή snapshot ή fixed queries). Τα ερωτήματα αυτά εκτελούνται μόνο μία φορά μόλις υποβληθούν στο σύστημα. Η δε εκτέλεσή τους γίνεται στις πλειάδες (tuples) που ορίζει το σχετικό χρονικό παράθυρο. Διαθέτουν χρονικό παράθυρο με συγκεκριμένες χρονικές στιγμές έναρξης και λήξης που βρίσκονται και οι δύο στο παρελθόν. Παράδειγμα: Επιλογή αντικειμένων που πληρούν τις τάδε προϋποθέσεις από τη χρονική στιγμή t_1 έως τη χρονική στιγμή t_2.
- Ερωτήματα χρονικού ορισήμου (landmark queries). Πρόκειται για συνδυασμό των δύο προηγουμένων και όπως προαναφέρθηκε, στην περίπτωση αυτή, παραμένει σταθερή στο χρόνο η θέση έναρξης του παραθύρου ενώ μεταβάλλεται η θέση λήξης αυτού η οποία ακολουθεί κάθε φορά τη χρονική στιγμή εκτέλεσης του ερωτήματος. Παράδειγμα. Συνάθροιση αντικειμένων που πληρούν τις τάδε προϋποθέσεις από τη χρονική στιγμή t_1 έως την τρέχουσα χρονική στιγμή.
- Ερωτήματα χρονικής σύνδεσης ζώνης (temporal band-join queries). Στην περίπτωση αυτή, γίνεται σύνδεση πλειάδων από πολλαπλά ρεύματα βάσει των χρονικών τους ορισήμων. Συγκεκριμένα χρονικά παράθυρα ορίζουν τις χρονικές ζώνες (bands) στις οποίες θα αναζητηθούν οι καταχωρημένες πλειάδες προς επεξεργασία. Συνήθως τα χρονικά παράθυρα είναι τα ίδια στα διάφορα ρεύματα που συνδέονται. Παράδειγμα: Ποια αντικείμενα του ρεύματος α πληρούν τις τάδε προϋποθέσεις συγκρινόμενα με τα αντικείμενα του ρεύματος β κάθε λεπτό.

3.1.3 Τα διάφορα Συστήματα Διαχείρισης Ρευμάτων Δεδομένων της αγοράς

Τα πιο γνωστά υλοποιημένα συστήματα διαχείρισης ρευμάτων δεδομένων είναι τα Aurora, TelegraphCQ και STREAM. Άλλες σχετικές προτάσεις ΣΔΡΔ αποτελούν τα Gigascope, OpenCQ, NiagaraCQ, Tribeca, Tapestry, Chronicle, Alert, Cougar, Hancock, Xfilter και Xyleme.

Το Aurora έχει σχεδιαστεί για την παρακολούθηση πληροφοριών αισθητήρων και μπορεί να δεχτεί εισερχόμενες πληροφορίες μόνο υπό τη μορφή ρευμάτων, δεν υποστηρίζει δηλαδή τη δημιουργία πινάκων με στατικά δεδομένα. Έχει δημιουργηθεί με σύμπραξη των Πανεπιστημίων M.I.T., Brandeis University και Brown University. Πρόκειται για ένα Σύστημα Διαχείρισης Ρευμάτων Δεδομένων γενικού σκοπού όπου η επεξεργασία των ρευμάτων γίνεται με μία προσέγγιση βασισμένη σε γραφικό περιβάλλον με 'κουτιά' και 'βέλη' (boxes and arrows). Ο χρονοπρογραμματιστής (scheduler) αποτελεί τον πυρήνα της αρχιτεκτονικής της λειτουργίας αυτού του συστήματος. Το Aurora διαθέτει σύστημα εποπτείας της ποιότητας των αποτελεσμάτων που εξάγει (Quality of Service Monitor), ενημερώνοντας τον χρονοπρογραμματιστή σχετικά με τις επιδόσεις του συστήματος. Διαθέτει επίσης μέθοδο αποβολής δεδομένων (load shedding) για τις περιπτώσεις υπερφόρτωσης του συστήματος. Το Aurora τέλος φροντίζει για τη βελτιστοποίηση της εκτέλεσης των διαφόρων συνεχών ερωτημάτων, συνεκτιμώντας την ποιότητα των αποτελεσμάτων, τον χρόνο εκτέλεσής τους και το ποσοστό των αποβαλλόμενων καταχωρήσεων. Υποστηρίζει τα παράθυρα στιγμιοτύπου, χρονικού ορόσημου και τα κυλιόμενα. Υποστηρίζει επίσης παράθυρα σταθερού πλήθους πλειάδων.

Το STREAM είναι ένα Σύστημα Διαχείρισης Ρευμάτων Δεδομένων γενικού σκοπού που έχει αναπτυθχεί από το 2001 στο Πανεπιστήμιο Stanford. Σχεδιάστηκε ως

πρωτότυπο ΣΔΡΔ χωρίς να γίνει προσαρμογή-τροποποίηση κάποιου υπάρχοντος ΣΔΒΔ. Τα ερωτήματα υποβάλλονται με χρήση της γλώσσας ερωταποκρίσεων CQL (Continuous Query Language) η οποία αποτελεί υπερσύνολο της SQL. Στον πυρήνα της αρχιτεκτονικής του βρίσκεται ένας καθολικός χρονοπρογραμματιστής (global scheduler). Υποστηρίζει τη διαχείριση ρευμάτων δεδομένων (data streams) αλλά και απλών πινάκων (relations) βάσεων δεδομένων. Από χρονικά παράθυρα, υποστηρίζει μόνο τα κυλιόμενα και όχι αυτά με σταθερή αρχή (χρονικού ορόσημου) ή/και σταθερού τέλους (στιγμιοτύπου), ούτε και τα παράθυρα χρονικής σύνδεσης ζώνης. Υποστηρίζει επίσης παράθυρα σταθερού πλήθους πλειάδων. Τέλος, διαθέτει γραφικό περιβάλλον επικοινωνίας με τον χρήστη.

3.2 Το TelegraphCQ

Το TelegraphCQ προέρχεται από το πρωτότυπο πρόγραμμα (project) Telegraph. Έχει ξεκινήσει η δημιουργία του στο Πανεπιστήμιο Berkeley το 2000 ενώ σταμάτησε το 2006 όταν αποφοίτησε και ο τελευταίος φοιτητής που ασχολήθηκε με αυτό. Πρόκειται για ένα Σύστημα Διαχείρισης Ρευμάτων Δεδομένων με έμφαση στη διαχείριση πληροφοριών δικτύων αισθητήρων το οποίο όμως έχει στόχο να γίνει γενικού σκοπού. Στο TelegraphCQ θεωρείται ότι τα ερωτήματα διαρκείας μπορούν να μεταβάλλονται με την πάροδο του χρόνου και μπορούν να παρομοιαστούν με ρεύματα, καθώς αλλάζει τόσο το πλήθος τους όσο και η δομή τους όπως μπορεί να συμβεί και με την περίπτωση των ρευμάτων.

Το TelegraphCQ έχει επιλεγεί ως το Σύστημα Διαχείρισης των Ρευμάτων Δεδομένων στην παρούσα εργασία. Οι λόγοι προτίμησής του έναντι άλλων πιθανών Συστημάτων Διαχείρισης Ρευμάτων Δεδομένων, όπως το Aurora και το STREAM είναι οι εξής:

Καθώς έχει αναπτυχθεί ως ειδική έκδοση της PostgreSQL η οποία υποστηρίζει χωρικές βάσεις δεδομένων, το TelegraphCQ υποστηρίζει και αυτό με την σειρά του τους αναγκαίους χωρικούς τύπους και τα χωρικά ερωτήματα της εφαρμογής. Έτσι λοιπόν τα ρεύματα χωρικών δεδομένων της παρούσας εφαρμογής υποστηρίζονται πλήρως από το TelegraphCQ και συνεπώς είναι δυνατή η παρακολούθηση των κινούμενων αντικειμένων σε πραγματικό χρόνο.

Επιπλέον, η γλώσσα ερωταποκρίσεων που χρησιμοποιεί αυτό το ΣΔΡΔ αποτελεί μια επέκταση της γνωστής SQL που χρησιμοποιεί η PostgreSQL για συνεχή χωρικά ερωτήματα ώστε να είναι εύκολη και απλή η υποβολή τους στο σύστημα.

Τέλος, το TelegraphCQ έχει σχεδιαστεί με έμφαση στην ευελιξία και στην προσαρμοστικότητα όπως θα δούμε παρακάτω, χαρακτηριστικά ιδιαιτέρως χρήσιμα για την παρούσα εφαρμογή.

Η μεταβλητότητα των εισερχόμενων δεδομένων και των αντίστοιχων ερωτημάτων οδήγησαν την προσέγγιση που ακολουθεί το TelegraphCQ στην προσαρμοστικότητα των τελεστών (adaptivity of operators). Έχει δοθεί ειδική έμφαση λοιπόν στην γρήγορη προσαρμογή του συστήματος σε αλλαγές που αναφέρονται στη διαθεσιμότητα δεδομένων, στο περιεχόμενό τους, στα χαρακτηριστικά του συστήματος, στις πηγές τροφοδοσίας πληροφοριών και τέλος στα υποβληθέντα ερωτήματα.

Όπως προαναφέρθηκε, η ανάπτυξη του TelegraphCQ έγινε επάνω στην PostgreSQL, ώστε να είναι δυνατή η υποβολή και εκτέλεση ερωτημάτων διαρκείας πάνω σε ρεύματα δεδομένων σε ένα γνωστό και ευρέως διαδεδομένο ΣΔΒΔ.

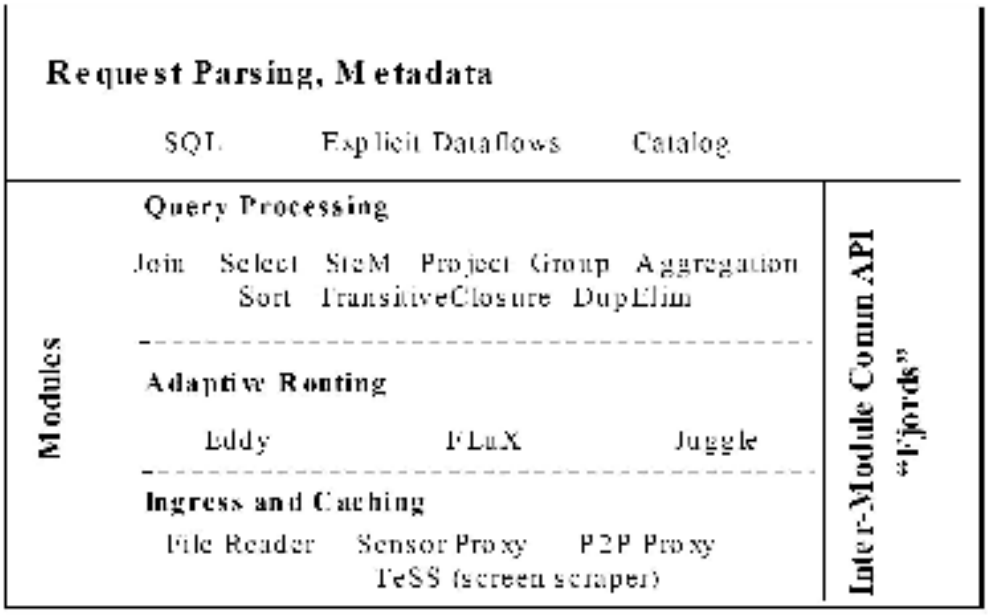
Για την δημιουργία του TelegraphCQ έχουν τροποποιηθεί αρκετά τμήματα του κώδικα της PostgreSQL, άλλα λιγότερο, άλλα περισσότερο. Το TelegraphCQ χρησιμοποιεί τη γλώσσα ερωταποκρίσεων StreaQuel η οποία μοιάζει με την SQL. Η σύνταξη των ερωτημάτων για παράδειγμα έχει επεκταθεί με τη χρήση παραθύρων επί

των ρευμάτων δεδομένων και φυσικά έχουν προστεθεί οι αναγκαίες οντότητες για την υποστήριξη των ρευμάτων.

Οι κύρια προσαρμοστικότητα δημιουργείται από το υποτμήμα (module) που ονομάζεται Eddy. Το υποτμήμα αυτό λαμβάνει και στέλνει πλειάδες που επεξεργάζονται από διαφορετικούς χειριστές (operators). Αυτό θέτει μια εναλλακτική προσέγγιση στο δένδρο στατικών ερωτημάτων που χρησιμοποιείται από άλλα ΣΔΡΔ. Η τεχνική χρήσης παραθύρων του TelegraphCQ κάνει εφικτή τη χρήση κυλιόμενων, τύπου Jumping και τύπου Tumbling παραθύρων. Σε αντίθεση με το STREAM, το TelegraphCQ διαχειρίζεται μόνο έναν τύπο ρευμάτων που δημιουργούνται με την πρόταση CREATE STREAM. Ωστόσο, το TelegraphCQ παρέχει τη δυνατότητα αποθήκευσης του κάθε ρεύματος σε στατικούς πίνακες ΣΔΒΔ στο δίσκο ώστε να μπορούν τα αποθηκευμένα δεδομένα να χρησιμοποιηθούν αργότερα για απάντηση διαφόρων ερωτημάτων.

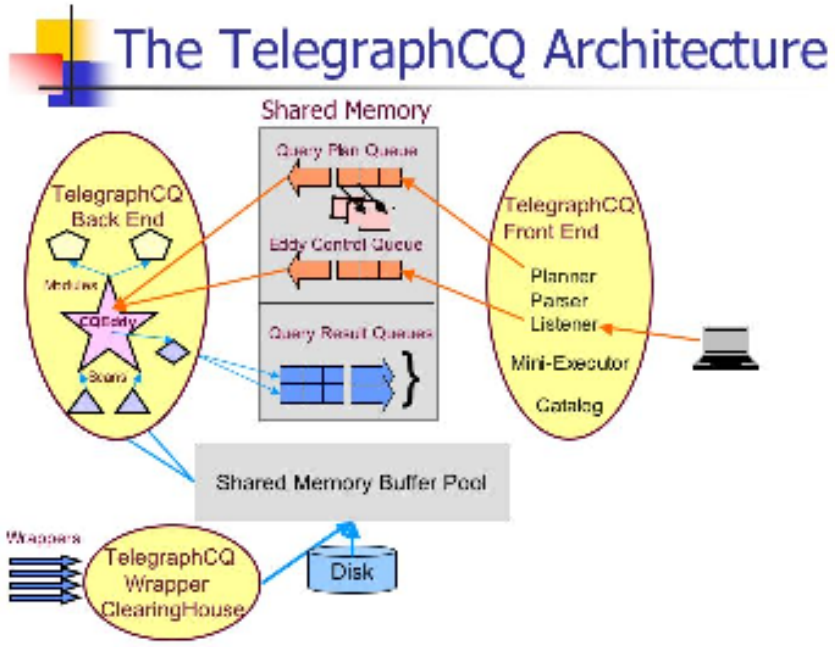
3.2.1 Επισκόπηση αρχιτεκτονικής λειτουργίας του TelegraphCQ

Όπως φαίνεται στην εικόνα 3.1 το Telegraph και κατ' επέκταση και το TelegraphCQ, βασίζεται σε υπομτήματα (modules) που συνδέονται μεταξύ τους χρησιμοποιώντας διασυνδέσεις (interface) ονομαζόμενα ως Fiords. Τα υποτμήματα χωρίζονται σε τρία τμήματα, *είσοδος και διατήρηση* (*ingress and caching*), *προσαρμοστική δρομολόγηση* (*adaptive routing*) και *επεξεργασία ερωτημάτων* (*query processing*). Τα υποτμήματα εισόδου και διατήρησης εστιάζονται στην περισυλλογή και μετατροπή των εισερχόμενων ρευμάτων δεδομένων σε πλειάδες που δέχεται το σύστημα. Οι πλειάδες τότε δρομολογούνται από τα υποτμήματα της *προσαρμοστικής δρομολόγησης* τα οποία βελτιστοποιούν την εκτέλεση κάθε ερωτήματος. Τα υποτμήματα της *προσαρμοστικής δρομολόγησης* στη συνέχεια στέλνουν τις πλειάδες στα υποτμήματα *επεξεργασίας ερωτημάτων*. Οι διασυνδέσεις τύπου Fiords χρησιμοποιούνται για να εισάγουν σε ουρές τις πλειάδες (tuples) μεταξύ των χειριστών (operators) ώστε να αποφεύγεται το κώλυμα (μπλοκάρισμα) των τελευταίων. Η αποφυγή των κωλυμάτων γίνεται χρησιμοποιώντας ουρές οι οποίες είτε ωθούν (push) είτε έλκουν (pull) τα δεδομένα από τις πηγές τους.



Σχήμα 3.1 Αρχιτεκτονική του Telegraph (Πηγή: [CCD+03])

Η αρχιτεκτονική του TelegraphCQ απεικονίζεται στο σχήμα 3.2 παρακάτω.



Σχήμα 3.2 Η αρχιτεκτονική λειτουργίας του συστήματος TelegraphCQ (Πηγή: http://telegraph.cs.berkeley.edu/telegraphcq/v2.1/)

Οι ελλείψεις απεικονίζουν τις κύριες διεργασίες που όλες μαζί αποτελούν τον εξυπηρετητή (server) του συστήματος (Front End, Back End ή Executor, Wrapper). Προβλέπεται ειδικός χώρος κοινής μνήμης (shared memory) για την προσωρινή αποθήκευση των δεδομένων προς επεξεργασία. Η Front End ενεργοποιείται σε κάθε νέα σύνδεση κάποιου τερματικού με το σύστημα. Ο Ακροατής (Listener) δέχεται τις αιτήσεις για πολλαπλά ερωτήματα διαρκείας από κάθε τερματικό και ο Αναλυτής λέξεων (Parser) τα αναλύσει συντακτικά. Τα ερωτήματα βελτιστοποιούνται σε πρώτο στάδιο από τον Mini-Executor ή Optimizer προκειμένου να προκύψει ένα προσαρμοζόμενο προσχέδιο εκτέλεσης αυτών και ο Σχεδιαστής (Planner) τα εντάσσει στο σχέδιο εκτέλεσης ερωτημάτων το οποίο με τη σειρά του προσαρμόζεται συνεχώς στις νέες συνθήκες. Τα ερωτήματα τοποθετούνται σε μία λίστα-ουρά με πλάνα εκτέλεσης αυτών (Query Plan Queue), στην κοινή μνήμη του συστήματος από όπου οδεύουν για εκτέλεση από το Back End (Executor).

Οι Συλλέκτες (Wrappers) παραλαμβάνουν τα διάφορα ρεύματα δεδομένων και μετατρέπουν τις πλειάδες (tuples) σε τύπους δεδομένων που κατανοεί το TelegraphCQ. Όλοι οι Συλλέκτες ελέγχονται από τη διεργασία που ονομάζεται Wrapper Clearing House, η οποία δέχεται συνδέσεις από εξωτερικές πηγές και στέλνει τις διαμορφωμένες πλειάδες στην *ενδιάμεση ζώνη διαμοιραζόμενης μνήμης* (*Shared Memory Buffer Pool*) - στο εξής *ενδιάμεση μνήμη*.

Η ενδιάμεση μνήμη παρέχει επίσης διασύνδεση με τους διάφορους σκληρούς δίσκους του συστήματος. Κληρονομώντας τις δυνατότητες της PostgreSQL, το TelegraphCQ δίνει στο χρήστη τη δυνατότητα να συνδέει ρεύματα δεδομένων με σχεσιακούς πίνακες εάν ζητηθεί κάτι τέτοιο.

Το Back End του TelegraphCQ παραλαμβάνει τις πλειάδες από την ενδιάμεση μνήμη και τις αποστέλλει στο υποτμήμα Eddy, το οποίο είναι ένας χρονοπρογραμματιστής που δρομολογεί πλειάδες σε διαφορετικά υποτμήματα χειριστές (operator modules), χρησιμοποιώντας συγκεκριμένη μεθοδολογία δρομολόγησης. Αυτοί

οι χειριστές εκτελούν επιλογές (selectons) και συνδέσεις (joins) μεταξύ των πλειάδων. Όταν ένας χειριστής τελειώσει την εργασία του, επιστρέφει την πλειάδα στο Eddy, το οποίο αποφασίζει εάν η πλειάδα είναι έτοιμη για εξαγωγή (output) ή πρέπει να σταλεί και σε άλλον χειριστή. Το υποτμήμα Eddy είναι και αυτό ένας χειριστής και συνεπώς ένα Eddy μπορεί να στείλει μια πλειάδα σε ένα άλλο Eddy. Η αρχιτεκτονική των χειριστών τύπου Eddy, αρχικά υλοποιήθηκε για απλά ερωτήματα και ύστερα επεκτάθηκε στη διαχείριση συνεχώς προσαρμοζόμενων ερωτημάτων διαρκείας (continuously adaptive continuous queries – CACQ) στο σύστημα TelegraphCQ. Το CACQ εισάγεται με την έννοια των SteMs, τα οποία είναι υποτμήματα που κάνουν πιο ανεξάρτητες τις αποφάσεις σχετικά με συνδέσεις πλειάδων απ' ότι οι απλοί χειριστές σύνδεσης.

Εάν μία πλειάδα είναι έτοιμη για έξοδο, αποστέλλεται στις *ουρές αποτελεσμάτων ερωτημάτων* (*Query Result Queues*) στην *διαμοιραζόμενη μνήμη* (*Shared Memory*). Από εκεί παραλαμβάνεται από το τερματικό του χρήστη (client) το οποίο έχει υποβάλλει το σχετικό ερώτημα. Αυτό συμβαίνει στις διαδικασίες μεταξύ server και client που εκκινούν από τον Postmaster.

Όταν ένας χρήστης υποβάλλει ένα ερώτημα στο Front End του TelegraphCQ στο τερματικό του, το ερώτημα αναλύεται και μετασχηματίζεται σε ένα πλάνο ερωτήματος το οποίο αποστέλλεται στη διαμοιραζόμενη μνήμη και παραλαμβάνεται από το Eddy. Το Eddy με τη σειρά του ενσωματώνει το ερώτημα στη λειτουργία του με τα υποτμήματα χειριστές.

3.2.2 Επισκόπηση οντοτήτων του TelegraphCQ

Το TelegraphCQ διαθέτει τρία βασικά είδη οντοτήτων, ως εξής (βλ. σχετικά Εικ.3.1):

Οντότητες πρόσβασης και αποθήκευσης (Συλλέκτες). Αυτές χρησιμεύουν για την επικοινωνία του συστήματος με τις διάφορες εξωτερικές πηγές δεδομένων. Οι οντότητες αυτές ονομάζονται wrappers (Συλλέκτες) και δύνανται να χρησιμοποιηθούν είτε για την ανάγνωση τοπικών αρχείων του συστήματος όπου μπορεί να βρίσκονται οι πληροφορίες, είτε ως HTML/XML screen scrappers ("Tess"), είτε ως αντιπρόσωποι για τη μεταφορά δεδομένων διαμέσου δικτύων peer-to-peer ("TeleNap").

Οντότητες εκτέλεσης ερωτημάτων. Η επεξεργασία διαφόρων ερωτημάτων γίνεται με τη δρομολόγηση των πλειάδων μεταξύ των διαφόρων οντοτήτων επεξεργασίας. Αυτές είναι ουσιαστικά εκδόσεις των κλασικών σχεσιακών τελεστών όπως η επιλογή (SELECT), η προβολή (PROJECT) και η σύνδεση (JOIN) με τη διαφορά ότι είναι διασωληνωμένες (pipelined) και δεν ανακόπτουν την εκτέλεση του ερωτήματος (non-blocking). Στις παραπάνω συμπεριλαμβάνεται και η οντότητα επεξεργασίας SteM. Τα SteMs (State Modules) απλοποιούν και επεκτείνουν τη δύναμη των Eddies αλλά ταυτόχρονα αυξάνουν το ενδεχόμενο πολλαπλών ταυτόσημων αποτελεσμάτων και τη μη εμφάνιση άλλων που θα έπρεπε. Το υποτμήμα SteM είναι παρόμοιο με μια σύνδεση κερματισμού (hash join) και προσφέρει διπλά διασωληνωμένες συνδέσεις για τη μεταφορά των πλειάδων. Κάθε πηγή δεδομένων έχει ένα αντιστοιχισμένο SteM.

Οντότητες δυναμικής δρομολόγησης πλειάδων. Η διεργασία Back End (tcqbackend) παραλαμβάνει πλειάδες από την ενδιάμεση μνήμη. Διαθέτει αρκετά υποτμήματα που στοχεύουν στην υποστήριξη τουTelegraphCQ με τέτοιον τρόπο ώστε το ΣΔΡΔ να προσαρμόζεται στα χαρακτηριστικά των ρευμάτων. Στο σχήμα 3.1 φαίνονται τρία υποτμήματα προσαρμοστικής δρομολόγησης (Adaptive Routing). Εστιαζόμαστε στη περιγραφή του Eddy και του Flux καθώς αυτά έχουν υλοποιηθεί στο TelegraphCQ. Όπως προαναφέρθηκε, το Eddy χρησιμοποιείται για τη δυναμική προσαρμογή του συστήματος σε αλλαγές του ρεύματος δεδομένων. Τα Eddies είχαν σχεδιαστεί αρχικά για επεξεργασία απλών ερωτημάτων, αλλά επεκτάθηκαν με σκοπό τη διαχείριση πολλαπλών συνεχών ερωτημάτων στο TelegraphCQ. Το Flux σκοπό έχει να παρεμβάλλεται μεταξύ

οντοτήτων παραγωγών – καταναλωτών, ρυθμίζοντας τη ροή των δεδομένων και τη διαδικασία δρομολόγησης. Ουσιαστικά επεκτείνει την Back End, επιτρέποντας την εξισορρόπηση του καταμερισμού εργασίας (load balancing) και την ανοχή σε σφάλματα (fault tolerance).

Κεφάλαιο 4

Εργαλεία εφαρμογής

4.1 Συνοπτική Αναφορά

Για την υλοποίηση της παρούσας εφαρμογής χρησιμοποιήθηκαν πολλά και διαφορετικά μεταξύ τους εργαλεία. Εν συντομία τα εργαλεία αυτά είναι τα εξής:

1. Λειτουργικό Σύστημα Linux
2. Σύστημα Διαχείρισης Ρευμάτων Δεδομένων TelegraphCQ
3. Σύστημα Διαχείρισης Χωρικών Βάσεων Δεδομένων PostgreSQL
4. Δημιουργία ιστοσελίδων με την HTML (HyperText Markup Language)
5. Γλώσσα προγραμματισμού PHP (Personal Home Page)
6. Καταχώρηση αποτελεσμάτων ερωτημάτων σε XML (Extensible Markup Language)
7. Γλώσσα προγραμματισμού JavaScript
8. Εμφάνιση αποτελεσμάτων σε εφαρμογή που χρησιμοποιεί GoogleMaps API
9. Γλώσσα προγραμματισμού C++
10. Γλώσσα προγραμματισμού PERL

4.2 Λειτουργικό Σύστημα Linux

Το Σύστημα Διαχείρισης Ρευμάτων Δεδομένων TelegraphCQ έχει δοκιμαστεί μόνο σε περιβάλλον Linux και συγκεκριμένα σε Linux Fedora Core 1. Μέχρι σήμερα δεν υπάρχει έκδοση του TelegraphCQ που να μπορεί να λειτουργήσει σε Windows. Στην παρούσα εφαρμογή χρησιμοποιήθηκε το λειτουργικό Red Hat Enterprise Linux 5.1. Η

Red Hat είναι η δημιουργός και του Linux Fedora και συνεπώς δεν αναμένονταν δυσλειτουργίες και προβλήματα συμβατότητας με την έκδοση Enterprise Linux 5.1 τα οποία και δεν υπήρξαν.

Ως γνωστόν, τα λειτουργικά συστήματα της κατηγορίας Linux είναι σχεδόν στο σύνολό τους ελεύθερου κώδικα και ως εκ τούτου χρησιμοποιούνται συχνά για τη δημιουργία νέων πρωτότυπων εφαρμογών όπως η παρούσα διότι εκτός του παραπάνω στοιχείου, παρέχουν καλύτερη υποστήριξη στις γλώσσες προγραμματισμού και στα διάφορα εργαλεία ανάπτυξης εφαρμογών και είναι πιο σταθερά και πιο ασφαλή για διαδικτυακές εφαρμογές.

4.3 Σύστημα Διαχείρισης Ρευμάτων Δεδομένων TelegraphCQ

Πρόκειται για το σύστημα που χρησιμοποιήθηκε στην παρούσα εφαρμογή για την διαχείριση των κινούμενων αντικειμένων.

Ο κύριος λόγος που προτιμήθηκε έναντι άλλων είναι ότι καθώς έχει αναπτυχθεί ως μία ειδική έκδοση της PostgreSQL υποστηρίζει εγγενώς χωρικά δεδομένα απαραίτητα για την παρούσα εφαρμογή.

Εκτενής αναφορά σε αυτό γίνεται στο Κεφάλαιο 3.

4.4 Σύστημα Διαχείρισης Χωρικών Βάσεων Δεδομένων PostgreSQL

Πρόκειται για σύστημα ανοιχτού κώδικα που υποστηρίζει όλους τους αναγκαίους για την υλοποίηση της παρούσας εργασίας, χωρικούς τύπους δεδομένων αλλά και πληθώρα άλλων λειτουργιών.

Πάνω σε αυτό έχει αναπτυχθεί το TelegraphCQ που διαχειρίζεται τα ρεύματα δεδομένων της παρούσας εφαρμογής.

Σχετική αναφορά γίνεται στο Κεφάλαιο 2.

4.5 Χρήση της γλώσσας HTML

Για την υλοποίηση της παρούσας εφαρμογής ήταν απαραίτητη η χρήση της γλώσσας HTML καθώς βασικός στόχος είναι να γίνεται η διαχείριση των κινούμενων αντικειμένων μέσω ιστοσελίδας ώστε αυτή να είναι εφικτή από οποιονδήποτε υπολογιστή στον κόσμο.

4.5.1 Βασικά ιστορικά στοιχεία της HTML

Το 1989 ο Tim Berners-Lee επινοεί την HTML στο CERN (European Laboratory for Particle Physics, Geneva, Switzerland)

Το 1993 κυκλοφορεί ο Mosaic, ο πρώτος browser με γραφικό περιβάλλον και δυνατότητα να διαβάζει ιστοσελίδες γραμμένες σε HTML.

Το 1994 ιδρύεται η Netscape Communications Corp. από τον δημιουργό του Mosaic, Marc Andreessen. Επίσης, η HTML επεκτείνεται με αποτέλεσμα ένα νέο πρότυπο, γνωστό ως HTML 2.

Το 1995 κυκλοφορεί ο browser Netscape, αλλά και ο Internet Explorer από τη Microsoft. Επίσης, κυκλοφορεί το πρότυπο HTML 3.

Το 1997 κυκλοφορεί η HTML 3.2, με δυνατότητες για έλεγχο της εμφάνισης των ιστοσελίδων.

Το 1999 κυκλοφορεί το τελευταίο πρότυπο της HTML – η HTML 4.01.

4.5.2 Λίγα λόγια για την HTML

Τα αρχικά HTML προέρχονται από τις λέξεις HyperText Markup Language. Η html δεν είναι μια γλώσσα προγραμματισμού. Είναι μια περιγραφική γλώσσα (markup language), δηλαδή ένας ειδικός τρόπος γραφής κειμένου. Αποτελεί υποσύνολο της γλώσσας SGML (Standard Generalized Markup Language) που επινοήθηκε από την IBM προκειμένου να λυθεί το πρόβλημα της μη τυποποιημένης εμφάνισης κειμένων στα διάφορα υπολογιστικά συστήματα. Όλοι οι browsers (φυλλομετρητές) αναγνωρίζουν αυτόν τον τρόπο γραφής και εκτελούν τις εντολές που περιέχονται σε αυτόν. Η HTML είναι η πρώτη και πιο διαδεδομένη γλώσσα περιγραφής της δομής μιας ιστοσελίδας. Χρησιμοποιεί ειδικές ετικέτες-στοιχεία (τα λεγόμενα tags) ώστε να δώσει τις απαραίτητες οδηγίες για την εμφάνιση της επιθυμητής ιστοσελίδας στον browser. Τα στοιχεία (tags) αποτελούν εντολές που συνήθως ορίζουν την αρχή ή το τέλος μιας λειτουργίας. Βρίσκονται πάντα μεταξύ των συμβόλων < και >. Οι οδηγίες είναι case insensitive, δεν επηρεάζονται δηλαδή από το αν έχουν γραφτεί με πεζά (μικρά) ή κεφαλαία. (Πηγή: http://el.wikipedia.org/wiki/HTML)

4.6 Η γλώσσα προγραμματισμού PHP

Η PHP είναι μία γλώσσα σεναρίων (scripting language) που είχε αρχικά σχεδιαστεί για την παραγωγή δυναμικών ιστοσελίδων. Πλέον έχει εξελιχθεί και περιέχει τη δυνατότητα γραμμής εντολών διασύνδεσης και μπορεί να χρησιμοποιηθεί σε αυτόνομες γραφικές εφαρμογές. Οι γλώσσες σεναρίων ανήκουν στην κατηγορία των γλωσσών προγραμματισμού.

Η PHP δημιουργήθηκε αρχικά από τον Rasmus Lerdorf το 1995, ενώ σήμερα η βασική υλοποίησή της δημιουργείται από την ομάδα PHP (PHP Group).

Η PHP είναι ελεύθερο λογισμικό υπό την άδεια της PHP (PHP License), αλλά είναι ασύμβατη με τη γενική δημόσια άδεια GNU (GNU General Public License) λόγω περιορισμών στη χρήση της από τον καθορισμό της άδειάς της.

4.6.1 Βασικά ιστορικά στοιχεία της PHP

Το 1995 ο Rasmus Lerdorf επινοεί την PHP τα αρχικά της οποίας σημαίνουν Personal Home Page. Κυκλοφορεί η έκδοση 1.0.0.

Το 1996 κυκλοφορεί η έκδοση 2.0.0, η οποία θεωρείται από τον δημιουργό της ως το ταχύτερο και απλούστερο εργαλείο για τη δημιουργία δυναμικών ιστοσελίδων.

Το 1998 κυκλοφορεί η έκδοση 3.0.0 από περισσότερους πλέον χρήστες-προγραμματιστές. Οι Zeev Suraski και Andi Gutmans ξαναγράφουν τη βάση για αυτήν την έκδοση.

Το 2000 κυκλοφορεί η έκδοση 4.0.0 με διάφορες προσθήκες και βελτιώσεις.

Τις επόμενες χρονιές κυκλοφορούν οι εκδόσεις 4.1.0, 4.2.0, 4.3.0, 4.4.0, 4.4.8, 4.4.9, 5.0.0, 5.1.0, 5.2.0 και 5.2.8, ενώ τη στιγμή συγγραφής της παρούσας εργασίας η τρέχουσα έκδοση ήταν η 5.2.9.

Πρόκειται λοιπόν για μία γλώσσα προγραμματισμού που εξελίσσεται συνεχώς. (Πηγή: http://en.wikipedia.org/wiki/Php)

4.6.2 Δυνατότητες της PHP

Η PHP σε γενικές γραμμές εκτελείται στον web server, ο οποίος δέχεται τον κώδικα PHP ως εισαγωγή δεδομένων (input) και δημιουργεί ιστοσελίδες ως αποτέλεσμα εκτέλεσης του κώδικα (output).

Ο βασικότερος λόγος που εντάχθηκε αυτή η γλώσσα προγραμματισμού στην μελετώμενη εφαρμογή είναι η δυνατότητα που παρέχει για διασύνδεση με διάφορα ΣΔΒΔ όπως το απαραίτητο στην περίπτωσή μας σύστημα της PostgreSQL και ταυτόχρονα, η δυνατότητά της να δημιουργεί αρχεία XML και να αποθηκεύει εκεί τα αποτελέσματα εκτέλεσης των ερωτημάτων που υποβάλλονται στο ΣΔΒΔ.

Επίσημος ιστοτόπος της PHP: http://www.php.net/.

4.7 Η γλώσσα XML

Η XML (eXtensible Markup Language – Επεκτάσιμη Γλώσσα Επισήμανσης) αποτελεί μία έκδοση της SGML (Standard Generalized Markup Language – Πρότυπη Γενικευμένη Γλώσσα Σήμανσης). Η SGML σχεδιάστηκε με σκοπό να μειώσει το κόστος (σε χρόνο κυρίως και κατ' επέκταση σε χρήμα) διαχείρισης εγγράφων. Η XML σχεδιάστηκε με σκοπό να εισάγει την SGML στο διαδίκτυο. Πρόκειται για μία περιγραφική γλώσσα σήμανσης δεδομένων που σκοπό έχει την επικοινωνία και τη μεταφορά πληροφοριών μεταξύ διαφορετικών προγραμμάτων, συστημάτων και εφαρμογών.

Όλες οι περιγραφικές γλώσσες σήμανσης (Markup Languages), όπως οι HTML, XML, SGML, GML (Geography Markup Language), KML και άλλες, έχουν το εξής κοινό χαρακτηριστικό, συμπεριλαμβάνουν ενσωματωμένες πληροφορίες μέσα στο κείμενό τους σχετικά με την ερμηνεία και το νόημα των πληροφοριών που περιέχουν.

4.7.1 Βασικά ιστορικά στοιχεία της XML

Το Νοέμβριο του 1996 παρουσιάζεται η πρώτη πρόχειρη έκδοση της XML.

Το Μάρτιο του 1997 διεξάγεται στο San Diego το πρώτο συνέδριο με θέμα την XML (XML Conference).

Το Δεκέμβριο του 1997 το World Wide Web Consortium (W3C) ανακοινώνει την κυκλοφορία της έκδοσης 1.0 της XML ως επίσημη πρότασή του.

Τον Οκτώβριο του 2000 το W3C ανακοινώνει την κυκλοφορία της XML 1.0 δεύτερη έκδοση.

Τον Οκτώβριο του 2003 ο ίδιος οργανισμός ανακοινώνει την κυκλοφορία της XML 1.0 τρίτη έκδοση.

Τον Ιούνιο του 2006 κυκλοφορεί η XML 1.0 τέταρτη έκδοση.

Τέλος, το Νοέμβριο του 2008 ανακοινώνεται η κυκλοφορία της πέμπτης έκδοσης της γλώσσας XML 1.0.

4.7.2 Στόχοι της XML

Επιγραμματικά, οι στόχοι που τέθηκαν κατά τον σχεδιασμό της XML είναι:

- Άμεση και εύκολη χρήση της μέσω του διαδικτύου

- Υποστήριξη ευρέως φάσματος εφαρμογών
- Συμβατότητα με την SGML
- Δυνατότητα εύκολης δημιουργίας προγραμμάτων που θα επεξεργάζονται αρχεία XML
- Ελάχιστο δυνατό πλήθος προαιρετικών χαρακτηριστικών της XML, ιδανικά μηδενικό
- Αρχεία XML εύκολα στην ανάγνωση από ανθρώπους και ξεκάθαρα ως προς το περιεχόμενο
- Γρήγορη πραγματοποίηση του σχεδιασμού της XML
- Επίσημος και περιεκτικός σχεδιασμός της XML
- Εύκολη δημιουργία αρχείων XML

4.7.3 Η XML στην παρούσα εργασία – Απόρριψη της KML

Η XML είναι απαραίτητη για την υλοποίηση της παρούσας εφαρμογής, διότι χρησιμοποιείται για την αποθήκευση των αποτελεσμάτων των ερωτημάτων από την PHP ώστε αυτά να αναγνωσθούν από την JavaScript και να εμφανιστούν στους χάρτες Google της εφαρμογής.

Το ενδεχόμενο δημιουργίας αρχείων KML (Keyhole Markup Language) απορρίφθηκε καθώς για να αναγνωσθούν αυτά και οι πληροφορίες να εμφανιστούν επάνω στους χάρτες Google, πρέπει να αρχεία KML να βρίσκονται σε web server της Google. Αυτό σημαίνει ότι δεν είναι εφικτή η δυναμική δημιουργία τους, αλλά αυτά πρέπει να δημιουργηθούν άπαξ και να καταχωρηθούν στον σχετικό web server της Google. Όπως θα αναλυθεί στο επόμενο κεφάλαιο, στην παρούσα εφαρμογή είναι απαραίτητη η δυναμική και επαναληπτική δημιουργία αρχείων με τα τρέχοντα κάθε φορά αποτελέσματα των ερωτημάτων και συνεπώς τα αρχεία με τα αποτελέσματα πρέπει να ανανεώνονται ανά τακτά χρονικά διαστήματα (λίγων δευτερολέπτων), όπως και η εμφάνισή τους στην ιστοσελίδα.

Από τα παραπάνω διαφαίνεται ότι είναι απαραίτητη η χρήση αρχείων XML που δεν χρειάζεται να βρίσκονται σε web server της Google αλλά μπορούν να βρίσκονται σε οποιονδήποτε web server ή database server.

4.8 Η γλώσσα προγραμματισμού JavaScript

4.8.1 Γενικά – ιστορικά στοιχεία

Η JavaScript αναπτύχθηκε από τη Netscape Communications Corporation, που είναι η κατασκευάστρια εταιρεία του Netscape Web Browser αλλά και του Mozilla Firefox. Η JavaScript ήταν η πρώτη γλώσσα script για το Web που υποστηρίχθηκε από Browser και μέχρι και σήμερα είναι η πιο δημοφιλής στο είδος της.

Η JavaScript είναι και αυτή μία γλώσσα σεναρίων (scripting language) και ταυτόχρονα μία γλώσσα προγραμματισμού. Σκοπός της είναι η παραγωγή δυναμικού περιεχομένου σε ιστοσελίδες. Προέρχεται από την ECMAscript, ενώ έχει επεκταθεί με αρκετές πρόσθετες ιδιότητες.

Ο τρόπος σύνταξης της JavaScript παρουσιάζει πολλές ομοιότητες με αυτόν της C στην οποία έχει βασιστεί. Η JavaScript, σε αντίθεση με την PHP που είναι μια server side γλώσσα προγραμματισμού, είναι client side γλώσσα. Η επεξεργασία του κώδικα JavaScript και η παραγωγή των αποτελεσμάτων δηλαδή, δεν πραγματοποιείται στον server αλλά στον browser των επισκεπτών (clients). Συνεπώς, η JavaScript δεν έχει καμία απαίτηση από πλευράς του server για να εκτελεστεί αλλά έχει απαίτηση από

πλευράς του υπολογιστή και του browser του κάθε επισκέπτη. Η JavaScript μπορεί να ενσωματωθεί και σε στατικές αλλά και σε δυναμικές σελίδες HTML.

Η JavaScript ονομάστηκε αρχικά LiveScript και παρουσιάστηκε στον Netscape Navigator 2.0 το 1995. Μετά από σύντομο χρονικό διάστημα μετονομάστηκε σε JavaScript υποδεικνύοντας μόνο τη σχέση marketing που είχε με τη γλώσσα Java της Sun, ενώ δε θα πρέπει να συγχέεται με αυτήν καθώς πρόκειται για διαφορετικές γλώσσες προγραμματισμού.

4.8.2 Δυνατότητες της JavaScript

Οι δυνατότητες της JavaScript είναι σημαντικά λιγότερες από αυτές της PHP με κύρια διαφορά ότι δεν παρέχει συνδεσιμότητα με βάσεις δεδομένων. Εντούτοις είναι μία πολύ χρήσιμη και δυνατή γλώσσα προγραμματισμού αναγκαία σε πολλές περιπτώσεις ιστοσελίδων. Παρακάτω δίδονται μερικά παραδείγματα λειτουργιών και εφαρμογών της JavaScript:

- Εμφάνιση μηνυμάτων στο χρήστη ως τμήμα μιας ιστοσελίδας, στη γραμμή κατάστασης του browser ή σε πλαίσια ειδοποίησης.
- Επικύρωση του περιεχομένου μιας φόρμας και εκτέλεση υπολογισμών
- Δημιουργία κινούμενων εικόνων ή εικόνων που αλλάζουν όταν μετακινούμε το ποντίκι επάνω τους
- Δημιουργία διαφημιστικών τίτλων που αλληλεπιδρούν με το χρήστη, αντί της απλής εμφάνισης ενός γραφικού
- Εντοπισμός του browser του χρήστη και των δυνατοτήτων του και εκτέλεση προχωρημένων λειτουργιών μόνο στους browsers που τις υποστηρίζουν
- Εντοπισμός εγκατεστημένων πρόσθετων προγραμμάτων και ειδοποίηση του χρήστη εάν απαιτείται κάποιο επιπλέον πρόσθετο
- Τροποποίηση ολόκληρης ή μέρους μίας ιστοσελίδας χωρίς την ανάγκη επαναφόρτωσης αυτής από τον χρήστη
- Εμφάνιση και αλληλεπίδραση με δεδομένα που ανακτώνται από απομακρυσμένο διακομιστή
- Συνδυασμός των παραπάνω και δημιουργία ολόκληρων εφαρμογών γραμμένων σε αυτή τη γλώσσα προγραμματισμού

Πηγές:

http://el.wikipedia.org/wiki/JavaScript και

[Michel Moncur] Μάθετε την JavaScript σε 24 ώρες, Εκδόσεις Μ. Γκιούρδας,2007

4.8.3 Η JavaScript στην παρούσα εργασία

Ο ρόλος της JavaScript στην παρούσα εργασία είναι καθοριστικός. Είναι η γλώσσα προγραμματισμού που αναλαμβάνει μερικές από τις βασικότερες λειτουργίες της παρούσας εφαρμογής.

Τα σχετικά script που έχουν αναπτυχθεί για τις ανάγκες της παρούσας εφαρμογής εμφανίζουν τους επιθυμητούς χάρτες Google στην ιστοσελίδα, διαβάζουν με επαναληπτικό τρόπο τα αρχεία XML, επεξεργάζονται τις πληροφορίες αυτών, εμφανίζουν τα αποτελέσματα των συνεχών ερωτημάτων στο χάρτη αλλά και σε σχετικούς πίνακες και εκτελούν πληθώρα άλλων λειτουργιών στις οποίες θα αναφερθούμε εκτενώς στο επόμενο κεφάλαιο.

4.9 Οι χάρτες Google σε ιστοσελίδες

4.9.1 Γενικές πληροφορίες

Οι Χάρτες Google (GoogleMaps) είναι μια υπηρεσία της Google που προσφέρει φιλική προς το χρήστη τεχνολογία χαρτογράφησης. Διατίθενται από την Google τουλάχιστον 4 είδη χαρτών που καλύπτουν ολόκληρη την επιφάνεια της γης σε διάφορες κλίμακες ως εξής:

- Οδικός χάρτης
- Φυσικός-εδαφικός χάρτης
- Εικόνες δορυφόρου
- Συνδυασμός εικόνων δορυφόρου και οδικού χάρτη (υβριδικός χάρτης)

Τους χάρτες αυτούς μπορεί ο καθένας να συμπεριλάβει στις ιστοσελίδες του με σκοπό να εμφανίσει επί χάρτου διάφορες επιθυμητές πληροφορίες.

Οι Χάρτες Google υποστηρίζονται προς το παρόν από τα ακόλουθα προγράμματα περιήγησης ιστού:

IE 6.0+ (σε Windows)
Firefox 0.8+ (σε Windows, Mac και Linux)
Safari 1.2.4+ (σε Mac)
Netscape 7.1+ (σε Windows, Mac και Linux)
Mozilla 1.4+ (σε Windows, Mac και Linux)
Opera 8.02+ (σε Windows, Mac και Linux)

4.9.2 Χρήση των χαρτών Google στην παρούσα εφαρμογή

Για την διαχείριση των κινούμενων αντικειμένων της παρούσας εφαρμογής επιλέχθηκε η οπτικοποίηση και εμφάνιση τους με τη βοήθεια της JavaScript σε χάρτες της Google.

Η χρήση χαρτών Google ήταν ένας από τους βασικούς στόχους της παρούσας εργασίας καθώς αυτοί μπορούν να ενσωματωθούν σε οποιαδήποτε ιστοσελίδα που φιλοξενείται σε οποιονδήποτε web server και κατ' επέκταση καθίσταται δυνατή η διαχείριση των κινούμενων αντικειμένων της εφαρμογής από οποιονδήποτε υπολογιστή από οποιοδήποτε σημείο του πλανήτη.

4.10 Η γλώσσα προγραμματισμού C++

4.10.1 Γενικές πληροφορίες

Ο Bjarne Stroustrup ανέπτυξε, το 1979 στα εργαστήρια Bell της AT&T, την C++ με πρωταρχικό σκοπό την προσθήκη αντικειμενοστραφών δομών στη γλώσσα C. Επειδή η αντικειμενοστραφής τεχνολογία ήταν καινούργια και όλες οι αντικειμενοστραφείς υλοποιήσεις που υπήρχαν ήταν αρκετά αργές και μη αποδοτικές, ένας δευτερεύων σκοπός της C++ ήταν να διατηρήσει την αποδοτικότητα της C.

Η C++ (C Plus Plus) είναι μια γενικού σκοπού γλώσσα προγραμματισμού Η/Υ. Θεωρείται γλώσσα μέσου επιπέδου, καθώς περιλαμβάνει έναν συνδυασμό χαρακτηριστικών από γλώσσες υψηλού και χαμηλού επιπέδου. Η C++ μπορεί να θεωρηθεί μια διαδικαστική γλώσσα με κάποιες επιπλέον δομές, μερικές από τις οποίες προστέθηκαν για αντικειμενοστραφή προγραμματισμό, ενώ άλλες για την βελτίωση του

συντακτικού της γλώσσας. Η C++ είναι ουσιαστικά μια επεκτάσιμη γλώσσα αφού μπορούμε να ορίσουμε νέους τύπους με τέτοιο τρόπο ώστε να λειτουργούν σαν τους προκαθορισμένους τύπους, που είναι τμήμα της γλώσσας. Η C++ σχεδιάστηκε για την ανάπτυξη μεγάλων προγραμμάτων. Είναι μια δακτυλογραφούμενη, ελεύθερης μορφής, πολλαπλών παραδειγμάτων, μεταφράσιμη γλώσσα όπου η μετάφρασή της (compilation) δημιουργεί κώδικα μηχανής για ένα συγκεκριμένο τύπο υλικού. Υποστηρίζει δομημένο, αντικειμενοστραφή και γενικό προγραμματισμό.

Αρχικά ονομάστηκε "C with Classes", δηλαδή C με Κλάσεις. Μετονομάστηκε σε C++ το 1983. Οι βελτιώσεις ξεκίνησαν με την προσθήκη κλάσεων, και ακολούθησαν, μεταξύ άλλων, εικονικές συναρτήσεις, υπερφόρτωση τελεστών, πολλαπλή κληρονομικότητα, πρότυπα κ.α.

Η γλώσσα ορίστηκε παγκοσμίως, το 1998, με το πρότυπο ISO/IEC 14882:1998. Η τρέχουσα έκδοση αυτού του προτύπου είναι αυτή του 2003, η ISO/IEC 14882:2003. Μια καινούρια έκδοση είναι υπό ανάπτυξη, γνωστή ανεπίσημα με την ονομασία C++0x.

Πηγή πληροφοριών:
http://www.it.uom.gr/project/cppmanual/cplman/index00.htm και
http://el.wikipedia.org/wiki/C%2B%2B

4.10.2 Χρήση της C++ στην παρούσα εφαρμογή

Κάθε ολοκληρωμένη διαδικτυακή εφαρμογή πρέπει να είναι σε θέση να λειτουργεί εσαεί χωρίς την υποχρέωση παρέμβασης του ανθρώπου. Για το σκοπό αυτό πρέπει εκτός των άλλων, ανά τακτά χρονικά διαστήματα να γίνεται μία εκκαθάριση της εφαρμογής. Για την εκκαθάριση αυτή, κρίθηκε καταλληλότερη η χρήση της C++ έναντι άλλων γλωσσών προγραμματισμού κυρίως λόγω των αυξημένων δυνατοτήτων της αλλά και της ευρείας χρήσης της σε σχέση με άλλες γλώσσες προγραμματισμού.

Διάφορα προβλήματα και περιορισμοί της PHP και της PostgreSQL κατέστησαν αναγκαία τη δημιουργία και άλλων προγραμμάτων σε C++ που εκτελούν συγκεκριμένες λειτουργίες και περιορίζουν σημαντικά τα προβλήματα και τους περιορισμούς αυτούς.

Πιο συγκεκριμένα, στην περίπτωση που ένα ερώτημα δεν επιστρέφει αποτελέσματα, προκύπτει η ανάγκη τερματισμού εκτέλεσης του ερωτήματος με χρήση συγκεκριμένων εντολών σε περιβάλλον Linux όπου εκτελείται η βάση δεδομένων αλλά και τα αρχεία PHP και οι συνδέσεις Apache. Ο τερματισμός αυτός κατέστη δυνατός με τη χρήση προγραμμάτων γραμμένων σε C++.

Περισσότερες πληροφορίες σχετικά με τα 8 συνολικά προγράμματα που είναι γραμμένα σε C++ υπάρχουν στο Κεφάλαιο 5 και συγκεκριμένα στις παραγράφους 5.14, 5.15, 5.16, 5.17 και 5.19.

4.11 Η γλώσσα προγραμματισμού PERL

4.11.1 Γενικές πληροφορίες – Ιστορικά στοιχεία

Η PERL (Practical Extraction and Report Language – Πρακτική Γλώσσα Αναφοράς και Εξαγωγής) είναι μια γλώσσα scripting προγραμματισμού σχεδιασμένη να εκτελείται από μία πληθώρα λειτουργικών συστημάτων και αρχιτεκτονικών και διατίθεται με την άδεια ανοικτού λογισμικού GPL. Δημιουργήθηκε από τον Larry Wall το 1987, οπότε και κυκλοφόρησε η πρώτη της έκδοση. Διαδόθηκε ταχύτατα και σύντομα

κυκλοφόρησαν δύο επόμενες εκδόσεις της: έκδοση 2 το 1988 και έκδοση 3 το 1989. Το 1991 κυκλοφόρησε η τέταρτη έκδοση, ενώ η πέμπτη έκδοση εκδόθηκε το 1994. Η Πέμπτη έκδοση έφερε σημαντικές αλλαγές και προσέθεσε αρκετές δυνατότητες ενώ ταυτόχρονα αύξησε τον αριθμό των υποστηριζόμενων λειτουργικών συστημάτων. Η έκδοση αυτή συνεχίζει να αναπτύσσεται ακόμα και σήμερα ευρισκόμενη στην 5.10.0. Η έκτη έκδοση έχει ανακοινωθεί ήδη από το 2000 αλλά δεν έχει κυκλοφορήσει μέχρι σήμερα.

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

Επιγραμματικά, μερικές από τις δυνατότητες της PERL είναι:

- Ταχύτητα Ανάπτυξης (Δεν χρειάζεται ξεχωριστός μεταγλωττιστής)
- Ισχύς (Από τις καλύτερες κανονικές εκφράσεις (regular expressions) που διατίθενται)
- Ευκολία Χρήσης (δεν απαιτείται γνώση αντικειμενοστραφούς προγραμματισμού)
- Μεταφερσιμότητα (τα ίδια scripts θα τρέξουν σε όλα τα λειτουργικά συστήματα)
- Εργαλεία Επιμέλειας (τα scripts γράφονται σε απλούς κειμενογράφους αλλά και σε σχετικά δωρεάν λογισμικά)
- Δυνατότητα εκτός των άλλων για σύνδεση και επικοινωνία με συγκεκριμένο απομακρυσμένο ή μη socket

Πηγές:
http://el.wikipedia.org/wiki/Perl και
http://www.it.uom.gr/project/perl/win32perltut.html

4.11.2 Χρήση της Perl στην παρούσα εφαρμογή

Ένα πολύ σημαντικό εργαλείο που χρησιμοποιήθηκε για την υλοποίηση της παρούσας εφαρμογής είναι ένα script γραμμένο στη γλώσσα προγραμματισμού Perl.

Πρόκειται για ένα πρόγραμμα που συνδέεται με το ρεύμα δεδομένων στο συγκεκριμένο port στη συγκεκριμένη IP που "ακούει" η PostgreSQL. Ανοίγει συγκεκριμένο εξωτερικό αρχείο με τις πληροφορίες για τα κινούμενα αντικείμενα και με προκαθορισμένο ρυθμό, αποστέλλει τις πληροφορίες στον σχετικό wrapper του TelegraphCQ ο οποίος με τη σειρά του αφού μετατρέψει κατάλληλα τα δεδομένα για εισαγωγή στο εκάστοτε ρεύμα, τα αποστέλλει σε αυτό.

Περισσότερες πληροφορίες σχετικά με το script αυτό υπάρχουν στο Κεφάλαιο 5 και συγκεκριμένα στην παράγραφο 5.20.

Κεφάλαιο 5

Υλοποίηση Εφαρμογής

5.1 Στόχος εφαρμογής

Στόχος της εφαρμογής είναι η διαχείριση πληροφοριών και παρακολούθηση συγκεκριμένης ομάδας κινούμενων αντικειμένων σε πραγματικό χρόνο από οποιονδήποτε υπολογιστή ανά την υφήλιο. Αυτή η διαχείριση και παρακολούθηση θα πρέπει να μπορεί να γίνει από οποιονδήποτε υπολογιστεί χωρίς να χρειάζεται η εγκατάσταση ειδικών προγραμμάτων, θα πρέπει να μπορεί να γίνεται απ' ευθείας μέσα από τον web browser (φυλλομετρητή διαδικτύου) που διαθέτει το κάθε μηχάνημα. Ακόμα και από κινητά, PDAs και άλλες συσκευές που διαθέτουν σύνδεση στο διαδίκτυο και έναν web browser θα πρέπει να είναι εφικτή αυτή η διαχείριση και παρακολούθηση των συγκεκριμένων κινούμενων αντικειμένων.

Η εφαρμογή λοιπόν θα πρέπει να ανοίγει μέσα από έναν web browser και η όποια επεξεργασία για την εμφάνιση των αποτελεσμάτων θα πρέπει να γίνεται εν μέρει απομακρυσμένα στον web server σε συνεργασία με έναν database server (server-side programming) και εν μέρει τοπικά σε κάθε υπολογιστή με τη βοήθεια υφιστάμενων προγραμμάτων (client-side programming).

Ο κάθε χρήστης θα πρέπει να μπορεί να παρακολουθεί σε πραγματικό χρόνο (με απόκλιση από την πραγματικότητα το πολύ μερικά δευτερόλεπτα) τη θέση όλων των κινούμενων αντικειμένων, τη θέση συγκεκριμένων αντικειμένων που θα πληρούν ορισμένες προϋποθέσεις ανάλογα με τα ερωτήματα που θα θέτει στην εφαρμογή, καθώς και πληροφορίες που επιθυμεί σχετικά με τα κινούμενα αντικείμενα θέτοντας τα αντίστοιχα ερωτήματα.

5.2 Εργαλεία ανάπτυξης εφαρμογής

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκαν διάφορα εργαλεία τα οποία έπαιξαν, το καθένα ξεχωριστά, πολύ σημαντικό ρόλο στην επίτευξη του στόχου που είχε τεθεί. Τα εργαλεία αυτά έχουν ήδη αναλυθεί εκτενώς στο προηγούμενο κεφάλαιο. Παρακάτω αναφέρονται επιγραμματικά:

Linux
TelegraphCQ
PostgreSQL
HTML (HyperText Markup Language)
PHP (Personal Home Page)
XML (Extensible Markup Language)
JavaScript
GoogleMaps API
C++
Perl

Η εφαρμογή περιλαμβάνει τέσσερις διαφορετικούς χάρτες της Google στους οποίους, εφ' όσον υπάρχει ροή δεδομένων στο TelegraphCQ, δύνανται να εμφανίζονται οι θέσεις όλων των κινούμενων αντικειμένων, διαφόρων υποσυνόλων αυτών και συνδυασμοί. Στον ίδιο χάρτη εμφανίζονται τα αποτελέσματα από πέντε συνολικά διαφορετικά ερωτήματα που μπορεί να θέσει ο χρήστης στην ιστοσελίδα της εφαρμογής.

Στο σημείο αυτό κρίνεται σκόπιμο να αναφερθεί η χρήση του καθενός από τα αναφερθέντα εργαλεία της προηγούμενης παραγράφου στη συγκεκριμένη εφαρμογή.

Όπως έχει προαναφερθεί, το TelegraphCQ λειτουργεί σε περιβάλλον Linux και συνεπώς εκτός από τον web server που δύναται να φιλοξενήσει την εφαρμογή, είναι απαραίτητη και η διασύνδεση με έναν database server ο οποίος θα διαθέτει λειτουργικό σύστημα Linux και στον οποίο θα "τρέχει" συνεχώς το TelegraphCQ χρησιμοποιώντας για την επικοινωνία με τον χρήστη μία ειδική έκδοση της PostgreSQL. Από τα παραπάνω διαφαίνεται η αναγκαιότητα χρήσης λειτουργικού συστήματος Linux. Το TelegraphCQ, όπως έχει ήδη αναφερθεί είναι το ΣΔΡΔ που επιλέχθηκε για την υλοποίηση της εφαρμογής. Το TelegraphCQ λειτουργεί σε συνεργασία με την PostgreSQL η οποία με τη σειρά της υποστηρίζει χωρικά ερωτήματα. Πιο συγκεκριμένα, προκειμένου ο χρήστης του TelegraphCQ να μπορεί στείλει ρεύματα δεδομένων σε αυτό και να θέσει ερωτήματα διαρκείας, έχει τροποποιηθεί συγκεκριμένη έκδοση της PostgreSQL ώστε πάνω σε αυτήν να λειτουργεί το ΣΔΡΔ. Ο συνδυασμός λοιπόν των δύο αυτών προγραμμάτων υποστηρίζει χωρικά και χρονικά ερωτήματα (τα αναφερόμενα και ως χωροχρονικά).

Η HTML είναι απαραίτητη για την αρχική καταχώρηση στοιχείων, μεταβλητών και παραμέτρων και για την τελική απεικόνιση των πληροφοριών στην ιστοσελίδα της εφαρμογής (front-end) και συνεπώς είναι απαραίτητη εάν επιθυμούμε ένα περιβάλλον φιλικό προς το χρήστη με δυνατότητες καταχώρησης πληροφοριών από αυτόν και εμφάνισης αποτελεσμάτων.

Η διασύνδεση με την PostgreSQL και το TelegraphCQ γίνεται μέσω της γλώσσας προγραμματισμού PHP, η οποία διαθέτει συγκεκριμένες εντολές διασύνδεσης με διάφορες βάσεις δεδομένων μεταξύ των οποίων και την PostgreSQL (και κατ' επέκταση και το TelegraphCQ).

Η απεικόνιση αντικειμένων στους χάρτες της Google, γενικά γίνεται είτε ορίζοντας απ' ευθείας σε σχετικό script τα στοιχεία του κάθε αντικειμένου ξεχωριστά, είτε συνδέοντας κάποιο αρχείο τύπου XML με συγκεκριμένο κώδικα στο script είτε συνδέοντας κάποιο αρχείο KML ή και με άλλους τρόπους στους οποίους δε θα αναφερθούμε στην παρούσα εργασία. Στην εφαρμογή, καθώς τα δεδομένα για τα σημεία λαμβάνονται από το TelegraphCQ, δε θα μπορούσαν να υπάρχουν απ' ευθείας σε script

και συνεπώς δημιουργείται διαφορετικό αρχείο XML με τις απαντήσεις του κάθε ερωτήματος και στη συνέχεια τα XML αυτά διαβάζονται από τον κώδικα JavaScript και απεικονίζονται στο χάρτη. Επίσης τα αρχεία XML ανανεώνονται ανά τακτά χρονικά διαστήματα. Αρχεία KML, μετά από σχετική διερεύνηση, αποφάνηκε ότι για να απεικονιστούν σε χάρτη GoogleMaps πρέπει να βρίσκονται σε web server της Google και συνεπώς παύει η δυνατότητα ανανέωσης του περιεχομένου ανά τακτά χρονικά διαστήματα μερικών δευτερολέπτων. Από τα παραπάνω διαφαίνεται η αναγκαιότητα χρήσης αρχείων XML σε ένα απόλυτα δυναμικό περιβάλλον όπως αυτό της παρούσας εφαρμογής.

Για την απεικόνιση του χάρτη στην ιστοσελίδα, την ανάγνωση αρχείων XML, την απεικόνιση σημείων και τροχιών επί του χάρτη, την αυτόματη ανανέωση των αποτελεσμάτων, τη δημιουργία πινάκων με αποτελέσματα ερωτημάτων, τον έλεγχο των πεδίων εισαγωγής πληροφοριών και την επίτευξη λοιπών δυναμικών λειτουργιών της εφαρμογής είναι απαραίτητη η χρήση της JavaScript.

Το GoogleMaps Application Programming Interface είναι το εργαλείο το οποίο, με τον κατάλληλο κώδικα στο σχετικό JavaScript, εμφανίζει τους χάρτες της Google στην εφαρμογή.

Η C++ έχει χρησιμοποιηθεί για τη δημιουργία 8 συνολικά προγραμμάτων τα οποία συνδράμουν στη διακοπή των ερωτημάτων που δεν επιστρέφουν απαντήσεις.

Τέλος η Perl είναι η γλώσσα προγραμματισμού που προτιμήθηκε για τη δημιουργία του script που αναλαμβάνει την αποστολή των πληροφοριών από εξωτερικό ASCII αρχείο μέσα στη βάση δεδομένων.

5.3 Περιγραφή εφαρμογής – Μια γενική εικόνα

Η εφαρμογή διατίθεται στην Ελληνική και στην Αγγλική γλώσσα.

Οι περιγραφές θα αφορούν στην Ελληνική έκδοση ενώ σε παρενθέσεις και όπου αυτό κρίνεται σκόπιμο, θα αναφέρεται το αντίστοιχο κείμενο της Αγγλικής έκδοσης. Η ιστοσελίδα χωρίζεται σε τρία τμήματα με σχετικές οριζόντιες διαχωριστικές γραμμές. Το πρώτο τμήμα περιέχει Τον τίτλο και ορισμένα βασικά – αρχικά στοιχεία. Το δεύτερο τμήμα περιέχει τον χάρτη και σχετικές με αυτόν πληροφορίες. Το τρίτο τμήμα περιέχει τα πλαίσια των ερωτημάτων και σχετικές με αυτά πληροφορίες.

5.3.1 Πρώτο τμήμα κεντρικής ιστοσελίδας εφαρμογής

Ο χρήστης ανοίγοντας την κεντρική ιστοσελίδα της εφαρμογής, πάνω πάνω θα διαβάσει τον τίτλο αυτής ο οποίος είναι:

Διαχείριση κινούμενων αντικειμένων με το TelegraphCQ και τους χάρτες της Google

(Moving objects management with TelegraphCQ and GoogleMaps)

Στη συνέχεια, υπάρχει μια προτροπή για ανάγνωση των οδηγιών χρήσης:

Παρακαλώ διαβάστε τις Οδηγίες χρήσης. (Please read Instructions of use.)

Η ιστοσελίδα με τις οδηγίες χρήσης παρατίθεται στην Ελληνική και στην Αγγλική έκδοση στο Παράρτημα 1. Μέσα από τις οδηγίες χρήσης υπάρχει δεσμός (link) προς τρίτη ιστοσελίδα με περισσότερες πληροφορίες σχετικά με την εφαρμογή. Η ιστοσελίδα με τις περισσότερες πληροφορίες (Ελληνική και Αγγλική έκδοση) παρατίθεται και αυτή στο Παράρτημα 1.

Κατόπιν, υπάρχουν δύο πλαίσια-κουμπιά τα οποία πατώντας ο χρήστης, θα ξεκινήσουν να αποστέλλονται Δεδομένα στο TelegraphCQ υπό μορφή ρεύματος. Αυτά τα πλαίσια-κουμπιά αναγράφουν επάνω τους:

Εκκίνηση ενός Ρεύματος Δεδομένων (Start of one Data Stream)

Εκκίνηση δύο Ρευμάτων Δεδομένων (Start of two Data Streams)

Προφανώς πατώντας επάνω στο πρώτο κουμπί ξεκινά ένα Ρεύμα Δεδομένων. Παράλληλα, μόλις ο χρήστης αφήσει το αριστερό κλικ του ποντικιού, εμφανίζεται μια ειδοποίηση ενημερώνοντας το χρήστη ότι το ρεύμα δεδομένων έχει μόλις ξεκινήσει και ότι θα διαρκέσει για τα επόμενα 6 λεπτά. Η παρούσα εφαρμογή, καθώς αποτελεί αντικείμενο μεταπτυχιακής διπλωματικής εργασίας, είναι σε πειραματικό-ερευνητικό στάδιο και τα δεδομένα δεν έρχονται σε πραγματικό χρόνο από κινούμενα αντικείμενα αλλά διαβάζονται από αρχείο απλού κειμένου και αφορούν 15 κινούμενα αντικείμενα για διάρκεια 6 λεπτών (360 δευτερολέπτων).

Προτού συνεχίσουμε την περιγραφή της εφαρμογής πρέπει να αναφέρουμε ότι το ΣΔΡΔ TelegraphCQ δεν μπορεί να κάνει σύνδεση ρεύματος δεδομένων με τον εαυτό του. Υπάρχουν ερωτήματα στα οποία θα αναφερθούμε εκτενέστερα παρακάτω, τα οποία απαιτούν τη σύνδεση του ρεύματος δεδομένων με τον εαυτό του. Καθώς κάτι τέτοιο δεν είναι εφικτό, δημιουργήθηκε η δυνατότητα εκτέλεσης ενός δεύτερου ρεύματος δεδομένων από την εφαρμογή με τα ίδια στοιχεία. Πρώτα ορίστηκε στο TelegraphCQ ένα δεύτερο ρεύμα (το ρεύμα στην περίπτωση αυτή είναι κάτι αντίστοιχο με τον πίνακα στα ΣΔΒΔ) με τα ίδια πεδία-χαρακτηριστικά που έχει το πρώτο ρεύμα.

Τα δύο ρεύματα λοιπόν έχουν τα εξής πεδία-χαρακτηριστικά:

- `vid` (Τύπος: ακέραιος αριθμός. Πρόκειται για τον μοναδικό κωδικό αριθμό του κάθε οχήματος(αντικειμένου).)
- `pos` (Τύπος: σημείο (point). Πρόκειται για την χωρική πληροφορία του ρεύματος δεδομένων η οποία αποθηκεύει τις συντεταγμένες του κάθε αντικειμένου. Οι συντεταγμένες είναι σε γεωγραφικό μήκος (longitude) και πλάτος(latitude) αντίστοιχα.)
- `t` (Τύπος: ακέραιος αριθμός. Πρόκειται για έναν αύξων αριθμό - μετρητή δευτερολέπτων από την πρώτη αποστολή της θέσης του αντικειμένου. Στην παρούσα εφαρμογή το πεδίο t παίρνει τις τιμές 0, 5, 10, 15 κ.ο.κ.)
- `tcqtime` (Τύπος: χρονική στιγμή χωρίς ζώνη ώρας. Στο πεδίο αυτό αποθηκεύεται η χρονική στιγμή κατά την οποία λήφθηκαν οι συντεταγμένες από τη συσκευή εντοπισμού. Η χρονική στιγμή περιέχει το δευτερόλεπτο, το λεπτό, την ώρα, την ημέρα του μήνα, το μήνα και το έτος λήψης της θέσης.)

Πατώντας λοιπόν το δεύτερο κουμπί εκκίνησης δύο ρευμάτων δεδομένων, ξεκινούν οι πληροφορίες υπό μορφή ρεύματος και αποστέλλονται και στα δύο ρεύματα (πίνακες του ΣΔΡΔ) ταυτόχρονα. Για να σταλούν βέβαια ταυτόχρονα οι ίδιες πληροφορίες και στα δύο ρεύματα, πρέπει να είναι και τα δύο σταματημένα πριν πατήσει ο χρήστης το κουμπί της εκκίνησης των δύο ρευμάτων δεδομένων.

Με το ένα ρεύμα δεδομένων μπορούν να εκτελεστούν όλα τα ερωτήματα που δεν απαιτούν σύνδεση του ρεύματος με τον εαυτό του. Τέτοια είναι η εμφάνιση όλων των κινούμενων αντικειμένων, η εμφάνιση αντικειμένων με συγκεκριμένο στατικό κριτήριο επιλογής, όπως συγκεκριμένο id, συγκεκριμένη απόσταση από σταθερό σημείο, συγκεκριμένη περιοχή κλπ.

Με την ταυτόχρονη αποστολή των ίδιων πληροφοριών στα δύο ρεύματα δεδομένων, μπορούν να εκτελεστούν και ερωτήματα σύνδεσης του ρεύματος με τον εαυτό του. Τέτοια είναι η εμφάνιση των αντικειμένων που ανά πάσα στιγμή βρίσκονται σε απόσταση μικρότερη από μια καθορισμένη τιμή από το πλησιέστερο άλλο κινούμενο

αντικείμενο και ο υπολογισμός ανά πάσα στιγμή των αποστάσεων όλων των κινούμενων αντικειμένων από συγκεκριμένο κινούμενο αντικείμενο.

Τα ερωτήματα που μπορούν να εκτελεστούν με ένα ρεύμα δεδομένων, μπορούν φυσικά να εκτελεστούν και με τα δύο ρεύματα ενεργά. Είναι λογικό να αναρωτηθεί κανείς, γιατί λοιπόν δεν υπάρχει μόνο το κουμπί εκτέλεσης των δύο ρευμάτων δεδομένων. Η συνεχής εισροή δεδομένων από εξωτερική πηγή στο ΣΔΡΔ είναι μια διαδικασία σχετικά απαιτητική σε επεξεργαστική ισχύ και προσωρινή μνήμη, ενώ οι απαιτήσεις αυξάνονται ανάλογα με το πλήθος των κινούμενων αντικειμένων και τη συχνότητα αποστολής πληροφοριών από αυτά στο σύστημα. Στην περίπτωση λοιπόν που τα δεδομένα εισρέουν ταυτόχρονα σε δύο ρεύματα (οντότητες αντίστοιχες με τους πίνακες στα ΣΔΒΔ) η απαιτούμενη επεξεργαστική ισχύς και προσωρινή μνήμη διπλασιάζονται. Καθώς λοιπόν υπάρχουν αρκετά είδη ερωτημάτων που απαιτούν μόνο ένα ρεύμα δεδομένων ενεργό, η καταλληλότερη λύση στη πρόβλημα θεωρείται η ύπαρξη της δυνατότητας εκτέλεσης ενός ή δύο ρευμάτων δεδομένων στην εφαρμογή.

Στα δεξιά από αυτά τα δύο πλαίσια-κουμπιά εκτέλεσης ρευμάτων δεδομένων και συγκεκριμένα στο μέσο της ιστοσελίδας, αποτυπώνεται η κατάσταση στην οποία ευρίσκεται το κάθε ρεύμα δεδομένων. Η κατάσταση μπορεί να πάρει την τιμή "Σταματημένο." ή "Ενεργό..." ανάλογα φυσικά με το εάν ένα ρεύμα δεδομένων είναι σταματημένο ή ενεργό.

Κατάσταση 1ου Ρεύματος Δεδομένων: **Σταματημένο**. / **Ενεργό**...
Κατάσταση 2ου Ρεύματος Δεδομένων: **Σταματημένο**. / **Ενεργό**...
(1st Data Stream Status: **Stopped**. / **Running**...)
(2nd Data Stream Status: **Stopped**. / **Running**...)

Η ιστοσελίδα περιέχει διάφορα δυναμικά στοιχεία όπως η κατάσταση των ρευμάτων και των ερωτημάτων και τα αποτελέσματα που εμφανίζονται στο χάρτη. Όλα τα δυναμικά στοιχεία της ιστοσελίδας ανανεώνονται κάθε πέντε δευτερόλεπτα. Έτσι λοιπόν και η κατάσταση των ρευμάτων δεδομένων ανανεώνεται κάθε πέντε δευτερόλεπτα.

Στο δεξί μέρος της κεντρικής ιστοσελίδας ακριβώς κάτω από τον τίτλο της εφαρμογής, υπάρχει ο δεσμός (link) αλλαγής της γλώσσας.

Στα σχήματα 5.1 και 5.2 φαίνεται το πρώτο αυτό τμήμα της εφαρμογής στην Ελληνική και Αγγλική έκδοση αντίστοιχα με το πρώτο ρεύμα δεδομένων να είναι ενεργό ενώ το δεύτερο σταματημένο. Παρατηρήστε ότι στην Ελληνική έκδοση υπάρχει δεσμός (link) προς την Αγγλική (English), ενώ στην Αγγλική έκδοση ο αντίστοιχος δεσμός (link) είναι αντίστροφος (Ελληνικά) και επιστρέφει στην Ελληνική έκδοση.

Σχήμα 5.1: 1ο τμήμα κεντρικής ιστοσελίδας εφαρμογής (Ελληνική έκδοση)

Σχήμα 5.2: 1ο τμήμα κεντρικής ιστοσελίδας εφαρμογής (Αγγλική έκδοση)

5.3.2 Δεύτερο τμήμα κεντρικής ιστοσελίδας εφαρμογής – Χάρτης και σχετικές πληροφορίες

Μία οριζόντια γραμμή ξεχωρίζει το πρώτο από το δεύτερο τμήμα της εφαρμογής. Στο δεύτερο τμήμα της εφαρμογής, κάτω από αυτήν την οριζόντια γραμμή και στο μέσο της ιστοσελίδας εμφανίζεται η χρονική στιγμή κατά την οποία λήφθηκαν τα πλέον πρόσφατα δεδομένα από το ΣΔΡΔ.

Χρόνος: … (Time: …)

Ακολουθεί το πεδίο όπου εμφανίζεται ο χάρτης της εφαρμογής με πλάτος αυτό του παραθύρου στο οποίο εμφανίζεται η κεντρική ιστοσελίδα και ύψος 420 pixels (εικονοστοιχεία). Είναι διαθέσιμα τα ακόλουθα τέσσερα είδη χαρτών:

Χάρτης (πρόκειται ουσιαστικά για οδικό χάρτη με ονόματα περιοχών (Πόλεις, Δήμοι, Νησιά, χωριά κ.α.), οδούς και ονόματα οδών και ορισμένες άλλες χρήσιμες πληροφορίες.)

Δορυφόρος (πρόκειται για τις γνωστές δορυφορικές εικόνες της Google.)

Υβριδικός (πρόκειται για έναν συνδυασμό οδικού χάρτη και δορυφορικών εικόνων.)

Εδαφικός (πρόκειται για εδαφικό χάρτη ο οποίος εμφανίζει και ονόματα περιοχών αλλά και οδικό δίκτυο της περιοχής που εμφανίζεται στο χάρτη.)

Προεπιλεγμένος είναι ο εδαφικός χάρτης καθώς εμφανίζει τις οδούς με τα ονόματά τους και ταυτόχρονα οι εικόνες του εδαφικού χάρτη είναι πιο μικρές σε όγκο πληροφορίας και συνεπώς έρχονται ευκολότερα στην ιστοσελίδα από τους υπολογιστές της Google (web servers).

Θα πρέπει στο σημείο αυτό να αναφερθεί ότι και τα τέσσερα είδη χαρτών καλύπτουν ολόκληρη τη γη σε ποικιλία κλιμάκων και είναι διαθέσιμα από την Google για δημιουργία ιστοσελίδων. Οι χάρτες έρχονται λοιπόν από τους web servers της Google ξεχωριστά σε κάθε χρήστη που ανοίγει την κεντρική ιστοσελίδα της εφαρμογής.

Στην αριστερή επάνω πλευρά του χάρτη υπάρχει ο έλεγχος κλίμακας και δυνατότητα μετακίνησης του χάρτη στις τέσσερις βασικές κατευθύνσεις (Επάνω-Βόρεια, Κάτω-Νότια, Αριστερά-Δυτικά, Δεξιά-Ανατολικά). Στη δεξιά επάνω πλευρά υπάρχει ο έλεγχος τύπου εμφανιζόμενου χάρτη με τις τέσσερις επιλογές που προαναφέραμε. Στην αριστερή κάτω πλευρά υπάρχει το λογότυπο της Google (Powered by Google). Στην δεξιά κάτω πλευρά τέλος υπάρχει μια επισκόπηση σε πολύ μικρότερη κλίμακα της περιοχής που φαίνεται στο χάρτη και δίπλα σε αυτήν, η Google μας ενημερώνει για το από πού προέρχονται τα δεδομένα που εμφανίζει και παρέχει έναν δεσμό (link) προς τους όρους χρήσης των χαρτών της.

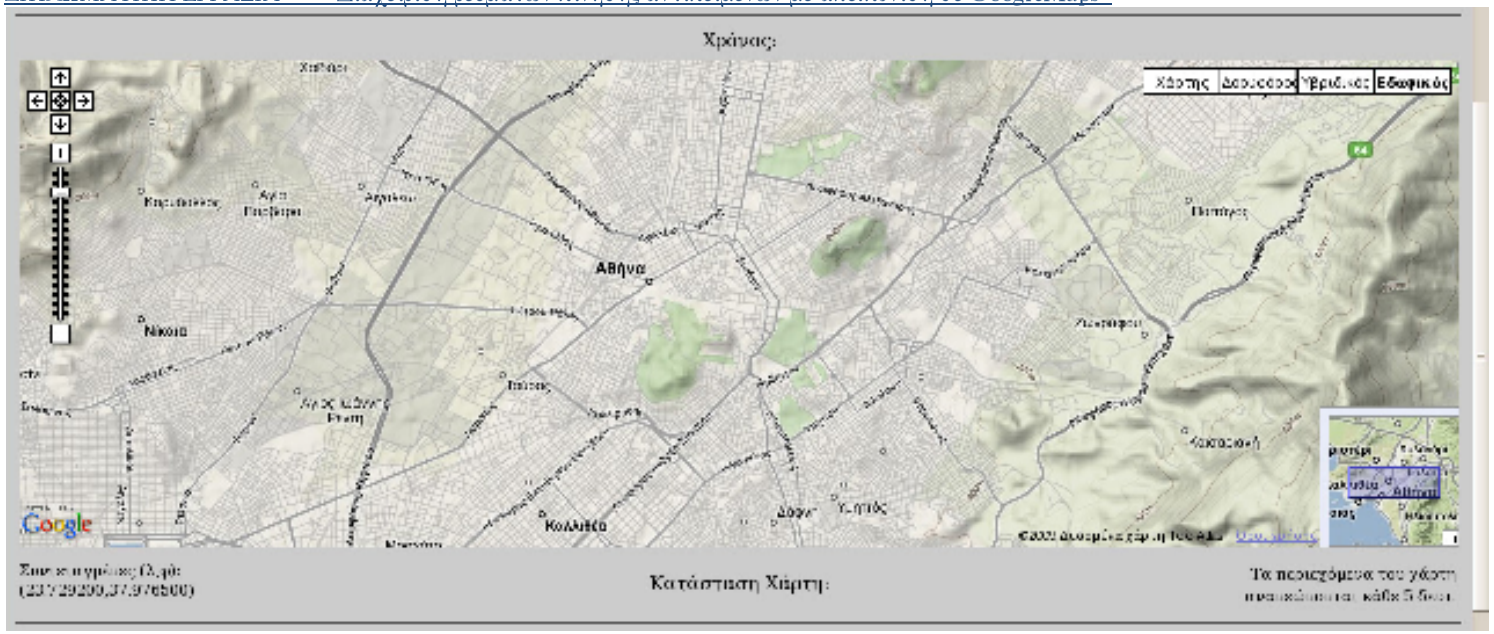
Στη συνέχεια, κάτω από τον χάρτη στα αριστερά υπάρχει πεδίο στο οποίο αναγράφονται οι "Συντεταγμένες (λ,φ): …" (Coordinates (long,lat): …) της θέσης όπου ο χρήστης έκανε τελευταία φορά κλικ επάνω στο χάρτη και αρχικά οι συντεταγμένες του κέντρου του χάρτη όταν ανοίγει η εφαρμογή. Αυτό σημαίνει ότι εάν θέλουμε να δούμε τις συντεταγμένες σε μια συγκεκριμένη θέση επάνω στον χάρτη, δεν έχουμε παρά να πάμε τον δείκτη του ποντικιού στο σημείο ενδιαφέροντος και να πατήσουμε το αριστερό κλικ. Όταν ανοίγει η εφαρμογή στη συγκεκριμένη θέση εμφανίζονται οι συντεταγμένες του κέντρου του χάρτη. Οι συντεταγμένες έχουν τη μορφή (γεωγραφικό μήκος, γεωγραφικό πλάτος). Υπενθυμίζουμε ότι το γεωγραφικό μήκος (longitude, λ) είναι η γωνία που σχηματίζει ο μεσημβρινός της τρέχουσας θέσης με το μεσημβρινό του Γκρίνουιτς (Greenwich) και είναι αντίστοιχο με τη μέτρηση στον άξονα Χ σε καρτεσιανά συστήματα. Το γεωγραφικό πλάτος (latitude, φ) είναι η γωνιακή απόσταση μεταξύ της παραλλήλου της συγκεκριμένης θέσης και του ισημερινού και είναι αντίστοιχο με τη μέτρηση στον άξονα Υ σε καρτεσιανά συστήματα. Συνηθίζεται οι συντεταγμένες να αναγράφονται αντίστροφα ((φ,λ) και όχι (λ,φ)) αλλά καθώς στο ερώτημα περιοχής και σε

κάθε ερώτημα καταχώρησης συντεταγμένων πρέπει να αναγραφούν ως (λ,φ) (κατ' αντιστοιχία με το ζεύγος (x,y) των καρτεσιανών συστημάτων), προτιμήθηκε αυτός ο τρόπος εμφάνισης κάτω από το χάρτη για την αποφυγή εσφαλμένων καταχωρήσεων στις συντεταγμένες των σχετικών ερωτημάτων. Στους χάρτες της Google θετικές τιμές στο γεωγραφικό μήκος σημαίνουν ότι η θέση ενδιαφέροντος ευρίσκεται Ανατολικά του Γκρίνουιτς (Greenwich) (μέχρι 180 μοίρες) ενώ αρνητικές τιμές σημαίνουν ότι η θέση ενδιαφέροντος ευρίσκεται Δυτικά του Γκρίνουιτς (Greenwich) (μέχρι 180 μοίρες). Θετικές τιμές έχει το γεωγραφικό πλάτος στο Βόρειο ημισφαίριο και αρνητικές στο Νότιο.

Δεξιά από τις συντεταγμένες και στο μέσο της ιστοσελίδας, ακριβώς κάτω από το χάρτη αναγράφεται η "Κατάσταση Χάρτη: ..." (Map Status: ...). Η κατάσταση αυτή ενημερώνει το χρήστη σχετικά με τα δεδομένα ποιας χρονική στιγμής εμφανίζονται επί του χάρτη ως αποτελέσματα των ερωτημάτων που εκτελούνται.

Κάτω από το χάρτη στα αριστερά παρέχεται στο χρήστη η πληροφορία ότι "Τα περιεχόμενα του χάρτη ανανεώνονται κάθε 5 δευτερόλεπτα." (Contents of map are refreshed every 5 seconds.). Ο κώδικας JavaScript που εκτελείται καθ' όλη τη διάρκεια που έχουμε ανοιχτή την εφαρμογή, έχει ρυθμιστεί έτσι ώστε να ανανεώνει όλα τα δυναμικά στοιχεία της εφαρμογής κάθε 5 δευτερόλεπτα. Δυναμικά στοιχεία της εφαρμογής είναι μεταξύ άλλων και οι πληροφορίες που εμφανίζονται στο χάρτη ως αποτελέσματα των ερωτημάτων που εκτελούνται ανά πάσα στιγμή.

Στα σχήματα 5.3 και 5.4 φαίνεται το δεύτερο αυτό τμήμα της εφαρμογής στην Ελληνική και Αγγλική έκδοση αντίστοιχα χωρίς οποιαδήποτε αποτελέσματα επί του χάρτη.



Σχήμα 5.3: 2ο τμήμα κεντρικής ιστοσελίδας εφαρμογής – Χάρτης και σχετικές πληροφορίες (Ελληνική έκδοση)

Σχήμα 5.4: 2ο τμήμα κεντρικής ιστοσελίδας εφαρμογής – Χάρτης και σχετικές πληροφορίες (Αγγλική έκδοση)

5.3.3 Τρίτο τμήμα κεντρικής ιστοσελίδας εφαρμογής – Ερωτήματα

Γενικά στοιχεία

Όπως έχει προαναφερθεί, η εφαρμογή έχει στόχο την παρακολούθηση και διαχείριση ομάδας κινούμενων αντικειμένων. Οι πληροφορίες για τις θέσεις των κινούμενων αντικειμένων εισρέουν υπό μορφή ρεύματος δεδομένων στο ΣΔΡΔ από εξωτερικό αρχείο τύπου απλού κειμένου. Μια ενδεικτική καταχώρηση είναι η εξής:

```
168;(23.720295,37.976281);0;'2004-01-31 22:00:00 EET'
```

Οι πληροφορίες που εισρέουν στο σύστημα διαχειρίζονται από ένα ρεύμα το οποίο ονομάζεται network.vehicles και ορίζεται ως εξής:

```
CREATE STREAM network.vehicles(
                vid integer,
                pos point,
                t integer,
                tcqtime timestamp TIMESTAMPCOLUMN
) TYPE ARCHIVED;
```

Βέβαια, όπως προαναφέρθηκε υπάρχει και ένα δεύτερο ρεύμα, το network.vehicles2 το οποίο έχει ακριβώς τα ίδια χαρακτηριστικά και ορίζεται με αντίστοιχο τρόπο.

Το τρίτο τμήμα της εφαρμογής διαχωρίζεται από το δεύτερο με μία οριζόντια γραμμή.

Ακριβώς κάτω από αυτήν τη γραμμή και στο αριστερό τμήμα της ιστοσελίδας υπάρχουν οι πληροφορίες, τα πεδία και τα πλαίσια-κουμπιά για την εκτέλεση του Βασικού Ερωτήματος.

Το **Βασικό Ερώτημα** ζητά από το TelegraphCQ να επιστρέψει ως απάντηση όλα τα κινούμενα αντικείμενα τα οποία θα εμφανιστούν τελικά στο χάρτη με τη βοήθεια της JavaScript. Το Βασικό Ερώτημα είναι το εξής:

```
SELECT vid, pos[0] AS lng, pos[1] AS lat, tcqtime FROM
network.vehicles [RANGE BY |5 seconds| SLIDE BY |5 seconds|
START AT |2004-01-31 22:00:00|];
```

Θα αναρωτηθεί κανείς γιατί δεν γράφουμε

"SELECT * FROM network.vehicles;"

αλλά θέτουμε ένα τέτοιο ερώτημα;

Στόχος μας είναι τα αποτελέσματα να εμφανίζονται στο χάρτη. Για να εμφανιστούν τα αποτελέσματα στον χάρτη θα πρέπει να τα προσπελάσει η JavaScript από εξωτερικό αρχείο, να τα επεξεργαστεί και να εμφανίσει τα επιθυμητά αποτελέσματα στο χάρτη. Προκειμένου να δημιουργηθούν εύκολα τα σημεία-σημάδια από την JavaScript και να εμφανιστούν στο χάρτη, θα πρέπει οι συντεταγμένες στις απαντήσεις των ερωτημάτων να διαχωριστούν σε δύο ξεχωριστά χαρακτηριστικά-πεδία. Έτσι λοιπόν, ο πρώτος αριθμός των συντεταγμένων που είναι το γεωγραφικό μήκος (longitude), επιλέγεται από τις απαντήσεις ως εξής: pos[0] και παίρνει το όνομα lng το οποίο στη συνέχεια θα αναζητήσει η JavaScript για τη δημιουργία των σημείων-σημαδιών επάνω στο χάρτη. Αντίστοιχα ο δεύτερος αριθμός των συντεταγμένων που είναι το γεωγραφικό πλάτος (latitude), επιλέγεται από τις απαντήσεις ως εξής: pos[1] και παίρνει το όνομα lat το οποίο στη συνέχεια θα αναζητήσει η JavaScript για τη δημιουργία των σημείων-σημαδιών επάνω στο χάρτη. Επίσης, προκειμένου να είναι διαθέσιμος στις απαντήσεις επί του χάρτη ο μοναδικός αριθμός του κάθε αντικειμένου (vid), πρέπει αυτός να ζητηθεί από το ερώτημα προς το TelegraphCQ. Ομοίως προκειμένου να είναι διαθέσιμη η χρονική στιγμή πρέπει αυτή να επιλεγεί στις απαντήσεις. Η χρονική στιγμή θα

αναζητηθεί από την JavaScript ως tcqtime και συνεπώς στο βασικό ερώτημα αυτή πρέπει να μείνει ως έχει.

Επίσης, στα συνεχή ερωτήματα πρέπει να ορίζουμε ένα παράθυρο που θα περιέχει τις τελευταίες καταχωρήσεις στο ρεύμα, πάνω στο οποίο θα εκτελείται το ερώτημα ανά πάσα στιγμή. Το παράθυρο σε όλα ουσιαστικά τα ερωτήματα που θα θέλει να θέσει κανείς στη συγκεκριμένη εφαρμογή, θα πρέπει να είναι χρονικό και μάλιστα, στα περισσότερα από αυτά, θα θέλει να αναφέρεται στα τελευταία δευτερόλεπτα.

Περιγραφή τρίτου τμήματος κεντρικής ιστοσελίδας εφαρμογής

Η παρούσα εφαρμογή υποστηρίζει την ταυτόχρονη εκτέλεση πέντε ερωτημάτων και την απεικόνιση των αποτελεσμάτων επί του χάρτη.

Για τη διαχείριση κάθε ερωτήματος, υπάρχουν σχετικά πλαίσια-κουμπιά στην κεντρική ιστοσελίδα. Τα στοιχεία που αφορούν κάθε ερώτημα είναι ομαδοποιημένα με βάση το ερώτημα στο οποίο αναφέρονται σε ειδικά χρωματισμένα πλαίσια ξεχωριστά για το καθένα. Κάθε πλαίσιο ερωτήματος εμφανίζει ομαδοποιημένες τις σχετικές πληροφορίες σε τρία μέρη από πάνω προς τα κάτω. Στο πρώτο μέρος του κάθε πλαισίου ερωτήματος αναγράφεται ο τίτλος του και για την περίπτωση του Βασικού Ερωτήματος και του Ερωτήματος Περιοχής, ορισμένες πολύ συνοπτικές πληροφορίες σχετικά με αυτά σε παρένθεση. Στο δεύτερο μέρος του κάθε πλαισίου υπάρχουν διάφορα πεδία για εισαγωγή διαφόρων πληροφοριών, ορισμένες απαραίτητες συνοπτικές οδηγίες-επεξηγήσεις-διευκρινήσεις σχετικά με τη διαχείριση του ερωτήματος και ορισμένα κουμπιά εκτέλεσης συγκεκριμένων ενεργειών. Στο δεύτερο μέρος του κάθε πλαισίου αναγράφεται η κατάσταση του κάθε ερωτήματος (εάν δηλαδή είναι ενεργό ή σταματημένο).

Βασικό ερώτημα

Έτσι λοιπόν, κάτω από την οριζόντια γραμμή που χωρίζει το δεύτερο από το τρίτο τμήμα της ιστοσελίδας στα αριστερά υπάρχει πλαίσιο χρωματισμένο με ένα ανοιχτό κόκκινο χρώμα για τη διαχείριση του Βασικού Ερωτήματος, στο επάνω μέρος του οποίου αναγράφεται:

Βασικό Ερώτημα (Επεξεργασία όλων των Κινούμενων Αντικειμένων)
(Basic Query (Process All Moving Objects))

Ακριβώς από κάτω, μετά από τη διαχωριστική γραμμή, υπάρχει πεδίο στο οποίο καταχωρείται η Μέγιστη Διάρκεια Εκτέλεσης του συγκεκριμένου ερωτήματος και η οποία έχει προκαθορισμένη τιμή τα 300 sec (όπως κάθε άλλο ερώτημα). Φυσικά ο χρήστης μπορεί να βάλει οποιαδήποτε ακέραια αριθμητική τιμή, αλλά προτείνεται αρχικά τουλάχιστον να χρησιμοποιήσει την προεπιλεγμένη τιμή.

Ακολουθεί πλαίσιο-κουμπί το οποίο ονομάζεται:

Εκτέλεση Βασικού Ερωτήματος (Run Basic Query)

Προφανώς πατώντας αυτό το κουμπί, ξεκινά η εκτέλεση του Βασικού Ερωτήματος και τα αποτελέσματά του θα εμφανιστούν μετά από μερικά δευτερόλεπτα στο χάρτη. Προϋπόθεση βέβαια για την επιτυχή εκτέλεση του Βασικού Ερωτήματος όπως και κάθε άλλου ερωτήματος στην παρούσα εφαρμογή είναι να είναι ενεργά τα ρεύματα δεδομένων που χρειάζονται για την εκτέλεσή του.

Ακολουθεί πλαίσιο-κουμπί το οποίο ονομάζεται:

Διακοπή Βασικού Ερωτήματος (Stop Basic Query)

Πατώντας αυτό το κουμπί σταματά η εκτέλεση του Βασικού Ερωτήματος.

Κάθε ερώτημα θα σταματήσει την εκτέλεσή του όταν πατηθεί το σχετικό κουμπί διακοπής ή όταν παρέλθει η μέγιστη διάρκεια εκτέλεσής του. Προϋπόθεση για την επιτυχή διακοπή του Βασικού Ερωτήματος όπως και κάθε άλλου ερωτήματος στην παρούσα εφαρμογή είναι να είναι ενεργά τα ρεύματα δεδομένων που χρειάζονται για την εκτέλεσή του. Αυτό συμβαίνει διότι η διακοπή εκτέλεσης του ερωτήματος σε κάθε περίπτωση γίνεται με έλεγχο εάν έχει παρέλθει ο μέγιστος χρόνος εκτέλεσής του ή εάν

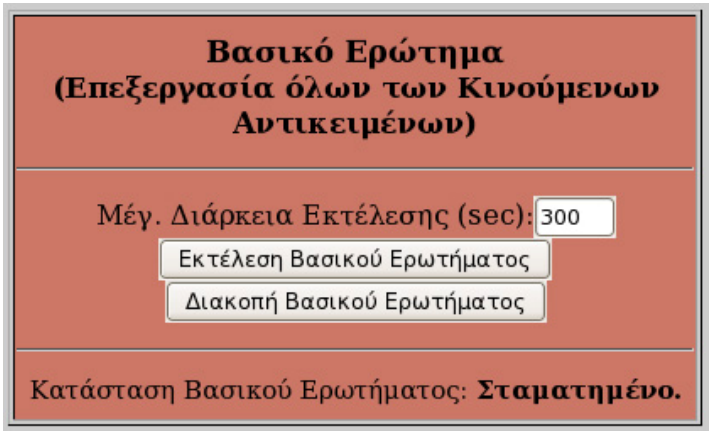
έχει πατηθεί το κουμπί διακοπής του. Ο έλεγχος γίνεται σε κάθε επανάληψη λήψης αποτελεσμάτων από το ΣΔΡΔ και για να πραγματοποιηθεί πρέπει να υπάρχουν αποτελέσματα. Έτσι, πρέπει να είναι ενεργά τα ρεύματα δεδομένων που χρειάζονται για την παραγωγή αποτελεσμάτων ώστε να μπορέσουν να διακοπούν σωστά τα ερωτήματα.

Ακριβώς από κάτω, μετά από τη δεύτερη διαχωριστική γραμμή εντός του πλαισίου του Βασικού Ερωτήματος, αναγράφεται η κατάσταση αυτού:

Κατάσταση Βασικού Ερωτήματος: Σταματημένο./Ενεργό...
(Basic Query Status: Stopped./Running...)

Προφανώς όταν το ερώτημα εκτελείται εμφανίζεται το κείμενο: "Ενεργό...", ενώ όταν δεν εκτελείται εμφανίζεται το κείμενο: "Σταματημένο.".

Στα σχήματα 5.5 και 5.6 φαίνεται το πλαίσιο του Βασικού Ερωτήματος στην Ελληνική και Αγγλική έκδοση αντίστοιχα με το ερώτημα να είναι σταματημένο.



Σχήμα 5.5: Πλαίσιο Βασικού Ερωτήματος με το ερώτημα σταματημένο (Ελληνική έκδοση)

Σχήμα 5.6: Πλαίσιο Βασικού Ερωτήματος με το ερώτημα σταματημένο (Αγγλική έκδοση)

Δεξιά από το πλαίσιο του Βασικού Ερωτήματος και στο μέσο της ιστοσελίδας, ακριβώς κάτω από τη διαχωριστική γραμμή μεταξύ δεύτερου και τρίτου τμήματος της ιστοσελίδας, υπάρχει ο δεσμός (link) προς τα Παραδείγματα Ερωτημάτων. Πιο συγκεκριμένα αναγράφεται:

Παραδείγματα Ερωτημάτων μπορούν να βρεθούν εδώ.
(Example Queries can be found here.)

Πατώντας λοιπόν ο χρήστης στη λέξη-δεσμό "εδώ" ("here") ανοίγει μια άλλη ιστοσελίδα η οποία περιέχει ορισμένα παραδείγματα ερωτημάτων. Πιο συγκεκριμένα, στην ιστοσελίδα αυτή αναγράφεται πρώτα το Βασικό Ερώτημα ενώ ακολουθούν 10 επιπλέον παραδείγματα ερωτημάτων τα οποία αναλύονται εκτενώς σε σχετική παράγραφο παρακάτω.

Από κάτω, μετά από μία οριζόντια διαχωριστική γραμμή, αναγράφεται το εξής κείμενο:

Πριν κλείσετε την εφαρμογή, παρακαλείσθε να σταματήσετε όλα τα ερωτήματα.

(Before Closing The Application, please Stop All Queries.)

και ακριβώς από κάτω υπάρχει σχετικό κουμπί με τίτλο:

Διακοπή Όλων των Ερωτημάτων

(Stop All Queries).

Η σημασία αυτής της προτροπής για διακοπή εκτέλεσης όλων των ερωτημάτων, αφορά στην διακοπή των ερωτημάτων ώστε να μην είναι ενεργά σε ενδεχόμενη επίσκεψη της ιστοσελίδας από άλλον χρήστη μέσα στα επόμενα λίγα λεπτά από το κλείσιμό της. Ταυτόχρονα πατώντας το κουμπί διακοπής όλων των ερωτημάτων γίνεται και μία μικρή εκκαθάριση στην εφαρμογή όπως περιγράφεται αναλυτικά στην παράγραφο 5.12. Τα αντίστοιχα βοηθητικά αρχεία του κάθε ερωτήματος, φροντίζει να σβήσει μετά από συγκεκριμένο χρόνο, το PHP αρχείο που εκτελείται κάθε φορά που ο χρήστης πατάει το σχετικό κουμπί διακοπής όλων των ερωτημάτων.

Η διακοπή όλων των ερωτημάτων με μία κίνηση είναι χρήσιμη και σε κάθε περίπτωση που ο χρήστης εκτελεί περισσότερα του ενός ερωτήματα και επιθυμεί να τα σταματήσει. Αντί να πατήσει δύο ή περισσότερα κουμπιά διακοπής ερωτημάτων μπορεί να πατήσει το περιγραφόμενο κουμπί και να έχει το ίδιο αποτέλεσμα.

Μετά το προπεριγραφόμενο κουμπί ακολουθεί οριζόντια διαχωριστική γραμμή και στη συνέχεια το κείμενο:

"Εκκαθάριση τροχιών από χάρτη"

("Cleanup Trajectories from map")

και ακολουθεί πλαίσιο-κουμπί με τίτλο "Εκκαθάριση τροχιών" ("Cleanup Trajectories").

Όταν εκτελούμε και σταματάμε διάφορα ερωτήματα η εφαρμογή έχει σχεδιαστεί έτσι ώστε οι τροχιές των αντικειμένων να διατηρούνται. Μετά από πολλαπλούς ελέγχους και δοκιμές διαφάνηκε η ανάγκη για εκκαθάριση των τροχιών κατά βούληση του χρήστη χωρίς να γίνει ανανέωση της ιστοσελίδας οπότε και δημιουργήθηκε το σχετικό πλαίσιο-κουμπί το οποίο κάνει ακριβώς αυτό το πράγμα.

Ερώτημα Περιοχής

Κάτω από την οριζόντια γραμμή που χωρίζει το δεύτερο από το τρίτο τμήμα της ιστοσελίδας στα δεξιά (δεξιά από τις παραπάνω περιγραφείσες πληροφορίες), υπάρχει πλαίσιο χρωματισμένο με ένα ανοιχτό ιώδες χρώμα για τη διαχείριση του Ερωτήματος Περιοχής, στο επάνω μέρος του οποίου αναγράφεται:

Ερώτημα Περιοχής (Απλώς *ορίστε* την περιοχή στο χάρτη)

Area Query (Just draw a polygon on the map)

Ακριβώς από κάτω, μετά από τη διαχωριστική γραμμή, αναγράφεται το κείμενο:

Συντ/νες Π.: (Pol. Coords:)

Δίπλα (από αριστερά προς τα δεξιά) υπάρχει πλαίσιο-κουμπί με τίτλο "On/Off", δίπλα αναγράφεται η λέξη "Off" και δίπλα υπάρχει πεδίο στο οποίο καταχωρούνται οι συντεταγμένες των κορυφών του πολυγώνου που ορίζει την περιοχή στην οποία επιθυμούμε να εμφανίσουμε τα όποια κινούμενα αντικείμενα.

Το "Συντ/νες Π." είναι συντομογραφία του Συντεταγμένες Πολυγώνου και φυσικά εννοούνται οι συντεταγμένες των κορυφών του πολυγώνου που ορίζει την περιοχή. Το "Pol. Coords" είναι συντομογραφία του Polygon Coordinates και δηλώνει τις παραπάνω συντεταγμένες.

Το πλαίσιο-κουμπί "On/Off" ενεργοποιεί και απενεργοποιεί τη δημιουργία πολυγώνου επί του χάρτη. Το "Off" που αναγράφεται δίπλα, θα αλλάξει σε "On" όταν ο χρήστης πατήσει το κουμπί "On/Off". Η λέξη δίπλα στο κουμπί "On/Off" δηλώνει δηλαδή εάν είναι ενεργοποιημένη ή όχι η σχεδίαση πολυγώνου επί του χάρτη.

Όταν λοιπόν είναι ενεργοποιημένη η σχεδίαση πολυγώνου, ο χρήστης πατώντας με το ποντίκι σε διάφορα σημεία επάνω στο χάρτη, δημιουργεί την περιοχή που επιθυμεί. Το πολύγωνο μπορεί να σχεδιαστεί δεξιόστροφα ή αριστερόστροφα χωρίς επιπτώσεις στην εκτέλεση του ερωτήματος.

Ενώ είναι ενεργοποιημένη η σχεδίαση του πολυγώνου, κάθε φορά που ο χρήστης πατάει με το ποντίκι επάνω στο χάρτη, οι αντίστοιχες συντεταγμένες καταχωρούνται αυτομάτως στο σχετικό πεδίο συντεταγμένων στο πλαίσιο του Ερωτήματος Περιοχής και μάλιστα με τη μορφή που χρειάζεται ώστε να εκτελεστεί το ερώτημα. Έτσι, το μόνο που πρέπει να κάνει ο χρήστης είναι αφού ενεργοποιήσει τη δημιουργία πολυγώνου, είναι να το σχεδιάσει στο χάρτη και να πατήσει το πλαίσιο-κουμπί με τίτλο "Εκτέλεση Ερωτήματος Περιοχής" που βρίσκεται λίγο παρακάτω μέσα στο πλαίσιο του συγκεκριμένου ερωτήματος. Σημειώνεται ότι μόλις ο χρήστης πατήσει την εκτέλεση του ερωτήματος, το πολύγωνο κλείνει αυτομάτως και συνεπώς δε χρειάζεται να προσπαθήσει να το κλείσει ο ίδιος με το ποντίκι. Ο χρήστης βέβαια μπορεί να εισάγει συντεταγμένες απ' ευθείας στο σχετικό πεδίο και να εκτελέσει το ερώτημα χωρίς να έχει σχεδιάσει κάτι επί του χάρτη.

Κάτω από το πεδίο συντεταγμένων, αναγράφεται το κείμενο:

Μέγ. Διάρκεια Εκτέλεσης (sec): (Set Max Exec. Time (sec):)

και δίπλα από αυτό υπάρχει πεδίο στο οποίο καταχωρείται σε δευτερόλεπτα ο χρονικός αυτός περιορισμός του συγκεκριμένου ερωτήματος και ο οποίος έχει προκαθορισμένη τιμή τα 300 sec (όπως κάθε άλλο ερώτημα). Ακολουθεί πλαίσιο-κουμπί το οποίο ονομάζεται:

Εκτέλεση Ερωτήματος Περιοχής (Run Area Query)

Από κάτω υπάρχει πλαίσιο-κουμπί το οποίο ονομάζεται:

Διακοπή Ερωτήματος Περιοχής (Stop Area Query)

ενώ κάτω από αυτό υπάρχει πλαίσιο-κουμπί το οποίο ονομάζεται:

Αφαίρεση Συντεταγμένων και Πολυγώνου
(Remove Coordinates and Polygon)

Τα κουμπιά "Εκτέλεση Ερωτήματος Περιοχής" και " Διακοπή Ερωτήματος Περιοχής " εκτελούν αυτό ακριβώς που αναφέρουν, όπως στο Βασικό Ερώτημα και σε κάθε άλλο ερώτημα.

Πατώντας ο χρήστης το κουμπί "Αφαίρεση Συντεταγμένων και Πολυγώνου", σβήνονται οι συντεταγμένες από το σχετικό πεδίο παραπάνω και αφαιρείται το πολύγωνο από το χάρτη.

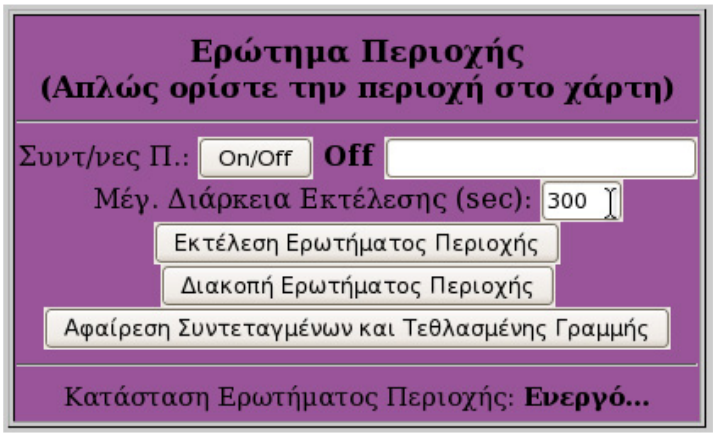
Ακολουθεί διαχωριστική γραμμή και κάτω από αυτήν, εμφανίζεται η "Κατάσταση Ερωτήματος Περιοχής: **Σταματημένο./Ενεργό...**" όπως αντίστοιχα συμβαίνει στο Βασικό Ερώτημα και στα Προσαρμοσμένα Ερωτήματα.

Το Ερώτημα Περιοχής είναι το εξής:

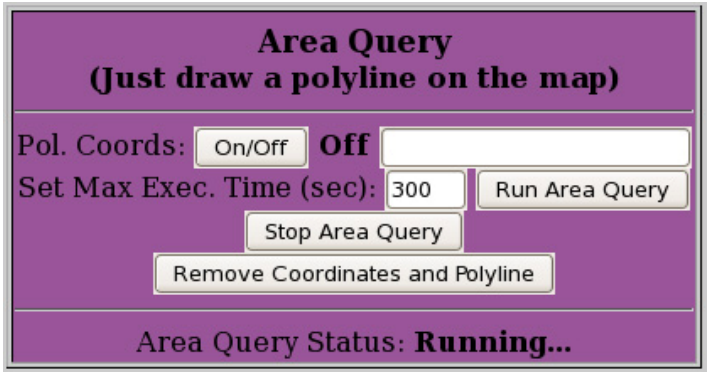
```
SELECT vid, pos[0] AS lng, pos[1] AS lat, tcqtime FROM
network.vehicles [RANGE BY |5 seconds| SLIDE BY |5 seconds|
START AT |2004-01-31 22:00:00|] WHERE pos@polygon|([οι
συντεταγμένες εισάγονται εδώ])|= TRUE;
```

Το πεδίο [οι συντεταγμένες εισάγονται εδώ] είναι η θέση όπου εισάγονται οι συντεταγμένες απ' ευθείας από το σχετικό πεδίο της ιστοσελίδας.

Στα σχήματα 5.7 και 5.8 φαίνεται το πλαίσιο του Ερωτήματος Περιοχής στην Ελληνική και Αγγλική έκδοση αντίστοιχα με το ερώτημα να είναι ενεργό.



Σχήμα 5.7: Πλαίσιο Ερωτήματος Περιοχής με το ερώτημα ενεργό (Ελληνική έκδοση)



Σχήμα 5.8: Πλαίσιο Ερωτήματος Περιοχής με το ερώτημα ενεργό (Αγγλική έκδοση)

Κάτω από το πλαίσιο του Βασικού Ερωτήματος, υπάρχει το πλαίσιο του 1ου Προσαρμοσμένου Ερωτήματος χρωματισμένο με ένα ανοιχτό πορτοκαλί χρώμα στο επάνω μέρος του οποίου αναγράφεται ο τίτλος του:

Προσαρμοσμένο Ερώτημα1 (Custom Query1)

Ακολουθεί 1η διαχωριστική γραμμή μέσα στο πλαίσιο του ερωτήματος και κάτω από αυτήν, υπάρχει το πεδίο εισαγωγής ερωτήματος από τον χρήστη. Μέσα στο πεδίο αυτό (όπως και σε κάθε πεδίο εισαγωγής ερωτήματος), όταν ανοίγει η ιστοσελίδα αναγράφονται κάποιες οδηγίες και πληροφορίες προς τον χρήστη της εφαρμογής οι οποίες είναι:

Εισάγετε εδώ το Ερώτημα1.
Προσοχή: Προτού εκτελέσετε οποιοδήποτε ερώτημα, παρακαλείσθε να διαβάσετε τις παραπάνω οδηγίες χρήσης διότι αλλιώς ίσως λάβετε αρκετά μηνύματα λάθους από το σύστημα.
Παρακαλείσθε επίσης να κοιτάξετε τα σχετικά παραδείγματα.

Ενώ στην Αγγλική έκδοση αναγράφεται:

Type Query1 here.
Important: Before executing any query, please read the instructions above or you may receive several Error messages from the system.
Please check also related examples.

Στο πεδίο αυτό λοιπόν, ο χρήστης μπορεί να εισάγει ένα δικό του ερώτημα προς εκτέλεση. Φυσικά θα πρέπει πρώτα να διαβάσει τις οδηγίες χρήσης και ενδείκνυται να συμβουλευτεί και τα σχετικά παραδείγματα.

Ακολουθεί η Μέγιστη Διάρκεια Εκτέλεσης για το συγκεκριμένο ερώτημα όπως συμβαίνει με κάθε ερώτημα άλλωστε και από κάτω ακολουθούν τα πλαίσια-κουμπιά "Εκτέλεση Ερωτήματος1" και "Διακοπή Ερωτήματος1" που εκτελούν και διακόπτουν αντίστοιχα το Προσαρμοσμένο Ερώτημα1.

Από κάτω ακολουθεί η δεύτερη διαχωριστική οριζόντια γραμμή του πλαισίου του ερωτήματος αυτού και κάτω από αυτήν, αναφέρεται η κατάσταση του Ερωτήματος1 κατ' αντιστοιχία με το Βασικό Ερώτημα και το Ερώτημα Περιοχής.

Ακριβώς κάτω από το πλαίσιο του Προσαρμοσμένου Ερωτήματος1 υπάρχει πλαίσιο χρωματισμένο με το ίδιο χρώμα με το Ερώτημα1 στο οποίο θα εμφανιστούν τα αποτελέσματα του ερωτήματος αυτού όταν εκτελείται. Πιο συγκεκριμένα, στο επάνω μέρος του πλαισίου αυτού αναγράφεται:

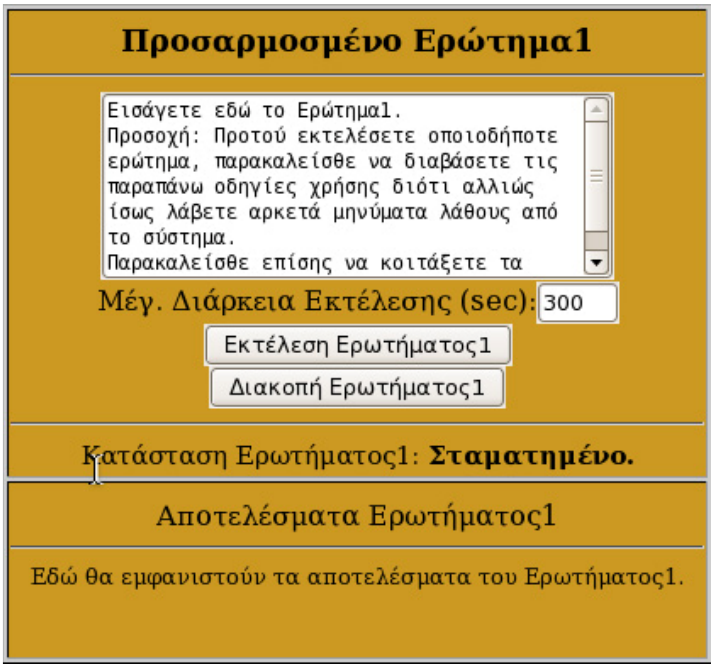
Αποτελέσματα Ερωτήματος1 (Results of Query1)

Ακολουθεί διαχωριστική οριζόντια γραμμή εντός του πλαισίου κάτω από την οποία, όταν δεν εκτελείται κάποιο ερώτημα (στο Προσαρμοσμένο Ερώτημα1) αναγράφεται:

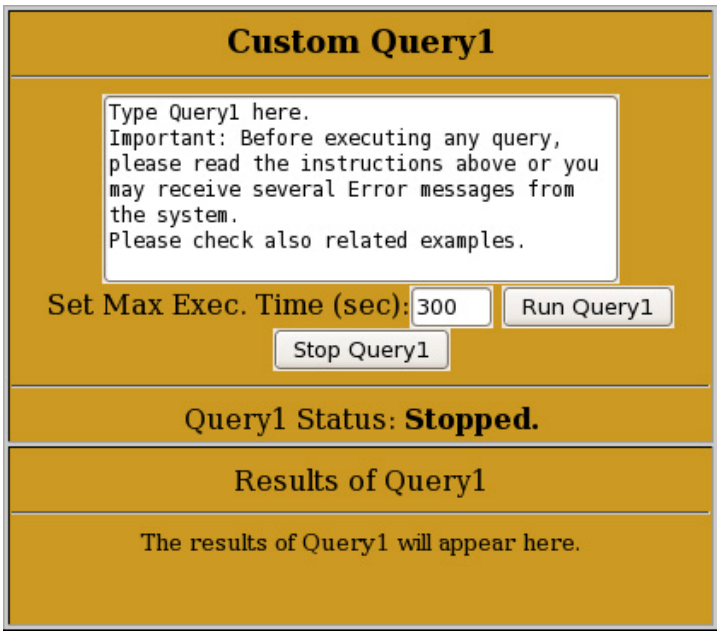
Εδώ θα εμφανιστούν τα αποτελέσματα του Ερωτήματος1.
(The results of Query1 will appear here.)

Όταν εκτελείται κάποιο ερώτημα στο Προσαρμοσμένο Ερώτημα1, αντί αυτού του κειμένου, εμφανίζεται ο πίνακας με τα αποτελέσματα του ερωτήματος.

Στα σχήματα 5.9 και 5.10 φαίνεται το πλαίσιο του Προσαρμοσμένου Ερωτήματος1 στην Ελληνική και Αγγλική έκδοση αντίστοιχα με το ερώτημα να είναι σταματημένο.



Σχήμα 5.9: Πλαίσιο Προσαρμοσμένου Ερωτήματος1 και πλαίσιο αποτελεσμάτων αυτού με το ερώτημα σταματημένο (Ελληνική έκδοση)

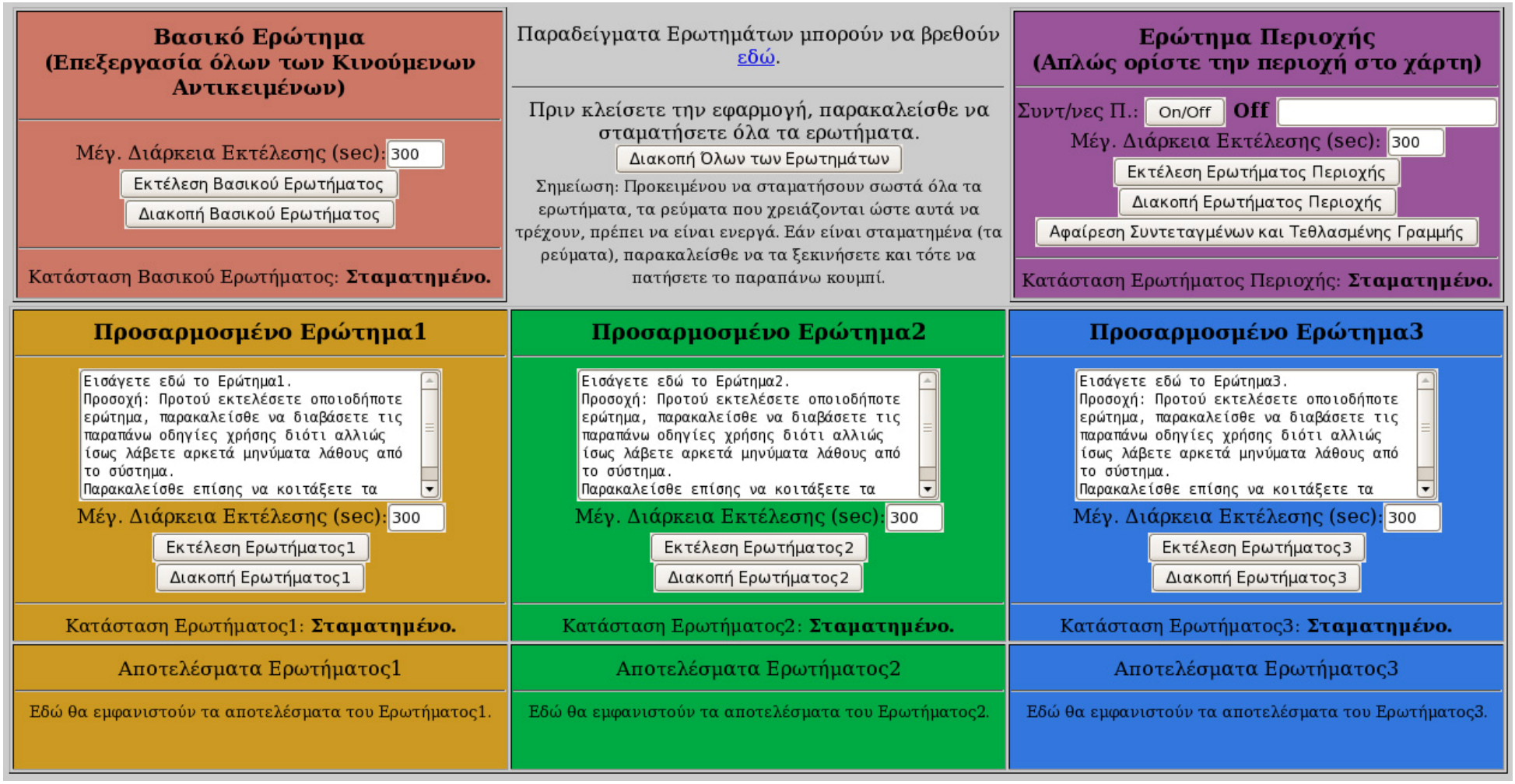


Σχήμα 5.10: Πλαίσιο Προσαρμοσμένου Ερωτήματος1 και πλαίσιο αποτελεσμάτων αυτού με το ερώτημα σταματημένο (Αγγλική έκδοση)

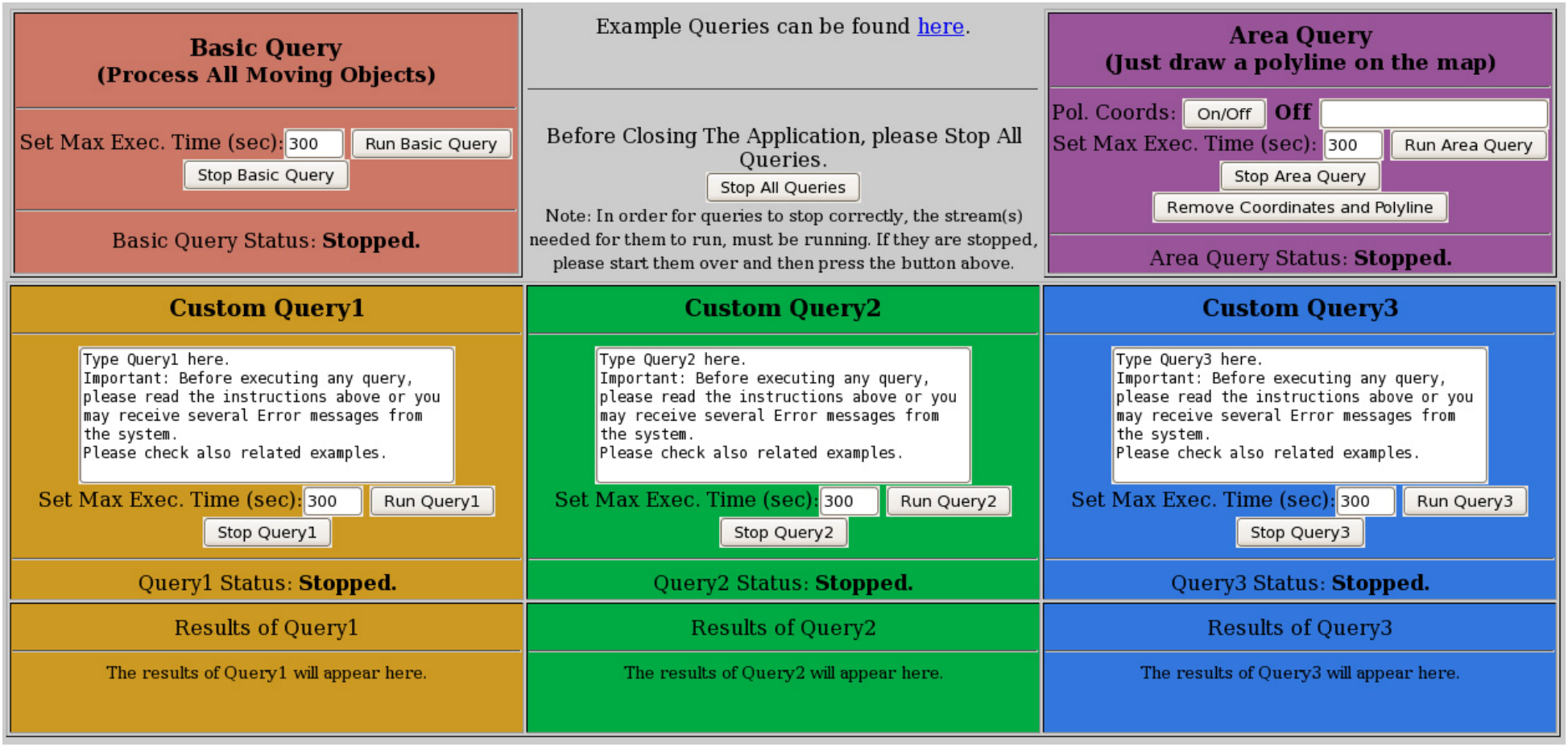
Δίπλα στο Προσαρμοσμένο Ερώτημα1 και στο μέσο της ιστοσελίδας, υπάρχει το πλαίσιο του Προσαρμοσμένου Ερωτήματος2 χρωματισμένο με ένα ανοιχτό πράσινο χρώμα και κάτω από αυτό υπάρχει το πλαίσιο των αποτελεσμάτων του στο ίδιο χρώμα με το πλαίσιο του ερωτήματος. Είναι δομημένο με τον ίδιο τρόπο με το Ερώτημα1, αναγράφοντας πρώτα τον τίτλο, μετά υπάρχει η αντίστοιχη οριζόντια διαχωριστική γραμμή, από κάτω το πεδίο εισαγωγής του ερωτήματος προς εκτέλεση, ακολουθεί το πεδίο της μέγιστης διάρκειας εκτέλεσης, τα πλαίσια-κουμπιά εκτέλεσης του ερωτήματος, η δεύτερη διαχωριστική γραμμή και τέλος η κατάσταση του συγκεκριμένου ερωτήματος. Το πλαίσιο αποτελεσμάτων του Ερωτήματος2 είναι και αυτό αντίστοιχα δομημένο με το πλαίσιο αποτελεσμάτων του $1^{ου}$ ερωτήματος.

Δίπλα στο Προσαρμοσμένο Ερώτημα2 στο δεξί μέρος της ιστοσελίδας, υπάρχει το πλαίσιο του Προσαρμοσμένου Ερωτήματος3 χρωματισμένο με ένα ανοιχτό μπλε χρώμα και κάτω από αυτό υπάρχει το πλαίσιο των αποτελεσμάτων του στο ίδιο χρώμα με το πλαίσιο του ερωτήματος. Ακολουθεί και αυτό την ίδια δομή με τα άλλα δύο προσαρμοσμένα ερωτήματα.

Στα σχήματα 5.11 και 5.12 φαίνεται (στην Αγγλική και Ελληνική έκδοση αντίστοιχα) ολόκληρο το τρίτο τμήμα της ιστοσελίδας με τα πλαίσια των πέντε ερωτημάτων, τον δεσμό (link) προς τα Παραδείγματα Ερωτημάτων και το πλαίσιο-κουμπί διακοπής όλων των ερωτημάτων με τις σχετικές του οδηγίες.



Σχήμα 5.11: 3ο τμήμα κεντρικής ιστοσελίδας εφαρμογής – Ερωτήματα και σχετικές πληροφορίες (Ελληνική έκδοση)



5.12: 3ο τμήμα κεντρικής ιστοσελίδας εφαρμογής – Ερωτήματα και σχετικές πληροφορίες (Αγγλική έκδοση)

Όπως έχει προαναφερθεί, όταν εκτελείται ένα ερώτημα, τα αποτελέσματά του εμφανίζονται στον χάρτη με συγκεκριμένα σημεία-σημάδια. Τα αποτελέσματα του Βασικού Ερωτήματος εμφανίζονται με ένα ανοιχτό κόκκινο χρώμα παρόμοιο με το φόντο του πλαισίου του Βασικού Ερωτήματος. Παρομοίως, τα αποτελέσματα του Ερωτήματος Περιοχής εμφανίζονται με ένα ιώδες χρώμα και τα αποτελέσματα του κάθε προσαρμοσμένου ερωτήματος στον αντίστοιχο χρωματισμό με το φόντο του κάθε πλαισίου ερωτήματος.

5.4 Παραδείγματα ερωτημάτων

Τα παραδείγματα ερωτημάτων είναι διαθέσιμα σε ξεχωριστή ιστοσελίδα και είναι τα εξής:

Βασικό Ερώτημα (Αυτό το ερώτημα εκτελείται όταν ζητάμε την εκτέλεση του Βασικού Ερωτήματος.):

```
SELECT vid, pos[0] AS lng, pos[1] AS lat, tcqtime FROM
network.vehicles [RANGE BY |5 seconds| SLIDE BY |5 seconds|
START AT |2004-01-31 22:00:00|];
```

Παράδειγμα 1 (Παρακολούθηση συγκεκριμένου κινούμενου αντικειμένου.):

```
SELECT vid, pos[0] AS lng, pos[1] AS lat, tcqtime FROM
network.vehicles [RANGE BY |5 seconds| SLIDE BY |5 seconds|
START AT |2004-01-31 22:00:00|] WHERE vid=587;
```

Παράδειγμα 2 (Ερώτημα περιοχής. Αντίστοιχο με αυτό που θα εκτελεστεί όταν ζητήσουμε την εκτέλεση του Ερωτήματος Περιοχής από την κεντρική ιστοσελίδα της εφαρμογής.):

```
SELECT vid, pos[0] AS lng, pos[1] AS lat, tcqtime FROM
network.vehicles [RANGE BY |5 seconds| SLIDE BY |5 seconds|
START AT |2004-01-31 22:00:00|] WHERE
pos@polygon|(23.710,37.984,23.728,37.985,23.747,37.975,23.744,
37.955,23.713,37.959)|= TRUE;
```

Παράδειγμα 3 (Παρακολούθηση συγκεκριμένου κινούμενου αντικειμένου – διαφορετικού από το Παράδειγμα 1.):

```
SELECT vid, pos[0] AS lng, pos[1] AS lat, tcqtime FROM
network.vehicles [RANGE BY |5 seconds| SLIDE BY |5 seconds|
START AT |2004-01-31 22:00:00|] WHERE vid=384;
```

Παράδειγμα 4 (Πληροφορίες για συγκεκριμένο κινούμενο αντικείμενο. Χωρίς latitude, longitude.):

```
SELECT vid AS id, tcqtime FROM network.vehicles [RANGE BY
|5 seconds| SLIDE BY |5 seconds| START AT |2004-01-31
22:00:00|] WHERE vid=703;
```

Παράδειγμα 5 (Ερώτημα περιοχής. Χωρίς latitude, longitude.):

```
SELECT vid, tcqtime FROM network.vehicles [RANGE BY |5
seconds| SLIDE BY |5 seconds| START AT |2004-01-31 22:00:00|]
WHERE
pos@polygon|(23.710,37.984,23.728,37.985,23.747,37.975,23.744,
37.955,23.713,37.959)|= TRUE;
```

Παράδειγμα 6 (Ερώτημα απόστασης από συγκεκριμένο σταθερό σημείο.):

```
SELECT vid, pos[0] AS lng, pos[1] AS lat, tcqtime FROM
network.vehicles [RANGE BY |5 seconds| SLIDE BY |5 seconds|
```

```
START AT |2004-01-31 22:00:00|] WHERE (pos <-> Point
|(23.728,37.985)|) <= 0.02;
```

Παράδειγμα 7 (Ερώτημα συνάθροισης. Μέτρηση πλήθους κινούμενων αντικειμένων που ανά πάσα στιγμή βρίσκονται εντός συγκεκριμένης περιοχής.):

```
SELECT tcqtime, COUNT(vid) AS cnt FROM network.vehicles
[RANGE BY |5 seconds| SLIDE BY |5 seconds| START AT |2004-01-
31 22:00:00|] WHERE
(pos@polygon|(23.710,37.984,23.728,37.985,23.747,37.975,23.744
,37.955,23.713,37.959)|)= TRUE GROUP BY tcqtime HAVING
COUNT(vid)>=2;
```

Παράδειγμα 8 (Ερώτημα με σύνδεση ρευμάτων – χρειάζεται δύο ρεύματα δεδομένων. Επιστρέφει ανά πάσα στιγμή πίνακα με τις αποστάσεις όλων των οχημάτων από το όχημα με αριθμό 427.):

```
SELECT V1.vid AS vehicle1, V2.vid AS vehicle2,
CAST((V1.pos <-> V2.pos) AS float) AS distance, V1.tcqtime
FROM network.vehicles V1 [RANGE BY |5 seconds| SLIDE BY |5
seconds| START AT |2004-01-31 22:00:00|], network.vehicles2 V2
[RANGE BY |5 seconds| SLIDE BY |5 seconds| START AT |2004-01-
31 22:00:00|] WHERE V1.vid = 427 AND V2.vid <> 427 AND
V1.t=V2.t;
```

Παράδειγμα 9 (Ερώτημα με σύνδεση ρευμάτων – χρειάζεται δύο ρεύματα δεδομένων. Επιστρέφει ζευγάρια οχημάτων που ανά πάσα στιγμή είναι σε απόσταση μικρότερη των 0.005 μοιρών. Εμφάνιση αποτελεσμάτων μόνο σε πίνακα.):

```
SELECT V1.vid AS veh1, V2.vid AS veh2, CAST((V1.pos <->
V2.pos) AS float) AS distance, V1.tcqtime FROM
network.vehicles V1 [RANGE BY '5 seconds' SLIDE BY '5 seconds'
START AT '2004-01-31 22:00:00'], network.vehicles2 V2 [RANGE
BY '5 seconds' SLIDE BY '5 seconds' START AT '2004-01-31
22:00:00'] WHERE CAST((V1.pos <-> V2.pos) AS float) < 0.005
AND V1.vid <> V2.vid AND V1.t=V2.t;
```

Παράδειγμα 10 (Ερώτημα με σύνδεση ρευμάτων – χρειάζεται δύο ρεύματα δεδομένων. Επιστρέφει ζευγάρια οχημάτων που ανά πάσα στιγμή είναι σε απόσταση μικρότερη των 0.005 μοιρών. Εμφάνιση αποτελεσμάτων στο χάρτη και σε πίνακα.):

```
SELECT V1.vid AS vid, V1.pos[0] AS lng, V1.pos[1] AS lat,
V2.vid AS veh2, CAST((V1.pos <-> V2.pos) AS float) AS
distance, V1.tcqtime FROM network.vehicles V1 [RANGE BY |5
seconds| SLIDE BY |5 seconds| START AT |2004-01-31 22:00:00|],
network.vehicles2 V2 [RANGE BY |5 seconds| SLIDE BY |5
seconds| START AT |2004-01-31 22:00:00|] WHERE CAST((V1.pos <-
> V2.pos) AS float) < 0.005 AND V1.vid <> V2.vid AND
V1.t=V2.t;
```

Πρώτα αναφέρεται το Βασικό Ερώτημα το οποίο μπορεί ο καθένας να εισάγει σε οποιοδήποτε από τα Προσαρμοσμένα Ερωτήματα. Όπως προαναφέρθηκε, τα αποτελέσματα του Βασικού Ερωτήματος φαίνονται μόνο στο χάρτη και όχι σε κάποιον πίνακα. Εκτελώντας λοιπόν το ερώτημα αυτό ως ένα προσαρμοσμένο ερώτημα, ο χρήστης μπορεί να δει τα αποτελέσματα και σε μορφή πίνακα.

Το Παράδειγμα 1 είναι ένα ερώτημα που στόχο έχει την εμφάνιση επί του χάρτη ενός συγκεκριμένου κινούμενου αντικειμένου με βάση τον μοναδικό κωδικό αριθμό του (vid=587).

Το Παράδειγμα 2 είναι ένα ερώτημα περιοχής με συγκεκριμένες προκαθορισμένες συντεταγμένες με παρόμοια σύνταξη με το Ερώτημα Περιοχής που μπορεί να εκτελέσει ο χρήστης από την ιστοσελίδα της εφαρμογής.

Το Παράδειγμα 3 είναι ένα ερώτημα όμοιο με το Παράδειγμα 1 με διαφορετικό μοναδικό κωδικό αριθμό που στόχο έχει να δείξει τη δυναμικότητα της εφαρμογής και να ωθήσει τον χρήστη να δοκιμάσει διαφορετικά vid για να παρακολουθήσει όποιο άλλο όχημα επιθυμεί. Τα 15 οχήματα του ρεύματος δεδομένων έχουν τα παρακάτω vid:

84, 168, 247, 303, 327, 343, 384, 392, 427, 532, 580, 587, 595, 703 και 709

Αλλάζοντας λοιπόν είτε στο Παράδειγμα 1 είτε στο 3 το vid με οποιοδήποτε από τα παραπάνω μπορεί ο χρήστης να παρακολουθήσει όποιο όχημα επιθυμεί.

Το Παράδειγμα 4 είναι ένα ερώτημα που επιστρέφει πληροφορίες για συγκεκριμένο κινούμενο αντικείμενο χωρίς όμως τις συντεταγμένες του. Σε αυτήν την περίπτωση τα αποτελέσματα φαίνονται μόνο στον πίνακα κάτω από το πλαίσιο του Προσαρμοσμένου Ερωτήματος.

Το Παράδειγμα 5 είναι αντίστοιχο με το Παράδειγμα 4 με κριτήριο επιλογής των αποτελεσμάτων το πολύγωνο του Παραδείγματος 2. Τα αποτελέσματα εμφανίζονται μόνο σε πίνακα.

Το Παράδειγμα 6 είναι ένα ερώτημα το οποίο επιστρέφει όλα τα κινούμενα αντικείμενα που ανά πάσα στιγμή βρίσκονται σε απόσταση μικρότερη από 0,02 μοίρες από συγκεκριμένο σταθερό σημείο. Τα αποτελέσματα εμφανίζονται στο χάρτη και σε πίνακα.

Το Παράδειγμα 7 είναι ερώτημα συνάθροισης. Μέσα στο χρονικό παράθυρο των 5 τελευταίων δευτερολέπτων μετρά πόσα αντικείμενα βρίσκονται εντός συγκεκριμένης πολυγωνικής περιοχής. Τα αποτελέσματα εμφανίζονται μόνο σε πίνακα. Δε θα μπορούσε άλλωστε η πληροφορία του πλήθους των αντικειμένων να εμφανίζεται στο χάρτη.

Το Παράδειγμα 8 είναι ένα ερώτημα που απαιτεί σύνδεση του ρεύματος με το εαυτό του. Καθώς το TelegraphCQ δεν υποστηρίζει σύνδεση ενός ρεύματος με τον εαυτό του, πρέπει τα δύο ρεύματα με τα ίδια χαρακτηριστικά που έχουν δημιουργηθεί γι' αυτόν το σκοπό να είναι ενεργά. Πρέπει μάλιστα τα δεδομένα να αρχίσουν να ρέουν και στα δύο ταυτόχρονα κάτι εφικτό στην παρούσα εφαρμογή χάρη στη συνδυασμένη χρήση της JavaScript, της PHP και της HTML. Το ερώτημα λοιπόν αυτό επιστρέφει ανά πάσα στιγμή πίνακα με τις αποστάσεις όλων των οχημάτων από το κινούμενο όχημα με αριθμό 427. Τα αποτελέσματα εμφανίζονται μόνο σε πίνακα.

Το Παράδειγμα 9 είναι και αυτό ένα ερώτημα που απαιτεί σύνδεση του ρεύματος με το εαυτό του. Όπως όλα τα ερωτήματα με σύνδεση ρευμάτων (ουσιαστικά το ίδιο ρεύμα δύο φορές), έτσι κι αυτό χρειάζεται δύο ρεύματα δεδομένων να είναι ενεργά. Το ερώτημα αυτό επιστρέφει ζευγάρια κινούμενων οχημάτων που ανά πάσα στιγμή βρίσκονται σε απόσταση μεταξύ τους μικρότερη των 0,005 μοιρών. Οι 0,005 μοίρες είναι περίπου 555 m σε γεωγρ. πλάτος (lat) και 438 m σε γεωγρ. μήκος (long) στην περιοχή της Αθήνας και στις διαγώνιους των αξόνων (45° γωνία) οι 0,005 μοίρες ανάγονται σε 500 m με πολύ καλή ακρίβεια. Στα αποτελέσματα του ερωτήματος περιέχεται ο αριθμός του πρώτου οχήματος, ο αριθμός του δεύτερου οχήματος, η μεταξύ τους απόσταση ως αριθμός κινητής υποδιαστολής (floating number) και η χρονική στιγμή στην οποία αναφέρονται. Δεν περιέχεται το γεωγραφικό μήκος και πλάτος των οχημάτων και συνεπώς αυτά δεν εμφανίζονται επί του χάρτη.

Τέλος, το Παράδειγμα 10 είναι επέκταση του 9 με επιλογή εκτός από τα παραπάνω χαρακτηριστικά και των συντεταγμένων των οχημάτων του πρώτου ρεύματος. Καθώς ο περιορισμός μεταξύ των οχημάτων του πρώτου ρεύματος και του δεύτερου είναι οι αριθμοί τους να είναι απλώς διαφορετικοί, στα αποτελέσματα εμφανίζεται το κάθε ζευγάρι δύο φορές με τους αριθμούς των οχημάτων αντεστραμμένους και επάνω στο χάρτη εμφανίζονται τα ζευγάρια μία φορά καθώς εμφανίζονται μόνο τα οχήματα του πρώτου ρεύματος.

Έχει δοκιμαστεί με επιτυχία η εκτέλεση δύο ερωτημάτων με βάση το Παράδειγμα 10 τα οποία έχουν ως περιορισμό μεταξύ των οχημάτων ο αριθμός του πρώτου να είναι μικρότερος και ο αριθμός του πρώτου να είναι μεγαλύτερος. Στην περίπτωση αυτή τα αποτελέσματα μοιράζονται στα δύο ερωτήματα και στους πίνακες και στον χάρτη. Ο χάρτης σε αυτήν την περίπτωση απεικονίζει με δύο διαφορετικά χρώματα τα μέλη του κάθε ζεύγους οχημάτων, τα μισά με το χρώμα των απαντήσεων του ενός ερωτήματος και τα άλλα μισά με το χρώμα των απαντήσεων του άλλου ερωτήματος.

5.5 Περιγραφή εφαρμογής – Βασικές λειτουργίες

Στην παράγραφο αυτή γίνεται μια πιο αναλυτική περιγραφή των βασικών λειτουργιών της εφαρμογής.

5.5.1 Εργαλεία εφαρμογής

Προτού προχωρήσουμε στην περιγραφή των βασικών λειτουργιών κρίνεται σκόπιμο να γίνει εκ νέου μια σύντομη αναφορά στα εργαλεία που χρησιμοποιεί αυτή η εφαρμογή προκειμένου να επιτύχει τον στόχο της.

Η παρούσα εφαρμογή χρησιμοποιεί τον κώδικα **HTML**, τη γλώσσα προγραμματισμού **JavaScript**, τη γλώσσα προγραμματισμού **PHP**, αρχεία **XML**, το Σύστημα Διαχείρισης Βάσεων Δεδομένων **PostgreSQL**, το Σύστημα Διαχείρισης Ρευμάτων Δεδομένων **TelegraphCQ**, τη γλώσσα προγραμματισμού **C++** και τη γλώσσα προγραμματισμού Perl. Το TelegraphCQ, απ' όσο γνωρίζουμε, μέχρι σήμερα έχει δοκιμαστεί με επιτυχία μόνο σε λειτουργικό σύστημα **Linux**, οπότε η βάση δεδομένων με το ρεύμα δεδομένων βρίσκονται σε database server με εγκατεστημένο λειτουργικό σύστημα Linux.

5.5.2 Εκκίνηση Ρευμάτων Δεδομένων

Όταν ο χρήστης πατήσει το πλαίσιο-κουμπί "Εκκίνηση Ενός Ρεύματος Δεδομένων", εκτελείται ένα συγκεκριμένο αρχείο PHP (“startStream.php”). Αυτό το αρχείο περιέχει την απαιτούμενη εντολή για την εκκίνηση του Ρεύματος Δεδομένων στο ρεύμα "network.vehicles". Δημιουργεί επίσης ένα αρχείο XML (“zstreamRunning.xml”) που χρησιμοποιείται από την JavaScript για να ελέγξει εάν το συγκεκριμένο Ρεύμα Δεδομένων είναι ενεργό. Όταν το αρχείο XML υπάρχει σημαίνει ότι το Ρεύμα Δεδομένων είναι ενεργό. Όταν δεν υπάρχει σημαίνει ότι το Ρεύμα είναι σταματημένο. Όταν τελειώνει το Ρεύμα Δεδομένων, σχετική εντολή στο ίδιο αρχείο PHP διαγράφει το XML αρχείο. Η κατάσταση του ρεύματος εμφανίζεται στην ιστοσελίδα με τη βοήθεια της JavaScript.

Όταν ο χρήστης πατήσει το πλαίσιο-κουμπί "Εκκίνηση δύο όμοιων Ρευμάτων Δεδομένων", εκτελούνται δύο συγκεκριμένα αρχεία PHP. Το πρώτο αρχείο PHP είναι το ίδιο με το παραπάνω που εκτελείται όταν ο χρήστης πατήσει το "Εκκίνηση Ενός Ρεύματος Δεδομένων" (“zstreamRunning.xml”). Το δεύτερο αρχείο PHP (“startStream2.php”) είναι παρόμοιο με το πρώτο αλλά ξεκινά το ίδιο Ρεύμα Δεδομένων στο ρεύμα "network.vehicles2" και δημιουργεί αντίστοιχα ένα δεύτερο αρχείο XML (“zstreamRunning2.xml”) για τον ίδιο λόγο που το πρώτο PHP δημιουργεί το σχετικό XML. Εάν υπάρχει το δεύτερο αρχείο XML, τότε η JavaScript το αναγνωρίζει και δίνει στην κατάσταση του δεύτερου Ρεύματος Δεδομένων την τιμή "Ενεργό" όπως αντίστοιχα κάνει και με το πρώτο αρχείο XML. Όταν τελειώνει το Ρεύμα Δεδομένων, σχετική εντολή στο συγκεκριμένο (δεύτερο) αρχείο PHP διαγράφει το αντίστοιχο XML αρχείο και η JavaScript αλλάζει την τιμή της κατάστασης του Ρεύματος σε "Σταματημένο" όπως αντίστοιχα γίνεται και με το πρώτο αρχείο XML.

Προκειμένου τα δύο ρεύματα δεδομένων να στέλνουν ταυτόχρονα τα ίδια δεδομένα στο σύστημα, πρέπει πριν τα εκτελέσουμε να βεβαιωθούμε ότι και τα δύο ρεύματα είναι σταματημένα. Σε πραγματικές συνθήκες λειτουργίας της εφαρμογής, θα είναι ενεργά συνεχώς και τα δύο ρεύματα δεδομένων και δε θα χρειάζεται να τα ξεκινήσουμε πατώντας σχετικά κουμπιά στην εφαρμογή.

5.5.3 Εκτέλεση Βασικού Ερωτήματος

Όταν ο χρήστης πατήσει το πλαίσιο-κουμπί "Εκτέλεση Βασικού Ερωτήματος", ένα συγκεκριμένο αρχείο PHP, διαφορετικό από τα δύο παραπάνω εκτελείται με συγκεκριμένες παραμέτρους (query=0, refreshT=5) που του δίνει ο κώδικας HTML. Το αρχείο ονομάζεται execQuery_getResults.php. Σε αυτήν την περίπτωση, το ερώτημα που εκτελείται είναι το Βασικό Ερώτημα το οποίο είναι αποθηκευμένο μέσα στο αρχείο PHP και μπορεί επίσης να βρεθεί στα Παραδείγματα Ερωτημάτων αλλά και στην ιστοσελίδα των οδηγιών χρήσης. Το συγκεκριμένο αρχείο PHP συνδέεται με την PostgreSQL και το TelegraphCQ και στέλνει το Βασικό Ερώτημα προς εκτέλεση. Τα αποτελέσματα αποστέλλονται από το TelegraphCQ και την PostgreSQL στο αρχείο PHP το οποίο με συγκεκριμένες εντολές δημιουργεί συγκεκριμένο αρχείο XML αποτελεσμάτων αποθηκεύοντάς τα εκεί. Το αρχείο XML, που ονομάζεται “results0.xml” διαβάζει η JavaScript και εμφανίζει τα αποτελέσματα επί του χάρτη.

Παρακάτω φαίνονται τυπικά αποτελέσματα του επαναληπτικά δημιουργούμενου “results0.xml”.

```
<?xml version="1.0" ?>
- <markers>
<marker vid="327" lng="23.697756" lat="37.993996" tcqtime="2004-01-31 22:01:30" />
<marker vid="303" lng="23.724296" lat="37.969959" tcqtime="2004-01-31 22:01:30" />
<marker vid="84" lng="23.706249" lat="37.988356" tcqtime="2004-01-31 22:01:30" />
<marker vid="427" lng="23.718778" lat="37.961198" tcqtime="2004-01-31 22:01:30" />
<marker vid="587" lng="23.748376" lat="37.964372" tcqtime="2004-01-31 22:01:30" />
<marker vid="168" lng="23.715002" lat="37.969345" tcqtime="2004-01-31 22:01:30" />
<marker vid="703" lng="23.710057" lat="37.979323" tcqtime="2004-01-31 22:01:30" />
<marker vid="709" lng="23.706903" lat="37.976543" tcqtime="2004-01-31 22:01:30" />
<marker vid="343" lng="23.730184" lat="37.967271" tcqtime="2004-01-31 22:01:30" />
<marker vid="392" lng="23.744885" lat="37.973205" tcqtime="2004-01-31 22:01:30" />
<marker vid="384" lng="23.720206" lat="37.976244" tcqtime="2004-01-31 22:01:30" />
<marker vid="247" lng="23.70969" lat="37.979006" tcqtime="2004-01-31 22:01:30" />
<marker vid="532" lng="23.701542" lat="37.975513" tcqtime="2004-01-31 22:01:30" />
<marker vid="580" lng="23.749706" lat="37.975161" tcqtime="2004-01-31 22:01:30" />
<marker vid="595" lng="23.751701" lat="37.977987" tcqtime="2004-01-31 22:01:30" />
</markers>
```

Τυπικά αποτελέσματα αρχείου “results0.xml” όπως εμφανίζονται εάν ανοίξουμε το αρχείο με τον Internet Explorer.

5.5.4 Εκτέλεση Ερωτήματος Περιοχής

Όταν ο χρήστης πατήσει το πλαίσιο-κουμπί "Εκτέλεση Ερωτήματος Περιοχής", το ίδιο με το Βασικό Ερώτημα αρχείο PHP (execQuery_getResults.php), εκτελείται με συγκεκριμένες διαφορετικές παραμέτρους (query=6, refreshT=5) που του δίνει ο κώδικας HTML. Φυσικά, για να εκτελεστεί το ερώτημα πρέπει ο χρήστης να έχει εισάγει συντεταγμένες στο σχετικό πεδίο της ιστοσελίδας. Το κουμπί "On/Off" ουσιαστικά εκτελεί μια συνάρτηση στην JavaScript η οποία αλλάζει την κατάσταση δημιουργίας πολυγώνου από On σε Off και αντίστροφα και δίνει συγκεκριμένη τιμή σε μία μεταβλητή. Όταν ο χρήστης κάνει κλικ επάνω στο χάρτη τότε ανάλογα με την τιμή αυτής της μεταβλητής θα δημιουργηθεί ή όχι το πολύγωνο και θα περαστούν ή όχι οι συντεταγμένες στο σχετικό πεδίο. Σε αυτήν την περίπτωση, το ερώτημα που εκτελείται

είναι το Ερώτημα Περιοχής το οποίο είναι αποθηκευμένο εν μέρει μέσα στο αρχείο PHP ενώ τις συντεταγμένες λαμβάνει από το σχετικό πεδίο του ερωτήματος. Το ερώτημα αυτό μπορεί να βρεθεί στην ιστοσελίδα των οδηγιών χρήσης και έχει παρόμοια δομή με το Παράδειγμα 2 που περιγράφηκε παραπάνω. Το συγκεκριμένο αρχείο PHP συνδέεται με το ΣΔΒΔ PostgreSQL και το ΣΔΡΔ TelegraphCQ και στέλνει το Ερώτημα Περιοχής προς εκτέλεση. Τα αποτελέσματα αποστέλλονται από το TelegraphCQ και την PostgreSQL στο αρχείο PHP το οποίο με συγκεκριμένες εντολές δημιουργεί συγκεκριμένο αρχείο XML ("results1.xml") αποθηκεύοντάς τα εκεί (όπως συμβαίνει και με το Βασικό Ερώτημα, μόνο που το αρχείο XML έχει διαφορετικό όνομα). Το αρχείο XML διαβάζει η JavaScript και εμφανίζει τα αποτελέσματα επί του χάρτη (όπως συμβαίνει και με το Βασικό Ερώτημα, αλλά με διαφορετικό χρώμα των σημείων-σημαδιών).

5.5.5 Εκτέλεση Προσαρμοσμένων Ερωτημάτων

Όταν ο χρήστης πατήσει το πλαίσιο-κουμπί "Εκτέλεση Ερωτήματος1", "Εκτέλεση Ερωτήματος2" ή "Εκτέλεση Ερωτήματος3", το ίδιο με το Βασικό Ερώτημα αρχείο PHP (execQuery_getResults.php), εκτελείται με συγκεκριμένες διαφορετικές παραμέτρους (refreshT=5 και query=1, query=2, query=3 αντίστοιχα) που του δίνει ο κώδικας HTML. Φυσικά, για να εκτελεστεί το κάθε ερώτημα πρέπει ο χρήστης να έχει εισάγει το ερώτημα προς εκτέλεση στα σχετικά πεδία και μάλιστα ακολουθώντας τους κανόνες που αναφέρονται στις οδηγίες χρήσης. Τότε, το αρχείο PHP ελέγχει πρώτα το ερώτημα, που στέλνει ο χρήστης προς εκτέλεση, εάν περνάει από τους ελέγχους που αναφέρθηκαν στις οδηγίες χρήσης (λέξη κλειδί "SELECT", λέξη κλειδί "FROM", άλλες λέξεις κλειδιά, κενά πεδία κλπ). Εάν το ερώτημα που εισήγαγε ο χρήστης δεν απορριφθεί από κανέναν έλεγχο, τότε η PHP συνδέεται με το σύστημα διαχείρισης βάσεων δεδομένων PostgreSQL και το σύστημα διαχείρισης ρευμάτων δεδομένων TelegraphCQ και στέλνει το προσαρμοσμένο ερώτημα προς εκτέλεση. Η διαδικασία από αυτό το στάδιο και μετά είναι η ίδια με το Βασικό Ερώτημα και το Ερώτημα Περιοχής με μία διαφορά: Όταν η JavaScript διαβάζει το αρχείο XML που δημιουργήθηκε από το κάθε προσαρμοσμένο ερώτημα, δημιουργεί και πίνακα με τα αποτελέσματα ο οποίος εμφανίζεται στο σχετικό πεδίο του κάθε προσαρμοσμένου ερωτήματος στην κεντρική ιστοσελίδα της εφαρμογής.

5.5.6 Διακοπή Εκτέλεσης Ερωτημάτων

Τα ερωτήματα διακόπτονται είτε όταν παρέλθει η μέγιστη διάρκεια εκτέλεσής τους, είτε όταν πατηθεί το σχετικό πλαίσιο-κουμπί διακοπής τους, είτε όταν πατηθεί το σχετικό πλαίσιο-κουμπί διακοπής όλων των ερωτημάτων, είτε αυτόματα μετά από 20 δευτερόλεπτα περίπου από την έναρξη εκτέλεσής τους, εάν διαπιστωθεί ότι δεν επιστρέφουν απαντήσεις. Στις τρεις πρώτες περιπτώσεις, για να διακοπεί η εκτέλεση οποιουδήποτε ερωτήματος πρέπει τα ρεύματα δεδομένων που χρειάζονται για την εκτέλεσή του να είναι ενεργά. Αυτό συμβαίνει διότι ο έλεγχος μέσα στον κώδικα PHP για την διακοπή ή όχι εκτέλεσης του ερωτήματος γίνεται στην επανάληψη ενός βρόχου κάθε φορά που θα επιστραφούν απαντήσεις από το ΣΔΡΔ, εάν δεν τρέχει το ρεύμα, δε θα επιστραφούν απαντήσεις και συνεπώς το αρχείο PHP θα σταματήσει την εκτέλεσή του περιμένοντας απαντήσεις από το TelegraphCQ.
Όταν ο χρήστης πατήσει κάποιο πλαίσιο-κουμπί διακοπής συγκεκριμένου ερωτήματος, εκτελείται συγκεκριμένο αρχείο PHP διαφορετικό από τα προηγούμενα ("stopQuery.php"). Αυτό το αρχείο PHP δέχεται μία παράμετρο με διαφορετική για κάθε ερώτημα τιμή και δημιουργεί συγκεκριμένο αρχείο XML για κάθε περίπτωση. Η παράμετρος ονομάζεται squery και για το Βασικό Ερώτημα παίρνει την τιμή 0, για το

Ερώτημα Περιοχής την τιμή 6 και για τα Προσαρμοσμένα Ερωτήματα 1,2 και 3 την τιμή 1,2 και 3 αντίστοιχα. Το αρχείο XML ονομάζεται "zstopQuery#.xml" όπου #=0,6,1,2,3 για το κάθε ένα από τα παραπάνω ερωτήματα αντίστοιχα. Όταν ανιχνευθεί το συγκεκριμένο δημιουργούμενο αρχείο XML από το "execQuery_getResults.php" που εκτελεί το κάθε ερώτημα, το τελευταίο αυτό αρχείο PHP απλά διακόπτει την εκτέλεση του ερωτήματος, σβήνοντας και το αρχείο XML με τα αποτελέσματα ("results#.xml") και το βοηθητικό αρχείο XML που χρησίμευσε για την διακοπή του ερωτήματος. Στο σημείο αυτό πρέπει να αναφερθεί ότι το execQuery_getResults.php αφού συνδεθεί με τη βάση δεδομένων και το ρεύμα δεδομένων, δημιουργεί ένα επιπλέον αρχείο XML ("zQuery#IsRunning.xml") το οποίο ανιχνεύεται από την JavaScript και εμφανίζεται στην ιστοσελίδα η κατάσταση του συγκεκριμένου ερωτήματος ως ενεργό. Έτσι, όταν σταματάει το ερώτημα διαγράφεται εκτός από τα "results#.xml" και "zstopQuery#.xml" και το "zQuery#IsRunning.xml".

Όταν ο χρήστης πατήσει το πλαίσιο-κουμπί διακοπής όλων των ερωτημάτων, εκτελείται συγκεκριμένο αρχείο PHP διαφορετικό από τα προηγούμενα ("stopAllQueries.php") το οποίο δημιουργεί και τα 5 αρχεία XML διακοπής όλων των ερωτημάτων. Όποιο ερώτημα είναι ενεργό και επιστέφει αποτελέσματα θα διακοπεί και το αρχείο PHP που το εκτελούσε execQuery_getResults.php θα σβήσει τα σχετικά αρχεία XML με τα αποτελέσματα ("results#.xml"), τα αρχεία XML που χρησίμευσαν για τη διακοπή εκτέλεσης των ερωτημάτων ("zstopQuery#.xml") και τα αρχεία που με την ύπαρξή τους δηλώνουν την κατάσταση του κάθε ερωτήματος ως ενεργό ("zQuery#IsRunning.xml"). Τα υπόλοιπα αρχεία XML διακοπής ερωτημάτων, καταχώρησης απαντήσεων και κατάστασης ερωτημάτων θα διαγραφούν μετά από μερικά δευτερόλεπτα από το αρχείο PHP που δημιούργησε τα αρχεία XML διακοπής δηλ. το "stopAllQueries.php". Το "stopAllQueries.php" κάνει και κάτι σαν εκκαθάριση της εφαρμογής φροντίζοντας να σταματήσει και πέντε πιθανά ερωτήματα τα οποία δεν επιστρέφουν απαντήσεις και έχουν κατά κάποιο τρόπο κολλήσει. Περισσότερες λεπτομέρειες σχετικά με τη λειτουργία αυτού του αρχείου PHP αναφέρονται στην παράγραφο 5.12 παρακάτω.

5.5.7 Απλά εισαγωγικά και κατακόρυφη μπάρα

Όταν η HTML στέλνει το Προσαρμοσμένο Ερώτημα στο σχετικό αρχείο PHP (execQuery_getResults.php), τότε, εάν υπάρχουν απλά εισαγωγικά, το κείμενο του πεδίου του ερωτήματος δεν φτάνει σωστά στο αρχείο PHP λόγω περιορισμών στην επικοινωνία και αποστολή δεδομένων μεταξύ HTML και PHP. Αυτό το πρόβλημα επικοινωνίας μεταξύ κώδικα HTML και κώδικα PHP έχει λυθεί αντικαθιστώντας τα απλά εισαγωγικά (') με την κατακόρυφη μπάρα (¦). Το αρχείο PHP έχει σχετική εντολή με την οποία, αφού λάβει το ερώτημα προς εκτέλεση, αντικαθιστά την κατακόρυφη μπάρα (¦) με απλά εισαγωγικά (') και το ερώτημα εκτελείται χωρία προβλήματα τέτοιου είδους.

5.5.8 Ερωτήματα χωρίς αποτελέσματα

Όταν δεν υπάρχουν αποτελέσματα, καθώς τα ερωτήματα είναι συνεχή, η PHP θα περιμένει για πάντα απάντηση από την PostgreSQL. Θα σταματήσει μάλιστα η εκτέλεση του αρχείου PHP στην αναμονή για απάντηση από την PostgreSQL. Προκειμένου λοιπόν να διακοπεί η αναμονή, να σταματήσει η εκτέλεση του αρχείου PHP αλλά κυρίως να κλείσει η σύνδεση με τη βάση δεδομένων και το ρεύμα δεδομένων, ενεργοποιείται συγκεκριμένο πρόγραμμα γραμμένο σε C++ το οποίο τερματίζει το μπλοκαρισμένο PHP και την μπλοκαρισμένη σύνδεση με τη β.δ. και το ρ.δ.. Προκειμένου να τερματιστεί η εκτέλεση του σωστού αρχείου PHP και η σωστή σύνδεση, πρέπει να

τηρηθούν τα αναγραφόμενα με έντονα γράμματα στις οδηγίες χρήσης. Πιο συγκεκριμένα, σχετικό πρόγραμμα σε C++ εκτελείται συνεχώς και ελέγχει ανά τακτά χρονικά διαστήματα εάν έχει ζητηθεί η διακοπή συγκεκριμένου ερωτήματος που δεν επιστρέφει αποτελέσματα.

Όταν ο χρήστης πατήσει το κουμπί εκτέλεσης ενός ερωτήματος, εκτός από το "execQuery_getResults.php" εκτελείται και το "stopQuery#WithNoResults.php". Το αρχείο αυτό ελέγχει μετά από την πάροδο 2 δευτερολέπτων εάν υπάρχει το "zQuery#IsRunning.xml" σε συνδυασμό με το εάν δεν υπάρχει το "results#.xml". Εάν λοιπόν το πρώτο υπάρχει και το δεύτερο δεν υπάρχει τότε μια μεταβλητή μετρητής αυξάνει την τιμή της κατά 1 (από 0 σε 1). Μετά από 5, 10 και 15 δευτερόλεπτα η διαδικασία επαναλαμβάνεται. Εάν λοιπόν ο μετρητής λάβει την τιμή 4 σημαίνει ότι στα προηγούμενα 17 δευτερόλεπτα (συνολικά 4 δοκιμές και 4 επαναλήψεις δημιουργίας του "results#"" εάν το ερώτημα είχε αποτελέσματα) το ερώτημα δεν έχει επιστρέψει κανένα αποτέλεσμα και θα πρέπει να διακοπεί διότι πιθανότατα δεν θα επιστρέψει κανένα αποτέλεσμα όσο και να περιμένουμε. Φυσικά οι έλεγχοι μπορούν να αυξηθούν αλλά κάτι τέτοιο υπό τις παρούσες συνθήκες της εφαρμογής, μάλλον δεν έχει νόημα.

Πιο συγκεκριμένα, εάν το ερώτημα δεν έχει επιστρέψει κανένα αποτέλεσμα τα τελευταία 17 δευτερόλεπτα, το PHP αρχείο δημιουργεί ένα βοηθητικό XML με όνομα "deleteFetch#.xml". Το πρόγραμμα της C++ που στην ουσία εκτελείται συνεχώς, κάθε 4 δευτερόλεπτα ελέγχει εάν υπάρχει το αρχείο "deleteFetch#.xml" και εάν ναι τότε διακόπτει την εκτέλεση της σχετικής σύνδεσης της PHP με την PostgreSQL αλλά και τη σχετική σύνδεση του Apache (httpd) που χρησιμοποιεί η PHP. Επίσης σβήνει το "deleteFetch#.xml" αλλά και το "zQuery#IsRunning.xml". Έτσι διακόπτεται η εκτέλεση του ερωτήματος που δεν επιστρέφει αποτελέσματα. Λόγω περιορισμών στις εντολές διακοπής διεργασιών, για να εντοπιστεί αυτόματα από κάποιο πρόγραμμα ο μοναδικός αριθμός της διεργασίας (process) συγκεκριμένης σύνδεσης PostgreSQL και συγκεκριμένης σύνδεσης Apache θα πρέπει ο χρήστης να περιμένει 4-5 δευτερόλεπτα ανάμεσα στην εκκίνηση εκτέλεσης ερωτημάτων που πιθανόν να μην επιστρέφουν αποτελέσματα. Έτσι το πρόγραμμα που έχει δημιουργηθεί με την C++ θα τερματίσει την σωστή σύνδεση της PHP με την PostgreSQL και τη σωστή σύνδεση του Apache (httpd). Συνεπώς ο χρήστης, στην περίπτωση το ερώτημα που υποβάλλει πιθανόν να μην επιστρέφει αποτελέσματα, θα πρέπει να περιμένει 4-5 δευτερόλεπτα προτού εκτελέσει οποιοδήποτε άλλο ερώτημα είτε σε αυτό είτε σε άλλο πεδίο.

5.5.9 Συντήρηση της εφαρμογής

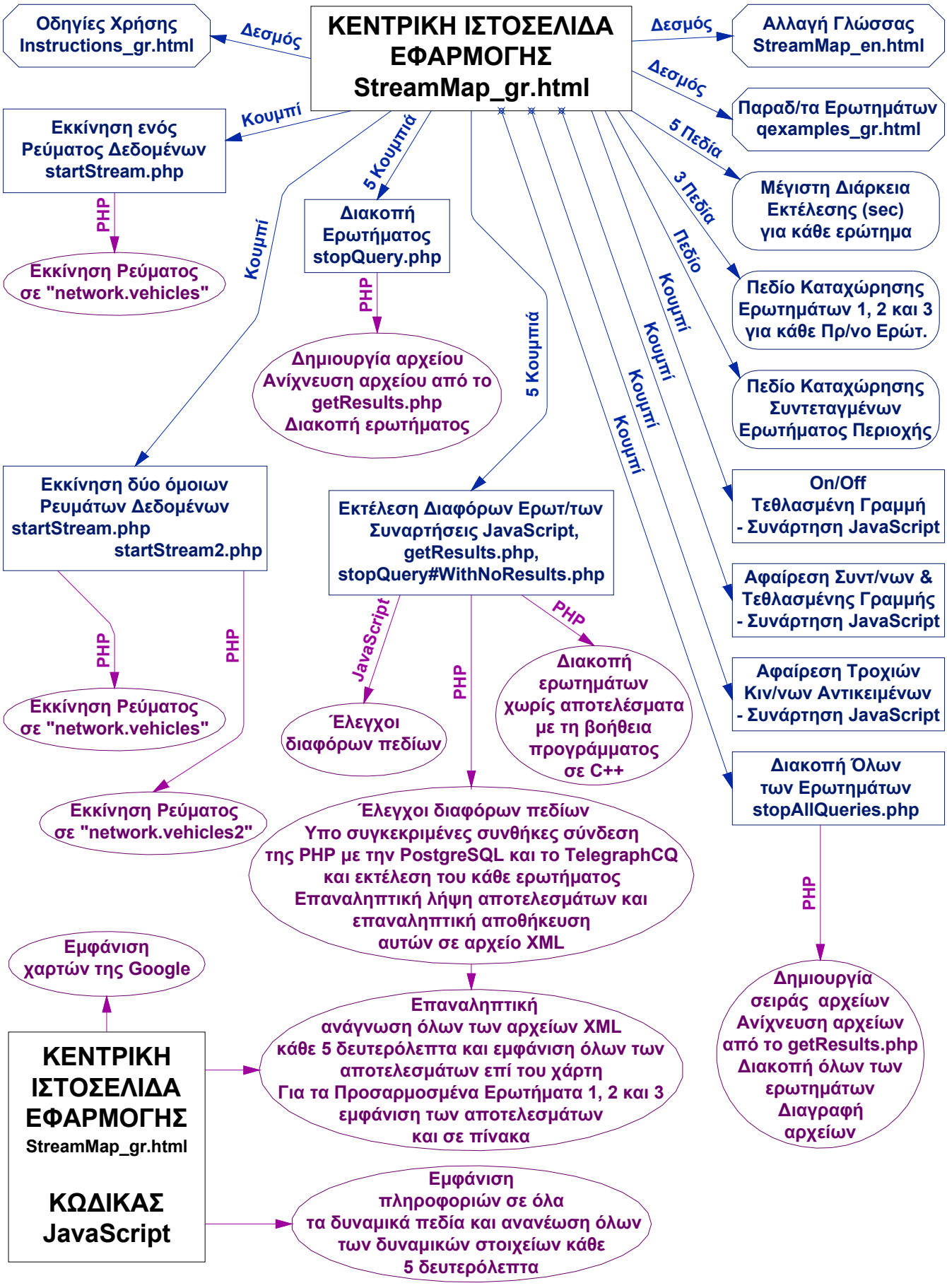
Προκειμένου να διασφαλιστεί εσαεί η κανονική λειτουργία της εφαρμογής, δημιουργήθηκε πρόγραμμα γραμμένο σε C++ το οποίο κάθε 24 ώρες τερματίζει όλες τις συνδέσεις της PHP με τη b.d. και το p.d., τερματίζει όλα τα εκτελούμενα αρχεία PHP, τερματίζει τα ρεύματα δεδομένων και σβήνει όσα αρχεία δημιουργούνται δυναμικά από την εφαρμογή (αρχεία με αποτελέσματα ερωτημάτων και αρκετά βοηθητικά αρχεία που δημιουργούνται και διαγράφονται από διάφορα εκτελούμενα scripts και προγράμματα). Τα αρχεία αυτά είναι η ομάδα των "results#.xml", η ομάδα "zQuery#IsRunning.xml", η ομάδα "zstopQuery#.xml", η ομάδα "deleteFetch#.xml" όπου η δίεση # παίρνει τις τιμές 0,1,2,3,6, η ομάδα "zq#dbconnID.log", η ομάδα zq#httpdID.log και τα αρχεία "zstreamRunning.xml" και "zstreamRunning2.xml".

5.5.10 Διάγραμμα βασικών οντοτήτων και λειτουργιών εφαρμογής

Παρακάτω παρουσιάζονται διαγραμματικά οι βασικές οντότητες και λειτουργίες της εφαρμογής:

Διπλωματική Εργασία: Διαχείριση ρευμάτων κίνησης αντικειμένων με απεικόνιση σε GoogleMaps

Διαγραμματική Παρουσίαση Βασικών Οντοτήτων και Λειτουργιών Εφαρμογής 1



Διάγραμμα 1 **Κωνσταντίνος Δ. Θεοφιλογιαννάκος**

Σημαντικότερες Λειτουργίες Εφαρμογής

Δεσμοί προς άλλες ιστοσελίδες με χρήση HTML και JavaScript

Δυνατότητα εκκίνησης ενός ή δύο Ρευμάτων Δεδομένων με χρήση PHP

Εμφάνιση κατάστασης Ρευμάτων Δεδομένων και ερωτημάτων με χρήση JavaScript

Εμφάνιση χαρτών της Google για απεικόνιση των κινούμενων αντικειμένων και των τροχιών τους με χρήση JavaScript

Εκκαθάριση εφαρμογής κάθε 24ώρες με χρήση C++ για την εύρυθμη και εσαεί λειτουργία αυτής

Δυνατότητα ταυτόχρονης υποβολής διαφόρων συνεχών ερωτημάτων (μέχρι 5 ταυτόχρονα) με χρήση PHP: Βασικό Ερώτημα, Ερώτημα Περιοχής και τρία Προσαρμοσμένα Ερωτήματα όπου ο χρήστης εισάγει ερωτήματα της αρεσκείας του

Έλεγχος των σχετικών με τα ερωτήματα πεδίων με χρήση JavaScript και PHP πριν την εκτέλεση των ερωτημάτων για την αποφυγή σφαλμάτων και ανεπιθύμητων ενεργειών

Επαναληπτική εμφάνιση αποτελεσμάτων ερωτημάτων επί των χαρτών Google με χρήση JavaScript και XML (κινούμενα αντικείμενα και τροχιές αυτών)

Επαναληπτική εμφάνιση αποτελεσμάτων ερωτημάτων σε σχετικούς πίνακες με χρήση JavaScript

Αυτόματη διακοπή ερωτημάτων που δεν επιστρέφουν αποτελέσματα με χρήση PHP και C++

Απλή χειροκίνητη διακοπή Ερωτημάτων με χρήση PHP

Αυτόματη διακοπή Ερωτημάτων μετά από προκαθορισμένο χρόνο με χρήση PHP

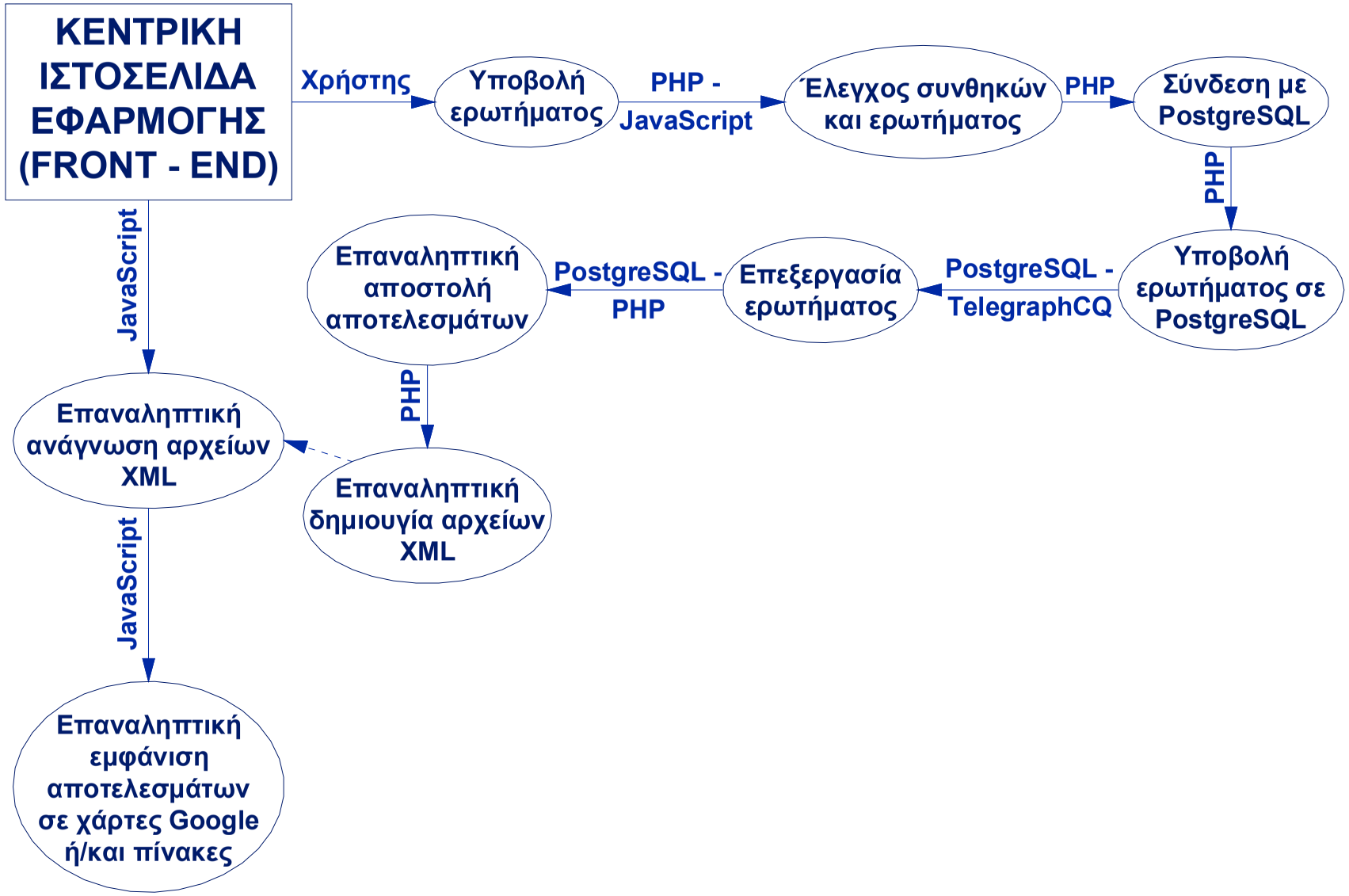
Γραφική υποβολή Ερωτήματος Περιοχής με χρήση JavaScript και PHP

Δυνατότητα εκκαθάρισης τροχιών με χρήση JavaScript

Δυνατότητα διακοπής όλων των Ερωτημάτων και μικρής κλίμακας εκκαθάρισης με χρήση PHP και C++

Διάγραμμα 2

Διάγραμμα ροής σημαντικότερων λειτουργιών εφαρμογής

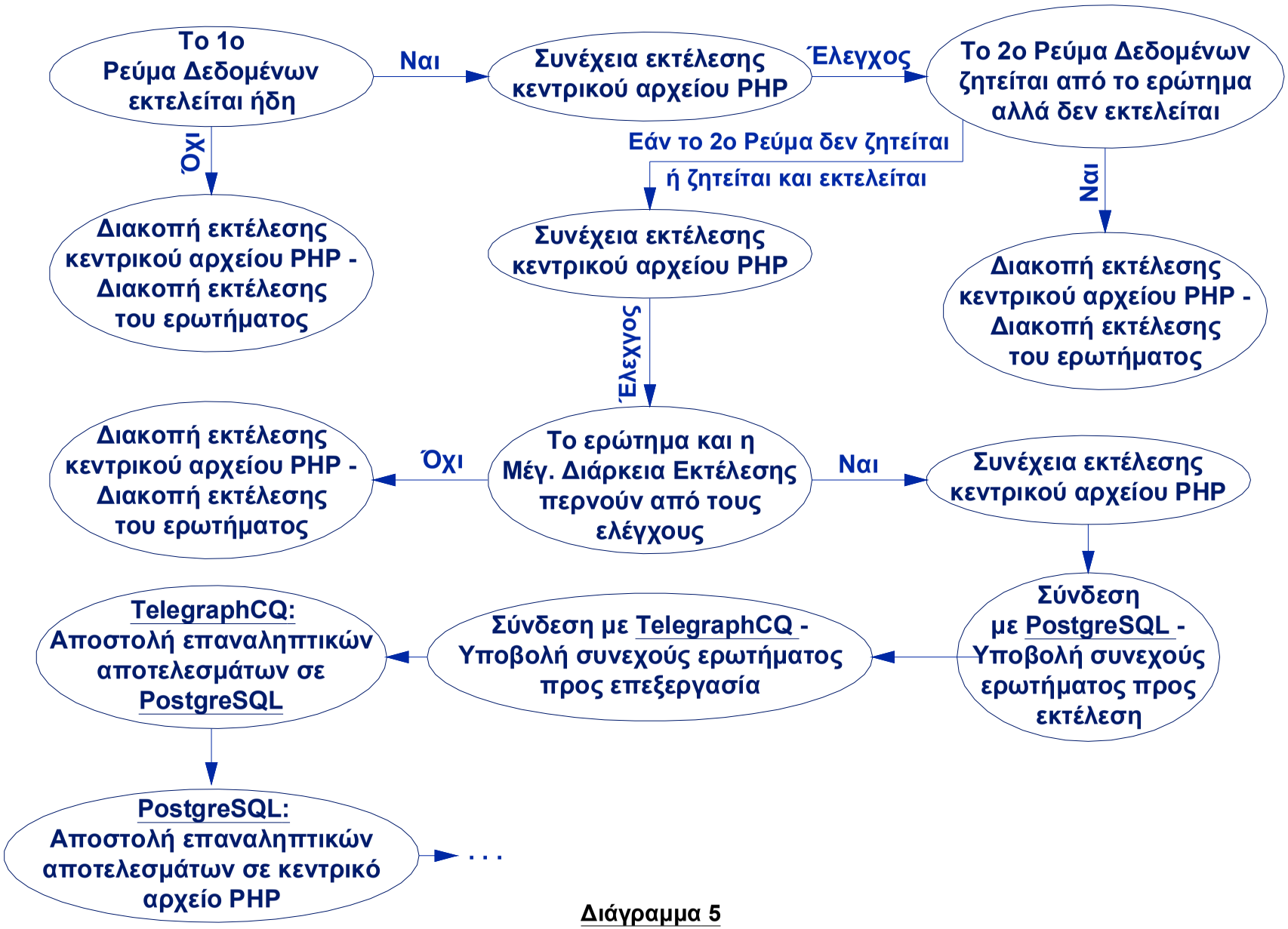


Διάγραμμα 3

Διαγραμματική Παρουσίαση Διαδικασιών Όταν Ζητείται η Εκτέλεση Ενός Ερωτήματος (1)

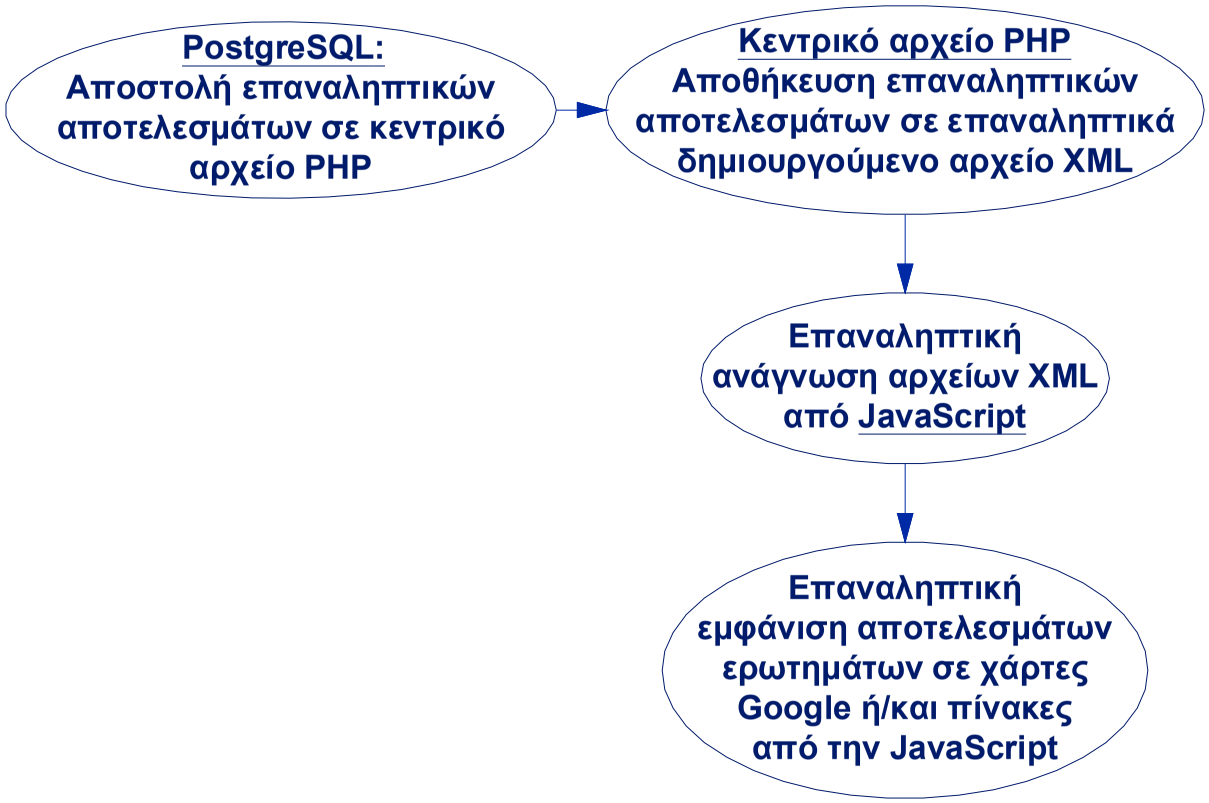
Διάγραμμα 4

Διαγραμματική Παρουσίαση Διαδικασιών Όταν Ζητείται η Εκτέλεση Ενός Ερωτήματος (2)



Διάγραμμα 5

Διαγραμματική Παρουσίαση Διαδικασιών Όταν Ζητείται η Εκτέλεση Ενός Ερωτήματος (3)



Διάγραμμα 6

5.6 Αναλυτική περιγραφή λειτουργίας αρχείου execQuery_getResults.php

Όπως έχει προαναφερθεί, το αρχείο execQuery_getResults.php είναι αυτό που περιέχει τον κώδικα για την εκτέλεση όλων των ερωτημάτων και την αποθήκευση των αποτελεσμάτων σε επαναληπτικά αρχεία XML. Παρακάτω θα αναφερθούμε εκτενώς σε όλες τις διαδικασίες που εκτελεί το συγκεκριμένο αρχείο.

Η εκτέλεση του execQuery_getResults.php ξεκινά όταν ο χρήστης της εφαρμογής πατήσει οποιοδήποτε κουμπί εκτέλεσης ερωτήματος στην κεντρική ιστοσελίδα της εφαρμογής. Σημαντικό ρόλο παίζουν οι παράμετροι που δίδονται από τον κώδικα HTML. Πιο συγκεκριμένα, όταν ο χρήστης πατήσει το κουμπί "Εκτέλεση Βασικού Ερωτήματος" οι παράμετροι τις οποίες δίνει η HTML φόρμα στην εκτέλεση του execQuery_getResults.php είναι refreshT=5 και query=0. Η παράμετρος refreshT δηλώνει τη συχνότητα επανάληψης της λήψης αποτελεσμάτων, ενώ η παράμετρος query δηλώνει το ερώτημα που πρόκειται να εκτελεστεί. Η τιμή 0 στην παράμετρο query δηλώνει το Βασικό Ερώτημα, η τιμή 6 το Ερώτημα Περιοχής, η τιμή 1 το Προσαρμοσμένο Ερώτημα 1, η τιμή 2 το Πρ. Ερώτημα 2 και η τιμή 3 το Πρ. Ερώτημα 3.

Οι ενέργειες που θα εκτελέσει το execQuery_getResults.php εξαρτώνται από τις παραμέτρους αλλά και από εξωτερικές συνθήκες όπως η ύπαρξη ή όχι διαφόρων βοηθητικών αρχείων που δημιουργούνται είτε από το ίδιο το execQuery_getResults.php σε προηγούμενες εκτελέσεις του είτε από την εκτέλεση διαφόρων άλλων αρχείων-προγραμμάτων.

Το αρχείο execQuery_getResults.php όπως και κάθε αρχείο PHP μπορεί να εκτελείται πολλές, ανεξάρτητες μεταξύ τους, φορές ταυτόχρονα. Έτσι, τα πέντε ερωτήματα της εφαρμογής μπορούν να εκτελούνται ταυτόχρονα.

Ξεκινώντας την εκάστοτε εκτέλεσή του το execQuery_getResults.php αποθηκεύει σε δικές του μεταβλητές τις τιμές των παραμέτρων refreshT και query. Πιο συγκεκριμένα η μεταβλητή της PHP $refreshTime λαμβάνει την τιμή της από την παράμετρο refreshT ενώ η μεταβλητή της PHP $qID λαμβάνει την τιμή της από την παράμετρο query.

Στη συνέχεια η PHP ελέγχει εάν υπάρχει το αρχείο "zstopQuery#.xml" το οποίο χρησιμεύει για την διακοπή του ερωτήματος. Καθώς το ερώτημα δεν έχει ξεκινήσει ακόμη, θεωρείται ότι εκ παραδρομής υπάρχει αυτό το αρχείο και η PHP στο σημείο αυτό το διαγράφει και συνεχίζει την εκτέλεση του script. Εάν δεν διαγραφεί σε αυτό το σημείο το αρχείο "zstopQuery#.xml", μόλις η εκτέλεση του "execQuery_getResults.php" φτάσει στον πρώτο βρόχο λήψης επαναληπτικών αποτελεσμάτων, αυτό θα σταματήσει διότι θα θεωρηθεί ότι ο χρήστης έχει πατήσει το κουμπί διακοπής του συγκεκριμένου ερωτήματος. Για να αποφευχθεί λοιπόν ένα τέτοιο ανεπιθύμητο ενδεχόμενο, εάν εντοπιστεί αυτό το αρχείο σε αυτό το σημείο εκτέλεσης της PHP αυτό θα διαγραφεί.

Ακολούθως η PHP ελέγχει εάν το συγκεκριμένο ερώτημα ήδη εκτελείται από κάποια άλλη εκτέλεση του "execQuery_getResults.php" με τις ίδιες παραμέτρους. Ο έλεγχος γίνεται με την ύπαρξη ή όχι συγκεκριμένου βοηθητικού αρχείου ("zQuery#IsRunning.xml" με #=0, 1, 2, 3 ή 6 ο αριθμός του ερωτήματος) που δημιουργεί το ίδιο το execQuery_getResults.php σε επόμενο στάδιο στο οποίο θα αναφερθούμε παρακάτω. Εάν λοιπόν υπάρχει το βοηθητικό αρχείο "zQuery#IsRunning.xml" η εκτέλεση του "execQuery_getResults.php" σταματά. Εάν δεν υπάρχει η εκτέλεση προχωρά στον επόμενο έλεγχο.

Στο σημείο αυτό πρέπει να αναφερθεί ότι η εφαρμογή έχει ρυθμιστεί έτσι ώστε να απαιτείται να είναι ενεργά τα απαραίτητα ρεύματα δεδομένων πριν την εκτέλεση των ερωτημάτων. Αυτό συμβαίνει για την αποφυγή άσκοπης εκτέλεσης ερωτημάτων από το χρήστη και διακοπής τους από την εφαρμογή καθώς δε θα υπάρχουν αποτελέσματα. Έτσι λοιπόν, ακολουθεί έλεγχος εάν το πρώτο ρεύμα δεδομένων, προς το network.vehicles,

είναι ήδη ενεργό όταν ο χρήστης εκτελέσει κάποιο ερώτημα (όταν δηλαδή η εκτέλεση του "execQuery_getResults.php" περάσει από τον προηγούμενο έλεγχο που αναφέρθηκε παραπάνω). Ο έλεγχος γίνεται με την ύπαρξη ή όχι συγκεκριμένου βοηθητικού αρχείου ("zstreamRunning.xml") το οποίο δημιουργείται όταν ξεκινά να εκτελείται το πρώτο ρεύμα δεδομένων και διαγράφεται όταν αυτό σταματά. Έτσι λοιπόν, εάν εκτελείται το πρώτο ρεύμα δεδομένων, η εκτέλεση του "execQuery_getResults.php" συνεχίζεται, αλλιώς σταματά.

Εφόσον συνεχιστεί η εκτέλεση του βασικού αρχείου PHP, ακολουθεί ο ορισμός του ερωτήματος προς εκτέλεση και ο ορισμός της μέγιστης διάρκειας εκτέλεσης αυτού με βάση την μεταβλητή $qID που έχει αποθηκεύσει την τιμή της παραμέτρου "query" από την HTML. Εάν λοιπόν είναι $qID = 0 τότε πρόκειται για το Βασικό Ερώτημα το οποίο επιστρέφει όλα τα κινούμενα αντικείμενα του ρεύματος και το οποίο ορίζεται και αποθηκεύεται στη σχετική μεταβλητή που ονομάζεται $qOrig από τα αρχικά query Original (αρχικό ερώτημα). Η τιμή της μέγιστης διάρκειας εκτέλεσης λαμβάνεται από την μεταβλητή της HTML (setlimit0) και αποθηκεύεται στην μεταβλητή $met από τα αρχικά των λέξεων maximum execution time.

Εάν είναι $qID = 1 ή 2 ή 3 τότε πρόκειται να εκτελεστεί το Προσαρμοσμένο Ερώτημα 1 ή 2 ή 3 αντίστοιχα. Το ερώτημα προς εκτέλεση για τις περιπτώσεις αυτές λαμβάνεται από τις μεταβλητές της HTML query1 ή query2 ή query3 αντίστοιχα οι οποίες διατηρούν το κάθε ερώτημα προς εκτέλεση και αποθηκεύεται στην παραπάνω αναφερθείσα μεταβλητή $qOrig. Αντίστοιχα, η μέγιστη διάρκεια εκτέλεσης λαμβάνεται από τις μεταβλητές της HTML setlimit1 ή setlimit2 ή setlimit3 και αποθηκεύεται στην μεταβλητή $met.

Εάν είναι $qID = 6 τότε πρόκειται να εκτελεστεί το Ερώτημα Περιοχής. Το ερώτημα που αποθηκεύεται στην μεταβλητή $qOrig λαμβάνεται εν μέρει από το ίδιο το execQuery_getResults.php και εν μέρει από την μεταβλητή της HTML qcoords που διατηρεί τις συντεταγμένες στο σχετικό πεδίο της ιστοσελίδας. Η μεταβλητή $met παίρνει την τιμή της από την μεταβλητή της HTML setlimit6.

Αφού λοιπόν έχουν οριστεί και έχουν λάβει τιμές οι μεταβλητές που ορίζουν τον αριθμό του ερωτήματος ($qID), τον χρόνο επανάληψης του βρόχου ($refreshTime), τη μέγιστη διάρκεια εκτέλεσης του ερωτήματος ($met) και το ερώτημα προς εκτέλεση, ακολουθεί ένας έλεγχος του ερωτήματος που αφορά στην ύπαρξη ή όχι του κειμένου "network.vehicles2" μέσα στο κείμενο του ερωτήματος. Γίνεται λοιπόν ο εξής έλεγχος: εάν υπάρχει το κείμενο "network.vehicles2" μέσα στο κείμενο του ερωτήματος και δεν υπάρχει το αρχείο "zstreamRunning2.xml", τότε σταματά η εκτέλεση του "execQuery_getResults.php". Όπως η ύπαρξη του αρχείου "zstreamRunning.xml" σημαίνει ότι το πρώτο ρεύμα εκτελείται, έτσι και η ύπαρξη του αρχείου "zstreamRunning2.xml" σημαίνει ότι το δεύτερο ρεύμα εκτελείται. Συνεπώς, εάν το ερώτημα που καταχωρήθηκε προς εκτέλεση πρόκειται να χρησιμοποιήσει πληροφορίες από το δεύτερο ρεύμα δεδομένων και αυτό δεν εκτελείται, τότε δε θα επιστρέφεται καμία απάντηση στο ερώτημα αυτό και συνεπώς δεν εκτελείται καν – τερματίζει την εκτέλεσή του το "execQuery_getResults.php". Στις υπόλοιπες περιπτώσεις, που απαιτείται το δεύτερο ρεύμα και είναι ενεργό, δεν απαιτείται το δεύτερο ρεύμα και είναι ενεργό και δεν απαιτείται το δεύτερο ρεύμα και δεν είναι ενεργό συνεχίζεται η εκτέλεση του "execQuery_getResults.php".

Μετά από τον παραπάνω έλεγχο, ακολουθεί ο ορισμός της μεταβλητής $bgtime που παίρνει την τιμή της συγκεκριμένης χρονικής στιγμής ($bgtime = time();). Παρομοίως ορίζεται η μεταβλητή $inittime (= time();) ως η συγκεκριμένη χρονική στιγμή.

Ακολουθεί η αντικατάσταση της κατακόρυφης γραμμής με τα απλά εισαγωγικά και στη συνέχεια οι έλεγχοι επί του ερωτήματος πριν την εκτέλεσή του.

Ο πρώτος έλεγχος είναι εάν υπάρχει κείμενο πληκτρολογημένο στο πεδίο του ερωτήματος προς εκτέλεση. Εάν δεν υπάρχει, τότε δεν υπάρχει και λόγος πλέον να

επιχειρηθεί να εκτελεστεί το ερώτημα καθώς θα επιστραφεί μήνυμα λάθους από την PostgreSQL.

Ο δεύτερος έλεγχος είναι εάν υπάρχουν οι λέξεις κλειδιά: ABORT, ALTER, ANALYZE, BEGIN, CHECKPOINT, CLOSE, CLUSTER, COMMENT, COMMIT, COPY, CREATE, DEALLOCATE, DECLARE, DELETE, DROP, END, EXECUTE, EXPLAIN, FETCH, GRANT, INSERT, LISTEN, LOAD, LOCK, MOVE, NOTIFY, PREPARE, REINDEX, RESET, REVOKE, ROLLBACK, SET, SHOW, START TRANSACTION, TRUNCATE, UNLISTEN, UPDATE και VACUUM στο υπό εκτέλεση ερώτημα. Με κατάλληλες εντολές λοιπόν, γίνεται αναζήτηση μέσα στο κείμενο του υποβληθέντος ερωτήματος για την ύπαρξη ή όχι των παραπάνω λέξεων κλειδιών. Εάν υπάρχει έστω και μία από τις παραπάνω, το ερώτημα δεν θα εκτελεστεί. Αυτό γίνεται για τη διασφάλιση της βάσης δεδομένων και των ρευμάτων από πιθανή κακόβουλη χρήση των δυνατοτήτων της εφαρμογής. Εάν δεν υπήρχε αυτός ο περιορισμός, θα μπορούσε κανείς να δημιουργήσει νέους δικούς του πίνακες, να εισάγει δικά του δεδομένα στους πίνακες και στα ρεύματα, να τροποποιήσει τα υφιστάμενα ρεύματα ακόμη και να διαγράψει τα υφιστάμενα ρεύματα “network.vehicles” και “network.vehicles2” κάτι καταστροφικό βέβαια για την εφαρμογή. Με τους ελέγχους αυτούς λοιπόν διασφαλίζεται η βάση δεδομένων και το ρεύμα δεδομένων.

Ακολουθεί ο έλεγχος για τη λέξη κλειδί “SELECT”. Αυτή θα πρέπει να υπάρχει και μάλιστα να ευρίσκεται στην αρχή του υποβληθέντος ερωτήματος. Το TelegraphCQ δεν υποστηρίζει φωλιασμένα (nested) ερωτήματα και συνεπώς η λέξη κλειδί “SELECT” δε θα μπορούσε να ευρίσκεται και σε άλλη θέση εκτός από την αρχή του ερωτήματος. Εάν λοιπόν δεν τηρείται αυτή η προϋπόθεση, δε θα εκτελεστεί το ερώτημα.

Ο τέταρτος έλεγχος αφορά στην ύπαρξη ή όχι της λέξης κλειδί “FROM” η οποία θα πρέπει να υπάρχει και μάλιστα να βρίσκεται μετά τη λέξη κλειδί “SELECT”. Εάν δεν τηρείται αυτός ο περιορισμός τότε δε θα εκτελεστεί το ερώτημα καθώς θα επιστρέψει μήνυμα λάθους.

Ο πέμπτος έλεγχος αφορά στην ύπαρξη ή όχι του κειμένου “network.vehicles” ή του κειμένου “network.vehicles2”. Στο υπό εκτέλεση ερώτημα θα πρέπει να υπάρχει είτε η αναφορά στο πρώτο ρεύμα δεδομένων “network.vehicles” είτε η αναφορά στο δεύτερο ρεύμα δεδομένων είτε η αναφορά και στα δύο. Μάλιστα οι συμβολοσειρές αυτές θα πρέπει να ευρίσκονται μετά τη λέξη κλειδί “FROM”. Δίδεται λοιπόν η δυνατότητα στο χρήστη να εκτελέσει ερώτημα με αναφορά είτε μόνο στο πρώτο ρ.δ. είτε μόνο στο δεύτερο ρ.δ. είτε και στα δύο. Εάν δεν υπάρχει αναφορά σε κανένα από τα δύο δε θα εκτελεστεί το ερώτημα. Φυσικά εάν υπάρχει αναφορά στο δεύτερο ρεύμα, θα πρέπει αυτό να είναι ενεργό διότι αλλιώς θα σταματήσει η εκτέλεση του “execQuery_getResults.php” σε προηγούμενο στάδιο ελέγχου εκτέλεσης των ρευμάτων δεδομένων.

Ο έκτος έλεγχος αφορά στην τιμή που έχει λάβει η μεταβλητή $met. Εάν δεν έχει οριστεί τιμή ή έχει οριστεί το κενό, τότε ο έλεγχος αυτός θα σταματήσει την εκτέλεση του ερωτήματος για την αποφυγή σφαλμάτων σε επόμενο στάδιο εκτέλεσης του “execQuery_getResults.php”.

Ο τελευταίος έλεγχος αφορά πάλι στην τιμή που έχει λάβει η μεταβλητή $met. Πιο συγκεκριμένα η μέγιστη διάρκεια εκτέλεσης πρέπει να είναι αριθμός με ή χωρίς δεκαδικά ψηφία (αριθμός κινητής υποδιαστολής). Δεν μπορεί δηλαδή το συγκεκριμένο πεδίο της ιστοσελίδας και κατ’ επέκταση η συγκεκριμένη μεταβλητή της PHP να περιέχει άλλα σύμβολα εκτός από αριθμούς (και το σύμβολο της υποδιαστολής φυσικά που θα πρέπει να είναι η τελεία (.) και όχι το κόμμα (,)). Εάν λοιπόν η μεταβλητή $met δεν είναι αριθμός κινητής υποδιαστολής, ο συγκεκριμένος έλεγχος θα αποτρέψει το execQuery_getResults.php από το να εκτελέσει τις υπόλοιπες λειτουργίες του και κατ’ επέκταση τη σύνδεση με την βάση δεδομένων και την υποβολή του ερωτήματος.

Ακολουθεί η επεξεργασία του υποβληθέντος ερωτήματος με σκοπό τον διαχωρισμό των χαρακτηριστικών που ζητούνται να επιστραφούν ως απαντήσεις στο

σχετικό ερώτημα. Κάθε συνεχές ερώτημα ξεκινά με τη λέξη κλειδί "SELECT", ακολουθεί ο ορισμός των χαρακτηριστικών που ζητούνται να επιστραφούν ως απαντήσεις στο ερώτημα, ακολουθεί η λέξη κλειδί "FROM" και ο ορισμός των ρευμάτων όπου θα αναζητήσει τις πληροφορίες που ζητούνται το ερώτημα και κατ' επέκταση το σύστημα διαχείρισης ρευμάτων δεδομένων, με ορισμό του παραθύρου (=υποσυνόλου) χρονικού ή όχι των στοιχείων του ρεύματος. Προαιρετικά ακολουθεί η λέξη κλειδί "WHERE" και οι περιορισμοί που επιθυμεί ο χρήστης για το συγκεκριμένο ερώτημα.

Προκειμένου λοιπόν η PHP να ζητήσει τα αποτελέσματα των πεδίων-χαρακτηριστικών που τίθενται με το συγκεκριμένο κάθε φορά ερώτημα, πρέπει να τα διαχωρίσει και να τα απομονώσει σε συγκεκριμένες μεταβλητές. Έτσι λοιπόν προκειμένου να έχει τη δυνατότητα ο χρήστης να ζητήσει ότι χαρακτηριστικά επιθυμεί, με όποια σειρά επιθυμεί και να τα ονομάσει όπως επιθυμεί, πρέπει αυτά να εντοπίζονται από την PHP και να αποθηκεύονται σε συγκεκριμένες μεταβλητές. Επίσης, καθώς η PostgreSQL και το TelegraphCQ μετατρέπουν σε μικρά τα ονόματα όλων των χαρακτηριστικών, πρέπει να πεδία-χαρακτηριστικά να μετατραπούν και αυτά σε μικρά.

Με συγκεκριμένες εντολές λοιπόν εύρεσης και αντικατάστασης κειμένου μέσα στο κείμενο του ερωτήματος, απομονώνονται σε συγκεκριμένη μεταβλητή όλα τα χαρακτηριστικά που ζητούνται ως απαντήσεις στο υποβληθέν ερώτημα.

Στη συνέχεια, με συγκεκριμένες πάλι εντολές εύρεσης και αντικατάστασης κειμένου μέσα στο κείμενο της μεταβλητής αυτής, απομονώνονται σε συγκεκριμένες μεταβλητές τα χαρακτηριστικά που ζητούνται να επιστραφούν ως απαντήσεις στο συγκεκριμένο κάθε φορά ερώτημα.

Έχει προβλεφθεί ο εντοπισμός, η απομόνωση και διαχείριση μέχρι 6 διαφορετικών χαρακτηριστικών για κάθε ερώτημα. Οι μεταβλητές της PHP $a1, $a2, $a3, $a4, $a5 και $a6 αποθηκεύουν τα ονόματα των χαρακτηριστικών με μικρά γράμματα ώστε στη συνέχεια να αναζητηθούν αυτά στις απαντήσεις που θα επιστρέφει η PostgreSQL και το TelegraphCQ.

Μετά από όλες τις παραπάνω διαδικασίες και εφ' όσον δεν έχει διακοπεί η εκτέλεση του execQuery_getResults.php από κάποιον περιορισμό, έρχεται η στιγμή που η PHP συνδέεται με την PostgreSQL και το TelegraphCQ.

Για την εκτέλεση συνεχών ερωτημάτων, πρέπει να εισάγουμε συγκεκριμένες εντολές στο σύστημα διαχείρισης δ.δ. και ρ.δ. ως εξής:

BEGIN WORK;

για την έναρξη συγκεκριμένης εργασίας όπως η εκτέλεση συγκεκριμένου συνεχούς ερωτήματος.

Το συνεχές ερώτημα πρέπει να καταχωρηθεί με κάποιον κέρσορα ως εξής:

DECLARE [όνομα κέρσορα] CURSOR FOR [ερώτημα];

Έτσι, η συμβολοσειρά που αποτελεί το ερώτημα που επιθυμούμε να εκτελέσουμε, ενώνεται με την παραπάνω συμβολοσειρά με συγκεκριμένο όνομα κέρσορα (το οποίο στην περίπτωση αυτής της εφαρμογής έχουμε ονομάσει Q1_portion) και η συγκεκριμένη εντολή DECLARE υποβάλλεται στην PostgreSQL αφού έχει προηγηθεί η εντολή BEGIN WORK;.

Από τη στιγμή που έχει υποβληθεί η εντολή DECLARE με το ερώτημα, αυτό εκτελείται συνεχώς χωρίς να επιστρέφει αποτελέσματα. Τα αποτελέσματα λαμβάνουμε με ξεχωριστή εντολή, ώστε να τα αποθηκεύουμε σε συγκεκριμένες μεταβλητές και να μην έρχονται ανεξέλεγκτα όποτε καταφθάνουν νέα δεδομένα στο σύστημα και προκύπτουν αποτελέσματα από την επεξεργασία.

Προτού προβούμε στην αναζήτηση αποτελεσμάτων στο ερώτημα που έχουμε υποβάλει, ζητούμε από την PHP να δημιουργήσει ένα βοηθητικό αρχείο xml το οποίο θα ανιχνεύει η JavaScript ώστε να δείχνει στην ιστοσελίδα ότι το συγκεκριμένο ερώτημα είναι ενεργό. Η στιγμή που έχει πλέον υποβληθεί το ερώτημα στη βάση δεδομένων είναι η κατάλληλη ώστε να δημιουργηθεί ένα τέτοιο βοηθητικό αρχείο και κατ' επέκταση να αλλάξει η κατάσταση του ερωτήματος στην ιστοσελίδα από σταματημένο σε ενεργό. Το

βοηθητικό αρχείο ονομάζεται "zQuery#IsRunning.xml", όπου # είναι ο αριθμός του ερωτήματος. Στον κώδικα JavaScript που θα περιγράψουμε αναλυτικά παρακάτω, θα αναφερθούμε σε αυτό το αρχείο και στον τρόπο με τον οποίο η ύπαρξή του μεταφράζεται σε αλλαγή της κατάστασης του κάθε ερωτήματος.

Για την επαναληπτική λήψη των αποτελεσμάτων και την αποθήκευση αυτών σε συγκεκριμένα αρχεία XML, χρησιμοποιείται βρόχος ο οποίος εκτελείται συνεχώς και ο οποίος υπό συγκεκριμένες μόνο συνθήκες σταματά. Η πρώτη περίπτωση διακοπής είναι όταν η παρούσα χρονική στιγμή παρέλθει της χρονικής στιγμής εκκίνησης εκτέλεσης του ερωτήματος προσαυξημένης κατά τη μέγιστη διάρκεια εκτέλεσης αυτού. Η δεύτερη περίπτωση διακοπής είναι όταν σε επανάληψη του βρόχου ανιχνευθεί η ύπαρξη του αρχείου "zstopQuery#.xml" το οποίο δημιουργείται από άλλα αρχεία PHP τα οποία εκτελούνται όταν ο χρήστης πατήσει το κουμπί διακοπής του συγκεκριμένου ερωτήματος ή το κουμπί διακοπής όλων των ερωτημάτων. Ο κεντρικός αυτός βρόχος, όπως θα αναλυθεί παρακάτω εκτελείται μία φορά κάθε $refreshTime δευτερόλεπτα (δηλαδή κάθε 5 δευτερόλεπτα).

Μέσα σε αυτόν τον κύριο βρόχο που θα εκτελείται για όσο χρονικό διάστημα δεν έχει παρέλθει η μέγιστη διάρκεια εκτέλεσης και δεν υπάρχει το βοηθητικό αρχείο διακοπής του, υπάρχει δεύτερος βρόχος ο οποίος φροντίζει για την επαναληπτική αποθήκευση των αποτελεσμάτων κάθε 5 δευτερόλεπτα. Καθώς ο χρήστης δύναται να καταχωρήσει δικά του ερωτήματα, το πλήθος των αποτελεσμάτων κάθε 5 δευτερόλεπτα θα διαφέρει ανάλογα με το ερώτημα αλλά και με τους περιορισμούς αυτού σε συνδυασμό με τη χρονική στιγμή. Έτσι, ένα ερώτημα μπορεί να επιστρέφει κάθε φορά 1 ή περισσότερα αντικείμενα. Ο μοναδικός τρόπος να αποθηκεύουμε τα ακριβή αποτελέσματα κάθε φορά στο σχετικό αρχείο XML είναι να ζητούμε από την PostgreSQL και το TelegraphCQ να τα επιστρέφει ένα – ένα και κάθε φορά να γίνεται έλεγχος εάν έχει παρέλθει συγκεκριμένο χρονικό διάστημα.

Τα αποτελέσματα του υποβληθέντος ερωτήματος αποθηκεύονται σε μεταβλητή που ονομάζεται $resFetch από τις λέξεις results Fetch. Η μεταβλητή αυτή αποθηκεύει το αποτέλεσμα της εκτέλεσης της εντολής FETCH FORWARD 1 IN Q1_portion; προς την PostgreSQL.

Η τελευταία αυτή εντολή ζητά από την PostgreSQL και κατ' επέκταση και το TelegraphCQ να επιστρέψει το επόμενο κάθε φορά αποτέλεσμα στο υποβληθέν ερώτημα.

Η κάθε σειρά αποτελεσμάτων (tuple) πρέπει να χωριστεί στα χαρακτηριστικά που την αποτελούν και να αποθηκευτούν αυτά σε σχετικό αρχείο XML πριν την επανάληψη της διαδικασίας. Έτσι, προκειμένου να αποφευχθούν λάθη στην εκτέλεση του κώδικα, διακρίνονται οι ακόλουθες έξι περιπτώσεις. Εάν το πλήθος των χαρακτηριστικών που ζητούνται στις απαντήσεις είναι 1 τότε οι μεταβλητές $a2, $a3, $a4, $a5 και $a6 δεν θα έχουν καθορισμένη τιμή και δεν πρέπει να αναζητηθούν στις απαντήσεις που λαμβάνει η PHP από την PostgreSQL σε άλλα χαρακτηριστικά εκτός από το πρώτο. Παρομοίως εάν το πλήθος των χαρακτηριστικών που ζητούνται στις απαντήσεις είναι 2 τότε οι μεταβλητές $a3, $a4, $a5 και $a6 δεν θα έχουν καθορισμένη τιμή και δεν πρέπει να αναζητηθούν στις απαντήσεις που λαμβάνει η PHP από την PostgreSQL. Αντίστοιχη διαδικασία πρέπει να ακολουθηθεί για τις περιπτώσεις που έχουμε ζητήσει 3, 4 ή 5 χαρακτηριστικά στις απαντήσεις των ερωτημάτων. Στην περίπτωση τέλος που έχουμε ζητήσει 6 χαρακτηριστικά το οποίο είναι και το μέγιστο, θα πρέπει η PHP να αναζητήσει και τα 6 στα αποτελέσματα που λαμβάνει από την PostgreSQL.

Έτσι, στην ανάγνωση των αποτελεσμάτων διακρίνονται οι 6 αυτές περιπτώσεις και η PHP εκτελεί τις ανάλογες εντολές. Η σχετική εντολή λήψης των αποτελεσμάτων θα εκτελεστεί όσες φορές χρειάζεται για την μεταφορά αυτών στην PHP (μέσα στο χρόνο επανάληψης του δεύτερου βρόχου) και θα εκτελεστεί μια ακόμη φορά καθώς το κριτήριο διακοπής εκτέλεσης του δεύτερου βρόχου είναι η πάροδος του μισού χρόνου ανανέωσης των αποτελεσμάτων και ελέγχεται μετά την λήψη κάθε αποτελέσματος. Στην περίπτωσή μας που ο χρόνος ανανέωσης των αποτελεσμάτων είναι τα 5 δευτερόλεπτα, ο

δεύτερος βρόχος, θα εκτελείται συνεχώς κάθε φορά για χρονικό διάστημα 2,5 δευτερολέπτων. Μέσα σε αυτά τα 2,5 δευτερόλεπτα θα επιστραφούν όλες οι απαντήσεις του ερωτήματος για την τρέχουσα χρονική στιγμή και η εντολή "FETCH FORWARD 1 IN Q1_portion;" μέσα από την εντολή pg_query της PHP θα εκτελεστεί μία ακόμη φορά καθώς δεν θα έχουν παρέλθει αυτά τα 2,5 sec (στην πραγματικότητα καθώς τα αποτελέσματα έρχονται σχεδόν ταυτόχρονα από την PHP, δεν θα έχει παρέλθει ούτε μισό δευτερόλεπτο). Εάν η καταγραφή των αποτελεσμάτων στο αρχείο XML γίνεται μέσα στον βρόχο μετά την εκτέλεση της εντολής λήψης του κάθε αποτελέσματος, το τελευταίο αυτό αποτέλεσμα θα καταχωρηθεί στο αρχείο XML με τα ήδη υπάρχοντα αποτελέσματα τα οποία θα αναφέρονται στην προηγούμενη χρονική στιγμή. Έτσι στην περίπτωση που η αποθήκευση των αποτελεσμάτων στο XML γίνεται μετά τη λήψη τους στον ίδιο βρόχο, παρατηρείται το φαινόμενο στα αποτελέσματα να υπάρχει ομάδα αντικειμένων με συγκεκριμένη χρονική στιγμή και ένα τελευταίο αντικείμενο με μεταγενέστερη χρονική στιγμή.

Για να αποφευχθεί το πρόβλημα αυτό η καταγραφή των αποτελεσμάτων στο αρχείο XML γίνεται μέσα στο βρόχο πριν τη λήψη τους από την PostgreSQL και το TelegraphCQ. Τα αποτελέσματα αποθηκεύονται στη μεταβλητή $resFetch όπως προαναφέρθηκε και στην επόμενη εκτέλεση του βρόχου καταχωρούνται στο αρχείο XML.

Την πρώτη φορά εκτέλεσης του συγκεκριμένου βρόχου που θα επιχειρήσει η PHP να αποθηκεύσει τις πληροφορίες που διαθέτει η μεταβλητή $resFetch, δεν θα υπάρχουν πληροφορίες προς αποθήκευση στο αρχείο XML. Για να αποφευχθεί αυτό το λάθος στον κώδικα, δίνεται μια αρχική τιμή στην μεταβλητή $resFetch πριν γίνει η σύνδεση με την βάση δεδομένων. Πιο συγκεκριμένα στην αρχή του κώδικα, αμέσως μετά τον καθορισμό των μεταβλητών $refreshTime και $qID στις οποίες έχουμε αναφερθεί παραπάνω, ορίζουμε την μεταβλητή $resFetch και της αποδίδουμε μια αρχική τιμή ($resFetch = "begin";). Έτσι, μέσα στον δεύτερο βρόχο γίνεται κάθε φορά έλεγχος εάν η μεταβλητή $resFetch έχει την τιμή "begin" και εάν συμβαίνει κάτι τέτοιο (δηλαδή μόνο την πρώτη φορά) δεν εκτελεί καθόλου τις εντολές καταχώρησης στο αρχείο XML των πληροφοριών που διαθέτει η μεταβλητή αυτή. Σε όλες τις άλλες περιπτώσεις (δηλαδή όλες εκτός από την πρώτη φορά) η PHP εκτελεί τις εντολές καταχώρησης στο αρχείο XML των πληροφοριών που διαθέτει η μεταβλητή $resFetch.

Η εντολή pg_query κάθε φορά περιμένει να λάβει κάποιο αποτέλεσμα και αφού το λάβει συνεχίζεται η εκτέλεση του κώδικα PHP. Έτσι, το πρώτο αποτέλεσμα από την επόμενη ομάδα δεδομένων θα είναι αυτό που μετά την λήψη του από το pg_query θα έχει περάσει ο χρόνος των 2,5 sec και δε θα επαναληφθεί ο δεύτερος βρόχος. Αφού λοιπόν συνεχίσει η εκτέλεση μετά τον δεύτερο βρόχο, αποθηκεύεται το αρχείο XML με όνομα "results#.xml" όπου # ο αριθμός του ερωτήματος.

Πριν επαναληφθεί ο πρώτος βρόχος, το σύστημα παραμένει σε κατάσταση αναμονής για τα υπόλοιπα 2,5 sec ώστε να συμπληρωθεί ο χρόνος ανανέωσης που είναι τα 5 δευτερόλεπτα. Επίσης, η μεταβλητή $bgtime που είχε πάρει την τιμή της νωρίτερα ως η τότε τρέχουσα χρονική στιγμή, λαμβάνει καινούργια τιμή την νέα τρέχουσα χρονική στιγμή. Ακριβώς μετά, ο πρώτος βρόχος επαναλαμβάνεται. Γίνεται έλεγχος εάν έχει παρέλθει ο μέγιστος χρόνος εκτέλεσης ή εάν υπάρχει το αρχείο διακοπής "zstopQuery#.xml" και εάν δεν συντρέχουν λόγοι διακοπής εκτέλεσης του πρώτου βρόχου, αυτός επαναλαμβάνεται.

Όταν σταματήσουν οι εκτελέσεις του πρώτου βρόχου σημαίνει ότι σταματά η λήψη αποτελεσμάτων για το συγκεκριμένο ερώτημα και η εκτέλεση του execQuery_getResults.php οδεύει προς το τέλος της. Πριν τον τερματισμό εκτέλεσης του αρχείου αυτού, θα πρέπει να αδειάσουν τα αποτελέσματα από την μεταβλητή $resFetch, να κλείσει ο κέρσορας ("CLOSE Q1_portion;"), να σταματήσει η εργασία που έχουμε ζητήσει από την PostgreSQL ("COMMIT WORK;") και να κλείσει η σύνδεση με την βάση δεδομένων (close db connection).

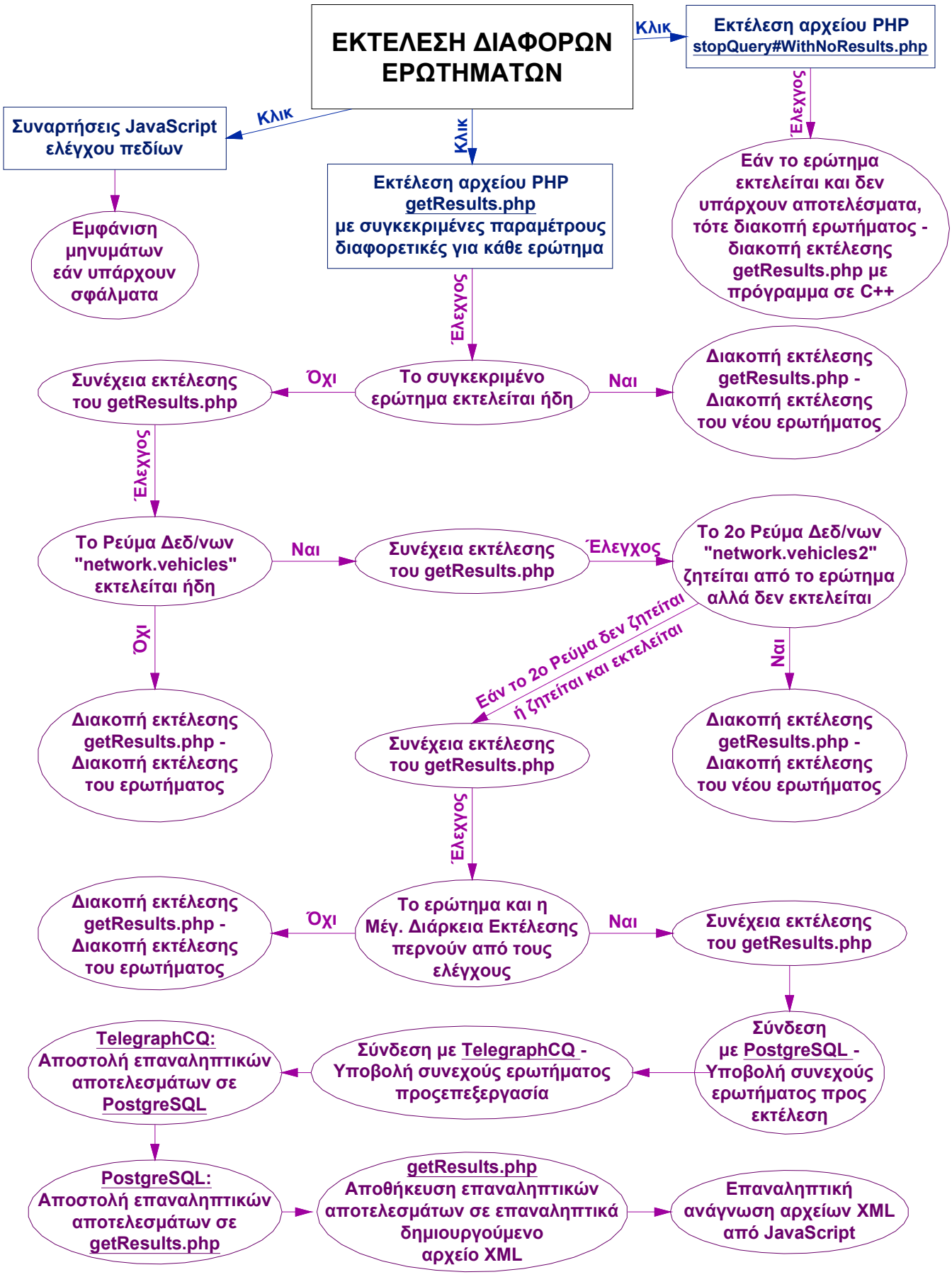
Προτού σταματήσει η εκτέλεση του “execQuery_getResults.php” διαγράφεται το αρχείο XML “results#.xml” με τα αποτελέσματα των ερωτημάτων ώστε αυτά να μην εμφανίζονται ξανά και ξανά στο χάρτη καθώς η JavaScript θα βρίσκει το αρχείο και θα το διαβάζει κάθε 5 δευτερόλεπτα. Επίσης, διαγράφεται το αρχείο “zQuery#IsRunning.xml” ώστε η JavaScript να αλλάξει την τιμή της κατάστασης του συγκεκριμένου ερωτήματος από ενεργό σε σταματημένο.

Στο σημείο αυτό τελειώνει η εκτέλεση του βασικότερου αρχείου PHP που χρησιμοποιεί η παρούσα εφαρμογή.

Ακολουθεί διάγραμμα το οποίο απεικονίζει τις κυριότερες λειτουργίες που εκτελούνται όταν ο χρήστης πατήσει κάποιο κουμπί εκτέλεσης ερωτήματος.

Διπλωματική Εργασία: Διαχείριση ρευμάτων κίνησης αντικειμένων με απεικόνιση σε GoogleMaps

Διαγραμματική Παρουσίαση Διαδικασιών Όταν Ζητείται η Εκτέλεση Ενός Ερωτήματος



* # = ο αριθμός του ερωτήματος

Διάγραμμα 7

Κωνσταντίνος Δ. Θεοφιλογιαννάκος

Στο Παράρτημα παρατίθεται αναλυτικός πίνακας με όλες τις μεταβλητές που χρησιμοποιεί το αρχείο execQuery_getResults.php και δεύτερος πίνακας με όλες τις συναρτήσεις αυτού.

Παρατίθεται επίσης ο συνολικός κώδικας με αρκετά σχόλια στην αγγλική γλώσσα.

Απαραίτητος βέβαια για την λειτουργία της παρούσας εφαρμογής είναι και ο κώδικας JavaScript καθώς αυτός καλείται να εμφανίσει το χάρτη στην ιστοσελίδα και φυσικά τα αποτελέσματα των διαφόρων ερωτημάτων.

Ακολουθεί αναλυτική περιγραφή του.

5.7 Αναλυτική περιγραφή λειτουργίας κώδικα JavaScript (αρχείο StrMap_JavaScript_code_gr.js)

5.7.1 Γενικές πληροφορίες

Ο κώδικας JavaScript εκτελεί σειρά λειτουργιών με τη βοήθεια συνολικά 34 προσαρμοσμένων συναρτήσεων ορισμένες από τις οποίες έχουν ελάχιστες εντολές ενώ άλλες θα λέγαμε ότι περιέχουν πραγματικά πολλές εντολές. Από αυτές τις συναρτήσεις, αρκετές ορίζονται εντός άλλων συναρτήσεων, ενώ οι περισσότερες ορίζονται απ' ευθείας στο σχετικό script. Εκτός από αυτές τις συναρτήσεις, ο κώδικας περιέχει συνολικά περίπου 95 μεταβλητές, 138 προτάσεις υπόθεσης if, else if και else, 6 βρόχους for, 12 βρόχους while και συνολικά μαζί με τα σχόλια περίπου 840 σειρές κώδικα.

57.2 Αρχικός καθορισμός μεταβλητών και πρώτη ομάδα συναρτήσεων

Το αρχείο StrMap_JavaScript_code_gr.js ξεκινά (γραμμές 2 έως 20) με τον ορισμό σειράς αναγκαίων γενικών (global) μεταβλητών και τον καθορισμό αρχικών τιμών σε ορισμένες από αυτές. Πιο συγκεκριμένα ορίζονται οι μεταβλητές: cur_ts, strStatus, text1, qpol, onoff, v_onoff, cleanT, polvertices, qpolygon, traject, QPArray, PArray, PolygArray, iq, ii, coordsq, ntNode και qpoint1. Η cur_ts αποθηκεύει τη χρονική στιγμή των αποτελεσμάτων (από τα αρχεία XML) για να εμφανιστεί αυτή στην ιστοσελίδα στο κατάλληλο πεδίο. Η strStatus αποθηκεύει την κατάσταση του χάρτη ως συμβολοσειρά που εμφανίζεται στην κατάλληλη θέση στην ιστοσελίδα. Η text1 αποθηκεύει τα χαρακτηριστικά των απαντήσεων των ερωτημάτων που θα εμφανιστούν στην πρώτη γραμμή του κάθε πίνακα αποτελεσμάτων. Η qpol παίρνει την τιμή 0 ή 1 και χρησιμοποιείται για την ενεργοποίηση της αφαίρεσης του πολυγώνου και των συντεταγμένων του. Η onoff διατηρεί την τιμή 1 ή 0 ανάλογα με το εάν είναι ενεργή η δημιουργία πολυγώνου ή όχι αντίστοιχα. Η v_onoff διατηρεί την τιμή " On " ή " Off " αντίστοιχα με την onoff και εμφανίζεται στη σχετική θέση στην ιστοσελίδα. Η cleanT χρησιμεύει για την εκκαθάριση των τροχιών των κινούμενων αντικειμένων και αρχικά παίρνει την τιμή 0 ενώ υπό συγκεκριμένες συνθήκες παίρνει την τιμή 1. Η traject χρησιμεύει και αυτή για την εκκαθάριση των τροχιών και αρχικά παίρνει την τιμή "begin" και υπό συγκεκριμένες συνθήκες τις τιμές "clear trajectories" και "stop clearing trajectories". Η polvertices αποθηκεύει τις συντεταγμένες των σημείων του πολυγώνου σε πίνακα. Η qpolygon αποθηκεύει το πολύγωνο που φαίνεται στο χάρτη και χρησιμοποιείται για το ερώτημα περιοχής. Η QPArray (πίνακας) διατηρεί τις τιμές της μεταβλητής qpolygon ώστε να είναι δυνατή η αφαίρεση του πολυγώνου από το χάρτη. Η PolygArray (πίνακας) αποθηκεύει τις τιμές της μεταβλητής qpolygon ώστε να είναι δυνατή η προσθήκη κάθε φορά που κάνουμε κλικ στο χάρτη του νέου ευθύγραμμου τμήματος που δημιουργείται στο ήδη υπάρχον πολύγωνο. Η iq είναι ένας μετρητής που αυξάνεται κάθε φορά που πατάμε κλικ επάνω στο χάρτη ενώ είναι ενεργή η δημιουργία

πολυγώνου. Η ii είναι και αυτή ένας μετρητής που παίρνει την τιμή της από την iq όπως θα δούμε πιο λεπτομερώς παρακάτω και χρησιμοποιείται από την QPArray[]. Η coordsq αποθηκεύει τα ζεύγη των συντεταγμένων του πολυγώνου με τρόπο ώστε να ενσωματώνονται με επιτυχία στο ερώτημα περιοχής και το τελευταίο να εκτελείται όπως θα έπρεπε. Η ntNode διατηρεί ένα αντικείμενο της DOM (Document Object Model) με κείμενο το περιεχόμενο της μεταβλητής text1. Η qpoint1 αποθηκεύει τις συντεταγμένες του πρώτου σημείου του πολυγώνου ώστε να είναι εφικτό το κλείσιμο αυτού όταν ζητάμε την εκτέλεση του ερωτήματος.

Μετά τον ορισμό των μεταβλητών αυτών, ακολουθεί ο καθορισμός ορισμένων συναρτήσεων. Πρώτα δημιουργείται η συνάρτηση f_onoff() η οποία καλείται κάθε φορά που ο χρήστης πατάει το κουμπί On/Off στην ιστοσελίδα της εφαρμογής. Ξεκινώντας η συνάρτηση αυτή εξετάζει την τιμή της μεταβλητής onoff. Εάν είναι onoff=0 τότε αλλάζει την τιμή της onoff σε 1. Επίσης δίνει καινούργια τιμή στη μεταβλητή v_onoff την " On ". Αλλιώς, εάν είναι onoff=1 τότε αλλάζει την τιμή της onoff σε 0. Επίσης δίνει καινούργια τιμή στη μεταβλητή v_onoff την " Off ". Μετά από αυτούς τους ελέγχους και ενέργειες φροντίζει να μεταφέρει την τιμή της v_onoff στο σχετικό πεδίο στην ιστοσελίδα. Το πεδίο αυτό ονομάζεται vv_onoff και εάν ο browser είναι ο Internet Explorer τότε χρησιμοποιώντας την μέθοδο innerText που αυτός αναγνωρίζει, η JavaScript δίδει την τιμή της v_onoff στο πεδίο με id vv_onoff και συνεπώς εμφανίζεται η αλλαγή στην ιστοσελίδα. Αντίστοιχα εάν ο browser δεν είναι ο Internet Explorer τότε χρησιμοποιώντας την μέθοδο textContent που αναγνωρίζει τουλάχιστον ο Firefox και ίσως και οι υπόλοιποι browsers, η JavaScript δίδει την τιμή της v_onoff στο πεδίο vv_onoff και συνεπώς εμφανίζεται η αλλαγή στην ιστοσελίδα.

Ακολουθεί η συνάρτηση με όνομα open_examples() αποστολή της οποίας είναι το άνοιγμα σε νέο παράθυρο με συγκεκριμένες διαστάσεις της ιστοσελίδας με τα Παραδείγματα Ερωτημάτων. Εκτελείται δε, όταν ο χρήστης πατήσει τον σχετικό δεσμό (link) στην ιστοσελίδα. Η συνάρτηση αυτή περιέχει μόνο μία εντολή και θα μπορούσε να παραλειφθεί βάζοντας την εντολή ως χειριστή συμβάντος στον κώδικα HTML (αντί τοποθέτησης της εντολής εκτέλεσης της συνάρτησης ως χειριστή συμβάντος) αλλά καθώς είναι σχετικά μακροσκελής εντολή με αρκετές παραμέτρους ο παρών τρόπος εκτέλεσής της προτιμήθηκε. Επίσης, έτσι ο κώδικας JavaScript είναι εξ' ολοκλήρου μέσα στο script ενώ στους χειριστές συμβάντων είναι μόνο εντολές εκτέλεσης συναρτήσεων.

Μετά καθορίζεται η συνάρτηση openInstr() η οποία αντίστοιχα με την παραπάνω περιέχει την εντολή ανοίγματος σε συγκεκριμένο καινούργιο παράθυρο με συγκεκριμένες διαστάσεις των οδηγιών χρήσης. Εκτελείται όταν ο χρήστης πατήσει τον δεσμό (link) οδηγιών χρήσης.

Ύστερα καθορίζεται η συνάρτηση remPolygon() η οποία περιέχει και αυτή μία μόνο εντολή, την qpol = 1;. Η συνάρτηση αυτή εκτελείται όταν ο χρήστης πατήσει το κουμπί Αφαίρεσης Συντεταγμένων και Πολυγώνου. Όταν λοιπόν εκτελείται αυτή η συνάρτηση, αλλάζει η τιμή της μεταβλητής qpol και από μηδέν (0) που είναι αρχικά γίνεται 1. Παρακάτω θα αναφερθούμε στη συνάρτηση removePolygon_traj η οποία χρησιμοποιεί αυτήν την μεταβλητή για να αφαιρέσει από τον χάρτη το πολύγωνο που δημιούργησε ο χρήστης και το οποίο χρησιμεύει για την εκτέλεση του ερωτήματος περιοχής.

Στη συνέχεια δημιουργείται η συνάρτηση cleanTraj() η οποία εκτελείται όταν ο χρήστης πατήσει το κουμπί Εκκαθάρισης των Τροχιών των κινούμενων αντικειμένων από τον χάρτη. Περιέχει την ακόλουθη εντολή: cleanT = 1; Η εντολή αυτή προφανώς δίνει την τιμή 1 στην cleanT και με την αλλαγή αυτή ενεργοποιείται τμήμα της removePolygon_traj το οποίο αφαιρεί τις τροχιές των αντικειμένων. Για να ξεκινήσουν οι τροχιές να δημιουργούνται από την αρχή χρησιμοποιείται η μεταβλητή traject η οποία για όσο χρόνο είναι ίση με "clear trajectories" κάνει τη συνάρτηση δημιουργίας τροχιών να ξεκινήσει από την αρχή τη δημιουργία τους.

Ακολουθεί ο ορισμός της συνάρτησης queryPolygon(qpoint) η οποία σκοπό έχει τη δημιουργία του πολυγώνου. Η συνάρτηση αυτή έχει μία μεταβλητή-παράμετρο, την qpoint. Ξεκινά με μία υπόθεση: Εάν η μεταβλητή qpolygon έχει τιμή διαφορετική από την "empty" τότε θα πάρει την τιμή PolygArray[0]. Ακολουθεί δεύτερη υπόθεση. Εάν η qpolygon έχει την τιμή null (κάτι που σημαίνει ότι η PolygArray[0] είχε την τιμή null) ή εάν έχει την τιμή "empty" τότε θα ξεκινήσει η δημιουργία του πολυγώνου από την αρχή. Η polvertices γίνεται πίνακας. Το πρώτο στοιχείο της polvertices παίρνει την τιμή της qpoint η οποία έχει αποθηκευμένες τις συντεταγμένες του πρώτου σημείου του πολυγώνου από την θέση που έκανε κλικ ο χρήστης επάνω στον χάρτη. Η qpolygon αποθηκεύει ένα καινούργιο αντικείμενο Πολυγώνου της Google με συντεταγμένες τα στοιχεία του πίνακα polvertices, συγκεκριμένο χρώμα γραμμής ορίων, με συγκεκριμένο ποσοστό αδιαφάνειας (opacity) γραμμής ορίων, συγκεκριμένο χρώμα εσωτερικού πολυγώνου, συγκεκριμένο ποσοστό αδιαφάνειας εσωτερικού πολυγώνου και με την παράμετρο clickable να είναι απενεργοποιημένη. Στο σημείο αυτό τελειώνει η τελευταία υπόθεση. Εάν αυτή η υπόθεση δεν ισχύει (κάτι που σημαίνει ότι το πολύγωνο υπάρχει ήδη), με συγκεκριμένη εντολή, απλώς προστίθεται ένα ακόμα σημείο στο υφιστάμενο πολύγωνο. Ακολούθως, ο πίνακας PolygArray στη θέση 0 αυτού αποθηκεύει την τιμή της qpolygon ώστε σε επόμενη εκτέλεση της συνάρτησης, η qpolygon να ξαναπάρει την τιμή που είχε και να προστεθεί ένα σημείο στο πολύγωνο. Η queryPolygon() εκτελείται όταν κάνουμε κλικ επάνω στο χάρτη, ενώ είναι ενεργοποιημένη η δημιουργία πολυγώνου. Η συνάρτηση αυτή καταλήγει δίδοντας την τιμή της qpolygon ως αποτέλεσμα της εκτέλεσής της.

Επόμενη συνάρτηση είναι η closePolygon() η οποία όταν εκτελείται κλείνει το πολύγωνο και απενεργοποιεί την δημιουργία πολυγώνου για την αποφυγή εσφαλμένης αύξησης των πλευρών αυτού. Φυσικά ο χρήστης μπορεί ανά πάσα στιγμή να ενεργοποιήσει ξανά τη δημιουργία πολυγώνου πατώντας το κουμπί On/Off. Η συνάρτηση αυτή έχει προγραμματιστεί να εκτελείται όταν ο χρήστης πατήσει το κουμπί εκτέλεσης του ερωτήματος περιοχής. Η closePolygon() λοιπόν ξεκινά με την απόδοση νέας τιμής στην onoff (onoff=0). Συνεχίζει με την καταχώρηση της τιμής " Off " στην v_onoff. Ακολουθεί η εμφάνιση της νέας τιμής της v_onoff στην ιστοσελίδα με τον ίδιο τρόπο που την εμφανίζει η συνάρτηση f_onoff(). Η πιο σημαντική εντολή της closePolygon είναι η επόμενη: qpolygon1 = queryPolygon(qpoint1) η οποία εκτελεί τη συνάρτηση δημιουργίας του πολυγώνου με το επόμενο σημείο να είναι το qpoint1 το οποίο διατηρεί το πρώτο σημείο του πολυγώνου. Με άλλα λόγια η εντολή αυτή κλείνει το πολύγωνο.

Μετά καθορίζεται η συνάρτηση OpenTwoLinks() η οποία εκτελείται όταν ο χρήστης πατήσει το κουμπί εκκίνησης δύο όμοιων Ρευμάτων Δεδομένων. Η συνάρτηση αυτή πρώτα θα επιχειρήσει να ανοίξει το αρχείο startStream2.php σε νέο παράθυρο το οποίο δεν θα εμφανιστεί καθόλου και ακριβώς μετά θα επιχειρήσει να ανοίξει το αρχείο startStream.php με τον ίδιο τρόπο. Έτσι, εκτελείται "αθόρυβα" το κάθε αρχείο PHP. Το startStream2.php όταν εκτελεστεί θα ελέγξει εάν είναι ήδη ενεργό το πρώτο ρεύμα δεδομένων και εάν όχι θα ξεκινήσει το δεύτερο ρεύμα. Εάν είναι ήδη ενεργό το πρώτο ρεύμα δεν θα κάνει τίποτε άλλο και θα σταματήσει την εκτέλεσή του. Ακριβώς μετά, τη εκκίνηση εκτέλεσης του startStream2.php, θα ξεκινήσει να εκτελείται και το startStream.php. Αυτό με αντίστοιχο τρόπο θα ελέγξει εάν είναι ήδη ενεργό το πρώτο ρεύμα δεδομένων και εάν δεν είναι θα το εκκινήσει. Ακριβώς πριν ξεκινήσει το κάθε ρεύμα δεδομένων και μετά από τους ελέγχους αυτούς, δημιουργείται σχετικό βοηθητικό αρχείο XML για κάθε περίπτωση το οποίο ανιχνεύεται από επόμενη συνάρτηση της JavaScript (που θα περιγραφεί παρακάτω) και η ύπαρξή του δηλώνει ότι το ρεύμα είναι ενεργό. Πιο συγκεκριμένα, όταν είναι ενεργό το πρώτο ρεύμα, υπάρχει στον ίδιο φάκελο με τα αρχεία της εφαρμογής βοηθητικό αρχείο XML με όνομα "zstreamRunning.xml". Αυτό είναι και το αρχείο του οποίου την ύπαρξη ελέγχει και το startStream.php και το startStream2.php. Αυτό το αρχείο XML δημιουργείται από κατά την εκκίνηση του

startStream.php αμέσως μετά τον έλεγχο φια την ύπαρξή του. Συνεπώς εάν εκτελείτο πρώτα το startStream.php στην OpenTwoLinks() θα είχε προλάβει να δημιουργηθεί το "zstreamRunning.xml" και το startStream2.php θα το εντόπιζε και δε θα ξεκινούσε την εκτέλεση του δεύτερου ρεύματος δεδομένων. Όταν εκτελούνται τα δύο ρεύματα δεδομένων, πρέπει να έχουν ανά πάσα στιγμή τα ίδια δεδομένα. Για να γίνει κάτι τέτοιο πρέπει το δοκιμαστικό ρεύμα δεδομένων να ξεκινήσει ταυτόχρονα ή σχεδόν ταυτόχρονα στα δύο ρεύματα (τα ρεύματα εδώ είναι αντίστοιχα των πινάκων στις βάσεις δεδομένων). Εάν λοιπόν εκτελείται ήδη το πρώτο ρεύμα δεδομένων από πάτημα του σχετικού κουμπιού σε προηγούμενη χρονική στιγμή, τότε δεν πρέπει να ξεκινήσει το δεύτερο ρεύμα δεδομένων αλλά ούτε και το πρώτο δεύτερη φορά. Με αυτή λοιπόν τη σειρά εκτέλεσης των συγκεκριμένων αρχείων PHP που κάνουν τις συγκεκριμένες ενέργειες επιτυγχάνεται ο στόχος μας που είναι όταν εκτελούνται και τα δύο ρεύματα δεδομένων, να έχουν ανά πάσα στιγμή τα ίδια δεδομένα. Η openTwoLinks() λοιπόν, αν και διαθέτει μόνο δύο εντολές, εκτελεί με τον τρόπο που θα έπρεπε μία πολύ σημαντική λειτουργία της εφαρμογής.

Στη συνέχεια καθορίζεται η συνάρτηση queryNoResults() η οποία επιτελεί και αυτή με τη σειρά την σημαντικό έργο στη σωστή λειτουργία της εφαρμογής. Η συνάρτηση αυτή έχει μία παράμετρο την qnr (από τις λέξεις query number) και δηλώνει τον αριθμό του ερωτήματος. Όπως γνωρίζουμε το Βασικό Ερώτημα έχει τον αριθμό 0, το Ερώτημα Περιοχής έχει τον αριθμό 6 και τα τρία Προσαρμοσμένα Ερωτήματα έχουν τον αριθμό 1, 2 και 3 αντίστοιχα με τον αριθμό του κάθε ερωτήματος. Έτσι η qnr μπορεί να πάρει τις τιμές 0, 1, 2, 3 ή 6. Όταν ο χρήστης ζητήσει την εκτέλεση του Βασικού Ερωτήματος εκτός από τις συναρτήσεις ελέγχου του πεδίου μέγιστης διάρκειας εκτέλεσης, εκτός από την εκτέλεση του σχετικού αρχείου PHP με τις αντίστοιχες παραμέτρους, εκτελείται και η queryNoResults() με την qnr να έχει την τιμή 0. Αντίστοιχα εκτελείται η queryNoResults() κατά το πάτημα του κουμπιού εκτέλεσης οποιουδήποτε άλλου ερωτήματος με την αντίστοιχη τιμή της παραμέτρου. Η συνάρτηση αυτή περιέχει δύο εντολές. Η πρώτη αποθηκεύει σε καινούργια συγκεκριμένη τοπική μεταβλητή τη συμβολοσειρά του αρχείου PHP προς εκτέλεση:

```
    var SQWNRFile =
"http://localhost/stopQuery"+qnr+"WithNoResults.php"
```

Τα αρχικά SQWNR προέρχονται από τις λέξεις Stop Query With No Results.

Η δεύτερη εντολή αποθηκεύει σε νέα μεταβλητή το αποτέλεσμα της εκτέλεσης του αρχείου PHP που έχει αποθηκευμένο η SQWNRFile σε νέο παράθυρο του browser που δεν εμφανίζεται στην οθόνη.

```
    var vqueryNoResults =
window.open (SQWNRFile,'RSIFrame','width=100,height=100');
```

Η ουσία είναι ότι αυτή η συνάρτηση εκτελεί το σχετικό για κάθε περίπτωση αρχείο PHP το οποίο ελέγχει εάν το υποβληθέν ερώτημα επιστρέφει αποτελέσματα. Ανάλογα με την τιμή της qnr εκτελείται και διαφορετικό αρχείο PHP. Υπάρχουν πέντε παρόμοια αρχεία PHP για τον έλεγχο ύπαρξης αποτελεσμάτων των πέντε ερωτημάτων. Το τι ακριβώς κάνουν αυτά τα αρχεία θα περιγραφεί αναλυτικά σε επόμενη παράγραφο.

Τέλος, καθορίζεται συνάρτηση η οποία εκτελείται όταν εστιάζουμε σε οποιοδήποτε πεδίο καταχώρησης διαφόρων τιμών στην ιστοσελίδα. Η ενέργειες που κάνει είναι απλά να επιλέγει όλο το κείμενο που είναι ήδη καταχωρημένο στο συγκεκριμένο πεδίο. Η συνάρτηση ονομάζεται SelectAll() και έχει μία παράμετρο την id στην οποία καταχωρούμε το id του HTML πεδίου το οποίο θέλουμε κάθε φορά να επιλέγεται. Οι εντολές που εκτελεί η συγκεκριμένη συνάρτηση είναι οι εξής δύο:

```
    document.getElementById(id).focus();
    document.getElementById(id).select();
```

Η πρώτη φροντίζει να είναι εστιασμένο το συγκεκριμένο πεδίο και η δεύτερη το επιλέγει.

Στο σημείο αυτό τελειώνει η πρώτη ομάδα συναρτήσεων.

5.7.3 Η συνάρτηση load()

Μετά τις πρώτες αυτές συναρτήσεις οι οποίες εκτελούνται σχεδόν όλες όταν ο χρήστης πατήσει κάποιο κουμπί στην ιστοσελίδα της εφαρμογής, ακολουθεί ο καθορισμός της συνάρτησης load η οποία εκτελείται όταν ανοίγει η ιστοσελίδα και περιέχει αρκετές συναρτήσεις και τον αναγκαίο κώδικα για την εμφάνιση του χάρτη και την απεικόνιση των αποτελεσμάτων επ' αυτού αλλά και για την εκτέλεση πλήθους άλλων δυναμικών λειτουργιών της εφαρμογής.

Η συνάρτηση αυτή ξεκινά με μία υπόθεση. Εάν ο browser (φυλλομετρητής ιστοσελίδων) είναι συμβατός με τους χάρτες της Google τότε συνεχίζεται η εκτέλεση της συνάρτησης αυτής αλλιώς εμφανίζεται παράθυρο ειδοποίησης που αναφέρει ότι obrowser δεν είναι συμβατός και δεν εκτελείται το κυρίως μέρος της συνάρτησης.

Εάν λοιπόν ο browser είναι συμβατός τότε ορίζεται μεταβλητή με όνομα map που αποθηκεύει νέο αντικείμενο τύπου GMap2 με παράμετρο το id (map_canvas) ενός DIV στοιχείου της HTML με τη βοήθεια της DOM. Ο χάρτης λοιπόν, θα εμφανιστεί στην ιστοσελίδα στη συγκεκριμένη θέση του στοιχείου DIV με αυτό το id.

Ακολουθεί ο καθορισμός 25 πινάκων που θα χρησιμοποιηθούν για τα αποτελέσματα των ερωτημάτων, την κατάσταση των ρευμάτων και των ερωτημάτων και τις τροχιές των κινούμενων αντικειμένων. Τα ονόματά τους συνοπτικά είναι τα εξής: MarkersArray, ResArray# όπου # = 1, 2, 3, 6, 10, 11, 12, 13, 14, 15 και 16, RoutesArray, RoutesArray# όπου # οι ίδιες τιμές με παραπάνω συν την τιμή 0. Ο αριθμός # αντιπροσωπεύει τον αριθμό του ερωτήματος ή του ρεύματος. Πιο συγκεκριμένα. Τιμή 0 δηλώνει το Βασικό Ερώτημα. Τιμή 1, 2 και 3 δηλώνει το Προσαρμοσμένο ερώτημα 1, 2 και 3 αντίστοιχα. Τιμή 4 και 5 δηλώνει βοηθητικούς πίνακες για την ενημέρωση της κατάστασης του Ρεύματος Δεδομένων 1 και 2 αντίστοιχα στην ιστοσελίδα. Τιμή 6 δηλώνει το Ερώτημα Περιοχής. Τιμή 10, 11, 12, 13 και 16 δηλώνει βοηθητικούς πίνακες για την ενημέρωση της κατάστασης του Βασικού Ερωτήματος, των Προσαρμοσμένων Ερωτημάτων 1, 2 και 3 και του Ερωτήματος Περιοχής αντίστοιχα.

Στη συνέχεια, με τη μέθοδο addControl() προστίθενται τα εξής τρία εργαλεία ελέγχου του χάρτη:

1. GLargeMapControl(). Πρόκειται για εργαλείο ελέγχου της κλίμακας και πεδίου θέασης του χάρτη.
2. GMapTypeControl(). Πρόκειται για εργαλείο ελέγχου του τύπου του χάρτη που εμφανίζεται στην ιστοσελίδα.
3. GOverviewMapControl().Πρόκειται για εργαλείο επισκόπησης σε πολύ μικρότερη κλίμακα του χάρτη που εμφανίζεται στην ιστοσελίδα.

Με τη μέθοδο addMapType() προστίθεται ο φυσικός – εδαφικός χάρτης (G_PHYSICAL_MAP).

Με τη μέθοδο enableScrollWheelZoom() ενεργοποιείται η αλλαγή κλίμακας με τη χρήση της ροδέλας που πιθανότατα διαθέτει το ποντίκι του χρήστη.

Με τη μέθοδο setCenter ορίζεται το κέντρο του χάρτη ως ένα σημείο GLatLng() με τις επιθυμητές συντεταγμένες. Ορίζεται επίσης η προεπιλεγμένη κλίμακα του χάρτη όταν ανοίγει η ιστοσελίδα.

Τέλος με τη μέθοδο setMapType αλλάζουμε τον προεπιλεγμένο χάρτη από τις δορυφορικές εικόνες στον φυσικό-εδαφικό χάρτη.

Στη συνέχεια, καθορίζουμε σε μεταβλητές τα εικονίδια (σημεία-σημάδια) που θα εμφανίζονται επάνω στο χάρτη. Πιο συγκεκριμένα, η μεταβλητή redIcon αποθηκεύει νέο αντικείμενο GIcon ως το προεπιλεγμένο εικονίδιο και με τη μέθοδο image του GIcon αλλάζει η εικόνα με ένα κόκκινο σημείο-σημάδι (red-dot.png). Με αντίστοιχο τρόπο οι μεταβλητές orangeIcon, greenIcon, blueIcon και purpleIcon αποθηκεύουν νέα

αντικείμενα GIcon και με τη μέθοδο image αλλάζει η κάθε εικόνα με πορτοκαλί σημάδι, πράσινο σημάδι, μπλε σημάδι και ιώδες σημάδι αντίστοιχα.

Ακολουθεί ο καθορισμός ενός χειριστή συμβάντος με τη μορφή του ακροατή (Listener):

```
GEvent.addListener(map, 'click', function(overlay,point) {
```

Όταν λοιπόν κάνουμε κλικ επάνω στο χάρτη ενεργοποιείται αυτός ο χειριστής συμβάντος και εκτελείται η συνάρτηση που περιέχει. Η συνάρτηση ξεκινά με τον καθορισμό της μεταβλητής XX η οποία αποθηκεύει ως αριθμό κινητής υποδιαστολής την τιμή του X της συγκεκριμένης θέσης. Με όμοιο τρόπο καθορίζεται η μεταβλητή YY. Ακολούθως τα XX και YY πολλαπλασιάζονται με 1000000 και στρογγυλοποιούνται στον κοντινότερο ακέραιο. Ύστερα τα XX και YY διαιρούνται με 1000000. Με τον τρόπο αυτό κρατάμε τα 6 πρώτα δεκαδικά ψηφία των αριθμών αυτών καθώς παραπάνω ψηφία δεν προσθέτουν καθόλου στην ακρίβεια της θέσης του σημείου αφού διαφορά 0,000001 της μοίρας αντιστοιχεί περίπου σε 11 cm απόσταση επί του εδάφους. Άλλωστε αυτή είναι και η ακρίβεια που παρέχεται για τα κινούμενα αντικείμενα στο ρεύμα δεδομένων.

Ακολουθεί η δημιουργία σημείου της Google με τις παραπάνω συντεταγμένες το οποίο αποθηκεύεται στη μεταβλητή qpoint.

Ύστερα καθορίζεται η μεταβλητή qqpoint ως ο συνδυασμός των συντεταγμένων XX και YY με ένα κόμμα ανάμεσά τους. Θα χρησιμοποιηθεί δε για την εμφάνιση των συντεταγμένων στο πεδίο του ερωτήματος περιοχής.

Η μεταβλητή latLngStr που καθορίζεται αμέσως μετά είναι μία συμβολοσειρά με τις συντεταγμένες όπως αυτές θέλουμε να εμφανίζονται στο σχετικό πεδίο συντεταγμένων κάτω από τον χάρτη στα αριστερά.

Ακολουθεί η εμφάνιση των συντεταγμένων στο σχετικό πεδίο κάτω από τον χάρτη στα αριστερά. Εάν ο browser είναι ο Internet Explorer η εμφάνιση γίνεται με τη μέθοδο του DOM innerText αλλιώς με τη μέθοδο textContent στο πεδίο με id= coordinatesAre. Η συμβολοσειρά προς εμφάνιση είναι το αυτή που έχει αποθηκεύσει η μεταβλητή latLngStr.

Ύστερα υπάρχει μία υπόθεση. Εάν η μεταβλητή onoff έχει τιμή διαφορετική από την τιμή 1 τότε η συνάρτηση αυτή του ακροατή σταματά και δεν εκτελεί τίποτε άλλο. Εάν η onoff είναι ίση με ένα, τότε καθώς είναι ενεργή η δημιουργία πολυγώνου ο ακροατής εκτελεί μια επιπλέον σειρά από εντολές. Πιο συγκεκριμένα: ο μετρητής iq αυξάνεται κατά 1. Η θέση iq του πίνακα PArray παίρνει την τιμή της qqpoint. Εάν ο iq έχει την τιμή 1 τότε η μεταβλητή qpoint1 παίρνει την τιμή qpoint ώστε να είναι καταχωρημένο σε ειδική μεταβλητή το πρώτο σημείο του πολυγώνου. Επίσης, εάν ο iq έχει την τιμή 1 τότε η μεταβλητή coordsq παίρνει την τιμή της θέσης 1 του πίνακα PArray. Εάν ο iq έχει μεγαλύτερη από 1 τιμή τότε η coordsq παίρνει την τιμή coordsq + "," + PArray[iq] οπότε έχει καταχωρημένη τη σειρά των σημείων που έχει πατήσει ο χρήστης με τρόπο ώστε να είναι έτοιμα για ενσωμάτωση με το υπόλοιπο Ερώτημα Περιοχής. Ακολουθεί η εντολή καταχώρησης της τιμής της coordsq στο σχετικό πεδίο συντεταγμένων του Ερωτήματος Περιοχής. Έπειτα εκτελείται η συνάρτηση queryPolygon(qpoint) με παράμετρο την qpoint και το αποτέλεσμα αυτής καταχωρείται στην μεταβλητή qpolygon. Ο μετρητής ii παίρνει την τιμή iq – 1 και η θέση ii του πίνακα QPArray παίρνει την τιμή της qpolygon.

Τέλος το πολύγωνο το οποίο είναι αποθηκευμένο στην qpolygon εμφανίζεται στο χάρτη με τη μέθοδο addOverlay του αντικειμένου GMap2. Με αυτή την εντολή σταματά ο κώδικας που εκτελείται στον ακροατή αυτόν όταν η onoff έχει την τιμή 1.

Ακολουθεί η δημιουργία 8 μεταβλητών οι οποίες θα χρησιμεύσουν για την κατασκευή και εμφάνιση των πινάκων με τα αποτελέσματα στα Προσαρμοσμένα Ερωτήματα 1, 2 και 3. Οι μεταβλητές ονομάζονται: tab0, tab1, tab2, tab3, tab6, tbo, row και cell. Θα αναφερθούμε σε αυτές εκτενέστερα παρακάτω.

Κατόπιν ορίζεται η συνάρτηση processPoints με παραμέτρους τις map, curArray, routeArray, xmlFile και optionIcon. Η map αναφέρεται στο αντικείμενο map – χάρτης της εφαρμογής. Η curArray (current Array) παίρνει τις τιμές MarkersArray για το Βασικό Ερώτημα, ResArray10 (Results Array10) για την κατάσταση του βασικού ερωτήματος, ResArray6 (Results Array6) για το Ερώτημα Περιοχής, ResArray16 (Results Array16) για την κατάσταση του Ερωτήματος Περιοχής, ResArray1 για το Προσαρμοσμένο Ερώτημα 1, ResArray11 για την κατάσταση του Προσαρμοσμένου Ερωτήματος 1, ResArray2 για το Προσαρμοσμένο Ερώτημα 2, ResArray12 για την κατάσταση του Προσαρμοσμένου Ερωτήματος 2, ResArray3 για το Προσαρμοσμένο Ερώτημα 3, ResArray13 για την κατάσταση του Προσαρμοσμένου Ερωτήματος 3, ResArray4 για την κατάσταση του πρώτου Ρεύματος Δεδομένων και τέλος ResArray5 για την κατάσταση του δεύτερου Ρεύματος Δεδομένων. Η παράμετρος routeArray αναφέρεται στους πίνακες με τις τροχιές των κινούμενων αντικειμένων. Οι RoutesArray0, RoutesArray6, RoutesArray1, RoutesArray2 και RoutesArray3 είναι οι πίνακες που αποθηκεύουν τις τροχιές του Βασικού Ερωτήματος, του Ερωτήματος Περιοχής και των Προσαρμοσμένων Ερωτημάτων 1, 2 και 3 αντίστοιχα. Οι RoutesArray10, RoutesArray11, RoutesArray12, RoutesArray13 , RoutesArray14, RoutesArray15 και RoutesArray16 είναι βοηθητικοί πίνακες οι οποίοι χρησιμεύουν για την εμφάνιση της κατάστασης των ερωτημάτων και των ρευμάτων (στην ουσία για τη σωστή εκτέλεση της processPoints() για την εμφάνιση της κατάστασης ερωτημάτων και ρευμάτων). Η παράμετρος xmlFile αντιστοιχεί στο XML αρχείο προς άνοιγμα από τη συνάρτηση και για κάθε μία από τις 12 εκτελέσεις της συνάρτησης processPoints() είναι διαφορετικό. Τέλος η παράμετρος optionIcon δηλώνει το εικονίδιο με το οποίο θα φαίνονται τα αποτελέσματα επί του χάρτη. Για το Βασικό Ερώτημα είναι η μεταβλητή redIcon, για το Ερώτημα Περιοχής είναι η purpleIcon και για το Προσαρμοσμένο Ερώτημα 1, 2 και τρία είναι η orangeIcon, greenIcon και blueIcon αντίστοιχα.

Η συνάρτηση processPoints() ξεκινά με τον καθορισμό της μεταβλητής qid (query id) ανάλογα με την τιμή της παραμέτρου curArray. Έτσι, εάν είναι curArray = MarkersArray τότε qid = 0, εάν curArray = ResArray1 τότε qid = 1, εάν curArray = ResArray2 τότε qid = 2 και πιο συγκεκριμένα το qid παίρνει την ίδια τιμή με τον αριθμό που συνοδεύει το ResArray. Η qid λοιπόν δηλώνει εν μέρει τον αριθμό του ερωτήματος προς ανάγνωση των αποτελεσμάτων και εμφάνιση αυτών στο χάρτη και σε πίνακες. Μπορεί να πάρει τις τιμές 0, 1, 2, 3, 6, 10, 11, 12, 13, 14, 15 και 16. Οι διψήφιες τιμές δεν δηλώνουν κάποιο ερώτημα αλλά βοηθητική εκτέλεση της processPoints ώστε να επιχειρηθεί η σύνδεση με συγκεκριμένα βοηθητικά αρχεία xml η ύπαρξη των οποίων δηλώνει ότι το ερώτημα εκτελείται ή ότι το ρεύμα εκτελείται. Με τις βοηθητικές αυτές εκτελέσεις της processPoints εμφανίζεται η κατάσταση των ερωτημάτων και των ρευμάτων στην ιστοσελίδα της εφαρμογής.

Μετά λοιπόν τον καθορισμό της qid ακολουθεί η αφαίρεση των αποτελεσμάτων από τον χάρτη και η αφαίρεση των τροχιών τους.

Μετά ξεκινά η διαδικασία σύνδεσης της JavaScript με τα διάφορα αρχεία XML με τη χρήση του αντικειμένου GXmlHttp. Η μεταβλητή request αποθηκεύει την αίτηση για επικοινωνία μέσω του αντικειμένου GXmlHttp και της μεθόδου create αυτού. Το άνοιγμα του εξωτερικού αρχείου γίνεται με τη μέθοδο GET ενώ στο όνομα του αρχείου προστίθεται μία παράμετρος που ονομάζεται RANDOM και παίρνει τυχαίες τιμές από σχετική μαθηματική συνάρτηση. Η προσθήκη της παραμέτρου RANDOM που κάθε φορά έχει διαφορετική τιμή συμβάλλει στο επιτυχημένο άνοιγμα του αρχείου κάθε φορά που επιχειρείται αυτό. Ακολουθεί ένας χειριστής συμβάντος ο οποίος όταν ενεργοποιείται εκτελεί συγκεκριμένη συνάρτηση. Ο χειριστής συμβάντος ονομάζεται onreadystatechange και εκτελείται ακριβώς όταν αλλάξει τιμή η readyState, η κατάσταση δηλαδή ετοιμότητας-επικοινωνίας μεταξύ JavaScript και XML. Η readyState (κατάσταση ετοιμότητας-επικοινωνίας) μπορεί να πάρει 5 πιθανές τιμές:

Τιμή της readyState	Επεξήγηση
0	Η αίτηση για σύνδεση δεν έχει ξεκινήσει
1	Η αίτηση για σύνδεση έχει ξεκινήσει
2	Η αίτηση για σύνδεση έχει αποσταλεί
3	Η αίτηση για σύνδεση είναι υπό επεξεργασία
4	Η αίτηση για σύνδεση έχει ολοκληρωθεί

Η συνάρτηση λοιπόν αυτού του ειδικού χειριστή συμβάντος ξεκινά με μία υπόθεση. Εάν η readyState έχει την τιμή 4 (εάν δηλαδή υπάρχει ολοκληρωμένη σύνδεση με το συγκεκριμένο αρχείο xml) τότε συνεχίζει ο κώδικας ως εξής.

Η μεταβλητή xmlDoc αποθηκεύει τις πληροφορίες του αρχείου XML χρησιμοποιώντας τη μέθοδο responseXML του αντικειμένου GXmlHttp. Η μεταβλητή markers (πίνακας) αποθηκεύει τις πληροφορίες που περιέχουν τα στοιχεία <marker> του αρχείου XML. Ορίζονται οι μεταβλητές ts, id, latitude και longitude που θα χρησιμοποιηθούν για την αποθήκευση των πληροφοριών που χρειάζονται για την εμφάνιση των αποτελεσμάτων επί του χάρτη. Ακολουθεί μία υπόθεση. Εάν το qid είναι 0 ή 1 ή 2 ή 3 ή 6 (εάν δηλαδή πρόκειται για ερώτημα με πιθανά αποτελέσματα και όχι για βοηθητική εκτέλεση της processPoints) τότε συνεχίζει η εκτέλεση της συνάρτησης ως εξής:

Επόμενη υπόθεση. Εάν το πλήθος των στοιχείων του πίνακα markers είναι μεγαλύτερο του μηδενός τότε:

Η μεταβλητή row που έχει δημιουργηθεί παραπάνω παίρνει τη μορφή στοιχείου <TR> της HTML, δηλαδή γραμμή πίνακα (table row). Η row παίρνει το χαρακτηριστικό στοίχισης στη μέση.

Ακολουθεί βρόχος ανάγνωσης των χαρακτηριστικών της πρώτης θέσης του πίνακα markers. Κάθε πίνακας στην JavaScript όπως και σε άλλες γλώσσες προγραμματισμού έχει μία διάσταση και συνεπώς υπάρχει ένας δείκτης προς κάθε στοιχείο αυτού. Το πρώτο στοιχείο έχει το δείκτη 0, το δεύτερο το δείκτη 1 κ.ο.κ. Έτσι λοιπόν το πρώτο στοιχείο του πίνακα markers έχει το δείκτη 0, περιέχει δε τα ονόματα των χαρακτηριστικών των στοιχείων marker του XML. Με την εντολή:

var text1 = markers[0].attributes[x].nodeName.toLowerCase();

αποθηκεύουμε στη μεταβλητή text1 το όνομα του χαρακτηριστικού x με μικρά γράμματα. Η μεταβλητή cell που έχει δημιουργηθεί παραπάνω παίρνει τη μορφή στοιχείου <TD> της HTML, δηλαδή κελί πίνακα (table data cell). Η cell παίρνει το χαρακτηριστικό στοίχισης στη μέση και μεγέθους γραμμάτων 2 (HTML ιδιότητα). Ακολούθως, η ntNode που έχει οριστεί στην αρχή του script περιέχει το κείμενο της text1. Η cell με τη μέθοδο appendChild αποκτά την πληροφορία της ntNode. Η row τέλος με την ίδια μέθοδο αποκτά την πληροφορία της cell. Στο σημείο αυτό κλείνει ο βρόχος ο οποίος επαναλαμβάνεται για όλα τα χαρακτηριστικά της πρώτης θέσης του πίνακα markers.

Οι μεταβλητές tab0, tab1, tab2, tab3 και tab6 παίρνουν τη μορφή στοιχείου πίνακα (<TABLE>) της HTML. Παίρνουν επίσης το χαρακτηριστικό border = 2 δηλαδή τα όρια του κάθε πίνακα έχουν πλάτος 2.

Η μεταβλητή tbo παίρνει τη μορφή στοιχείου σώματος πίνακα (<TBODY>) της HTML. Με τη μέθοδο appendChild αποθηκεύει τις πληροφορίες της row.

Ακολουθεί για τα ερωτήματα 1, 2 και 3 η αφαίρεση των πληροφοριών που πιθανώς οι αντίστοιχοι πίνακές τους έχουν με τη σχετική υπόθεση. Εάν το ερώτημα είναι το ερώτημα 1 και ο πίνακας tab1 έχει πληροφορίες τότε αυτές αφαιρούνται. Ομοίως και για τα ερωτήματα 2 και 3.

Ακολουθεί η καταχώρηση στον αντίστοιχο πίνακα ανάλογα με το ερώτημα των πληροφοριών της tbo με τη μέθοδο appendChild.

Για την εμφάνιση των αποτελεσμάτων επί του χάρτη αναζητούνται συγκεκριμένα χαρακτηριστικά στο αρχείο XML. Έτσι, η πληροφορία του χαρακτηριστικού "tcqtime" εάν υπάρχει αποθηκεύεται στη μεταβλητή ts που ορίστηκε νωρίτερα. Στη συνέχεια γίνεται η εμφάνισή της επί της ιστοσελίδας (στη θέση με id=txtTime) με τη βοήθεια των μεθόδων innerText και textContent για την περίπτωση που ο browser είναι ο Internet Explorer και για τις υπόλοιπες περιπτώσεις αντίστοιχα. Μετά καθορίζεται η μεταβλητή strStatus η οποία παίρνει την τιμή:

"Χάρτης ανανεωμένος με δεδομένα ληφθέντα τη χρονική στιγμή '" + cur_ts + "'"

και με αντίστοιχη διαδικασία εμφανίζεται αυτή στην ιστοσελίδα στην κατάλληλη θέση (με id=statusLine). Στο σημείο αυτό τελειώνει η υπόθεση εάν το πλήθος των στοιχείων του πίνακα markers είναι μεγαλύτερο του μηδενός.

Ακολουθεί βρόχος ο οποίος επαναλαμβάνεται όσο ο μετρητής i ξεκινώντας από το μηδέν έχει τιμή μικρότερη από το πλήθος των στοιχείων του πίνακα markers. Μέσα σε αυτόν τον βρόχο εκτελούνται τα εξής:

Η μεταβλητή latitude κρατάει σε μορφή αριθμού κινητής υποδιαστολής το χαρακτηριστικό lat του στοιχείου i του πίνακα markers. Η μεταβλητή longitude κρατάει σε μορφή αριθμού κινητής υποδιαστολής το χαρακτηριστικό lng του στοιχείου i του πίνακα markers. Η μεταβλητή id κρατάει σε μορφή αριθμού κινητής υποδιαστολής το χαρακτηριστικό vid του στοιχείου i του πίνακα markers. Τέλος, η μεταβλητή time κρατάει σε μορφή αριθμού κινητής υποδιαστολής το χαρακτηριστικό tcqtimetou στοιχείου i του πίνακα markers. Δημιουργείται μεταβλητή με όνομα point η οποία αποθηκεύει αντικείμενο GLatLng() με συντεταγμένες αυτές που διατηρούν τα latitude και longitude. Ουσιαστικά πρόκειται για σημείο που αναγνωρίζουν οι χάρτες της Google και σχετικά αντικείμενα και μέθοδοι αυτής.

Ακολουθεί ο καθορισμός της συνάρτησης crMarker η οποία δημιουργεί τα σημεία-σημάδια που θα εμφανιστούν στο χάρτη. Η συνάρτηση αυτή έχει 5 παραμέτρους. Πρώτη είναι η qqid που δηλώνει ουσιαστικά το qid για το κάθε marker. Δεύτερη είναι η iid που δηλώνει το id του αντικειμένου που εμφανίζεται ως αποτέλεσμα στο χάρτη. Τρίτη είναι η llat που δηλώνει το lat (latitude) του κάθε σημείου. Τέταρτη είναι η llng που δηλώνει αντίστοιχα το lng (longitude) του κάθε σημείου. Πέμπτη είναι ttime που δηλώνει τη χρονική στιγμή στην οποία αναφέρεται η συγκεκριμένη θέση του συγκεκριμένου σημείου (που είναι απάντηση συγκεκριμένου ερωτήματος).

Μέσα στη συνάρτηση crMarker λοιπόν πρώτα καθορίζεται μεταβλητή marker ως αντικείμενο GMarker με σημείο το point που δημιουργήθηκε νωρίτερα και εικονίδιο το optionIcon που αντιστοιχεί σε redIcon, purpleIcon, orangeIcon, greenIcon και blueIcon ανάλογα με το ερώτημα. Ακολούθως ορίζεται ένας ακροατής (listener) ως χειριστής συμβάντος ο οποίος όταν κάνουμε κλικ επάνω σε κάποιο marker εκτελεί τις παρακάτω ενέργειες:

Εάν το qqid είναι ίσο με 0 τότε με τη μέθοδο openInfoWindowHtml εμφανίζει παράθυρο πληροφοριών με τα εξής:

"Αποτέλεσμα Βασικού Ερωτήματος
ID = "+iid+"
Συντ/νες = ("+llat+", "+llng+")"+"
Χρόνος = '"+ttime+"'"

όπου iid είναι το vid του κάθε οχήματος, llat το γεωγραφικό πλάτος, llng το γεωγραφικό μήκος και ttime η χρονική στιγμή.

Εάν το qqid είναι ίσο με 0 τότε με τη μέθοδο openInfoWindowHtml εμφανίζει παράθυρο πληροφοριών με τα εξής:

"Αποτέλεσμα Ερωτήματος Περιοχής
ID = "+iid+"
Συντ/νες = ("+llat+", "+llng+")"+"

Χρόνος = "'+ttime+""

Στις υπόλοιπες περιπτώσεις (όπου το qqid είναι ή 1 ή 2 ή 3) με τη μέθοδο openInfoWindowHtml εμφανίζει παράθυρο πληροφοριών με τα εξής:

"Αποτέλεσμα Ερωτήματος "+qqid+"
ID = "+iid+"
Συντ/νες = ("+llat+", "+llng+")"+"
Χρόνος = "'+ttime+""

όπου qqid είναι ο αριθμός του Προσαρμοσμένου Ερωτήματος.

Ο ακροατής (listener) κλείνει στο σημείο αυτό.

Η συνάρτηση crMarker πριν κλείσει επιστρέφει ως αποτέλεσμα εκτέλεσής της την μεταβλητή marker.

Μετά τον καθορισμό της συνάρτησης crMarker ακολουθεί η εκτέλεσή της επάνω στη μεταβλητή marker η οποία αποθηκεύει το αποτέλεσμα της εκτέλεσης της παραπάνω συνάρτησης. Η παράμετρος-πίνακας curArray (που παίρνει τις τιμές MarkersArray ή ResArray# με # τον αριθμό του ερωτήματος εκτός του 0) στη θέση i αποθηκεύει την τρέχουσα τιμή της μεταβλητής marker ώστε να είναι εφικτή η αφαίρεση των σημείων σημαδιών από το χάρτη σε κάθε ανανέωση αυτών.

Αντίστοιχα με την μεταβλητή marker καθορίζεται και η μεταβλητή polyline η οποία εκτελεί τη συνάρτηση adjustTrajectory η οποία καθορίζεται παρακάτω. Η παράμετρος-πίνακας routeArray (που παίρνει τις τιμές RoutesArray# με # τον αριθμό του ερωτήματος) στη θέση i αποθηκεύει την τρέχουσα τιμή της μεταβλητής polyline ώστε να είναι εφικτή η αφαίρεση των τροχιών από το χάρτη σε κάθε ανανέωση αυτών αλλά και όταν πατήσει ο χρήστης το σχετικό κουμπί.

Ακολουθούν δύο σημαντικές εντολές οι οποίες εκτελούνται υπό συγκεκριμένη προϋπόθεση: Εάν η προσπάθεια ανάγνωσης του χαρακτηριστικού lat επιστρέψει τιμή διαφορετική από null και η προσπάθεια ανάγνωσης του χαρακτηριστικού lng επιστρέψει τιμή διαφορετική από null, τότε και μόνο τότε τα δημιουργηθέντα σημεία και οι αντίστοιχες τροχιές εμφανίζονται στο χάρτη με τη μέθοδο addOverlay του αντικειμένου GMap2 αντίγραφο του οποίου έχει αποθηκεύσει η μεταβλητή map.

Ακολουθεί η κατασκευή του υπόλοιπου τμήματος του κάθε πίνακα με τα αποτελέσματα. Καθορίζεται μεταβλητή με όνομα nrow (new row) και παίρνει τη μορφή στοιχείου <TR> της HTML, δηλαδή γραμμή πίνακα (table row). Η nrow παίρνει το χαρακτηριστικό στοίχισης στη μέση. Καθορίζεται βρόχος ο οποίος επαναλαμβάνεται όσο ο μετρητής j ξεκινώντας από το μηδέν έχει τιμή μικρότερη από το πλήθος των στοιχείων του πίνακα attributes ο οποίος είναι αποθηκευμένος στη θέση i του πίνακα markers. Μέσα στο βρόχο αυτό καθορίζεται η μεταβλητή ncell (new cell) και παίρνει τη μορφή στοιχείου <TD> της HTML, δηλαδή κελί πίνακα (table data cell). Η ncell παίρνει το χαρακτηριστικό στοίχισης στη μέση και μεγέθους γραμμάτων 2. Γίνεται η ανάγνωση της πληροφορίας που έχει στη θέση j ο πίνακας attributes που είναι αποθηκευμένος στη θέση i του πίνακα markers. Η πληροφορία αυτή ως κείμενο καταχωρείται με τη μέθοδο appendChild στη μεταβλητή ncell. Το περιεχόμενο της ncell αποθηκεύεται ως πληροφορία στη μεταβλητή nrow. Στο σημείο αυτό κλείνει ο βρόχος ανάγνωσης των στοιχείων του πίνακα attributes.

Ακριβώς μετά γίνεται η καταχώρηση της πληροφορίας που διαθέτει η nrow στη μεταβλητή tbo που έχει οριστεί παραπάνω. Εάν το qid είναι 0, 1, 2, 3 ή 6 η πληροφορία του tbo αποθηκεύεται στη μεταβλητή tab0, tab1, tab2, tab3 ή tab6 αντίστοιχα. Στο σημείο αυτό κλείνει ο βρόχος ανάγνωσης των στοιχείων του πίνακα markers. Οι πίνακες με τα αποτελέσματα είναι πλέον έτοιμοι προς εμφάνιση επί του χάρτη.

Κλείνει η υπόθεση εάν το qid είναι 0 ή 1 ή 2 ή 3 ή 6.

Κλείνει η υπόθεση εάν η readyState έχει την τιμή 4.

Μετά, υπάρχει υπόθεση εάν το qid έχει την τιμή 0. Τότε:

Εάν η readyState έχει την τιμή 3 γίνονται τα εξής:

Για όσο ισχύει η υπόθεση ότι το αντικείμενο DOM με id=txtTime έχει πληροφορίες, αφαιρείται η τελευταία πληροφορία που έχει αυτό το αντικείμενο (ουσιαστικά μέχρι να μην έχει πλέον πληροφορίες το συγκεκριμένο αντικείμενο).

Ομοίως όσο το αντικείμενο DOM με id=statusLine έχει πληροφορίες, αφαιρείται η τελευταία.

Τελειώνει η υπόθεση ότι η readyState έχει την τιμή 3.

Τελειώνει η υπόθεση ότι το qid έχει την τιμή 0.

Πρέπει να αναφερθεί στο σημείο αυτό ότι η readyState για να πάρει την τιμή 4 περνάει διαδοχικά από τις τιμές 1, 2 και 3. Επίσης, εάν υπάρχει το αρχείο XML προς ανάγνωση, τότε η readyState φτάνει μέχρι την τιμή 4 ενώ εάν δεν υπάρχει το XML προς ανάγνωση, τότε η readyState φτάνει μέχρι την τιμή 3. Έτσι λοιπόν, το readyState θα πάρει για κάθε περίπτωση σύνδεσης με οποιοδήποτε XML τις τιμές 1, 2 και 3 από μία φορά και εάν υπάρχει το αρχείο θα πάρει και την τιμή 4 αλλιώς όχι.

Ακολουθεί κώδικας που θα εκτελεστεί εάν το qid είναι ίσο με 10. Το qid παίρνει την τιμή 10 όταν επιχειρείται η ανάγνωση του βοηθητικού XML "zQuery0IsRunning.xml" η ύπαρξη του οποίου δηλώνει ότι το Βασικό Ερώτημα είναι ενεργό. Στην περίπτωση αυτή όσο το αντικείμενο DOM με id=stat0 έχει πληροφορίες, αφαιρείται η τελευταία. Εάν η readyState πάρει την τιμή 4 πράγμα που σημαίνει ότι υπάρχει το "zQuery0IsRunning.xml" τότε η τιμή "Ενεργό..." ("Running...") αποδίδεται στο αντικείμενο DOM με id=stat0 είτε με τη μέθοδο innerText είτε με τη μέθοδο textContent ανάλογα με τον browser. Αλλιώς εάν η readyState πάρει την τιμή 3, η τιμή "Σταματημένο." ("Stopped.") αποδίδεται στο ίδιο αντικείμενο DOM. Φυσικά όταν υπάρχει το αρχείο η readyState θα πάρει διαδοχικά και την τιμή 3 και την τιμή 4. Η τελευταία όμως τιμή θα είναι η 4 και έτσι η κατάσταση του Βασικού Ερωτήματος όταν αυτό είναι ενεργό θα πάρει στιγμιαία την τιμή Σταματημένο και αμέσως μετά και για τα επόμενα 5 δευτερόλεπτα την τιμή Ενεργό. Η τιμή Σταματημένο θα αποδοθεί στο αντικείμενο DOM για τόσο μικρό χρονικό διάστημα που δεν θα εμφανιστεί καν στην ιστοσελίδα. Σε αυτό το σημείο τελειώνει ο κώδικας που εκτελείται όταν το qid είναι ίσο με 10. Προφανώς το στοιχείο της HTML με id=stat0 είναι στη θέση όπου εμφανίζεται η κατάσταση του Βασικού Ερωτήματος.

Μετά, εάν το qid είναι ίσο με 11 ακολουθείται παρόμοια διαδικασία με την περίπτωση που το qid είναι ίσο με 10 με ορισμένες όμως προσθήκες. Αρχικά, όσο το αντικείμενο DOM με id=stat1 έχει πληροφορίες, αφαιρείται η τελευταία. Εάν η readyState πάρει την τιμή 4 πράγμα που σημαίνει ότι υπάρχει το "zQuery1IsRunning.xml" τότε αντίστοιχα με παραπάνω, η τιμή "Ενεργό..." αποδίδεται στο αντικείμενο DOM με id=stat1 είτε με τη μέθοδο innerText είτε με τη μέθοδο textContent ανάλογα με τον browser. Αλλιώς εάν η readyState πάρει την τιμή 3, η τιμή "Σταματημένο." αποδίδεται στο ίδιο αντικείμενο DOM. Το στοιχείο της HTML με id=stat1 είναι στη θέση όπου εμφανίζεται η κατάσταση του Προσαρμοσμένου Ερωτήματος 1. Μετά, όσο το αντικείμενο DOM με id=hhook1 έχει πληροφορίες, αφαιρείται η τελευταία. Το στοιχείο της HTML με id=hhook1 είναι στη θέση όπου εμφανίζεται ο πίνακας με τα αποτελέσματα του Προσαρμοσμένου Ερωτήματος 1. Εάν η readyState πάρει την τιμή 4 πράγμα που σημαίνει ότι υπάρχει το "zQuery1IsRunning.xml" και το ερώτημα είναι ενεργό, καταχωρούνται στο αντικείμενο DOM με id=hhook1 οι πληροφορίες που διαθέτει η μεταβλητή tab1 οι οποίες δεν είναι άλλες από τον πίνακα των αποτελεσμάτων για το Προσαρμοσμένο Ερώτημα 1. Εάν το ερώτημα είναι ενεργό αλλά δεν υπάρχουν αποτελέσματα τότε στο πεδίο αυτό δεν εμφανίζεται τίποτα καθώς η tab1 δεν έχει πληροφορίες να εμφανίσει. Με αυτόν τον τρόπο καταλαβαίνει ο χρήστης ότι το ερώτημα δεν επιστρέφει αποτελέσματα. Φυσικά δεν θα

υπάρχουν ούτε στο χάρτη σχετικά σημεία-σημάδια εάν το ερώτημα δεν επιστρέφει αποτελέσματα. Εάν η readyState σταματήσει στην τιμή 3, τότε το κείμενο:

"Εδώ θα εμφανιστούν τα αποτελέσματα του Ερωτήματος1."
("The results of Query1 will appear here.")
θα εμφανιστεί στη θέση του πίνακα αποτελεσμάτων.
Στο σημείο αυτό κλείνει η υπόθεση ότι το qid είναι ίσο με 11.

Αντίστοιχη υπόθεση υπάρχει για το εάν το qid είναι ίσο με 12. Οι διαφορές είναι φυσικά ότι το στοιχείο HTML στο οποίο φαίνεται η κατάσταση του Προσαρμοσμένου Ερωτήματος 2 είναι αυτό με id=stat2, το στοιχείο HTML στο οποίο εμφανίζεται ο πίνακας με τα αποτελέσματα είναι αυτό με id=hhook2, οι πληροφορίες που καταχωρούνται στο αντικείμενο DOM με id=hhook2 είναι αυτές της μεταβλητής tab2 και όταν το ερώτημα δεν εκτελείται, το κείμενο που εμφανίζεται στη θέση του πίνακα αποτελεσμάτων είναι αντίστοιχα το παρακάτω:

"Εδώ θα εμφανιστούν τα αποτελέσματα του Ερωτήματος2."
("The results of Query2 will appear here.")

Με αντίστοιχο τρόπο, αντίστοιχες υποθέσεις (qid=13), αντίστοιχες μεταβλητές (tab3, id=stat3, id=hhook3) και αντίστοιχες πληροφορίες, εμφανίζεται με τον επόμενο κώδικα η κατάσταση του Προσαρμοσμένου Ερωτήματος 3 και ο πίνακας αποτελεσμάτων ή το σχετικό μήνυμα. Το αρχείο XML η ύπαρξη του οποίου δηλώνει ότι εκτελείται το Προσαρμοσμένο Ερώτημα 3 είναι το " zQuery3IsRunning.xml".

Μετά υπάρχει η υπόθεση εάν το qid είναι ίσο με 14. Το αρχείο XML που επιχειρείται να ανοίξει για ανάγνωση, στην περίπτωση αυτή είναι το "zstreamRunning.xml" και επομένως, εάν αυτό υπάρχει θα εμφανιστεί η τιμή "Ενεργό..." στο πρώτο Ρεύμα Δεδομένων καθώς αυτό δηλώνει η ύπαρξη αυτού του αρχείου. Πιο συγκεκριμένα λοιπόν, όσο το αντικείμενο DOM με id=firStreamStat έχει πληροφορίες, αφαιρείται η τελευταία. Εάν η readyState πάρει την τιμή 4 πράγμα που σημαίνει ότι υπάρχει το "zstreamRunning.xml" τότε η τιμή "Ενεργό..." ("Running...") αποδίδεται στο αντικείμενο DOM με id=firStreamStat είτε με τη μέθοδο innerText είτε με τη μέθοδο textContent ανάλογα με τον browser. Αλλιώς εάν η readyState πάρει την τιμή 3, η τιμή "Σταματημένο." ("Stopped.") αποδίδεται στο ίδιο αντικείμενο DOM. Σε αυτό το σημείο τελειώνει ο κώδικας που εκτελείται όταν το qid είναι ίσο με 14. Προφανώς το στοιχείο της HTML με id=firStreamStat είναι στη θέση όπου εμφανίζεται η κατάσταση του πρώτου Ρεύματος Δεδομένων.

Ακολουθεί υπόθεση εάν το qid είναι ίσο με 15. Αντίστοιχα με την προηγούμενη περίπτωση λοιπόν, το αρχείο XML που επιχειρείται να ανοίξει για ανάγνωση, στην περίπτωση αυτή είναι το "zstreamRunning2.xml" και επομένως, εάν αυτό υπάρχει θα εμφανιστεί η τιμή "Ενεργό..." στο δεύτερο Ρεύμα Δεδομένων καθώς αυτό δηλώνει η ύπαρξη αυτού του αρχείου. Οι εντολές είναι αντίστοιχες με αυτές της προηγούμενης περίπτωσης. Το αντίστοιχο στοιχείο της HTML στο οποίο εμφανίζεται η τιμή "Ενεργό..." ή "Σταματημένο." Είναι αυτό με id=secStreamStat.

Στη συνέχεια υπάρχει υπόθεση εάν το qid είναι ίσο με 16. Στην περίπτωση αυτή εκτελείται το Ερώτημα Περιοχής. Με αντίστοιχες εντολές με την περίπτωση που εκτελείται το Βασικό Ερώτημα εμφανίζεται η κατάσταση του Ερωτήματος Περιοχής στην ιστοσελίδα της εφαρμογής. Πιο συγκεκριμένα όσο το αντικείμενο DOM με id=stat6 έχει πληροφορίες, αφαιρείται η τελευταία πληροφορία. Εάν η readyState γίνει ίση με 4 που σημαίνει ότι υπάρχει το αρχείο "zQuery6IsRunning.xml", τότε η τιμή "Ενεργό..." εμφανίζεται στο στοιχείο της HTML με id=stat6 που βρίσκεται στη θέση εμφάνισης της κατάστασης του Ερωτήματος Περιοχής. Σε άλλη περίπτωση το παραπάνω στοιχείο της

HTML παίρνει την τιμή "Σταματημένο.". Σε αυτό το σημείο τελειώνει η υπόθεση εάν το qid είναι ίσο με 16.

Τελειώνει επίσης η συνάρτηση που εκτελείται από τον ειδικό χειριστή συμβάντος onreadystatechange.

Καθώς η σύνδεση με το κάθε αρχείο XML έγινε με τη μέθοδο open της GXmlHttp και καθώς η αίτηση επικοινωνίας ήταν τύπου "GET" πρέπει κλείνοντας την επικοινωνία να στείλουμε την πληροφορία null στον web server κάτι το οποίο κάνουμε με την εντολή

request.send(null);

Μετά από αυτήν την εντολή, τελειώνει και η συνάρτηση processPoints.

Στη συνέχεια και ενώ είμαστε μέσα στη load, δημιουργούμε τη συνάρτηση adjustTrajectory η οποία δέχεται δύο παραμέτρους. Πρώτη παράμετρος είναι η point και δεύτερη η id. Η point όπως αναφέρει και το όνομά της δέχεται μεταβλητή που έχει αποθηκευμένο κάποιο σημείο ενώ η id δέχεται ως μεταβλητή την id που έχει οριστεί μέσα στη συνάρτηση processPoints και κρατά αποθηκευμένο τον μοναδικό κωδικό αριθμό κάθε κινούμενου αντικειμένου. Όπως έχει προαναφερθεί, η συνάρτηση adjustTrajectory καλείται μέσα από την processPoints και αποθηκεύει το αποτέλεσμα της εκτέλεσής της στη μεταβλητή polyline ου ορίζεται εκεί. Η polyline δεν είναι γενική μεταβλητή καθώς αυτό θα δημιουργούσε πρόβλημα στη λειτουργία της adjustTrajectory και κατ' επέκταση στην εμφάνιση των τροχιών.

Η adjustTrajectory ξεκινά με έναν έλεγχο:

Εάν η μεταβλητή traject δεν έχει την τιμή "empty" τότε ορίζεται η τοπική μεταβλητή polyline ως το στοιχείο id του πίνακα RoutesArray. Αυτό το στοιχείο έχει κρατήσει την πληροφορία που είχε από την προηγούμενη εκτέλεση η τοπική αυτή μεταβλητή polyline. Εάν αυτή είναι η πρώτη φορά που εκτελείται η συνάρτηση για το συγκεκριμένο id τότε η τιμή null αποθηκεύεται στη μεταβλητή polyline. Ακολουθεί ο ορισμός της μεταβλητής vertices. Εάν η polyline που ορίστηκε νωρίτερα έχει την τιμή null κάτι που σημαίνει ότι η RoutesArray στη θέση id είχε την τιμή null, ή η μεταβλητή traject έχει την τιμή "empty" τότε:

Δημιουργείται νέος πίνακας με κορυφές τεθλασμένης γραμμής (ο πίνακας vertices) και η πρώτη θέση αυτού (η θέση 0) αποκτά την τιμή της παραμέτρου point που είναι ουσιαστικά η μεταβλητή point που έχει οριστεί στην συνάρτηση processPoints παραπάνω. Μετά, η μεταβλητή polyline αποθηκεύει νέο αντικείμενο τύπου GPolyline με σημεία αυτά του πίνακα vertices, συγκεκριμένο χρώμα (ανοιχτό γαλάζιο) και πάχος γραμμής 4. Κλείνει η υπόθεση εάν η polyline έχει την τιμή null ή η traject έχει την τιμή "empty". Στην περίπτωση που δεν ισχύει αυτή η υπόθεση, εκτελείται εντολή προσθήκης σημείου στην υφιστάμενη τεθλασμένη γραμμή – τροχιά του συγκεκριμένου αντικειμένου. Μετά αποθηκεύεται η πληροφορία της polyline στη θέση id της RoutesArray ώστε να καταχωρηθεί ξανά στη μεταβλητή polyline σε επόμενη εκτέλεση της συνάρτησης για το ίδιο κινούμενο αντικείμενο (με το ίδιο id). Η adjustTrajectory καταλήγει δίνοντας ως αποτέλεσμα της εκτέλεσής της την πληροφορία που διαθέτει η μεταβλητή polyline. Με τον τρόπο αυτό αποθήκευσης της πληροφορίας για κάθε ξεχωριστό id σε σχετικό πίνακα είναι εφικτή η δημιουργία άπειρου πλήθους διαφορετικών τροχιών και η ενημέρωσή τους κάθε φορά με τις νέες θέσεις των κινούμενων αντικειμένων. Επίσης όταν μέσω άλλης συνάρτησης η μεταβλητή traject πάρει την τιμή "empty", τότε η δημιουργία των τροχιών ξεκινά από την αρχή, όπως συμβαίνει την πρώτη φορά που εκτελείται η adjustTrajectory για κάθε διαφορετικό αντικείμενο.

Μετά τη συνάρτηση adjustTrajectory, καθορίζεται η removePolygon_traj η οποία έχει μια ιδιαιτερότητα. Στο τέλος της εκτέλεσής της καλεί η ίδια τον εαυτό της

μετά από ένα δευτερόλεπτο ακριβώς. Συνεπώς, αυτή η συνάρτηση, αφότου εκτελεστεί για πρώτη φορά, μετά εκτελείται μία φορά κάθε δευτερόλεπτο. Εκτελείται δε συνεχώς μέχρι το κλείσιμο της ιστοσελίδας της εφαρμογής. Η συνάρτηση αυτή δεν είναι απαιτητική καθώς στην πλειονότητα των εκτελέσεών της κάνει δύο ελέγχους οι οποίοι δεν τηρούνται και περιμένει 1 δευτερόλεπτο μέχρι την επανεκτέλεσή της.

Ο πρώτος έλεγχος είναι εάν η μεταβλητή qpol έχει την τιμή 1. Σε αυτήν την περίπτωση εκτελείται ένας βρόχος όσο η τοπική μεταβλητή – μετρητής mn έχει τιμή μικρότερη από το πλήθος των στοιχείων του πίνακα QPArray. Η ενέργεια που κάνει κάθε φορά είναι η αφαίρεση από τον χάρτη της πληροφορίας που έχει η QPArray στη θέση mn. Υπενθυμίζεται ότι η QPArray διατηρεί αποθηκευμένο στις θέσεις της το πολύγωνο που κατασκευάζει ο χρήστης για το ερώτημα περιοχής. Η αφαίρεση του πολυγώνου είναι και η βασική λειτουργία που εκτελεί η removePolygon_traj όταν ο χρήστης πατήσει το κουμπί Αφαίρεσης συντεταγμένων και πολυγώνου (οπότε η qpol παίρνει την τιμή 1). Ο παραπάνω βρόχος εκτελεί μόνο μία ενέργεια κάθε φορά, αυτήν της αφαίρεσης του πολυγώνου. Εκτός από την εκτέλεση του βρόχου, όταν η qpol πάρει την τιμή 1 εκτελούνται επίσης τα εξής: Το πεδίο με τις συντεταγμένες στο ερώτημα περιοχής παίρνει την τιμή "" δηλαδή δεν δείχνει τίποτα, δηλαδή έχουν αφαιρεθεί οι όποιες πληροφορίες από αυτό. Η qpol παίρνει πάλι την τιμή 0 ώστε να μην εκτελείται πλέον κάθε δευτερόλεπτο αυτό το τμήμα κώδικα αλλά να εκτελεστεί μόνο μία φορά. Ο μετρητής iq παίρνει την τιμή 0 ώστε να ξεκινήσει από την αρχή η καταχώρηση στη μεταβλητή coordsq των συντεταγμένων του πολυγώνου. Τέλος η γενική μεταβλητή qpolygon παίρνει την τιμή "empty". Εφόσον η qpolygon πάρει αυτήν την τιμή, σε επόμενη εκτέλεση της συνάρτησης queryPolygon η μεταβλητή qpolygon δεν θα πάρει την τιμή που έχει ο πίνακας PolygArray στη θέση 0 αλλά θα ξεκινήσει η δημιουργία του πολυγώνου από την αρχή και η μεταβλητή αυτή θα αποθηκεύσει νέο αντικείμενο τύπου πολύγωνο (GPolygon) της Google. Στο σημείο αυτό τελειώνουν οι εντολές που εκτελούνται όταν ικανοποιηθεί ο πρώτος έλεγχος. Υπενθυμίζεται ότι αυτές οι εντολές εκτελούνται όταν ο χρήστης πατήσει το κουμπί "Αφαίρεσης Συντεταγμένων και Πολυγώνου" οπότε εκτελείται η συνάρτηση remPolygon η οποία δίδει την τιμή 1 στη μεταβλητή qpol.

Μετά τον πρώτο έλεγχο, υπάρχει ένας δεύτερος. Εάν η μεταβλητή cleanT έχει την τιμή 1 τότε εκτελούνται τα παρακάτω: Εκτελείται βρόχος ο οποίος στόχο έχει την αφαίρεση των τροχιών του ερωτήματος 0 (Βασικό Ερώτημα). Ο μετρητής mn (τοπική μεταβλητή) αυξάνεται κατά 1 σε κάθε εκτέλεση του βρόχου μέχρι να μην ισχύει πλέον η συνθήκη: ο μετρητής mn έχει τιμή μικρότερη από το πλήθος των στοιχείων του πίνακα RoutesArray0. Για όσο λοιπόν ισχύει αυτή η συνθήκη, αφαιρούνται από το χάρτη οι πληροφορίες που διαθέτει ο πίνακας RoutesArray0 στη θέση mn. Έτσι, αφαιρούνται οι τροχιές των αντικειμένων που προκύπτουν ως απαντήσεις στο Βασικό Ερώτημα. Ακολουθεί αντίστοιχος βρόχος ο οποίος καθώς εκτελείται αφαιρούνται από το χάρτη οι τροχιές των αντικειμένων που αποτελούν απαντήσεις στο Προσαρμοσμένο Ερώτημα 1 (πίνακας RoutesArray1). Με τον επόμενο βρόχο εκτελείται αντίστοιχη διαδικασία για την περίπτωση των αποτελεσμάτων του Προσαρμοσμένου Ερωτήματος 2 (πίνακας RoutesArray2). Ο τέταρτος βρόχος αφαιρεί τις τροχιές των κινούμενων αντικειμένων του Προσαρμοσμένου Ερωτήματος 3 (πίνακας RoutesArray3). Ο πέμπτος και τελευταίος βρόχος που εκτελείται όταν η μεταβλητή cleanT πάρει την τιμή 1 είναι αντίστοιχος με τους προηγούμενους και αφαιρεί τις τροχιές των σημείων που αποτελούν απαντήσεις στο Ερώτημα Περιοχής (πίνακας RoutesArray6). Μετά την εκτέλεση των 5 παραπάνω βρόχων, οι τροχιές έχουν αφαιρεθεί από όλα τα κινούμενα αντικείμενα. Εάν δεν κάνουμε καμία άλλη ενέργεια στην περίπτωση που η cleanT πάρει την τιμή 1 τότε a) κάθε δευτερόλεπτο, καθώς θα επαναλαμβάνεται η removePolygon_traj() θα αφαιρούνται οι τροχιές όλων των κινούμενων αντικειμένων b) Ο πίνακας RoutesArray στην επόμενη εκτέλεση της adjustTrajectory θα δώσει την πληροφορία που διαθέτει για την τροχιά κάθε αντικειμένου, στην τοπική μεταβλητή polyline και οι τροχιές θα εμφανιστούν ξανά

ολόκληρες χωρίς να αφαιρεθεί το τμήμα που δημιουργήθηκε από την αρχή εκτέλεσης του ερωτήματος μέχρι τη στιγμή που η cleanT πήρε την τιμή 1. Το πρώτο θέμα που δημιουργείται λύνεται όπως στην περίπτωση της qpol = 1 με αλλαγή τιμής της γενικής μεταβλητής cleanT σε 0. Έτσι οι παραπάνω βρόχοι εκτελούνται μία φορά ο καθένας (Στην ουσία θα εκτελεστούν μία σειρά από φορές ο καθένας καθώς θα εκτελεστούν για mn = 0, mn=1, mn=2 κ.ο.κ.). Το δεύτερο θέμα λύνεται με τη βοήθεια μίας ξεχωριστής μεταβλητής η οποία όταν πάρει συγκεκριμένη τιμή η συνάρτηση δημιουργίας τροχιών adjustTrajectory() ξεκινά τη δημιουργία τους από την αρχή. Η μεταβλητή αυτή είναι η traject στην οποία έχουμε αναφερθεί και παραπάνω και η οποία υπάρχει για τον λόγο αυτό ακριβώς. Συνεπώς η traject παίρνει την τιμή "clear trajectories". Όταν λοιπόν συμβεί κάτι τέτοιο, η συνάρτηση adjustTrajectory() που περιγράφηκε παραπάνω, δεν δίνει στην polyline την τιμή που έχει αποθηκευμένη ο πίνακας RoutesArray στη θέση id, αλλά αποθηκεύει καινούργιο αντικείμενο τύπου GPolyline και η τροχιά δημιουργείται από την αρχή. Εάν βέβαια δεν αλλάξει τιμή η traject, θα δημιουργείται κάθε φορά που θα εκτελείται η συνάρτηση, νέα τροχιά με μόλις ένα σημείο με αποτέλεσμα να μην φαίνεται αυτή στο χάρτη. Εάν μέσα στην adjustTrajectory(), στο τέλος της εκτέλεσης των διαφόρων εντολών αποδοθεί άλλη τιμή στην γενική μεταβλητή traject, τότε θα δημιουργηθεί καινούργια τροχιά μόνο για την πρώτη εκτέλεση της συνάρτησης δημιουργίας τροχιών. Για όλες τις άλλες εκτελέσεις, θα εμφανιστεί πάλι ολόκληρη η τροχιά που υπάρχει από την αρχή της εμφάνισης του συγκεκριμένου κινούμενου αντικειμένου. Καθώς δεν γνωρίζουμε τον αριθμό των κινούμενων αντικειμένων (στην περίπτωσή μας είναι 15 αλλά με διαφορετικό ρεύμα θα είναι διαφορετικό-άγνωστο το πλήθος τους), ούτε είναι συγκεκριμένο το πλήθος των ερωτημάτων που εκτελούνται, δεν μπορούμε να βάλουμε έναν μετρητή που μετά από συγκεκριμένο πλήθος εκτελέσεων της adjustTrajectory() να αλλάζει την τιμή της traject σε κάτι διαφορετικό από την τιμή "clear trajectories". Αυτό που είναι συγκεκριμένο και μπορούμε να χρησιμοποιήσουμε είναι η συχνότητα ανανέωσης των δυναμικών στοιχείων του χάρτη. Όπως θα δούμε στην επόμενη συνάρτηση, η processPoints() εκτελείται για κάθε περίπτωση κάθε 5 δευτερόλεπτα. Έτσι, εάν μετά από συγκεκριμένο χρόνο, η traject αλλάξει τιμή πετυχαίνουμε αυτό που θέλουμε. Για να κάνουμε κάτι τέτοιο χρησιμοποιούμε τη μέθοδο setTimeout η οποία μετά από συγκεκριμένο χρόνο σε millisecond εκτελεί συγκεκριμένες εντολές. Η μέθοδος αυτή είναι διαφορετική από τη συνάρτηση sleep που χρησιμοποιεί η PHP, η C++ και άλλες γλώσσες προγραμματισμού. Η setTimeout δεν ανακόπτει την εκτέλεση του κώδικα μέχρι να περάσει το συγκεκριμένο χρονικό διάστημα αλλά σε καινούργιο thread (νήμα) εκτελεί το συγκεκριμένο κώδικα που έχει καταχωρηθεί σε αυτήν μετά από συγκεκριμένο χρονικό διάστημα. Συνεπώς η setTimeout αποτελεί αυτό ακριβώς που χρειαζόμαστε για την συγκεκριμένη περίπτωση. Πιο συγκεκριμένα τη χρησιμοποιούμε ως εξής:

```
setTimeout('traject = "stop clearing trajectories"',5100);
```

Μετά λοιπόν από 5100 millisecond θα εκτελεστεί η εντολή

```
traject = "stop clearing trajectories"
```

η οποία θα δώσει νέα τιμή στη μεταβλητή traject. Μέχρι τότε, θα έχουν προλάβει όλα τα κινούμενα αντικείμενα να ανανεώσουν τη θέση τους, η adjustTrajectory() θα έχει εκτελεστεί μία φορά για το κάθε κινούμενο αντικείμενο και συνεπώς όλες οι τροχιές θα έχουν ξεκινήσει τη δημιουργία τους από την αρχή. Από τη στιγμή που η traject θα πάρει την τιμή "stop clearing trajectories" (ουσιαστικά μια οποιαδήποτε τιμή διαφορετική από την "clear trajectories"), σε επόμενες εκτελέσεις της η adjustTrajectory() θα προσθέτει στην ήδη υφιστάμενη τροχιά κάθε κινούμενου αντικειμένου το τελευταίο ευθύγραμμο τμήμα μεταξύ της προηγούμενης και της νέας θέσης του κάθε αντικειμένου, με αποτέλεσμα να σχηματίζεται η τροχιά του.

Με την τελευταία αυτή εντολή που χρησιμοποιεί τη μέθοδο setTimeout τελειώνει η ομάδα εντολών που εκτελείται όταν η μεταβλητή cleanT πάρει την τιμή 1.

Πριν σταματήσει την εκτέλεσή της η συνάρτηση removePolygon_traj() έχει όπως έχουμε προαναφέρει σχετική εντολή επανεκτέλεσής της η οποία είναι άλλη μία περίπτωση όπου χρησιμοποιείται η μέθοδος setTimeout. Καλείται λοιπόν μετά από 1 sec, σε καινούργιο thread κάθε φορά, η ίδια η removePolygon_traj() και συνεπώς εκτελείται συνεχώς.

Στο σημείο αυτό τελειώνει ο καθορισμός της παραπάνω συνάρτησης και αμέσως μετά υπάρχει εντολή εκτέλεσής της για πρώτη φορά στο script. Η εντολή είναι η εξής:

```
removePolygon_traj();
```

Από τη στιγμή λοιπόν που εκτελεστεί μία φορά η συγκεκριμένη συνάρτηση, εκτελείται συνεχώς: μία φορά κάθε δευτερόλεπτο.

Ακολουθεί η συνάρτηση ανανέωσης των περισσοτέρων δυναμικών στοιχείων της εφαρμογής. Λέγεται refreshMap() και έχει μία παράμετρο, την map. Η συνάρτηση αυτή δεν κάνει τίποτε άλλο παρά να εκτελεί συνολικά 12 φορές την processPoints() με διαφορετικές παραμέτρους κάθε φορά και να επαναλαμβάνει τον εαυτό της κάθε 5 δευτερόλεπτα. Η processPoints υπενθυμίζουμε ότι έχει 5 παραμέτρους, την map, την curArray, την routeArray, την xmlFile και την optionIcon. Η πρώτη εκτέλεση της processPoints() γίνεται ως εξής:

```
processPoints(map, MarkersArray, RoutesArray0,
"results0.xml", redIcon);
```

και θα εμφανίσει τα αποτελέσματα εκτέλεσης του Βασικού Ερωτήματος στο χάρτη.

Η μεταβλητή map δηλώνει το χάρτη που θα εμφανιστεί στην ιστοσελίδα. Η MarkersArray είναι μεταβλητή με μορφή πίνακα και αποθηκεύει σε κάθε θέση της τις πληροφορίες που έχει κάθε φορά η μεταβλητή marker ώστε να είναι δυνατή η αφαίρεση όλων των σημείων που είναι αποτέλεσμά του Βασικού Ερωτήματος πριν την εμφάνιση των νέων αποτελεσμάτων. Η RoutesArray0 είναι και αυτή μεταβλητή με μορφή πίνακα και αντίστοιχα με την MarkersArray αποθηκεύει σε κάθε θέση της τις πληροφορίες που έχει κάθε φορά η μεταβλητή polyline ώστε να είναι δυνατή η αφαίρεση όλων των τροχιών που είναι αποτέλεσμά του Βασικού Ερωτήματος πριν την εμφάνιση των νέων τροχιών. Το "results0.xml" είναι το όνομα του αρχείου XML που η JavaScript θα επιχειρήσει να διαβάσει με σκοπό την εμφάνιση των αποτελεσμάτων του Βασικού Ερωτήματος τα οποία περιέχονται σε αυτό το αρχείο. Εάν δεν υπάρχουν αποτελέσματα στο Βασικό Ερώτημα (από την εκτέλεση του σχετικού αρχείου PHP), τότε το αρχείο αυτό δεν δημιουργείται και συνεπώς η ειδική μεταβλητή readyState δεν φτάνει ποτέ την τιμή 4. Έτσι, εάν δεν υπάρχει το "results0.xml" δεν εκτελείται ο σχετικός κώδικας που προορίζεται για την περίπτωση κατά την οποία το αρχείο υπάρχει. Τέλος η παράμετρος optionIcon λαμβάνει τις πληροφορίες που είναι αποθηκευμένες στη μεταβλητή redIcon οι οποίες δεν είναι άλλες από τις πληροφορίες για την εμφάνιση κόκκινων σημείων-σημαδιών επάνω στο χάρτη ως αποτελέσματα του Βασικού Ερωτήματος.

Η επόμενη εκτέλεση της processPoints() είναι η εξής:

```
processPoints(map, ResArray10, RoutesArray10,
"zQuery0IsRunning.xml", redIcon);
```

Κατά την εκτέλεση της συνάρτησης με τις παραπάνω μεταβλητές και τιμές στη θέση των 5 παραμέτρων, επιχειρείται το άνοιγμα και η ανάγνωση των πληροφοριών που διαθέτει το αρχείο "zQuery0IsRunning.xml". Το αρχείο αυτό, όταν υπάρχει περιέχει απλά ένα στοιχείο με όνομα markers και τίποτε άλλο, χωρίς απαντήσεις σε ερωτήματα ή άλλες πληροφορίες. Πρόκειται για ένα βοηθητικό αρχείο το οποίο χρησιμεύει για την επικοινωνία διαφορετικών προγραμμάτων. Δημιουργείται κατά την έναρξη εκτέλεσης του Βασικού Ερωτήματος, από το σχετικό αρχείο PHP, με σκοπό να δείξει απλά ότι το ερώτημα εκτελείται. Εάν λοιπόν υπάρχει το αρχείο τότε η readyState θα πάρει εκτός των άλλων τιμών και την τιμή 4. Στην περίπτωση αυτή λοιπόν, όπως έχει προαναφερθεί, η τιμή "Ενεργό..." θα εμφανιστεί στο σχετικό πεδίο δίπλα στο κείμενο "Κατάσταση

Βασικού Ερωτήματος" στην ιστοσελίδα της εφαρμογής. Η εκτέλεση της processPoints() με αυτά τα στοιχεία δεν κάνει κάτι παραπάνω. Για την αποφυγή εκτέλεσης άσκοπων εντολών από την JavaScript, έχει τοποθετηθεί σχετική προϋπόθεση μέσα στον κώδικα της συνάρτησης (εάν το qid είναι ίσο με 0 ή 1 ή 2 ή 3 ή 6), η οποία έχει αναφερθεί κατά την περιγραφή των λειτουργιών που εκτελούνται από τη συνάρτηση αυτή.

Ακολουθεί η εκτέλεση της ίδιας συνάρτησης με τα εξής στοιχεία:

```
processPoints(map, ResArray6, RoutesArray6, "results6.xml",
purpleIcon);
```

Πρόκειται για την ανάγνωση των αποτελεσμάτων της PHP για το Ερώτημα Περιοχής. Η ResArray6 αποθηκεύει τα σημεία-αποτελέσματα ένα προς ένα σε ξεχωριστή θέση του πίνακα με σκοπό την αφαίρεσή τους κάθε φορά πριν την εμφάνιση των καινούργιων αποτελεσμάτων (επί του χάρτη). Αντίστοιχα η RoutesArray6 αποθηκεύει τις τροχιές του κάθε αντικειμένου για την αφαίρεσή τους σε κάθε ανανέωση των αποτελεσμάτων αλλά και σε περίπτωση διακοπής του ερωτήματος αυτού. Το αρχείο "results6.xml" είναι αυτό που δημιουργείται από την εκτέλεση του PHP το οποίο υποβάλλει το σχετικό ερώτημα στην PostgreSQL και στο TelegraphCQ και αποθηκεύει σε αυτό το αρχείο τα αποτελέσματα. Από αυτό το αρχείο λοιπόν η JavaScript διαβάζοντάς το θα εμφανίσει τα σχετικά αποτελέσματα επί του χάρτη. Η μεταβλητή purpleIcon αντίστοιχα με τη redIcon για το Βασικό Ερώτημα, θα συμβάλλει στην εμφάνιση σημείων – σημαδιών ιώδους χρώματος στο χάρτη ως αποτελέσματα του Ερωτήματος Περιοχής.

Η τέταρτη φορά εκτέλεσης της processPoints() γίνεται ως εξής:

```
processPoints(map, ResArray16, RoutesArray16,
"zQuery6IsRunning.xml", redIcon);
```

Η μεταβλητή map υπάρχει σε όλες τις εκτελέσεις της συνάρτησης αυτής και διατηρεί τις πληροφορίες για την εμφάνιση του χάρτη στην ιστοσελίδα. Οι μεταβλητές ResArray16, RoutesArray16 και redIcon στην εκτέλεση αυτή έχουν βοηθητικό ρόλο και στην ουσία δεν χρησιμοποιούνται καθώς το αρχείο "zQuery6IsRunning.xml" εάν υπάρχει, δεν περιέχει απαντήσεις επί ερωτημάτων προς εμφάνιση. Το αρχείο αυτό αντίστοιχα με το "zQuery0IsRunning.xml" δημιουργείται από την PHP κατά την εκτέλεση του Ερωτήματος Περιοχής και χρησιμεύει μόνο για την εμφάνιση της κατάσταση του ερωτήματος αυτού ως "Ενεργό..." από την JavaScript στην ιστοσελίδα. Αντίστοιχα λοιπόν με την δεύτερη εκτέλεση της processPoints(), σε αυτήν την περίπτωση η εκτέλεσή της έχει ως αποτέλεσμα την εμφάνιση της κατάστασης του Ερωτήματος Περιοχής ως ενεργό εάν υπάρχει το αρχείο XML ή ως σταματημένο εάν το αρχείο δεν υπάρχει.

Μετά, εκτελείται για πέμπτη φορά η processPoints() με τα εξής στοιχεία:

```
processPoints(map, ResArray1, RoutesArray1, "results1.xml",
orangeIcon);
```

Στην περίπτωση αυτή επιχειρείται η ανάγνωση του αρχείου "results1.xml" και συνεπώς η εμφάνιση επί του χάρτη των αποτελεσμάτων εκτέλεσης του Προσαρμοσμένου Ερωτήματος 1. Επίσης δημιουργείται ο HTML πίνακας tab1 με τα αποτελέσματα του ερωτήματος αυτού. Οι μεταβλητές ResArray1, RoutesArray1 και orangeIcon εκτελούν τις αντίστοιχες λειτουργίες με τις ResArray6, RoutesArray6 και purpleIcon του Ερωτήματος Περιοχής για την περίπτωση του Προσαρμοσμένου Ερωτήματος 1. Κατά την εκτέλεση της παραπάνω συνάρτησης με τα στοιχεία αυτά, δημιουργούνται και εμφανίζονται επί του χάρτη τα σημεία σημάδια με τα αποτελέσματα του συγκεκριμένου ερωτήματος και αποθηκεύονται οι απαραίτητες πληροφορίες για την εμφάνιση του HTML πίνακα με τα ίδια αποτελέσματα στη μεταβλητή tab1. Με την εκτέλεση της συνάρτησης με αυτά τα στοιχεία, δεν γίνεται η εμφάνιση του HTML πίνακα με τα αποτελέσματα επί της ιστοσελίδας. Αυτή γίνεται στην επόμενη εκτέλεση της συνάρτησης.

Η processPoints() όταν εκτελείται για έκτη φορά μέσα από τη συνάρτηση refreshMap() έχει τα εξής στοιχεία στις θέσεις των παραμέτρων αυτής:

```
processPoints(map, ResArray11, RoutesArray11,
"zQuery1IsRunning.xml", redIcon);
```

Όπως στη δεύτερη και την τέταρτη εκτέλεση της συνάρτησης, οι αντίστοιχες μεταβλητές ResArray11, RoutesArray11 και redIcon έχουν βοηθητικό ρόλο. Στην περίπτωση αυτή η συνάρτηση επιχειρεί τη σύνδεση με το αρχείο "zQuery1IsRunning.xml" το οποίο όταν υπάρχει σημαίνει ότι το Προσαρμοσμένο Ερώτημα 1 εκτελείται. Όταν δεν υπάρχει το αρχείο αυτό σημαίνει ότι το παραπάνω ερώτημα δεν εκτελείται. Κατά τη συγκεκριμένη εκτέλεση της processPoints() εάν υπάρχει το αρχείο "zQuery1IsRunning.xml", η readyState παίρνει την τιμή 4, οπότε εμφανίζεται η τιμή "Ενεργό..." στην κατάσταση του Ερωτήματος 1 και ο πίνακας των αποτελεσμάτων του Προσαρμοσμένου Ερωτήματος 1 στη σχετική θέση στην ιστοσελίδα (κάτω από το πλαίσιο του συγκεκριμένου ερωτήματος).

Η έβδομη εκτέλεση της processPoints() γίνεται ως εξής:

```
processPoints(map, ResArray2, RoutesArray2, "results2.xml",
greenIcon);
```

Είναι δε απολύτως αντίστοιχη με την πέμπτη εκτέλεση αλλά για την περίπτωση του Προσαρμοσμένου Ερωτήματος 2.

Η όγδοη εκτέλεση της συνάρτησης γίνεται με τα παρακάτω στοιχεία:

```
processPoints(map, ResArray12, RoutesArray12,
"zQuery2IsRunning.xml", redIcon);
```

Είναι και αυτή αντίστοιχη με την έκτη εκτέλεση της συνάρτησης για την περίπτωση όμως του Προσαρμοσμένου Ερωτήματος 2. Εμφανίζει την κατάσταση του ερωτήματος στην ιστοσελίδα και τον πίνακα των αποτελεσμάτων αυτού στη σχετική θέση.

Η ένατη εκτέλεση της processPoints() γίνεται με την παρακάτω εντολή:

```
processPoints(map, ResArray3, RoutesArray3, "results3.xml",
blueIcon);
```

Είναι δε απολύτως αντίστοιχη με την πέμπτη και την έβδομη εκτέλεση αλλά για την περίπτωση του Προσαρμοσμένου Ερωτήματος 3.

Ομοίως, η δέκατη εκτέλεση της συνάρτησης γίνεται ως εξής:

```
processPoints(map, ResArray13, RoutesArray13,
"zQuery3IsRunning.xml", redIcon);
```

Είναι και αυτή αντίστοιχη με την έκτη και την όγδοη εκτέλεση της συνάρτησης για την περίπτωση όμως του Προσαρμοσμένου Ερωτήματος 3.

Η ενδέκατη εκτέλεση της processPoints() γίνεται με την παρακάτω εντολή:

```
processPoints(map, ResArray14, RoutesArray14,
"zstreamRunning.xml", redIcon);
```

και είναι αντίστοιχη με τη δεύτερη και την τέταρτη εκτέλεση της συνάρτησης. Εμφανίζει την κατάσταση του πρώτου ρεύματος δεδομένων στη σχετική θέση στην ιστοσελίδα. Εάν υπάρχει το "zstreamRunning.xml" τότε εμφανίζεται η τιμή "Ενεργό..." στην κατάσταση του $1^{ου}$ ρεύματος δεδομένων. Εάν δεν υπάρχει το αρχείο αυτό, τότε εμφανίζεται η τιμή "Σταματημένο." στην κατάσταση του $1^{ου}$ ρεύματος δεδομένων. Το πώς ακριβώς γίνεται αυτή η εμφάνιση έχει αναλυθεί κατά την περιγραφή της processPoints() (είναι η περίπτωση όπου το qid έχει την τιμή 14).

Η δωδέκατη και τελευταία εκτέλεση της συνάρτησης processPoints() γίνεται ως εξής:

```
processPoints(map, ResArray15, RoutesArray15,
"zstreamRunning2.xml", redIcon);
```

Είναι απολύτως αντίστοιχη με την ενδέκατη εκτέλεση για την περίπτωση όμως του $2^{ου}$ Ρεύματος Δεδομένων.

Στο σημείο αυτό πρέπει να αναφερθεί ότι η σειρά με την οποία εκτελούνται οι διάφορες περιπτώσεις της processPoints() δημιουργεί εμφανείς διαφορές στο αποτέλεσμα που φαίνεται στο χάρτη. Όταν τα αποτελέσματα δύο ή περισσοτέρων ερωτημάτων

περιλαμβάνουν ορισμένα ίδια κινούμενα αντικείμενα, το αποτέλεσμα του ενός ερωτήματος θα βρίσκεται ακριβώς επάνω στο αποτέλεσμα του άλλου. Αυτό που θα φαίνεται στο χάρτη είναι το αποτέλεσμα του τελευταίου ερωτήματος που εκτελέστηκε από την παραπάνω σειρά των processPoints(). Έχει επιλεγεί λοιπόν πρώτα να εμφανίζονται τα αποτελέσματα του Βασικού ερωτήματος, μετά τα αποτελέσματα του Ερωτήματος Περιοχής, μετά τα αποτελέσματα του Προσαρμοσμένου Ερωτήματος 1, μετά του 2 και μετά του 3. Έτσι, εάν εκτελούνται όλα τα ερωτήματα με το Ερώτημα Περιοχής να περιλαμβάνει όλη την περιοχή στην οποία κινούνται τα οχήματα και τα τρία Προσαρμοσμένα Ερωτήματα να εκτελούν ουσιαστικά το ίδιο ερώτημα με το Βασικό, τότε κάθε ερώτημα θα εμφανίζει και τα 15 οχήματα (ή όσα είναι γενικότερα) και τα σημεία επάνω στο χάρτη θα είναι ακριβώς στις ίδιες θέσεις. Σε αυτήν την περίπτωση στην ιστοσελίδα θα φαίνονται μόνο τα αποτελέσματα του Προσαρμοσμένου Ερωτήματος 3 καθώς αυτό είναι το τελευταίο που εκτελείται κάθε φορά από την refreshMap() χρησιμοποιώντας την processPoints() ανάλογα. Γενικά, τα αποτελέσματα του Προσαρμοσμένου Ερωτήματος 3 θα εμφανίζονται πάνω από όλα τα άλλα, τα αποτελέσματα του Προσαρμοσμένου Ερωτήματος 2 θα εμφανίζονται πάνω από όλα εκτός από αυτά του Ερωτήματος 3, τα αποτελέσματα του Ερωτήματος 1 θα εμφανίζονται πάνω αυτά του Ερωτήματος Περιοχής και αυτά του Βασικού Ερωτήματος, τα αποτελέσματα του Ερωτήματος Περιοχής θα εμφανίζονται μόνο πάνω αυτά του Βασικού Ερωτήματος και τα αποτελέσματα του Βασικού Ερωτήματος θα εμφανίζονται κάτω από όλα. Ειδικά τα αποτελέσματα του τελευταίου αυτού ερωτήματος δε θα έπρεπε να εμφανίζονται πάνω από οποιοδήποτε άλλο καθώς στην περίπτωση αυτή, εάν εκτελείται το Βασικό Ερώτημα (που επιστρέφει όλα τα κινούμενα αντικείμενα), τότε οι απαντήσεις του άλλου ερωτήματος θα κρυβόντουσαν από τις απαντήσεις του Βασικού. Με την ίδια λογική, θεωρώντας ότι το πολύγωνο που θα σχεδιαστεί για το Ερώτημα Περιοχής θα είναι αρκετά μεγάλο ώστε στις απαντήσεις να υπάρχουν τουλάχιστον 3-4 αντικείμενα, αυτό έχει μπει δεύτερο στη σειρά εμφάνισης ενώ τα υπόλοιπα για λόγους ομοιομορφίας έχουν επιλεγεί να εμφανίζονται με τη σειρά. Το τελευταίο που έχει επιλεχθεί να εμφανίζεται είναι το Προσαρμοσμένο Ερώτημα 3 και γι' αυτόν τον λόγο τα αποτελέσματά του θα εμφανίζονται πάντα πάνω από τα άλλα.

Ακολουθεί η εντολή επανεκτέλεσης της συνάρτησης refreshMap() μετά από 5 δευτερόλεπτα.

Στο σημείο αυτό τελειώνει ο κώδικας της συνάρτησης refreshMap(). Η συνάρτηση αυτή όπως είναι φανερό είναι η αρμόδια για την ανανέωση όλων των δυναμικών στοιχείων επάνω στο χάρτη και στην ιστοσελίδα γενικότερα.

Ακριβώς μετά τον καθορισμό της, υπάρχει η εντολή της πρώτης εκτέλεσής της:

```
refreshMap(map);
```

Αρκεί κατά την εκτέλεση της load μέσα στην οποία βρίσκονται όλες αυτές οι συναρτήσεις να υπάρχει μία φορά η παραπάνω εντολή και η refreshMap() θα επαναλαμβάνει την εκτέλεσή της κάθε 5 δευτερόλεπτα.

Στο σημείο αυτό τελειώνουν οι εντολές που εκτελούνται όταν browser είναι συμβατός με τους χάρτες της Google.

Εάν ο browser δεν είναι συμβατός εμφανίζεται το εξής μήνυμα στο χρήστη:

"The Google Maps API is not compatible with this browser. We are sorry for the inconvenience. Το Google Maps API δεν είναι συμβατό με αυτόν τον browser. Λυπούμαστε για την αναστάτωση."

Στο σημείο αυτό τελειώνει η συνάρτηση load.

5.7.4 Δεύτερη ομάδα συναρτήσεων κώδικα JavaScript – Έλεγχος πεδίων

Ακολουθούν οι συναρτήσεις ελέγχου των τιμών των διαφόρων πεδίων.

Κάθε έλεγχος χρησιμοποιεί μία συγκεκριμένη συνάρτηση για την εκτέλεσή του. Η συνάρτηση κάνει το σχετικό έλεγχο και φροντίζει για την εμφάνιση του ανάλογου μηνύματος στο χρήστη με τη μέθοδο alert (εμφάνιση νέου παραθύρου ειδοποίησης με συγκεκριμένο μήνυμα). Κάθε συνάρτηση εκτελείται περισσότερες από μία φορές σε διαφορετικό πεδίο και με διαφορετικό σχετικό μήνυμα.

Ο πρώτος έλεγχος που καθορίζεται είναι εάν το συγκεκριμένο πεδίο ελέγχου είναι κενό. Χρησιμοποιεί δε τη συνάρτηση validate_required() η οποία έχει δύο παραμέτρους: την field και την alerttxt. Οι παράμετροι καθιστούν ικανή την εκτέλεση της ίδιας συνάρτησης για πολλά διαφορετικά πεδία και όπου αυτή χρειάζεται. Η παράμετρος field αναμένει το όνομα του πεδίου της HTML το οποίο θα ελέγξει η συνάρτηση, όπως ορίζεται με την παράμετρο name μέσα στον καθορισμό του στοιχείου HTML. Η παράμετρος alerttxt αναμένει το μήνυμα προς εμφάνιση στην περίπτωση που ο έλεγχος βρει λάθος στο πεδίο. Για παράδειγμα, στην περίπτωση εκτέλεσης του Προσαρμοσμένου Ερωτήματος 1, όταν υποβάλλουμε το ερώτημα γίνεται έλεγχος με την συγκεκριμένη συνάρτηση εάν το πεδίο εισαγωγής του ερωτήματος είναι κενό. Το πεδίο εισαγωγής του ερωτήματος είναι ένα στοιχείο <TEXTAREA> της HTML και έχει το όνομα query1. Το μήνυμα εάν το query1 κενό είναι το εξής:

"Εσφαλμένη εισαγωγή δεδομένων! Το πεδίο του Ερωτήματος1 δεν μπορεί να είναι κενό."

("Typing Error! Query1 Field must be filled out.")

Η ίδια συνάρτηση, στην περίπτωση του Ερωτήματος 1 ελέγχει επίσης εάν είναι κενό το πεδίο εισαγωγής μέγιστης διάρκειας εκτέλεσης.

Η validate_required() αποτελείται από τον ακόλουθο κώδικα:

Ξεκινά με τη λέξη-κλειδί της JavaScript, with:

```
with (field) {
```

Η λέξη-κλειδί with καθορίζει ένα αντικείμενο και στη συνέχεια έχει κώδικα μέσα σε άγκιστρα. Μέσα στα άγκιστρα, οι πιθανές ιδιότητες και μέθοδοι που αναφέρουμε χωρίς να προσδιορίσουμε το αντικείμενο αναφέρονται αυτομάτως στο αντικείμενο που έχουμε ορίσει σε παρενθέσεις ακριβώς μετά τη λέξη-κλειδί with. Το όνομα του πεδίου λοιπόν της HTML είναι στην περίπτωσή μας το αντικείμενο που εισάγεται μετά τη with. Μέσα στα άγκιστρα υπάρχει μία υπόθεση:

Εάν η ιδιότητα value του πεδίου έχει την τιμή null ή είναι κενή τότε να εμφανιστεί το μήνυμα που διαθέτει η παράμετρος alerttxt. Η ιδιότητα value ενός πεδίου είναι ουσιαστικά το περιεχόμενο του πεδίου. Εάν λοιπόν το περιεχόμενο του πεδίου έχει την τιμή null ή είναι κενό τότε θα εμφανιστεί στην οθόνη το μήνυμα της alerttxt. Αυτή η συνάρτηση εκτελείται για όλα τα πεδία της ιστοσελίδας.

Ακολουθεί ο καθορισμός της συνάρτησης ελέγχου ενός πεδίου για θετική αριθμητική τιμή. Ονομάζεται validate_number και έχει και αυτή τις ίδιες δύο παραμέτρους με την προηγούμενη. Συγκεκριμένα όλες οι συναρτήσεις ελέγχου πεδίων έχουν τις ίδιες παραμέτρους, την field και την alerttxt. Με τον τρόπο αυτό ορίζεται το πεδίο ελέγχου κάθε φορά και το μήνυμα λάθους που θα εμφανιστεί εάν πληρούνται οι προϋποθέσεις για την εμφάνισή του. Η χρήση της συνάρτησης ελέγχου θετικής αριθμητικής τιμής γίνεται στα πέντε πεδία μέγιστης διάρκειας εκτέλεσης. Στα πεδία αυτά, για την εκτέλεση του ερωτήματος πρέπει να έχει εισαχθεί θετική αριθμητική τιμή. Η συνάρτηση περιέχει τη λέξη κλειδί with και αντικείμενο το όνομα του πεδίου "field" όπως η validate_required() και μέσα στα άγκιστρα το σχετικό έλεγχο. Συγκεκριμένα όλες οι συναρτήσεις ελέγχου ξεκινούν την εκτέλεσή τους με τη λέξη-κλειδί with και αντικείμενο το όνομα του πεδίου ("field"), ο δε έλεγχος γίνεται μέσα στα άγκιστρα του with. Ο έλεγχος στην περίπτωση της validate_number() είναι εάν η τιμή

της ιδιότητας value του πεδίου ελέγχου είναι διαφορετική από την απόλυτη τιμή του αριθμού κινητής υποδιαστολής που προκύπτει διαβάζοντας την τιμή της ιδιότητας value. Εάν ο χρήστης έχει καταχωρήσει κείμενο, συνδυασμό κειμένου και αριθμών, αρνητική αριθμητική τιμή, μη αριθμητικά σύμβολα ή οτιδήποτε που δεν είναι θετική αριθμητική τιμή, τότε η ανάγνωση του αριθμητικού τμήματος της καταχώρησης και μετέπειτα η απόλυτή του τιμή, θα είναι μια τιμή οπωσδήποτε διαφορετική από την τιμή που έχει καταχωρηθεί στο σχετικό πεδίο. Στην περίπτωση λοιπόν αυτή θα πρέπει να εμφανιστεί μήνυμα λάθους.

Στη συνέχεια γίνεται ο καθορισμός της συνάρτησης ελέγχου της λέξης-κλειδί "SELECT". Η συνάρτηση ονομάζεται validate_select() και έχει τις ίδιες παραμέτρους με τις προηγούμενες. Μέσα στα άγκιστρα που υπάρχουν μετά τη λέξη-κλειδί "with" της JavaScript, γίνεται αναζήτηση της συμβολοσειράς "SELECT " στη συμβολοσειρά που έχει καταχωρηθεί στο πεδίο ελέγχου. Προφανώς αυτός ο έλεγχος γίνεται στα τρία πεδία καταχώρησης προσαρμοσμένου ερωτήματος προς εκτέλεση. Για τη σωστή αναζήτηση της λέξης-κλειδί "SELECT" όπως και κάθε άλλης λέξης-κλειδί, η συμβολοσειρά πρέπει να διαθέτει ένα σύμβολο κενού τουλάχιστον μετά τη λέξη-κλειδί. Σε πολλές περιπτώσεις, το σύμβολο κενού θα υπάρχει και πριν αλλά όχι πάντα και όχι κατ' ανάγκη. Χωρίς το κενό στη συμβολοσειρά αναζήτησης το μήνυμα λάθους θα έβγαινε και στην περίπτωση που ανιχνευόταν π.χ. η λέξη selection ή selected ή SELECT1AB λέξεις οι οποίες είναι διαφορετικές από τη λέξη κλειδί "SELECT". Έτσι, σε κάθε συμβολοσειρά αναζήτησης λέξεων-κλειδιών υπάρχει ένα κενό μετά τα σύμβολα της λέξης-κλειδί. Κατά την αναζήτηση της συγκεκριμένης κάθε φορά συμβολοσειράς (ανάλογα με τη συνάρτηση ελέγχου), οι λέξεις κλειδιά μπορεί να είναι με κεφαλαία γράμματα, με μικρά γράμματα ή και με συνδυασμούς κεφαλαίων και μικρών, συνεπώς η αναζήτηση πρέπει να γίνεται χωρίς διάκριση πεζών-κεφαλαίων. Η παρακάτω εντολή αποθηκεύει στη μεταβλητή selectSearch1 τη θέση που βρέθηκε η συμβολοσειρά "SELECT " χωρίς διάκριση πεζών-κεφαλαίων στην τιμή της ιδιότητας value του πεδίου αναζήτησης:

```
selectSearch1=value.search(/SELECT /i);
```

Το i είναι μία παράμετρος που δηλώνει case-insensitive αναζήτηση (χωρίς ευαισθησία-διάκριση σε μικρά και κεφαλαία).

Τα μηνύματα λάθους καταχωρούνται στη θέση της παραμέτρου alerttxt κατά την εκτέλεση οποιασδήποτε συνάρτησης ελέγχου. Όλες οι συναρτήσεις ελέγχου εκτελούνται μέσα από χειριστές συμβάντων στον κώδικα HTML της ιστοσελίδας. Ο κώδικας JavaScript μέσα στους χειριστές συμβάντων βρίσκεται μέσα σε διπλά εισαγωγικά. Το μήνυμα λάθους βρίσκεται και αυτό μέσα σε εισαγωγικά τα οποία είναι κατ' ανάγκη μονά καθώς εάν ήταν διπλά η HTML θα τα αντιλαμβανόταν ως το τέλος του κώδικα JavaScript και θα προέκυπταν λάθη κατά το άνοιγμα της ιστοσελίδας. Με τα μονά εισαγωγικά πετυχαίνουμε το στόχο μας. Στην περίπτωση όμως που θέλουμε το μήνυμα λάθους να περιέχει είτε μονά είτε διπλά εισαγωγικά υπάρχει πρόβλημα. Εάν βάλουμε μονά εισαγωγικά η JavaScript θα τα αντιληφθεί ως το τέλος του μηνύματος και θα προκύψουν λάθη κατά την εκτέλεση της εντολής. Εάν βάλουμε στο μήνυμα λάθους διπλά εισαγωγικά, θα δημιουργηθεί πρόβλημα στην εκτέλεση του χειριστή συμβάντος. Στην περίπτωση λοιπόν που θέλουμε το μήνυμα λάθους να έχει σε κάποιο σημείο εισαγωγικά (π.χ. η λέξη-κλειδί "SELECT" θα πρέπει να μπει σε διπλά εισαγωγικά) πρέπει να εργαστούμε με διαφορετικό τρόπο. Στο μήνυμα λάθους που επιθυμούμε να εμφανιστεί βάζουμε κάποιο άλλο σύμβολο στη θέση των εισαγωγικών και στη συνέχεια κάνουμε αντικατάσταση αυτού του συμβόλου με τα διπλά εισαγωγικά. Η επόμενη εντολή λοιπόν, μετά τον καθορισμό της μεταβλητής selectSearch1, κάνει αντικατάσταση του συμβόλου (`) με το σύμβολο (") όσες φορές το βρει μέσα στη συμβολοσειρά του μηνύματος λάθους:

```
alerttxt=alerttxt.replace(/`/g,'"');
```

Το αποτέλεσμα της αντικατάστασης αποθηκεύεται πάλι στην παράμετρο alerttxt και αντικαθιστά το αρχικό μήνυμα. Η παράμετρος g στην παραπάνω εντολή δηλώνει

ότι η συμβολοσειρά (` ) όσες φορές βρεθεί στη συμβολοσειρά alerttxt, θα αντικατασταθεί όλες τις φορές από τη συμβολοσειρά ( " ). Αν δεν μπει αυτή η παράμετρος, θα γίνει αντικατάσταση μόνο της πρώτης εμφάνισης της συμβολοσειράς ( `) στη alerttxt. Στην περίπτωση αυτή η συμβολοσειρά που αναζητείται και η συμβολοσειρά με την οποία αντικαθίσταται, είναι μόνο ένα σύμβολο και θα μπορούσαμε να αναφέρουμε ότι αναζητείται και αντικαθίσταται το σύμβολο (`) από το σύμβολο ("). Χρησιμοποιούμε τον όρο συμβολοσειρά θέλοντας να δείξουμε ότι δεν είναι ανάγκη να αναζητείται ένα σύμβολο και να αντικαθίσταται με ένα άλλο σύμβολο αλλά οι δυνατότητες της replace είναι για σειρές συμβόλων ήτοι συμβολοσειρές.

Ακολουθεί η υπόθεση:

Εάν η θέση εύρεσης της συμβολοσειράς “SELECT ” είναι αριθμός μεγαλύτερος από το 0 ή εάν δεν βρεθεί η συμβολοσειρά αυτή, τότε εμφάνισε το μήνυμα λάθους που έχει η παράμετρος alerttxt. Όταν δεν βρεθεί η συμβολοσειρά αναζήτησης, η ιδιότητα search επιστρέφει την τιμή -1. Συνεπώς η εντολή με την υπόθεση που περιγράφεται εδώ έχει ως εξής:

```
if (selectSearch1>0||selectSearch1==-1) alert(alerttxt);
```

Εάν (πρώτος έλεγχος) ή (δεύτερος έλεγχος) τότε εμφάνισε μήνυμα με το κείμενο της alerttxt.

Μετά την validate_select() ακολουθεί η συνάρτηση validate_keywords() η οποία έχει τις ίδιες παραμέτρους με τις προηγούμενες συναρτήσεις. Η συνάρτηση αυτή ελέγχει για την ύπαρξη ή όχι μίας ομάδας λέξεων-κλειδιών της PostgreSQL v7.4, η οποία χρησιμοποιείται στην παρούσα εργασία, στο υπό εκτέλεση ερώτημα. Η PostgreSQL χρησιμοποιεί διάφορες λέξεις-κλειδιά για την εκτέλεση συγκεκριμένων εντολών. Για παράδειγμα η λέξη-κλειδί DROP όταν εκτελεστεί στην PostgreSQL μαζί φυσικά με συγκεκριμένες άλλες πληροφορίες έχει ως αποτέλεσμα τη διαγραφή ενός πίνακα, ενός ρεύματος, ενός σχήματος κ.ο.κ. . Με την εντολή CREATE δημιουργούμε καινούργιες οντότητες, με την εντολή ALTER τροποποιούμε τα στοιχεία διαφόρων οντοτήτων και ούτω καθεξής. Καθώς ο χρήστης έχει τη δυνατότητα εκτέλεσης οποιουδήποτε ερωτήματος και οποιασδήποτε εντολής μέσω της ιστοσελίδας στη β.δ. και το ρ.δ., κρίθηκε σκόπιμο να τοποθετηθεί έλεγχος και περιορισμός στις δυνατότητες εκτέλεσης εντολών από τον χρήστη. Έτσι, όλες οι λέξεις-κλειδιά που δεν χρειάζονται για την εκτέλεση ερωτημάτων της εφαρμογής, δεν επιτρέπεται να χρησιμοποιηθούν στο προς εκτέλεση ερώτημα. Η validate_keywords() λοιπόν, ελέγχει για την ύπαρξη ή όχι κάθε τέτοιας λέξης-κλειδί. Οι λέξεις-κλειδιά που βρέθηκαν από την ιστοσελίδα των εντολών της PostgreSQL v7.4 (http://www.postgresql.org/docs/7.4/static/sql-commands.html) και κρίθηκε σκόπιμο να μην επιτρέπεται η χρήση τους, είναι στο σύνολο 38 και είναι οι εξής: ABORT, ALTER, ANALYZE, BEGIN, CHECKPOINT, CLOSE, CLUSTER, COMMENT, COMMIT, COPY, CREATE, DEALLOCATE, DECLARE, DELETE, DROP, END, EXECUTE, EXPLAIN, FETCH, GRANT, INSERT, LISTEN, LOAD, LOCK, MOVE, NOTIFY, PREPARE, REINDEX, RESET, REVOKE, ROLLBACK, SET, SHOW, START TRANSACTION, TRUNCATE, UNLISTEN, UPDATE και VACUUM. Το αποτέλεσμα της αναζήτησης της κάθε μίας αποθηκεύεται σε ξεχωριστή μεταβλητή. Οι 38 αυτές μεταβλητές έχουν το όνομα της λέξης-κλειδί με το πρώτο γράμμα κεφαλαίο και τα υπόλοιπα πεζά, ενώ προηγείται το γράμμα “v” δηλώνοντας ότι πρόκειται για μεταβλητή. Π.χ. αρχικά αναζητείται η συμβολοσειρά “ABORT ” και η θέση εύρεσής της αποθηκεύεται στη μεταβλητή “vAbort”. Αφού λοιπόν καταχωρηθούν σε ξεχωριστές μεταβλητές οι θέσεις εύρεσης των 38 λέξεων-κλειδιών ακολουθεί πρόταση υπό όρους:

```
if (vAbort!=-1||vAlter!=-1||vAnalyze!=-1||vBegin!=-1||
vCheckpoint!=-1||vClose!=-1||vCluster!=-1||vComment!=-1||
vCommit!=-1||vCopy!=-1||vCreate!=-1||vDeallocate!=-1||
vDeclare!=-1||vDelete!=-1||vDrop!=-1||vEnd!=-1||vExecute!=-1||
vExplain!=-1||vFetch!=-1||vGrant!=-1||vInsert!=-1||
```

```
vListen!=-1||vLoad!=-1||vLock!=-1||vMove!=-1||vNotify!=-1||
vPrepare!=-1||vReindex!=-1||vReset!=-1||vRevoke!=-1||
vRollback!=-1||vSet!=-1||vShow!=-1||vStartTr!=-1||
vTruncate!=-1||vUnlisten!=-1||vUpdate!=-1||vVacuum!=-1)
    alert(alerttxt);
```

Εάν οποιαδήποτε από τις παραπάνω μεταβλητές έχει τιμή διαφορετική από το -1, τότε εμφάνισε μήνυμα λάθους με το περιεχόμενο της alerttxt. Αυτό σημαίνει απλά ότι εάν υπάρχει έστω και μία από αυτές τις λέξεις-κλειδιά, τότε θα εμφανιστεί μήνυμα λάθους.

Με αυτό τον έλεγχο τελειώνει η validate_keywords(). Η συνάρτηση αυτή εκτελείται πάνω στα πεδία εισαγωγής των Προσαρμοσμένων Ερωτημάτων και στο πεδίο εισαγωγής των συντεταγμένων του Ερωτήματος Περιοχής.

Μετά από την validate_keywords() καθορίζεται η συνάρτηση validate_from() με τις ίδιες πάντα παραμέτρους. Η συνάρτηση αυτή ελέγχει εάν υπάρχει η λέξη-κλειδί "FROM" και μάλιστα εάν είναι μέσα στο ερώτημα μετά τη λέξη-κλειδί "SELECT". Αντίστοιχα λοιπόν με τη συνάρτηση validate_select() αναζητείται η θέση εμφάνισης της συμβολοσειράς "FROM " και η θέση εμφάνισης της "SELECT ". Δεν μπορεί να χρησιμοποιηθεί η τιμή της selectSearch1 που υπολογίζεται στην validate_select() διότι η μεταβλητή είναι τοπική και υπάρχει μόνο εντός της τελευταίας αυτής συνάρτησης. Έτσι, η τοπική μεταβλητή selectSearch2 αποθηκεύει τη θέση που βρέθηκε η συμβολοσειρά "SELECT " και η τοπική μεταβλητή fromSearch1 αποθηκεύει τη θέση που βρέθηκε η συμβολοσειρά "FROM ". Ακολουθεί η αντικατάσταση του συμβόλου (`) με το σύμβολο (") στο μήνυμα λάθους για τον ίδιο λόγο που αναφέρθηκε παραπάνω. Μετά εκτελείται ο εξής έλεγχος:

```
    if (fromSearch1==-1||selectSearch2-fromSearch1>0)
alert(alerttxt);
```

Εάν δεν βρεθεί η λέξη-κλειδί "FROM" ή βρεθεί μετά την λέξη-κλειδί "SELECT" τότε εμφάνισε μήνυμα λάθους. Μετά από αυτόν τον έλεγχο τελειώνει η εκτέλεση της validate_from(). Η συνάρτηση αυτή εκτελείται όταν ο χρήστης πατήσει την εκτέλεση οποιουδήποτε Προσαρμοσμένου Ερωτήματος και ελέγχει το ερώτημα που έχει εισαχθεί στο σχετικό πεδίο <TEXTAREA> της ιστοσελίδας.

Μετά καθορίζεται η συνάρτηση validate_vehicles() η οποία έχει τις ίδιες παραμέτρους με όλες τις συναρτήσεις ελέγχου πεδίων. Χρησιμοποιείται δε για να ελέγξει ότι το κείμενο "network.vehicles" ή το κείμενο "network.vehicles2" υπάρχει στο υπό εκτέλεση ερώτημα και μάλιστα ότι βρίσκεται μετά τη λέξη-κλειδί "FROM". Αντίστοιχα λοιπόν με την validate_from() αποθηκεύονται σε συγκεκριμένες μεταβλητές οι θέσεις εντοπισμού των αντίστοιχων συμβολοσειρών (εάν δεν εντοπιστούν οι σχετικές συμβολοσειρές τότε οι μεταβλητές παίρνουν την τιμή -1 όπως έχει περιγραφεί σε προηγούμενη συνάρτηση). Ο έλεγχος στην περίπτωση αυτή είναι λίγο πιο σύνθετος και έχει ως εξής:

```
    if
(((vehicles1Search==-1)||(fromSearch2>vehicles1Search))&&
((vehicles2Search==-1)||(fromSearch2>vehicles2Search)))
alert(alerttxt);
```

Εάν δεν βρεθεί η συμβολοσειρά "network.vehicles " ή βρεθεί αλλά είναι πριν τη συμβολοσειρά "FROM " και εάν δεν βρεθεί η συμβολοσειρά "network.vehicles2 " ή βρεθεί αλλά είναι πριν τη συμβολοσειρά "FROM " τότε εμφάνισε το σχετικό μήνυμα λάθους. Το σχετικό μήνυμα λοιπόν θα εμφανιστεί όταν δεν υπάρχει ούτε το πρώτο ούτε το δεύτερο ρεύμα δεδομένων. Εάν υπάρχει έστω και ένα από τα δύο δεν θα εμφανιστεί το μήνυμα. Η εφαρμογή έχει σχεδιαστεί έτσι ώστε όταν εκτελούνται και τα δύο ρεύματα δεδομένων, να μπορεί ο χρήστης να εκτελέσει ερωτήματα μόνο με το πρώτο ρεύμα, διαφορετικά-ξεχωριστά ερωτήματα εάν θέλει μόνο με το δεύτερο ρεύμα και ερωτήματα

και με τα δύο ρεύματα δεδομένων. Η συνάρτηση αυτή εκτελείται στις ίδιες περιπτώσεις με την προηγούμενη, όταν δηλαδή ο χρήστης πατήσει την εκτέλεση οποιουδήποτε Προσαρμοσμένου Ερωτήματος και ελέγχει το ερώτημα που έχει εισαχθεί στο σχετικό πεδίο της ιστοσελίδας.

Στη συνέχεια καθορίζεται η τελευταία συνάρτηση ελέγχου που λέγεται validate_quotes() και έχει τις ίδιες παραμέτρους με τις παραπάνω αντίστοιχες συναρτήσεις, που είναι το πεδίο εκτέλεσης του ελέγχου και το μήνυμα που θα εμφανιστεί στην περίπτωση που πληρούνται οι σχετικές προϋποθέσεις. Η συνάρτηση αυτή βγάζει μήνυμα λάθους εάν βρεθούν απλά εισαγωγικά στο υπό εκτέλεση ερώτημα. Όπως έχει προαναφερθεί τα απλά εισαγωγικά αν και χρειάζονται για τον καθορισμό του χρονικού παραθύρου και πιθανώς και σε άλλα μέρη του ερωτήματος, προκαλούν πρόβλημα στη μεταφορά της συμβολοσειράς που αποτελεί το υπό εκτέλεση ερώτημα από την HTML στην PHP. Για να αντιμετωπιστεί αυτό το θέμα ο χρήστης, αντί των απλών εισαγωγικών (‘) θα πρέπει να χρησιμοποιήσει κατακόρυφες μπάρες (¦) οι οποίες κατά την εκτέλεση του σχετικού αρχείου PHP αντικαθίστανται από απλά εισαγωγικά και το ερώτημα εκτελείται κανονικά. Ο κώδικας της τελευταίας αυτής συνάρτησης ελέγχου έχει ως εξής: Αντίστοιχα με τις προηγούμενες λέξεις-κλειδιά, η μεταβλητή searchQuotes αποθηκεύει τη θέση εμφάνισης των απλών εισαγωγικών μέσα στο υπό εκτέλεση ερώτημα και φυσικά εάν αυτά δεν υπάρχουν πουθενά, παίρνει την τιμή -1. Μετά γίνεται η αντικατάσταση του συμβόλου (`) με το (‘) και όχι με διπλά εισαγωγικά στο υπό εμφάνιση μήνυμα. Τέλος εκτελείται η πρόταση ελέγχου:

if (quotesSearch!=-1) alert(alerttxt);

που πολύ απλά λέει ότι εάν βρεθεί έστω και ένα απλό εισαγωγικό τότε εμφάνισε μήνυμα λάθους.

Στο σημείο αυτό τελειώνει ο καθορισμός της τελευταίας συνάρτησης ελέγχου και μαζί του τελειώνει ο κώδικας JavaScript.

5.7.5 Διαγραμματική παρουσίαση λειτουργιών κώδικα JavaScript

Παρακάτω ακολουθεί διαγραμματική παρουσίαση των βασικών λειτουργιών του κώδικα JavaScript:

Διπλωματική Εργασία: Διαχείριση ρευμάτων κίνησης αντικειμένων με απεικόνιση σε GoogleMaps

Διαγραμματική Παρουσίαση Συναρτήσεων JavaScript που Εκτελούνται Όταν Πατάμε Κάποιο Κουμπί

* # = ο αριθμός του ερωτήματος

Διάγραμμα 8

Κωνσταντίνος Δ. Θεοφιλογιαννάκος

Διπλωματική Εργασία: Διαχείριση ρευμάτων κίνησης αντικειμένων με απεικόνιση σε GoogleMaps

Διαγραμματική Παρουσίαση Συναρτήσεων JavaScript που Εκτελούνται Υπό Διαφορες Άλλες Συνθήκες (Συνέχεια Διαγράμματος 8)

* # = ο αριθμός του ερωτήματος

Διάγραμμα 9

Κωνσταντίνος Δ. Θεοφιλογιαννάκος

5.8 Αναλυτική περιγραφή λειτουργίας κώδικα HTML (αρχείο StreamMap_gr.html)

Πρόκειται για το αρχείο εμφάνισης της κεντρικής ιστοσελίδας της εφαρμογής. Ξεκινά με τον καθορισμό του τύπου του αρχείου HTML:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Ακολουθούν σχετικά σχόλια και στη συνέχεια η αρχή του κώδικα HTML με το σχετικό στοιχείο <HTML> μέσα στον καθορισμό του οποίου υπάρχουν σχετικές πληροφορίες για τον κώδικα που ακολουθεί:

```
<HTML xmlns="http://www.w3.org/1999/xhtml"
xmlns:v="urn:schemas-microsoft-com:vml">
```

Μέσα στο στοιχείο <HEAD> που ακολουθεί, ορίζεται ο τίτλος της ιστοσελίδας, ορισμένα μεταδεδομένα όπως η κωδικοποίηση (encoding) της ιστοσελίδας, και δύο scripts της JavaScript. Το πρώτο από αυτά περιέχει το κλειδί για τη χρήση των χαρτών της Google και το δεύτερο την αναφορά στο αρχείο "StrMap_JavaScript_code_gr.js" με τον κώδικα JavaScript που περιγράφηκε παραπάνω.

Ακολουθεί το στοιχείο <BODY> μέσα στο οποίο περιέχονται οι πληροφορίες για την εμφάνιση σε συγκεκριμένες θέσεις και με συγκεκριμένο τρόπο των επιθυμητών στοιχείων – πληροφοριών της ιστοσελίδας.

Μέσα στο <BODY> υπάρχει πρώτα το στοιχείο <NOSCRIPT> το οποίο εμφανίζει ένα ενημερωτικό κείμενο στο χρήστη στην περίπτωση που ο browser (φυλλομετρητής) του δεν υποστηρίζει την JavaScript ή αυτή είναι απενεργοποιημένη.

Μετά ορίζεται το στοιχείο <IFRAME> το οποίο καθορίζει πλαίσιο μηδενικών διαστάσεων στο οποίο θα ανοίγουν και θα "εμφανίζονται" τα αποτελέσματα εκτέλεσης του κάθε αρχείου PHP. Στην ουσία με τη χρήση του <IFRAME> και την αναφορά του κάθε φορά που εκτελείται ένα αρχείο PHP φροντίζουμε να εκτελείται το PHP και τα αποτελέσματα εκτέλεσής του να μην εμφανίζονται πουθενά και συνεπώς πετυχαίνουμε το στόχο μας που είναι ακριβώς αυτός (να εκτελούνται τα PHP αλλά τα αποτελέσματα των εκτελέσεών τους να μην εμφανίζονται πουθενά). Η δημιουργία αρχείων XML και η εκτέλεση διαφόρων εντολών είναι τα αποτελέσματα της εκτέλεσης των αρχείων PHP και συνεπώς δε χρειάζεται να εμφανίζονται κάπου τα αποτελέσματα εκτέλεσής τους. Το στοιχείο <IFRAME> λοιπόν με id="RSIFrame", name="RSIFrame" και μηδενικές διαστάσεις ορίζεται στο σημείο αυτό και χρησιμοποιείται ως η θέση εμφάνισης των αποτελεσμάτων εκτέλεσης των διαφόρων αρχείων PHP.

Ακολουθεί η ίδρυση HTML Πίνακα με το στοιχείο <TABLE>. Οι HTML Πίνακες βοηθούν στην καλύτερη οργάνωση και εμφάνιση των πληροφοριών στην ιστοσελίδα. Όπως έχει προαναφερθεί, αυτή χωρίζεται σε τρία τμήματα. Όλες οι πληροφορίες βρίσκονται σε Κελιά Πινάκων, οι περισσότεροι από τους οποίους δεν έχουν ορατά όρια.

Κατά τον ορισμό του στοιχείου Πίνακας μπορούμε να ορίσουμε τις διαστάσεις αυτού. Οι διαστάσεις που ορίζουμε είναι οι ελάχιστες δυνατές για τον πίνακα. Εάν οι πληροφορίες που αυτός περιέχει χρειάζονται περισσότερο χώρο, ο πίνακας θα μεγαλώσει ανάλογα. Αφού ορίσουμε το στοιχείο <TABLE> ορίζουμε το σώμα του Πίνακα με το στοιχείο <TBODY>. Μέσα σε αυτό ορίζουμε διάφορες Γραμμές Πίνακα με το στοιχείο <TR> (Table Row). Τέλος μέσα σε κάθε Γραμμή Πίνακα ορίζουμε ένα ή περισσότερα Κελιά Πίνακα <TD>, μέσα στα οποία καταχωρούμε τις επιθυμητές πληροφορίες.

Έτσι, ο τίτλος της εφαρμογής βρίσκεται μέσα σε πίνακα με πλάτος το 100% του πλάτους του παραθύρου του browser, χωρίς φανερά όρια και χωρίς συγκεκριμένο καθορισμένο ελάχιστο ύψος, με μία γραμμή και ένα κελί.

Ο δεύτερος πίνακας περιέχει μία γραμμή και τρία κελιά στα δύο από τα οποία υπάρχουν πληροφορίες και στο ένα όχι. Το πρώτο κελί περιέχει το κείμενο "Παρακαλώ

διαβάστε τις" και τον δεσμό (link) "Οδηγίες χρήσης." ενώ το τρίτο τον δεσμό προς την Αγγλική έκδοση της Εφαρμογής.

Ο τρίτος πίνακας περιέχει και αυτός μία γραμμή και τρία κελιά. Στο πρώτο κελί καθορίζονται τα κουμπιά με τίτλους: "Εκκίνηση ενός Ρεύματος Δεδομένων" και "Εκκίνηση δύο όμοιων Ρευμάτων Δεδομένων" σε ξεχωριστές φόρμες το καθένα. Όταν ο χρήστης πατήσει το πρώτο κουμπί, εκτελείται το "startStream.php" με προορισμό το RSIFrame. Επίσης εκτελείται και ένας χειριστής συμβάντος που εμφανίζει σε μήνυμα πληροφορίες σχετικά με το ρεύμα δεδομένων. Όταν ο χρήστης πατήσει το δεύτερο κουμπί, εκτελούνται συγκεκριμένοι χειριστές συμβάντων. Ο "onMouseDown" και ο "onKeyDown" εκτελούν τη συνάρτηση OpenTwoLinks() που περιγράφηκε παραπάνω. Ο "onsubmit" εμφανίζει μήνυμα με σχετικές πληροφορίες για την εκτέλεση των δύο ρευμάτων δεδομένων. Το δεύτερο κελί περιέχει το κείμενο "Κατάσταση 1ου Ρεύματος Δεδομένων: " και στη συνέχεια ορίζεται στοιχείο κειμένου με έντονα γράμματα και id="firStreamStat" στο οποίο εμφανίζεται η κατάσταση του πρώτου ρεύματος δεδομένων από την εκτέλεση του αντίστοιχου κώδικα JavaScript. Μετά από αλλαγή γραμμής χρησιμοποιώντας το στοιχείο
 υπάρχει το κείμενο "Κατάσταση 2ου Ρεύματος Δεδομένων: " και έπεται αντίστοιχο στοιχείο κειμένου με έντονα γράμματα και id="secStreamStat" στο οποίο εμφανίζεται η κατάσταση του δεύτερου ρεύματος δεδομένων. Το τρίτο κελί δεν περιέχει τίποτα. Σε προηγούμενα στάδια της εφαρμογής το κελί αυτό περιείχε κουμπί διακοπής εκτέλεσης των ρευμάτων δεδομένων αλλά αφαιρέθηκε καθώς είναι μία ενέργεια που δεν μπορεί να εκτελέσει ο απομακρυσμένος χρήστης και δεν χρειάζεται κιόλας καθώς τα ρεύματα δεδομένων σταματούν ούτως ή άλλως μετά από 6 λεπτά από την εκκίνησή τους.

Ο επόμενος πίνακας περιέχει δύο γραμμές με ένα κελί σε κάθε μία. Στο πρώτο κελί υπάρχει το κείμενο "Χρόνος: " και ακολουθείται από το στοιχείο <FONT> με id="txtTime". Το στοιχείο αυτό εμφανίζει τη χρονική στιγμή των αντικειμένων που υπάρχουν στο χάρτη. Εάν δεν εκτελείται κανένα ερώτημα δε θα υπάρχουν αντικείμενα στο χάρτη και δε θα εμφανίζεται καμία χρονική στιγμή στην παραπάνω θέση. Το δεύτερο κελί διαθέτει στοιχείο <DIV> με id="map_canvas" και στη θέση του εμφανίζεται ο χάρτης τον οποίο έχει αποθηκευμένο η μεταβλητή map της JavaScript. Αυτή λοιπόν είναι η θέση όπου εμφανίζεται ο χάρτης.

Ο πέμπτος πίνακας περιέχει μία γραμμή και τρία κελιά. Στο πρώτο κελί υπάρχει νέος εσωτερικός πίνακας με μία γραμμή και δύο κελιά. Μέσα στο πρώτο κελί, που καταλαμβάνει, το 70% του πλάτους του πίνακα (ο οποίος με τη σειρά του καταλαμβάνει το 100% του πλάτους του κελιού μέσα στο οποίο βρίσκεται) υπάρχει με μέγεθος γραμμάτων 2 το κείμενο "Συντεταγμένες (λ,φ): " ("Coordinates (long,lat):") και στη συνέχεια, μετά από αλλαγή σειράς, στοιχείο <DIV> με id="coordinatesAre" το οποίο έχει αρχικά την τιμή (23.729200, 37.976500) που είναι οι συντεταγμένες του κέντρου του χάρτη που εμφανίζεται στην ιστοσελίδα. Στο σημείο αυτό εμφανίζονται οι συντεταγμένες της θέσης του κέρσορα όταν ο χρήστης κάνει κλικ επάνω στο χάρτη. Στο επόμενο κελί του εσωτερικού πίνακα δεν υπάρχει κάποια πληροφορία και χρησιμοποιείται για να υπάρχει σε κάθε διάσταση του παραθύρου της ιστοσελίδας αναλογική απόσταση ανάμεσα στις πληροφορίες του πρώτου κελιού του εξωτερικού πίνακα και στις πληροφορίες του δεύτερου κελιού του εξωτερικού πίνακα. Στο διπλανό κελί υπάρχει το κείμενο "Κατάσταση Χάρτη: " ("Map Status: ") και στη συνέχεια στοιχείο <FONT> με id="statusLine" και συγκεκριμένο στυλ. Στη θέση αυτή εμφανίζεται δυναμικά, όταν εκτελείται τουλάχιστον ένα ερώτημα, η κατάσταση του χάρτη. Αν δεν εκτελείται κανένα ερώτημα, δεν εμφανίζεται τίποτα στη θέση αυτή. Το τρίτο κελί περιέχει και αυτό εσωτερικό πίνακα για την εμφάνιση με τον επιθυμητό τρόπο συγκεκριμένης πληροφορίας. Ο εσωτερικός αυτός πίνακας περιέχει μία γραμμή και δύο κελιά. Το πρώτο κελί δεν περιέχει δεδομένα και καταλαμβάνει το 30% του πλάτους του πίνακα ο οποίος καταλαμβάνει το 100% του πλάτους του κελιού μέσα στο οποίο έχει οριστεί. Το δεύτερο κελί καταλαμβάνει το υπόλοιπο 70% του πλάτους του πίνακα και περιέχει το κείμενο:

"Τα περιεχόμενα του χάρτη ανανεώνονται κάθε 5 δευτ."
("Contents of map are refreshed every 5 seconds.")

Ο εσωτερικός αυτός πίνακας έχει δημιουργηθεί ώστε να υπάρχει πάντα (σε κάθε διάσταση του παραθύρου της ιστοσελίδας) αναλογική απόσταση μεταξύ του κειμένου που εμφανίζεται στο μεσαίο κελί με την "Κατάσταση χάρτη: " και του κειμένου που εμφανίζεται στο δεξί κελί ("Τα περιεχόμενα του χάρτη ανανεώνονται κάθε 5 δευτ.").

Ο έκτος πίνακας περιέχει και αυτός μία γραμμή και τρία κελιά και τα όριά του δεν εμφανίζονται στην ιστοσελίδα. Στο πρώτο κελί υπάρχει εσωτερικός δεύτερος πίνακας με φανερά όρια πάχους μίας μονάδας και συγκεκριμένο χρώμα φόντου ανοιχτό κόκκινο. Ο πίνακας αυτός έχει μία γραμμή και ένα κελί και έχει δημιουργηθεί μόνο και μόνο για την εμφάνιση του περιγράμματος. HHTML δεν υποστηρίζει την ύπαρξη περιγράμματος σε συγκεκριμένα κελιά πίνακα. Το περίγραμμα είτε θα εμφανίζεται σε ολόκληρο τον πίνακα είτε δε θα εμφανίζεται καθόλου. Η HTML όμως, όπως γίνεται φανερό από τα παραπάνω, υποστηρίζει τη δημιουργία πινάκων μέσα σε κελιά άλλων πινάκων. Το αποτέλεσμα που φαίνεται στην ιστοσελίδα βέβαια δεν είναι ακριβώς το ίδιο αλλά θεωρείται αρκετά κοντινό με το επιθυμητό και υιοθετείται αυτή η μέθοδος εμφάνισης ορίων σε συγκεκριμένα κελιά ενός πίνακα. Ο δεύτερος αυτός πίνακας αποτελεί και το πλαίσιο του Βασικού Ερωτήματος. Το τι περιέχει έχει περιγραφεί σε προηγούμενη παράγραφο. Το πεδίο καταχώρησης της Μέγιστης Διάρκειας Εκτέλεσης έχει name="setlimit0" και γίνεται έλεγχος την τιμής που διαθέτει από τις συναρτήσεις validate_required() και validate_number() μέσα από τον χειριστή συμβάντος onsubmit ως εξής:

```
onsubmit="validate_required(setlimit0,'Εσφαλμένη εισαγωγή δεδομένων! Η Μέγιστη Διάρκεια Εκτέλεσης του Βασικού Ερωτήματος πρέπει να συμπληρωθεί.');
validate_number(document.f0.setlimit0,'Εσφαλμένη εισαγωγή δεδομένων! Η Μέγιστη Διάρκεια Εκτέλεσης του Βασικού Ερωτήματος πρέπει να είναι θετικός αριθμός.'); queryNoResults(0);"
(onsubmit="validate_required(setlimit0,'Typing Error! Maximum Execution Time of Basic Query must be filled out!');
validate_number(document.f0.setlimit0,'Typing Error! Maximum Execution Time of Basic Query must be a positive number.');
queryNoResults(0);")
```

Ο χειριστής συμβάντος, ο οποίος εκτελείται όταν ο χρήστης πατήσει το κουμπί εκτέλεσης του Βασικού Ερωτήματος, απ' ότι φαίνεται παραπάνω εκτός από τις δύο συναρτήσεις ελέγχου εκτελεί και τη συνάρτηση queryNoResults(0) με την παράμετρο qnr να έχει την τιμή 0 που δηλώνει το Βασικό Ερώτημα. Οι ενέργειες που εκτελεί αυτή η συνάρτηση έχουν περιγραφεί παραπάνω. Το κουμπί διακοπής του Βασικού Ερωτήματος, όταν πατηθεί έχει ως αποτέλεσμα την εκτέλεση του αρχείου stopQuery.php με την παράμετρο squery να έχει την τιμή 0 που δηλώνει το Βασικό Ερώτημα. Μετά από τη διαχωριστική γραμμή υπάρχει το κείμενο "Κατάσταση Βασικού Ερωτήματος: " ("Basic Query Status: ") και δίπλα στοιχείο έντονου κειμένου <B> με id="stat0" στο οποίο η JavaScript εμφανίζει τη λέξη "Ενεργό..." ή "Σταματημένο." ανάλογα με την κατάσταση του ερωτήματος.

Στο μεσαίο κελί του εξωτερικού πίνακα υπάρχει στοιχείο <DIV> με το κείμενο "Παραδείγματα Ερωτημάτων μπορούν να βρεθούν " δεσμός (link) με το κείμενο "εδώ" και προορισμό της υπερσύνδεσης τη συνάρτηση JavaScript: open_examples() η οποία έχει περιγραφεί σε προηγούμενη παράγραφο.

Μετά από μία οριζόντια γραμμή υπάρχει το κείμενο-προτροπή προς τον χρήστη: "Πριν κλείσετε την εφαρμογή, παρακαλείσθε να σταματήσετε όλα τα ερωτήματα." και ακολουθείται από μία φόρμα της HTML που περιέχει το κουμπί με τίτλο: "Διακοπή Όλων των Ερωτημάτων". Όταν ο χρήστης πατήσει αυτό το κουμπί, η φόρμα εκτελεί το αρχείο "stopAllQueries.php" οι λειτουργίες του οποίου θα περιγραφούν αναλυτικά σε επόμενο κεφάλαιο.

Μετά από τη δεύτερη οριζόντια γραμμή του μεσαίου αυτού κελιού του έκτου πίνακα, υπάρχει το κείμενο "Εκκαθάριση τροχιών από χάρτη." ("Cleanup Trajectories from map") και ακολουθεί κουμπί με τίτλο "Εκκαθάριση Τροχιών" ("Cleanup Trajectories"). Όταν ο χρήστης πατήσει αυτό το κουμπί εκτελείται μέσω χειριστών συμβάντος η JavaScript συνάρτηση cleanTraj() ως εξής:

```
onMouseUp="cleanTraj();" onKeyUp="cleanTraj();"
```

Στο τρίτο κελί του εξωτερικού πίνακα υπάρχει, αντίστοιχα με το Βασικό Ερώτημα, το Ερώτημα Περιοχής τα βασικά στοιχεία του οποίου έχουν περιγραφεί σε προηγούμενη παράγραφο. Το κουμπί "On/Off" ενεργοποιεί και απενεργοποιεί τη δημιουργία πολυγώνου επάνω στο χάρτη εκτελώντας τη συνάρτηση f_onoff(). Η κατάσταση δημιουργίας ή όχι πολυγώνου φαίνεται δίπλα στο κουμπί "On/Off" με στοιχείο έντονου κειμένου <B> με id="vv_onoff" το οποίο ενημερώνει σχετικά η f_onoff() όπως έχει περιγραφεί σε προηγούμενο κεφάλαιο. Δίπλα βρίσκεται το πεδίο καταχώρησης των συντεταγμένων του πολυγώνου με id="coordsqq" το οποίο ενημερώνεται από τη συνάρτηση του ακροατή (listener) που ενεργοποιείται όταν κάνουμε κλικ επάνω στο χάρτη με ενεργή τη δημιουργία πολυγώνου. Στην επόμενη σειρά υπάρχει το πεδίο καταχώρησης της Μέγιστης Διάρκειας εκτέλεσης του Ερωτήματος Περιοχής με name="setlimit6", η τιμή του οποίου ελέγχεται από τις σχετικές συναρτήσεις. Όταν ο χρήστης πατήσει την εκτέλεση του Ερωτήματος Περιοχής, εκτός από την εκτέλεση του execQuery_getResults.php με τις παραμέτρους refreshT=5 και query=6, μέσω χειριστή συμβάντος εκτελείται και μία σειρά συναρτήσεων της JavaScript ως εξής:

```
onsubmit="validate_required(qcoords,'Εσφαλμένη εισαγωγή δεδομένων! Οι συντεταγμένες πρέπει να συμπληρωθούν.');
validate_keywords(qcoords,'Εσφαλμένη εισαγωγή δεδομένων! Οι συντεταγμένες δεν μπορεί να περιλαμβάνουν καμία από τις παρακάτω λέξεις κλειδιά: ABORT, ALTER, ANALYZE, BEGIN, CHECKPOINT, CLOSE, CLUSTER, COMMENT, COMMIT, COPY, CREATE, DEALLOCATE, DECLARE, DELETE, DROP, END, EXECUTE, EXPLAIN, FETCH, GRANT, INSERT, LISTEN, LOAD, LOCK, MOVE, NOTIFY, PREPARE, REINDEX, RESET, REVOKE, ROLLBACK, SET, SHOW, START TRANSACTION, TRUNCATE, UNLISTEN, UPDATE, VACUUM (Χωρίς διάκριση πεζών-ΚΕΦΑΛΑΙΩΝ).');
validate_required(setlimit6,'Εσφαλμένη εισαγωγή δεδομένων! Η Μέγιστη Διάρκεια Εκτέλεσης του Ερωτήματος Περιοχής πρέπει να συμπληρωθεί.');
validate_number(document.f6.setlimit6,'Εσφαλμένη εισαγωγή δεδομένων! Η Μέγιστη Διάρκεια Εκτέλεσης του Ερωτήματος Περιοχής πρέπει να είναι θετικός αριθμός.');
queryNoResults(6);"
```

Εκτελούνται λοιπόν τέσσερις συναρτήσεις ελέγχου με τις σχετικές παραμέτρους και η συνάρτηση διακοπής του ερωτήματος εάν αυτό δεν επιστρέφει αποτελέσματα. Οι ενέργειες που κάνουν αυτές οι συναρτήσεις έχουν περιγραφεί στην παράγραφο 5.7.4 (Δεύτερη ομάδα συναρτήσεων κώδικα JavaScript – Έλεγχος πεδίων) και στην παράγραφο 5.7.2 (Αρχικός καθορισμός μεταβλητών και πρώτη ομάδα συναρτήσεων).

Στην επόμενη φόρμα καθορίζεται το κουμπί Διακοπής εκτέλεσης του Ερωτήματος Περιοχής, αντίστοιχα με το κουμπί Διακοπής εκτέλεσης Βασικού Ερωτήματος που περιγράφηκε νωρίτερα.

Το πλαίσιο του Ερωτήματος Περιοχής Διαθέτει άλλη μία φόρμα. Μέσα σε αυτήν υπάρχει το κουμπί "Αφαίρεση Συντεταγμένων και Πολυγώνου". Όταν λοιπόν ο χρήστης ζητήσει την αφαίρεση των συντεταγμένων των ορίων της περιοχής και του πολυγώνου που ορίζει την περιοχή, εκτελείται συγκεκριμένος χειριστής συμβάντος ως εξής:

```
onsubmit="remPolygon();"
```

Εκτελείται λοιπόν η συνάρτηση remPolygon() η οποία αφαιρεί το πολύγωνο από το χάρτη και τις συντεταγμένες του από το σχετικό πεδίο.

Μετά από μία οριζόντια γραμμή μέσα στο κελί, υπάρχει το κείμενο "Κατάσταση Ερωτήματος Περιοχής: " και στη συνέχεια υπάρχει στοιχείο έντονου κειμένου <B> με id="stat6". Στη θέση του στοιχείου αυτού εμφανίζεται η κατάσταση του Ερωτήματος Περιοχής από την JavaScript όπως έχει περιγραφεί παραπάνω.

Ο έβδομος και τελευταίος πίνακας περιέχει δύο γραμμές με τρία κελιά στην καθεμία. Τα δε όριά του είναι ορατά και έχουν πλάτος μία μονάδα. Το πρώτο κελί αποτελεί το πλαίσιο του Προσαρμοσμένου Ερωτήματος 1 και περιέχει δύο πεδία εισαγωγής δεδομένων και δύο κουμπιά εκτέλεσης συγκεκριμένων ενεργειών, το δε χρώμα του φόντου είναι ένα ανοιχτό πορτοκαλί. Το πρώτο πεδίο είναι αυτό στο οποίο ο χρήστης καλείται να εισάγει το επιθυμητό ερώτημα προς εκτέλεση και είναι στοιχείο <TEXTAREA> της HTML με name="query1". Όταν ανοίγει η ιστοσελίδα αναγράφεται συγκεκριμένο κείμενο οδηγιών σε κάθε πεδίο εισαγωγής ερωτήματος, το οποίο ο χρήστης θα πρέπει να σβήσει προτού εισάγει το επιθυμητό ερώτημα προς εκτέλεση. Μόλις ο χρήστης κάνει κλικ σε οποιοδήποτε πεδίο της εφαρμογής, αυτόματα επιλέγεται όλο το υπάρχον κείμενο και συνεπώς είναι πολύ εύκολο να σβηστεί από τον χρήστη. Το δεύτερο πεδίο είναι αυτό της εισαγωγής της μέγιστης διάρκειας εκτέλεσης του συγκεκριμένου ερωτήματος και είναι στοιχείο <INPUT type="text"> με name="setlimit1" size="3" value="300" και id="setlimit01". Η επιλογή του κειμένου γίνεται με τη χρήση χειριστή συμβάντος onFocus ο οποίος όταν εστιαστεί το συγκεκριμένο πεδίο εκτελεί τη JavaScript συνάρτηση SelectAll που έχει περιγραφεί σε προηγούμενη παράγραφο. Όταν πατηθεί το κουμπί εκτέλεσης του Προσαρμοσμένου Ερωτήματος 1, εκτελείται το "execQuery_getResults.php" με τις παραμέτρους refreshT=5 και query=1 και εκτελείται και μία σειρά από συναρτήσεις JavaScript μέσα από τον χειριστή συμβάντος ως εξής:

```
onsubmit="validate_required(query1,'Εσφαλμένη εισαγωγή δεδομένων! Το πεδίο του Προσαρμοσμένου Ερωτήματος1 δεν μπορεί να είναι κενό.'); validate_select(query1,'Εσφαλμένη εισαγωγή δεδομένων! Το Προσαρμοσμένο Ερώτημα1 πρέπει να ξεκινά με τη λέξη κλειδί `SELECT` (Χωρίς διάκριση πεζών-ΚΕΦΑΛΑΙΩΝ).'); validate_keywords(query1,'Εσφαλμένη εισαγωγή δεδομένων! Το Προσαρμοσμένο Ερώτημα1 δεν μπορεί να περιλαμβάνει καμία από τις παρακάτω λέξεις κλειδιά: ABORT, ALTER, ANALYZE, BEGIN, CHECKPOINT, CLOSE, CLUSTER, COMMENT, COMMIT, COPY, CREATE, DEALLOCATE, DECLARE, DELETE, DROP, END, EXECUTE, EXPLAIN, FETCH, GRANT, INSERT, LISTEN, LOAD, LOCK, MOVE, NOTIFY, PREPARE, REINDEX, RESET, REVOKE, ROLLBACK, SET, SHOW, START TRANSACTION, TRUNCATE, UNLISTEN, UPDATE, VACUUM (Χωρίς διάκριση πεζών-ΚΕΦΑΛΑΙΩΝ).'); validate_from(query1,'Εσφαλμένη εισαγωγή δεδομένων! Το Προσαρμοσμένο Ερώτημα1 πρέπει να περιέχει τη λέξη κλειδί `FROM` και αυτή πρέπει να βρίσκεται μετά τη λέξη κλειδί `SELECT` (Χωρίς διάκριση πεζών-ΚΕΦΑΛΑΙΩΝ).'); validate_vehicles(query1,'Εσφαλμένη εισαγωγή δεδομένων! Το Προσαρμοσμένο Ερώτημα1 πρέπει να περιέχει το κείμενο `network.vehicles` ή `network.vehicles2` και αυτό πρέπει να βρίσκεται μετά τη λέξη κλειδί `FROM` (Χωρίς διάκριση πεζών-ΚΕΦΑΛΑΙΩΝ).'); validate_quotes(query1,'Εσφαλμένη εισαγωγή δεδομένων! Το Προσαρμοσμένο Ερώτημα1 δεν μπορεί να περιέχει απλά εισαγωγικά(sq). Αντί αυτών, παρακαλώ χρησιμοποιήστε κατακόρυφες μπάρες (|).'); validate_required(setlimit1,'Εσφαλμένη εισαγωγή δεδομένων! Η Μέγιστη Διάρκεια Εκτέλεσης του Προσαρμοσμένου Ερωτήματος1 πρέπει να συμπληρωθεί.'); validate_number(document.f1.setlimit1,'Εσφαλμένη εισαγωγή δεδομένων! Η Μέγιστη Διάρκεια Εκτέλεσης του Προσαρμοσμένου Ερωτήματος1 πρέπει να είναι θετικός αριθμός.'); queryNoResults(1);"
```

Πρόκειται για οκτώ διαφορετικές συναρτήσεις μία εκ των οποίων (validate_required()) εκτελείται δύο φορές, μία φορά για τον έλεγχο του πεδίου εισαγωγής του ερωτήματος και μία φορά για το πεδίο εισαγωγής της μέγιστης διάρκειας εκτέλεσης. Οι συναρτήσεις αυτές έχουν περιγραφεί στη σχετική παράγραφο του κώδικα JavaScript.

Κατ' αντίστοιχο τρόπο με το Βασικό Ερώτημα και το Ερώτημα Περιοχής, ορίζεται το κουμπί διακοπής του Προσαρμοσμένου Ερωτήματος 1 και εμφανίζεται η κατάσταση του ερωτήματος στη σχετική θέση παρακάτω.

Το πρώτο κελί της δεύτερης γραμμής θα φιλοξενήσει τον πίνακα με τα αποτελέσματα του Προσαρμοσμένου Ερωτήματος 1. Για να το πετύχει αυτό, υπάρχει συγκεκριμένο στοιχείο <DIV> (division) με το επίσης συγκεκριμένο id="hhook1". Αυτή είναι και η θέση που η JavaScript θα εμφανίσει τον HTML πίνακα που δημιουργεί κάθε φορά. Το δε χρώμα φόντου του συγκεκριμένου κελιού είναι το ίδιο με αυτό του πλαισίου του Ερωτήματος 1.

Το δεύτερο κελί στην πρώτη γραμμή του έβδομου πίνακα αποτελεί το πλαίσιο του Προσαρμοσμένου Ερωτήματος 2. Καταχωρούνται τα ίδια HTML στοιχεία με το πρώτο κελί της πρώτης γραμμής με διαφορετικές όμως τιμές στα id, name, value και στους χειριστές συμβάντων έτσι που να αντιστοιχούν στην περίπτωση του Ερωτήματος 2. Παρομοίως το δεύτερο κελί της δεύτερης γραμμής είναι αντίστοιχο με το πρώτο κελί της δεύτερης γραμμής με διαφορετικές τιμές στα id και name των στοιχείων HTML έτσι που να αντιστοιχούν στην περίπτωση του Ερωτήματος 2.

Με όμοιο τρόπο καταχωρούνται οι πληροφορίες και στο τρίτο κελί της πρώτης και της δεύτερης γραμμής του έβδομου πίνακα. Τα δύο αυτά κελιά αφορούν στο Προσαρμοσμένο Ερώτημα 3, το οποίο έχει τα δικά του αντίστοιχα ids, names, values και χειριστές συμβάντων. Όπως έχει προαναφερθεί, σε όλα τα πεδία εισαγωγής δεδομένων υπάρχει εκτός των άλλων και χειριστής συμβάντος ο οποίος όταν εστιαστεί το πεδίο αυτομάτως επιλέγει το κείμενο που περιέχουν για την εύκολη διαγραφή του από τον χρήστη.

Στο σημείο αυτό τελειώνει η αναλυτική περιγραφή του κώδικα HTML.

Φυσικά στο παράρτημα υπάρχει ο πλήρης κώδικας HTML με αρκετές επεξηγήσεις, στον οποίο μπορεί να ανατρέξει κανείς να δει πώς ακριβώς έχει συνταχθεί το κάθε στοιχείο και η κάθε πληροφορία αυτού.

5.9 Αναλυτική περιγραφή λειτουργίας αρχείου startStream.php

Το αρχείο αυτό έχει περιγραφεί στην παράγραφο 5.7.2 στον καθορισμό της συνάρτησης OpenTwoLinks(). Πιο αναλυτικά λοιπόν στο σημείο αυτό αναφέρουμε:

Πρώτα καθορίζεται μεταβλητή με όνομα $xmlFile10 και τιμή τη συμβολοσειρά "zstreamRunning.xml".

Μετά ακολουθεί έλεγχος: Εάν ΔΕΝ υπάρχει αυτό το αρχείο τότε:

Δημιουργείται ένα καινούργιο XML αρχείο και αποθηκεύεται με τον τίτλο "zstreamRunning.xml". Μετά εκτελείται η εντολή:

```
exec('cat points5_wgs84.log | ./sliderate.pl localhost 5533
csvwrapper,network.vehicles 15 5');
```

Η εντολή αυτή ξεκινά το πρώτο ρεύμα δεδομένων. Η εκτέλεση της εντολής θα διαρκέσει 6 λεπτά. Το χρονικό αυτό διάστημα το αρχείο PHP θα τελεί σε αναμονή. Μόλις τελειώσει το ρεύμα δεδομένων το PHP θα διαγράψει το "zstreamRunning.xml".

Εάν από την άλλη ήδη υπάρχει το βοηθητικό αυτό αρχείο XML, τότε η εκτέλεση του αρχείου PHP θα σταματήσει εκεί. Εάν εκτελείται ήδη κάποιο ρεύμα δεδομένων δεν πρέπει να ξεκινήσει να εκτελείται δεύτερη φορά το ίδιο ρεύμα διότι οι απαντήσεις των ερωτημάτων θα περιλαμβάνουν απαντήσεις και στα δύο ρεύματα δηλαδή ανά πάσα στιγμή δύο διαφορετικές χρονικές στιγμές κάτι που θα δημιουργούσε υπερφόρτωση και

λάθη στην εφαρμογή. Αυτό λοιπόν το ενδεχόμενο αποφεύγεται με τον αρχικό αυτό έλεγχο κατά την εκτέλεση του startStream.php.

5.10 Αναλυτική περιγραφή λειτουργίας αρχείου startStream2.php

Το αρχείο αυτό μοιάζει αρκετά με το προηγούμενο. Φροντίζει δε για την εκτέλεση του δεύτερου ρεύματος δεδομένων. Αρχικά κάνει έναν πιο σύνθετο έλεγχο. Λογικά σκεπτόμενοι, θέλουμε το δεύτερο ρεύμα να ξεκινήσει μόνο εάν δεν εκτελείται ούτε το πρώτο ρεύμα ούτε το δεύτερο. Εάν εκτελείται ήδη το πρώτο ρεύμα θα το ανιχνεύσει το startStream2.php και δεν θα ξεκινήσει το δεύτερο. Εάν εκτελείται ήδη το δεύτερο ρεύμα πάλι θα το ανιχνεύσει το "startStream2.php" και δεν θα το ξεκινήσει. Αυτό συμβαίνει διότι όταν εκτελούνται και τα δύο ρεύματα δεδομένων πρέπει να έχουν ξεκινήσει σχεδόν ταυτόχρονα. Ο χρήστης έχει τη δυνατότητα να ξεκινήσει είτε μόνο το πρώτο ρεύμα δεδομένων είτε και τα δύο, όχι μόνο του το δεύτερο ρεύμα δεδομένων. Δεν υπάρχει κάποιο νόημα στην εκτέλεση μόνο του δεύτερου ρεύματος καθώς πρόκειται για αντίγραφο του πρώτου.

Έτσι λοιπόν η μεταβλητή $xmlFile10 αποθηκεύει το όνομα του "zstreamRunning.xml" και η μεταβλητή $xmlFile20 αποθηκεύει το όνομα του "zstreamRunning2.xml", αρχεία η ύπαρξη των οποίων δηλώνει ότι τα αντίστοιχα ρεύματα εκτελούνται.

Ακολουθεί ο έλεγχος:

Εάν δεν υπάρχει το πρώτο αρχείο και δεν υπάρχει ούτε το δεύτερο, τότε προχωρά η εκτέλεση του script, αλλιώς αυτή σταματά εδώ.

Εάν λοιπόν δεν υπάρχει κανένα από τα βοηθητικά αυτά αρχεία, δημιουργείται νέο αρχείο XML που αποθηκεύεται με το όνομα "zstreamRunning2.xml" και εκτελείται η εντολή:

```
exec('cat points5_wgs84.log | ./sliderate.pl localhost 5533 csvwrapper,network.vehicles2 15 5');
```

Με αυτήν, ξεκινά το δεύτερο ρεύμα δεδομένων το οποίο θα διαρκέσει για τα επόμενα 6 λεπτά. Αντίστοιχα με την περίπτωση του πρώτου ρεύματος, το "startStream2.php" θα τελεί σε αναμονή μέχρι τη παύση εκτέλεσης του ρεύματος η οποία θα γίνει μετά από 6 λεπτά. Εάν με οποιονδήποτε τρόπο διακοπεί η εκτέλεση του ρεύματος πριν τη λήξη του, το αρχείο PHP θα συνεχίσει την εκτέλεσή του. Μετά λοιπόν από την παύση ή διακοπή της εκτέλεσης του δεύτερου ρεύματος, αντίστοιχα με το "startStream.php", θα σβηστεί το "zstreamRunning2.xml" και θα τελειώσει η εκτέλεση αυτού του αρχείου PHP.

5.11 Αναλυτική περιγραφή λειτουργίας αρχείου stopQuery.php

Το αρχείο stopQuery.php εκτελείται, με συγκεκριμένη παράμετρο, κάθε φορά που ο χρήστης πατήσει κάποιο κουμπί διακοπής ενός ερωτήματος. Η παράμετρος (squery) ουσιαστικά δηλώνει τον αριθμό του ερωτήματος προς διακοπή. Ξεκινώντας την εκτέλεσή του, το αρχείο αυτό, αποθηκεύει σε δική του μεταβλητή ($sqID) τον αριθμό του ερωτήματος προς διακοπή τον οποίο βρίσκει στην παράμετρο:

```
$sqID = $HTTP_GET_VARS['squery'];
```

Ακολουθεί η δημιουργία ενός βοηθητικού αρχείου XML με δυναμικό τίτλο:

```
$xmlFile = "zstopQuery".$sqID.".xml"
```

Ο τίτλος αυτού του αρχείου εξαρτάται από την τιμή της μεταβλητής $sqID.

Το αρχείο αυτό, εφόσον εκτελείται το συγκεκριμένο ερώτημα από το "execQuery_getResults.php" και εφόσον η PostgreSQL επιστρέφει απαντήσεις σε αυτό το ερώτημα, θα εντοπιστεί από το παραπάνω αρχείο PHP στην επόμενη ανανέωση του

βρόχου αναζήτησης και αποθήκευσης αποτελεσμάτων και θα το PHP διακόψει την εκτέλεση του βρόχου διαγράφοντας και το συγκεκριμένο αρχείο XML. Στη συνέχεια το "execQuery_getResults.php" θα σταματήσει την εκτέλεση του ερωτήματος, θα κλείσει τη σύνδεση με τη βάση δεδομένων και θα διαγράψει τα βοηθητικά αρχεία με τίτλο "zQuery#IsRunning.xml" και results#.xml.

Όπως γνωρίζουμε, η ανανέωση των αποτελεσμάτων και η επανεκτέλεση των βρόχων και των ελέγχων γίνεται κάθε 5 δευτερόλεπτα.

Το stopQuery.php δε θα σταματήσει στο σημείο αυτό την εκτέλεσή του. Αφού μείνει σε αδράνεια για 6 δευτερόλεπτα θα αναζητήσει το αρχείο results#.xml το οποίο εάν υπάρχει θα σβηστεί. Μετά από άλλο ένα δευτερόλεπτο θα αναζητήσει το zstopQuery#.xml και το zQuery#IsRunning.xml και εάν υπάρχουν θα τα σβήσει και αυτά. Όπως έχουμε εξηγήσει στον κώδικα JavaScript, η εμφάνιση στην ιστοσελίδα της κατάστασης του κάθε ερωτήματος εξαρτάται από την ύπαρξη ή όχι του αρχείου zQuery#IsRunning.xml και η εμφάνιση των αποτελεσμάτων επί του χάρτη εξαρτάται και αυτή από την ύπαρξη ή όχι του results#.xml. Αυτά τα δύο αρχεία, όσο υπάρχουν, θα φαίνεται ότι το ερώτημα εκτελείται και θα φαίνονται τα αποτελέσματα στο χάρτη και στους πίνακες για τις περιπτώσεις των Προσαρμοσμένων Ερωτημάτων. Βέβαια εάν δεν εκτελείται το ερώτημα ή δεν επιστρέφει αποτελέσματα η PostgreSQL ή έχει σταματήσει το ρεύμα δεδομένων, τα αποτελέσματα θα παραμένουν τα ίδια. Εάν σβηστούν αυτά τα αρχεία, στην ιστοσελίδα θα φανεί ότι έχει σταματήσει το ερώτημα. Εάν λοιπόν το ερώτημα δεν επιστρέφει αποτελέσματα ή δεν εκτελείται πρέπει και πάλι να σβηστούν, εάν υπάρχουν, τα αρχεία results#.xml και zQuery#IsRunning.xml τα οποία διαβάζει η JavaScript ώστε να φανεί ότι έχει σταματήσει το ερώτημα. Εάν το ερώτημα εκτελείται αλλά δεν επιστρέφει αποτελέσματα, έχει σχεδιαστεί διαφορετική μέθοδος διακοπής του που θα περιγραφεί παρακάτω.

Συνοψίζοντας, το stopQuery.php όταν εκτελεστεί επιχειρεί να σταματήσει κανονικά το ερώτημα αλλά εάν αυτό δεν επιστρέφει αποτελέσματα τότε δε θα μπορέσει να σταματήσει και το PHP σβήνει απλά τα αρχεία που διαβάζει η JavaScript ώστε να φαίνεται στην ιστοσελίδα ότι το ερώτημα έχει σταματήσει και να είναι δυνατή η εκτέλεση άλλου ερωτήματος στη θέση του. Όπως έχει προαναφερθεί, η ύπαρξη του zQuery#IsRunning.xml εμποδίζει την εκτέλεση άλλου ερωτήματος ταυτόχρονα στην ίδια θέση.

Η διαχείριση των ερωτημάτων που δεν επιστρέφουν απαντήσεις γίνεται με άλλους τρόπους που θα περιγραφούν στη συνέχεια.

5.12 Αναλυτική περιγραφή λειτουργίας αρχείου stopAllQueries.php

Το αρχείο αυτό εκτελείται όταν ο χρήστης πατήσει το κουμπί διακοπής όλων των ερωτημάτων και δεν έχει παραμέτρους. Καλείται δε να σταματήσει όλα τα ερωτήματα και αναμφίβολα πετυχαίνει το στόχο του. Μοιάζει με την ταυτόχρονη εκτέλεση του stopQuery.php και για τα πέντε ερωτήματα αλλά εκτελεί και μία ακόμη σημαντική ενέργεια.

Πιο συγκεκριμένα, αρχικά δημιουργεί και τα πέντε βοηθητικά αρχεία XML zstopQuery0.xml, zstopQuery1.xml, zstopQuery2.xml, zstopQuery3.xml και zstopQuery6.xml. Όποιο ερώτημα εκτελείται και επιστρέφει αποτελέσματα θα σταματήσει για τον ίδιο λόγο που θα σταματήσει ένα συγκεκριμένο ερώτημα με το stopQuery.php.

Μετά από 6 δευτερόλεπτα, ώστε να έχει γίνει τουλάχιστον μία επανεκτέλεση του βρόχου ανανέωσης αποτελεσμάτων στις εκτελέσεις του "execQuery_getResults.php", εξετάζεται η ύπαρξη των results0.xml, results1.xml, results2.xml, results3.xml και results6.xml. Όποια από αυτά υπάρχουν σβήνονται. Μετά από άλλο ένα δευτερόλεπτο

σβήνονται και όσα από τα zstopQuery#.xml εξακολουθούν να υπάρχουν. Στη συνέχεια, δεν σβήνονται τα αλλά αντ' αυτών δημιουργούνται τα deleteFetch0.xml, deleteFetch1.xml, deleteFetch2.xml, deleteFetch3.xml και deleteFetch6.xml. Τα αρχεία αυτά ανιχνεύονται από πρόγραμμα γραμμένο σε C++ το οποίο όταν αυτά υπάρχουν φροντίζει για την αναγκαστική διακοπή των ερωτημάτων και εκτός των άλλων, σβήνει τα zQuery#IsRunning.xml και τα deleteFetch#.xml. Περισσότερα για το πρόγραμμα αυτό θα αναφερθούν στη σχετική παράγραφο παρακάτω.

5.13 Αναλυτική περιγραφή λειτουργίας ομάδας αρχείων stopQuery#WithNoResults.php

Εκτός των παραπάνω αρχείων PHP υπάρχουν και τα εξής πέντε αρχεία: stopQuery0WithNoResults.php, stopQuery1WithNoResults.php, stopQuery2WithNoResults.php, stopQuery3WithNoResults.php και stopQuery6WithNoResults.php.

Παρακάτω θα αναφερθούμε ταυτόχρονα και στα πέντε αρχεία χρησιμοποιώντας τη δίεση (#) αντί των πέντε αριθμών.

Ξεκινώντας το script, ορίζεται μεταβλητή με όνομα $c# και τιμή 0.

Αποθηκεύεται η συμβολοσειρά "zQuery#IsRunning.xml" στη μεταβλητή $xmlFile# και η "results#.xml" στη μεταβλητή $xmlFile1#.

Το πρόγραμμα τελεί σε αναμονή για 2 δευτερόλεπτα με τη συνάρτηση sleep() και ακολούθως γίνεται ο πρώτος έλεγχος:

```
if (file_exists($xmlFile#)&&!file_exists($xmlFile1#)) $c#++;
```

Εάν υπάρχει το αρχείο, το όνομα του οποίου έχει αποθηκευτεί στην $xmlFile# και δεν υπάρχει το αρχείο, το όνομα του οποίου έχει αποθηκευτεί στην $xmlFile1#, τότε ο μετρητής $c# αυξάνει την τιμή του κατά 1. Αυτός ο έλεγχος πολύ απλά σημαίνει: εάν μετά από δύο δευτερόλεπτα από την εκκίνηση εκτέλεσης του script εκτελείται το ερώτημα και δεν υπάρχουν αποτελέσματα, αυξάνουμε τον μετρητή κατά 1.

Ακολουθεί αναμονή του script μέχρι να περάσουν επιπλέον 5 δευτερόλεπτα. Μετά επαναλαμβάνει τον ίδιο έλεγχο. Εάν το ερώτημα εκτελείται και δεν υπάρχουν αποτελέσματα, τότε αυξάνει την τιμή του μετρητή κατά 1.

Η διαδικασία αναμονής 5 sec και ελέγχου ύπαρξης των αρχείων επαναλαμβάνεται άλλες δύο φορές. Αφού περάσουν συνολικά 17 δευτερόλεπτα (2+5+5+5) και γίνει και ο τέταρτος έλεγχος, τότε γίνεται ο τελευταίος έλεγχος του script:

```
if ($c#==4) { //κώδικας }
```

Εάν ο μετρητής έχει την τιμή 4 τότε θα εκτελεστεί συγκεκριμένος κώδικας. Εάν σε καμία χρονική στιγμή (μετά από 2, μετά από 7, μετά από 12 και μετά από 17 δευτερόλεπτα) δεν υπήρχαν αποτελέσματα ενώ το ερώτημα εκτελείτο, τότε ο μετρητής θα έχει πάρει σε αυτό το σημείο του script την τιμή 4. Στην περίπτωση αυτή θα πρέπει να διακοπεί το ερώτημα. Το πρόβλημα όμως είναι ότι αυτό δεν επιστρέφει αποτελέσματα και συνεπώς δεν επανεκτελείται ο εξωτερικός αλλά ούτε και ο εσωτερικός βρόχος του execQuery_getResults.php διότι αυτό έχει σταματήσει στην εντολή pg_query("FETCH FORWARD 1 IN Q1_portion;"), περιμένοντας εσαεί από την PostgreSQL απάντηση.

Το stopQuery#WithNoResults.php λοιπόν στο σημείο αυτό έχει προγραμματιστεί να δημιουργεί ένα βοηθητικό αρχείο XML με όνομα "deleteFetch#.xml". Μετά τη δημιουργία αυτού του αρχείου XML, το stopQuery#WithNoResults.php τερματίζει την εκτέλεσή του.

Το "deleteFetch#.xml" μέσα στα επόμενα 4 δευτερόλεπτα θα ανιχνευθεί από ειδικό πρόγραμμα γραμμένο σε C++ το οποίο εκτελείται συνεχώς και το οποίο θα εκτελέσει συγκεκριμένες ενέργειες που θα αναλύσουμε στην επόμενη παράγραφο.

5.14 Αναλυτική περιγραφή λειτουργίας προγράμματος με όνομα killQueryNoResults

Το killQueryNoResults είναι πρόγραμμα που έχει δημιουργηθεί στη γνωστή γλώσσα προγραμματισμού C++ και χρησιμεύει για την διακοπή εκτέλεσης ερωτημάτων που έχουν "κολλήσει" και δεν επιστρέφουν αποτελέσματα. Το πρόγραμμα αυτό λοιπόν, σε συνεργασία με διάφορα scripts της εφαρμογής, καλείται να λύσει ένα πρόβλημα το οποίο περιγράφεται ως εξής:

Εάν ένα ερώτημα είναι σωστά συντεταγμένο (ακολουθεί τους κανόνες που έχουν περιγραφεί σε προηγούμενη παράγραφο) και περνάει από τους ελέγχους της PostgreSQL και του TelegraphCQ, θα ξεκινήσει οπωσδήποτε η εκτέλεσή του από την PostgreSQL χωρίς φυσικά να εμφανιστεί κανένα μήνυμα λάθους. Το ερώτημα αυτό μπορεί κάλλιστα να επιστρέφει αποτελέσματα, εάν υπάρχουν καταχωρήσεις που πληρούν τις προϋποθέσεις αυτού. Από την άλλη, το ερώτημα, μπορεί κάλλιστα να μην επιστρέφει αποτελέσματα διότι πολύ απλά μπορεί να μην υπάρχουν καταχωρήσεις που να πληρούν τις προϋποθέσεις αυτού αλλά και για άλλους λόγους (π.χ. να μην εκτελείται το ρεύμα δεδομένων). Για τον περιορισμό διαφόρων περιπτώσεων όπου το ερώτημα εκτελείται αλλά δεν επιστρέφει αποτελέσματα, έχουν δημιουργηθεί διαφόρων ειδών έλεγχοι σε διάφορα σημεία της εφαρμογής τα οποία έχουν περιγραφεί παραπάνω. Υπάρχουν όμως πάντα περιπτώσεις όπου ένα ερώτημα μπορεί να μην επιστρέφει αποτελέσματα. Στην περίπτωση αυτή, η σύνδεση με τη βάση δεδομένων και μία σύνδεση Apache θα παραμείνουν ανοιχτές εσαεί, κάτι φυσικά μη επιθυμητό. Ειδικά οι συνδέσεις με τη βάση δεδομένων είναι αρκετά περιορισμένες και πρέπει να βρεθεί τρόπος εάν το ερώτημα δεν επιστρέφει αποτελέσματα, αυτές να κλείνουν, όχι από το PHP αλλά με οποιονδήποτε άλλο τρόπο. Πρέπει να αναφερθεί ότι οι συνδέσεις με τη βάση δεδομένων ανοίγονται από τον χρήστη daemon της PHP μέσα από την εκτέλεση του PHP script και ότι δεν μπορούν να κλείσουν κανονικά παρά μόνο από την εκτέλεση του ίδιου αρχείου PHP που τις άνοιξε. Έχουν γίνει πολλές δοκιμές σε αυτό το θέμα και σχολαστική διερεύνηση χωρίς επιτυχία. Ο μόνος τρόπος λοιπόν που φαίνεται να υπάρχει για το κλείσιμο των συνδέσεων με την PostgreSQL είναι αυτές να τερματιστούν με την εντολή kill ή την εντολή pkill σε τερματικό του Linux. Έχει βρεθεί λοιπόν τρόπος σαφούς εντοπισμού του κωδικού αριθμού της κάθε σύνδεσης ώστε να είναι δυνατός ο τερματισμός της. Μετά λοιπόν από τα 17 δευτερόλεπτα ελέγχου του ερωτήματος και των αποτελεσμάτων από το stopQuery#WithNoResults.php, εάν δεν υπάρχουν αποτελέσματα, τερματίζεται από το πρόγραμμα αυτό η συγκεκριμένη σύνδεση με την PostgreSQL και η συγκεκριμένη σύνδεση Apache. Ο εντοπισμός και η αποθήκευση σε εξωτερικά αρχεία των ID των διεργασιών γίνεται από άλλο πρόγραμμα γραμμένο σε C++ το οποίο περιγράφεται αμέσως μετά.

Το killQueryNoResults λοιπόν είναι το πρόγραμμα που τερματίζει υπό συγκεκριμένες συνθήκες τη σύνδεση με την PostgreSQL και τη σύνδεση Apache κάθε ερωτήματος το οποίο δεν επιστρέφει αποτελέσματα. Λειτουργεί δε ως εξής:

Αφού οριστούν οι αναγκαίες βιβλιοθήκες συναρτήσεων, καθορίζονται οι μεταβλητές που θα χρησιμοποιηθούν από το πρόγραμμα:

```
   int fexists0; int fexists1; int fexists2; int fexists3;
int fexists6; int fexists7; int n = 0; string myFileName0;
string myFileName1; string myFileName2; string myFileName3;
string myFileName6; string stopCpp;
```

Η ομάδα μεταβλητών fexists# και ο μετρητής n είναι τύπου ακέραιου αριθμού. Η ομάδα μεταβλητών myFileName# και η stopCpp είναι τύπου συμβολοσειράς.

Αφού δοθούν τιμές στις μεταβλητές τύπου συμβολοσειράς:

```
myFileName0 = "killProcQ0.xml";
myFileName1 = "killProcQ1.xml";
myFileName2 = "killProcQ2.xml";
myFileName3 = "killProcQ3.xml";
```

```
myFileName6 = "killProcQ6.xml";
stopCpp = "stopKillQueries.txt";
```

καταχωρούνται ορισμένα σχόλια σχετικά με το πρόγραμμα τα οποία θα εμφανιστούν εάν αυτό εκτελεστεί μέσα από τερματικό. Στη συνέχεια γίνεται έλεγχος εάν υπάρχει το αρχείο stopKillFile.xml και εάν υπάρχει θα διαγραφεί προτού ξεκινήσει ο βρόχος του προγράμματος.

Ακολούθως, ξεκινά ένας βρόχος ο οποίος επαναλαμβάνεται κάθε τέσσερα δευτερόλεπτα, εφόσον δεν υπάρχει το αρχείο stopKillFile.xml. Το πρόγραμμα λοιπόν αυτό εκτελείται συνεχώς και οι μόνοι τρόποι για να σταματήσει είναι είτε να δημιουργηθεί μέσα στον ίδιο φάκελο αρχείο με όνομα stopKillFile.xml (δημιουργείται με τη βοήθεια του προγράμματος killQueryNoResultsStop) είτε να τερματιστεί με εντολή τύπου kill η λειτουργία του από τον διαχειριστή του database server όπου θα εκτελείται η PostgreSQL και το TelegraphCQ.

Μέσα στον βρόχο, κάθε 4 δευτερόλεπτα, γίνεται μία σειρά ελέγχων εάν υπάρχουν τα αρχεία της ομάδας killProcQ#.xml (kill Process Query #). Εάν υπάρχει το killProcQ#.xml (με τη # να είναι ο αριθμός οποιουδήποτε ερωτήματος), τότε η μεταβλητή fexists# παίρνει την τιμή 1 και εκτελούνται οι ακόλουθες ενέργειες:

Η C++ ανοίγει το αρχείο "zq#dbconnID.log" στο οποίο είναι αποθηκευμένο το ID της διεργασίας σύνδεση με την PostgreSQL για το ερώτημα #, και αποθηκεύει σε μεταβλητή τύπου συμβολοσειράς το περιεχόμενο του αρχείου που δεν είναι άλλο από έναν ακέραιο αριθμό που δηλώνει το ID. Στη συνέχεια, νέα μεταβλητή αποθηκεύει τη συμβολοσειρά "kill " και το ID που καταχωρήθηκε νωρίτερα σε ένα ενιαίο κείμενο. Αυτό, μετά από κατάλληλη μετατροπή, εκτελείται με τη βοήθεια της συνάρτησης system της C++ ως εντολή στον πυρήνα του λειτουργικού. Το αποτέλεσμα είναι να τερματιστεί η διεργασία σύνδεσης της PHP με τη βάση δεδομένων.

Μετά, με αντίστοιχο τρόπο, αποθηκεύεται σε μεταβλητή της C++ η τιμή του ID της σχετικής διεργασίας σύνδεσης του Apache. Στη συνέχεια, πάλι με αντίστοιχο τρόπο, εκτελείται στον πυρήνα του λειτουργικού η εντολή "kill ID" με ID τον αριθμό της διεργασίας αυτής.

Μετά από την εκτέλεση αυτών των εντολών, που είναι και ο λόγος ύπαρξης του προγράμματος αυτού, εάν ο χρήστης εκτελεί το πρόγραμμα μέσα από τερματικό, θα ενημερωθεί σχετικά με τις ενέργειες που μόλις έγιναν με σχετικό κείμενο. Μετά από το κείμενο αυτό, θα διαγραφούν τα αρχεία: killProcQ#.xml, zQuery#IsRunning.xml, zq#dbconnID.log και zq#httpdID.log.

Αυτή η διαδικασία γίνεται για κάθε ερώτημα ξεχωριστά.

Πριν τελειώσει ο βρόχος, εάν υπάρχει το αρχείο "stopKillQueries.txt" στον ίδιο φάκελο με το πρόγραμμα, η μεταβλητή fexists7 θα έχει πάρει την τιμή 1 (από προηγούμενο έλεγχο) και θα εμφανιστεί σχετικό μήνυμα στο τερματικό και θα διαγραφεί το αρχείο "stopKillQueries.txt". Εάν λοιπόν η fexists7 έχει την τιμή 1, ο βρόχος θα σταματήσει να εκτελείται.

Μετά το βρόχο σταματά η εκτέλεση του προγράμματος.

5.15 Αναλυτική περιγραφή λειτουργίας προγράμματος με όνομα storeProcessIDs

Το πρόγραμμα αυτό, όπως δηλώνει και το όνομά του, αποθηκεύει τα IDs ορισμένων διεργασιών. Οι διεργασίες αυτές δεν είναι άλλες από τις συνδέσεις της PHP με την PostgreSQL για κάθε ερώτημα και οι αντίστοιχες συνδέσεις του Apache για κάθε ερώτημα.

Η χρήση αυτού του προγράμματος είναι απαραίτητη εάν θέλουμε μετά από συγκεκριμένο χρόνο να σταματάμε όποιο ερώτημα δεν επιστρέφει απαντήσεις.

Ο τρόπος λειτουργίας του προγράμματος αυτού είναι παρόμοιος με το killQueryNoResults:

Πρώτα ορίζονται οι αναγκαίες βιβλιοθήκες συναρτήσεων, ύστερα οι μεταβλητές τύπου ακεραίου και μετά οι μεταβλητές τύπου συμβολοσειράς ως εξής:

```
int fexists0; int fexists1; int fexists2; int fexists3;
int fexists6; int fexists7; int n = 0;
int n0 = 0; int n1 = 0; int n2 = 0; int n3 = 0; int n6 = 0;
string myFileName0; string myFileName1; string myFileName2;
string myFileName3; string myFileName6; string stopCpp;
```

Τα αρχεία, των οποίων ελέγχεται η ύπαρξη, είναι σε αυτήν την περίπτωση τα:

"zQuery0IsRunning.xml", "zQuery1IsRunning.xml", "zQuery2IsRunning.xml", "zQuery3IsRunning.xml", "zQuery6IsRunning.xml", "stopStoreIDs.txt".

Μετά λοιπόν την αποθήκευση των ονομάτων των αρχείων στις παραπάνω μεταβλητές, υπάρχουν πληροφορίες σχετικές με το πρόγραμμα που εμφανίζονται εάν ο χρήστης το εκτελέσει μέσα από τερματικό.

Μετά γίνεται έλεγχος εάν υπάρχει ήδη το “stopStoreIDs.txt” η ύπαρξη του οποίου, σε επόμενη χρονική στιγμή, θα διακόψει την λειτουργία του προγράμματος. Για να μην διακοπεί λοιπόν αμέσως το συγκεκριμένο πρόγραμμα, ένα υπάρχει ήδη αυτό το αρχείο, τότε διαγράφεται. Εάν λοιπόν το αρχείο αυτό δημιουργηθεί ξανά μέσα στο φάκελο ενώ το πρόγραμμα εκτελεί τον βρόχο που υπάρχει στο τελευταίο, τότε θα διακοπεί η εκτέλεση του βρόχου και η εκτέλεση του προγράμματος.

Ο βρόχος λοιπόν που ακολουθεί εκτελείται κάθε μισό δευτερόλεπτο και διακόπτεται μόνο εάν δημιουργηθεί το παραπάνω αναφερόμενο αρχείο. Κάθε φορά που εκτελείται ο βρόχος, κατ’ αντιστοιχία με το killQueryNoResults ελέγχεται η ύπαρξη των έξι συνολικά αρχείων. Εάν υπάρχει το αρχείο zQuery#IsRunning.xml, η μεταβλητή fexists# παίρνει την τιμή 1.

Στη συνέχεια, γίνεται ο εξής έλεγχος:

Εάν η fexists# έχει την τιμή 1 και η n# έχει την τιμή 0, τότε:
η n# παίρνει την τιμή 1 ώστε να μην εκτελεστεί πάλι αυτό το τμήμα του κώδικα,
το πρόγραμμα τελεί σε αναμονή για 0,6 δευτερόλεπτα,
εκτελείται η εντολή εύρεσης και αποθήκευσης της τελευταίας διεργασίας της postgres με χρήστη τον kostas (ο διαχειριστής της PostgreSQL και του TelegraphCQ),
το πρόγραμμα τελεί σε αναμονή για 1 δευτερόλεπτο,
εκτελείται η εντολή εύρεσης και αποθήκευσης της τελευταίας διεργασίας του Apache (httpd) με χρήστη τον daemon (ο διαχειριστής των εκτελέσεων της PHP),
εάν το πρόγραμμα εκτελείται σε τερματικό, τότε εμφανίζονται σχετικές πληροφορίες για τις ενέργειες που μόλις εκτελέστηκαν.

Εάν τέλος υπάρχει το αρχείο stopStoreIDs.txt, τότε η fexists7 παίρνει την τιμή 1, εμφανίζονται σχετικές πληροφορίες στο τερματικό, διαγράφεται το αρχείο αυτό και τερματίζεται η εκτέλεση του βρόχου. Ακριβώς πριν τον τερματισμό του βρόχου, γίνεται ο έλεγχος εάν έστω και μία από τις μεταβλητές fexists# (με # = 0ή 1 ή 2 ή 3 ή 6). Στην περίπτωση που ισχύει τότε θα εμφανιστούν σχετικές πληροφορίες στο τερματικό. Το τερματικό φυσικά, σε όλες τις περιπτώσεις, μπορεί να δει μόνο ο διαχειριστής του database server και όχι οι απομακρυσμένοι χρήστες της εφαρμογής.

Μετά τον βρόχο τελειώνει και η εκτέλεση του προγράμματος.

5.16 Αναλυτική περιγραφή λειτουργίας προγράμματος με όνομα killFetch

Το πρόγραμμα αυτό έχει παρόμοια αρχιτεκτονική λειτουργίας με τα δύο προηγούμενα. Σκοπός του είναι να ελέγχει για την ύπαρξη ή όχι της ομάδας αρχείων deleteFetch#.xml τα οποία δημιουργούνται από την εκτέλεση του "stopAllQueries.php" και εάν αυτά υπάρχουν, να τερματίζει για κάθε περίπτωση την πιο πρόσφατη διεργασία σύνδεσης της PHP με την PostgreSQL και την πιο πρόσφατη διεργασία του Apache.

Όταν ο χρήστης ζητήσει τη διακοπή όλων των ερωτημάτων πατώντας το σχετικό κουμπί στην ιστοσελίδα, όπως έχουμε προαναφέρει, εκτελείται το αρχείο "stopAllQueries.php" το οποίο έχει σχεδιαστεί για να κάνει ακριβώς αυτό, να σταματά όλα τα ερωτήματα. Όταν λέμε όλα τα ερωτήματα, εννοούμε αυτά που εκτελούνται και επιστρέφουν απαντήσεις αλλά και αυτά που ενώ εκτελούνται δεν επιστρέφουν απαντήσεις. Τα πέντε "deleteFetch#.xml" που δημιουργούνται ούτως ή άλλως από την εκτέλεση του "stopAllQueries.php", τερματίζουν (εάν υπάρχουν) μέχρι πέντε πρόσφατες συνδέσεις της PHP με την PostgreSQL που δεν επιστρέφουν αποτελέσματα και τις πέντε πρόσφατες συνδέσεις του Apache με χρήστη τον daemon.

Ο κώδικας του killFetch ξεκινά και αυτός όπως στις δύο προηγούμενες περιπτώσεις με τον καθορισμό των αναγκαίων βιβλιοθηκών και τον ορισμό των αναγκαίων μεταβλητών. Αφού αποθηκευτούν οι συμβολοσειρές "deleteFetch#.xml" στις αντίστοιχες μεταβλητές, εμφανίζονται πληροφορίες στο σχετικό τερματικό εάν το πρόγραμμα εκτελείται από εκεί. Στη συνέχεια ελέγχεται η ύπαρξη ή όχι του αρχείου "stopKillFetch.txt" και εάν υπάρχει διαγράφεται.

Ακολουθεί βρόχος με αντίστοιχο τρόπο λειτουργίας με τα δύο παραπάνω προγράμματα. Εκτελείται μία φορά κάθε 4 δευτερόλεπτα και θα σταματήσει την εκτέλεσή του μόνο εάν βρεθεί, στον ίδιο φάκελο με το πρόγραμμα, το αρχείο "stopKillFetch.txt". Μέσα στον βρόχο εκτελούνται οι εξής ενέργειες:

Έλεγχος για την ύπαρξη των 6 αρχείων (ομάδα "deleteFetch#.xml" και το "stopKillFetch.txt").

Εάν υπάρχει οποιοδήποτε "deleteFetch#.xml", τότε το πρόγραμμα τερματίζει την πιο πρόσφατη διεργασία που στην περιγραφή της έχει τη λέξη-κλειδί FETCH (εξ' ου και το όνομα του προγράμματος). Τερματίζει επίσης την πιο πρόσφατη διεργασία του προγράμματος httpd στην οποία χρήστης είναι ο daemon.

Εάν δημιουργηθεί, αντιγραφεί ή εμφανιστεί με οποιονδήποτε τρόπο το "stopKillFetch.txt", τότε η εκτέλεση του βρόχου σταματά.

Μετά τον τερματισμό εκτέλεσης του βρόχου, σταματά και η εκτέλεση του προγράμματος.

Έτσι, κατά κάποιον τρόπο, το "stopAllQueries.php" σε συνεργασία με το πρόγραμμα killFetch συντελεί στην εκκαθάριση διεργασιών που επιβαρύνουν το σύστημα χωρίς λόγο και στην εύρυθμη λειτουργία της εφαρμογής.

5.17 Αναλυτική περιγραφή λειτουργίας προγράμματος με όνομα cleanup0600

Το πρόγραμμα αυτό στόχο έχει την εκκαθάριση αρχείων και συνδέσεων της εφαρμογής ανά τακτά χρονικά διαστήματα. Έχει επιλεγεί η εκκαθάριση να γίνεται κάθε 24 ώρες αλλά φυσικά αυτό μπορεί να ρυθμιστεί ανάλογα με τις απαιτήσεις της εφαρμογής σε πραγματικές συνθήκες λειτουργίας. Για την επίτευξη του στόχου αυτού, θα μπορούσε κάλλιστα να υπάρχει ένας βρόχος ο οποίος κάθε 86400 δευτερόλεπτα (24 ώρες) θα εκτελούσε την εκκαθάριση. Στην περίπτωση αυτή όμως, εάν ο διαχειριστής του

database server αποφάσιζε ότι επιθυμεί να σταματήσει την εκτέλεση αυτού του αρχείου, θα έπρεπε να περιμένει έως και 24 ώρες μέχρι την επόμενη επανάληψη του βρόχου ώστε να διακοπεί αυτός. Εάν ο βρόχος εκτελείται ανά τακτά μικρά χρονικά διαστήματα π.χ. των 10 δευτερολέπτων και η εκκαθάριση γίνεται όταν σχετικός μετρητής φτάσει από την τιμή 0 στην τιμή 8640, πετυχαίνεται ο στόχος της γρήγορης διακοπής του βρόχου, εάν το επιθυμεί ο χρήστης, δημιουργώντας ένα σχετικό βοηθητικό αρχείο, αλλά η εκκαθάριση δεν θα γίνεται κάθε 86400 δευτερόλεπτα ακριβώς διότι η κάθε εκτέλεση του βρόχου διαρκεί ορισμένο ελάχιστο χρονικό διάστημα. Έτσι, στις 8640 εκτελέσεις του βρόχου μέσα σε ένα 24ώρο, έχει διαπιστωθεί ότι δεν θα έχουν περάσει ακριβώς 86400 δευτερόλεπτα αλλά μερικά παραπάνω ανάλογα με τις επεξεργαστικές δυνατότητες του database server και το βαθμό φόρτωσής του με εκτέλεση διαφόρων προγραμμάτων. Βρέθηκε λοιπόν μέθοδος ώστε κάθε επανάληψη του βρόχου να εκτελείται ακριβώς μία συγκεκριμένη ώρα της ημέρας χωρίς να εξαρτάται από τον υπολογιστή στον οποίο εκτελείται το πρόγραμμα. Το αποτέλεσμα είναι ότι πραγματικά κάθε 86400 ±2-3 δευτερόλεπτα γίνεται και μία εκκαθάριση. Η εκκαθάριση θα πρέπει να γίνεται συγκεκριμένη ώρα που θα επιλέξει ο διαχειριστής του database server, λογικά την στιγμή της ημέρας που οι λιγότεροι χρήστες θα είναι συνδεδεμένοι με την εφαρμογή ή και με οποιαδήποτε άλλα κριτήρια. Έχει επιλεγεί η εκκαθάριση να γίνεται στις 06:00 κάθε πρωί. Για την ακρίβεια, αυτή έχει προγραμματιστεί να γίνεται είτε στις 06:00:00, είτε στις 06:00:01, είτε στις 06:00:02, είτε στις 06:00:03, είτε στις 06:00:04, είτε στις 06:00:05.

Ο σχετικός βρόχος εκτελείται κάθε 5 δευτερόλεπτα και ελέγχει για δύο πράγματα, είτε εάν είναι η κατάλληλη ώρα για εκκαθάριση είτε εάν υπάρχει το αρχείο "stopCleanup.txt". Στην περίπτωση που υπάρχει το αρχείο αυτό σημαίνει ότι ο διαχειριστής επιθυμεί τη διακοπή εκτέλεσης του βρόχου και του προγράμματος και αυτό γίνεται στην επόμενη επανάληψη του βρόχου.

Η εκκαθάριση περιλαμβάνει τις ακόλουθες ενέργειες:

1. Τερματισμός διεργασιών του Apache (httpd) με χρήστη τον daemon.
2. Τερματισμός διεργασιών που στον τίτλο τους έχουν τη λέξη-κλειδί "FETCH" με χρήστη τον kostas ο οποίος είναι ο χρήστης που συνδέεται με την PostgreSQL.
3. Τερματισμός διεργασιών που στον τίτλο τους έχουν τη λέξη-κλειδί "transaction" με χρήστη τον kostas.
4. Τερματισμός διεργασιών του προγράμματος sliderate.pl (perl script). Το sliderate.pl είναι το πρόγραμμα που διαβάζει τα δεδομένα από συγκεκριμένο ASCII αρχείο που θα ορίσουμε στη σχετική εντολή και τα στέλνει με συγκεκριμένο ρυθμό στο ρεύμα δεδομένων (το αντίστοιχο του πίνακα στις βάσεις δεδομένων). Ουσιαστικά όταν εκτελείται το sliderate.pl είναι ενεργό και το ρεύμα δεδομένων.
5. Εκτέλεση του αρχείου deleteVariousFiles.php. Το αρχείο αυτό που θα περιγραφεί στη συνέχεια φροντίζει για την διαγραφή όλων των βοηθητικών αρχείων της εφαρμογής ώστε η εκκαθάριση να ολοκληρωθεί.

Πιο συγκεκριμένα λοιπόν, το cleanup06.00 περιέχει τον παρακάτω κώδικα:

- Καθορισμός βιβλιοθηκών που θα χρειαστούν για την εκτέλεση του προγράμματος.
- Καθορισμός μεταβλητών.
- Εμφάνιση στο τερματικό (εάν εκτελείται από εκεί το πρόγραμμα) σχετικών πληροφοριών για το συγκεκριμένο πρόγραμμα.
- Έλεγχος ύπαρξης του αρχείου "stopCleanup.txt". Εάν το αρχείο αυτό υπάρχει στο αρχικό αυτό στάδιο εκτέλεσης του cleanup06.00, διαγράφεται ώστε να μη σταματήσει αμέσως η εκτέλεση του βρόχου που ακολουθεί και έχει ως μοναδικό κριτήριο επανάληψής του την μη ύπαρξη αυτού του αρχείου.

- Δημιουργία βρόχου ο οποίος θα επαναλαμβάνεται κάθε πέντε δευτερόλεπτα και για όσο χρονικό διάστημα δεν υπάρχει το αρχείο "stopCleanup.txt".

Μέσα στο βρόχο:

- Καταχώρηση στη μεταβλητή rawtime τύπου μεταβλητή χρόνου (time_t) της τιμής του χρόνου τη στιγμή που εκτελείται η εντολή. Πρόκειται για έναν αριθμό που δηλώνει τον χρόνο που έχει περάσει σε χιλιοστά του δευτερολέπτου από την 01/01/1970 και ώρα 00:00.
- Καταχώρηση στη μεταβλητή χρόνου timeinfo του αποτελέσματος εκτέλεσης της συνάρτησης χρόνου localtime με παράμετρο την τιμή της rawtime. Η timeinfo περιέχει τη χρονική στιγμή στη μορφή: ημέρα της εβδομάδας, μήνας, ημέρα του μήνα, ώρα, λεπτό, δευτερόλεπτο, έτος.
- Μετατροπή του περιεχομένου της timeinfo σε συμβολοσειρά και αποθήκευση αυτής στην stringtime τύπου συμβολοσειράς.
- Απομόνωση μόνο της ώρας, του λεπτού και του δευτερολέπτου σε νέα μεταβλητή τύπου συμβολοσειράς με όνομα onlytime.

Μετά από αυτές τις ενέργειες γίνεται ο πρώτος έλεγχος:

Εάν η μεταβλητή onlytime έχει την τιμή 06:00:00 ή την τιμή 06:00:01 ή 06:00:02 ή 06:00:03 ή 06:00:04 ή 06:00:05 τότε θα γίνει η εκκαθάριση. Πιο συγκεκριμένα σε αυτήν την περίπτωση γίνονται τα εξής:

Εάν το πρόγραμμα εκτελείται σε τερματικό, τότε εμφανίζονται σχετικές πληροφορίες σε αυτό αναφέροντας την ώρα και ότι θα γίνει εκκαθάριση.

Ακολουθούν οι τέσσερις εντολές τερματισμού των συγκεκριμένων διεργασιών που αναφέρθηκαν παραπάνω.

Μετά από παύση ενός δευτερολέπτου ώστε να έχει προλάβει ο Apache να επαναδομήσει τις αναγκαίες συνδέσεις που μόλις τερματίστηκαν, εκτελείται το deleteVariousFiles.php. Η εντολή "pkill -u daemon httpd" τερματίζει όλες τις συνδέσεις του Apache με χρήστη τον daemon. Έτσι τερματίζονται οι διάφορες συνδέσεις που έχουν ανοίξει τα PHP κατά την εκτέλεσή τους αλλά και διάφορες άλλες συνδέσεις Apache με χρήστη τον daemon που υπάρχουν ούτως ή άλλως. Οι συνδέσεις αυτές δημιουργούνται από την αρχή χωρίς να προκαλείται κάποιο πρόβλημα στη λειτουργία της εφαρμογής και του Linux ούτε άλλων προγραμμάτων. Για να διασφαλιστεί η σωστή εκτέλεση του deleteVariousFiles.php προηγείται μία παύση ενός δευτερολέπτου που χρησιμεύει και για άλλο λόγο στην εκτέλεση του cleanup0600, ο οποίος θα αναλυθεί παρακάτω.

Μετά, εμφανίζονται σχετικές πληροφορίες (εάν το πρόγραμμα εκτελείται από τερματικό) για την εκκαθάριση που μόλις ολοκληρώθηκε.

Μετά από παύση πέντε δευτερολέπτων, γίνεται έλεγχος για την ύπαρξη ή όχι του "stopCleanup.txt" και εάν υπάρχει, η μεταβλητή fexists7 παίρνει την τιμή 1. Στο σημείο αυτό τελειώνει ο κώδικας που περιέχεται μέσα στο βρόχο. Κατά την επανάληψη του βρόχου γίνεται έλεγχος εάν η fexists έχει την τιμή 0 και εάν όχι τότε σταματά η εκτέλεσή του.

Μετά το βρόχο σταματά και η εκτέλεση του προγράμματος.

Μέσα στο βρόχο υπάρχει εκτός των άλλων η εντολή sleep(5) η οποία αναστέλλει την εκτέλεση του προγράμματος για πέντε δευτερόλεπτα. Έτσι, ο βρόχος θεωρούμε ότι επαναλαμβάνεται κάθε πέντε δευτερόλεπτα. Στην πραγματικότητα όμως δεν επαναλαμβάνεται ο βρόχος ακριβώς μετά από το χρόνο αυτό διότι υπάρχει και ένας ελάχιστος χρόνος τον οποίο χρειάζεται ο υπολογιστής για την εκτέλεση του κώδικα που βρίσκεται εντός του βρόχου. Μετά από δοκιμές για μεγάλα χρονικά διαστήματα φαίνεται ότι σε κάποια στιγμή η επανάληψη έγινε μετά από 6 δευτερόλεπτα και πιο συγκεκριμένα ότι η μεταβλητή onlytime στην τιμή δευτερολέπτου είχε διαφορά 6 δευτ. Εάν τύχει η onlytime να πάρει την τιμή 05:59:59 και η επόμενη τιμή είναι μετά από 6 δευτερόλεπτα, τότε η onlytime θα μεταπηδήσει στην 06:00:05 και εάν αυτή η τιμή δεν είναι μέσα στις προϋποθέσεις για την εκτέλεση της εκκαθάρισης, δεν θα γίνει εκκαθάριση. Για αυτόν τον

λόγο στις προϋποθέσεις για αυτήν δεν είναι μόνο οι πέντε τιμές 06:00:00 – 06:00:04 αλλά και αυτή.

Στην περίπτωση που κάποια στιγμή η onlytime πάρει την τιμή 06:00:00 τότε το πιθανότερο είναι ότι στην επόμενη επανάληψη του βρόχου αυτή θα πάρει την τιμή 06:00:05 και θα γίνει πάλι εκκαθάριση. Για να αποφευχθεί αυτή η περίπτωση, όταν γίνεται εκκαθάριση, υπάρχει μία επιπλέον αναστολή της λειτουργίας του προγράμματος για 1 δευτερόλεπτο. Έτσι ακόμα και στην παραπάνω περίπτωση που στατιστικά θα συμβαίνει μία στις πέντε φορές, δεν θα γίνεται εκκαθάριση ξανά μετά από πέντε δευτ. καθώς η επόμενη τιμή της onlytime θα είναι υποχρεωτικά μετά την 06:00:05 (πιθανότερα η 06:00:06 και σπάνια η 06:00:07).

Με τον τρόπο λοιπόν αυτό διασφαλίζεται ότι θα γίνεται ακριβώς μία εκκαθάριση των αρχείων και διεργασιών της εφαρμογής κάθε 24 ώρες.

5.18 Αναλυτική περιγραφή λειτουργίας αρχείου με όνομα deleteVariousFiles.php

Το αρχείο αυτό στόχο έχει την διαγραφή όλων των δυναμικά δημιουργούμενων αρχείων της εφαρμογής είτε πρόκειται για βοηθητικά αρχεία είτε για τα XML με τις απαντήσεις όλων των ερωτημάτων. Εκτελείται δε κατά την εκκαθάριση της εφαρμογής από το cleanup0600.

Ο λόγος που τα αρχεία αυτά δεν διαγράφονται από το παραπάνω πρόγραμμα αλλά από κάποιο αρχείο PHP είναι ότι η C++ δεν έχει κάποιον εύκολο και απλό τρόπο να εντοπίσει εάν κάποιο αρχείο υπάρχει ή όχι. Προκειμένου να μην δημιουργούνται λάθη κατά την προσπάθεια διαγραφής του κάθε αρχείου πρέπει να προηγείται έλεγχος για την ύπαρξή του και εάν αυτό υπάρχει τότε να διαγράφεται. Η PHP διαθέτει την ενσωματωμένη συνάρτηση file_exists() η οποία επιστρέφει την τιμή true εάν υπάρχει το αρχείο και την τιμή false εάν αυτό δεν υπάρχει. Έτσι λοιπόν, προτιμήθηκε η διαγραφή των διαφόρων αρχείων από script γραμμένο στην PHP.

Το deleteVariousFiles.php λοιπόν εκτελεί τα παρακάτω:

- Αποθήκευση σε μεταβλητές του τύπου $xmlFile1# των ονομάτων των αρχείων “results#.xml”.
- Διαγραφή των αρχείων αποτελεσμάτων “results#.xml” εάν υπάρχουν.
- Αποθήκευση σε μεταβλητές του τύπου $xmlFile# των ονομάτων των αρχείων “zstopQuery#.xml”.
- Διαγραφή των αρχείων της ομάδας “zstopQuery#.xml” εάν υπάρχουν.
- Αποθήκευση σε μεταβλητές του τύπου $xmlFile2# των ονομάτων των αρχείων “zQuery#IsRunning.xml”.
- Διαγραφή των αρχείων της ομάδας “zQuery#IsRunning.xml” εάν υπάρχουν.
- Αποθήκευση σε μεταβλητές του τύπου $xmlFile3# των ονομάτων των αρχείων “deleteFetch#.xml”.
- Διαγραφή των αρχείων της ομάδας “deleteFetch#.xml” εάν υπάρχουν.
- Αποθήκευση σε μεταβλητές του τύπου $xmlFile4# των ονομάτων των αρχείων “zq#dbconnID.log”.
- Διαγραφή των αρχείων της ομάδας “zq#dbconnID.log” εάν υπάρχουν.
- Αποθήκευση σε μεταβλητές του τύπου $xmlFile5# των ονομάτων των αρχείων “zq#httpdID.log”.
- Διαγραφή των αρχείων της ομάδας “zq#httpdID.log” εάν υπάρχουν.
- Αποθήκευση στη μεταβλητή $xmlFileRmSR του ονόματος του αρχείου “zstreamRunning.xml”.
- Διαγραφή του αρχείου “zstreamRunning.xml” εάν υπάρχει.
- Αποθήκευση στη μεταβλητή $xmlFileRmSR2 του ονόματος του αρχείου “zstreamRunning2.xml”.

- Διαγραφή του αρχείου "zstreamRunning2.xml" εάν υπάρχει.

Στο σημείο αυτό τελειώνει ο κώδικας του deleteVariousFiles.php.

5.19 Αναλυτική περιγραφή προγραμμάτων διακοπής των τεσσάρων προγραμμάτων της C++

Όπως έχει προαναφερθεί, η διακοπή καθενός από τα προγράμματα killQueryNoResults, storeProcessIDs, killFetch και cleanup0600 γίνεται δημιουργώντας ένα συγκεκριμένο βοηθητικό αρχείο για κάθε πρόγραμμα στο φάκελο όπου αυτά εκτελούνται. Πιο συγκεκριμένα, για τη διακοπή του killQueryNoResults χρειάζεται η δημιουργία του βοηθητικού αρχείου "stopKillQueries.txt", για τη διακοπή του storeProcessIDs χρειάζεται η δημιουργία του αρχείου "stopStoreIDs.txt", για τη διακοπή του killFetch χρειάζεται η δημιουργία του "stopkillFetch.txt" και για τη διακοπή του cleanup0600 χρειάζεται η δημιουργία του "stopcleanup.txt".

Για τη δημιουργία κάθε βοηθητικού αρχείου διακοπής των παραπάνω προγραμμάτων της C++, έχουν δημιουργηθεί τέσσερα ειδικά προγράμματα. Κάθε πρόγραμμα, όταν εκτελείται από τον διαχειριστή του συστήματος, κάνει μόνο μία ενέργεια, δημιουργεί το βοηθητικό αρχείο διακοπής του αντίστοιχου προγράμματος. Μέσα στο αρχείο αυτό αποθηκεύεται ως κείμενο το όνομα του αρχείου ώστε αυτό να μην είναι κενό. Τα προγράμματα διακοπής, έχουν αντίστοιχο όνομα με το πρόγραμμα το οποίο θα σταματήσουν ως εξής:

Το killQueryNoResultsStop θα σταματήσει το killQueryNoResults,
το StoreProcessIDsStop θα σταματήσει το storeProcessIDs,
το KillFetchStop θα σταματήσει το killFetch και
το CleanupStop θα σταματήσει το cleanup0600.

Οι προτάσεις που περιέχει κάθε ένα από αυτά τα προγράμματα έχουν ως εξής:

- Ορίζονται οι αναγκαίες βιβλιοθήκες.
- Μέσα στη βασική συνάρτηση του κάθε προγράμματος δημιουργείται στιγμιότυπο της τάξης ofstream με όνομα σχετικό με το επιθυμητό αρχείο προς δημιουργία και καθορισμό του ονόματος του αρχείου προς δημιουργία-άνοιγμα με σχετική συμβολοσειρά.
- Προστίθεται κείμενο μέσα στο αρχείο (το κείμενο είναι απλά το όνομα του ίδιου του αρχείου).
- Τερματίζεται η επεξεργασία του αρχείου κλείνοντάς το.

Ο κώδικας του κάθε προγράμματος στο σημείο αυτό τελειώνει.

5.20 Αναλυτική περιγραφή λειτουργίας αρχείου με όνομα sliderate.pl

Πρόκειται για script γραμμένο στη γλώσσα προγραμματισμού perl (Practical Extraction and Report Language).

Σκοπός και στόχος του προγράμματος αυτού είναι να αποστέλλει δεδομένα σε συγκεκριμένη διεύθυνση, σε συγκεκριμένο port, σε συγκεκριμένη οντότητα του ΣΔΡΔ και με συγκεκριμένο ρυθμό. Τα στοιχεία αυτά δίδονται ως παράμετροι κατά την εκτέλεση του script μέσα από τερματικό του Linux.

Το sliderate.pl λοιπόν αποστέλλει τα δεδομένα που έχει συγκεκριμένο ASCII αρχείο ("points5_wgs84.log") στον csvwrapper του TelegraphCQ ώστε να αποτελέσουν το ρεύμα δεδομένων για αυτήν εφαρμογή.

Η εντολή εκκίνησης του ρεύματος δεδομένων λοιπόν είναι η εξής:

```
cat points5_wgs84.log | ./sliderate.pl localhost 5533 csvwrapper,network.vehicles 15 5
```

και εκτελείται σε τερματικό του Linux.

Με την εντολή cat συνδέεται το πρώτο αρχείο με το δεύτερο. Το sliderate.pl δέχεται στις παραμέτρους του κατά σειρά τα εξής:

Η localhost, είναι η διεύθυνση στην οποία αποστέλλονται τα δεδομένα που περιέχει το ASCII αρχείο "points5_wgs84.log".

Το 5533 είναι το port στο οποίο αποστέλλονται τα δεδομένα αυτά.

Το csvwrapper,network.vehicles, δηλώνει τον wrapper που παραλαμβάνει τα δεδομένα και το ρεύμα στο οποίο αυτός θα τα στείλει με τη σειρά του.

Το 15 είναι το πλήθος των γραμμών της πηγής της πληροφορίας που θα αποστέλλονται κάθε φορά.

Το 5 είναι ο ρυθμός επανάληψης της αποστολής σε δευτερόλεπτα.

Συνεπώς κάθε 5 δευτερόλεπτα αποστέλλονται 15 καταχωρήσεις από το αρχείο "points5_wgs84.log" μέσω του sliderate.pl στον csvwrapper ο οποίος με τη σειρά του τα αποστέλλει στο ρεύμα network.vehicles. Φυσικά στην περίπτωση αποστολής των δεδομένων στο network.vehicles2 η εντολή είναι παρόμοια (αλλάζει μόνο το ρεύμα που θα παραλαμβάνει τα δεδομένα).

Πιο συγκεκριμένα, μέσα στον κώδικα του sliderate.pl γίνονται κατά σειρά τα εξής:

Καθορίζεται μεταβλητή που αποθηκεύει συμβολοσειρά με πληροφορίες για τη χρήση του συγκεκριμένου προγράμματος. Οι πληροφορίες είναι οι εξής:

```
source.pl host port wrappername number_of_items frequency
 where
   host = The name or IP address of the TelegraphCQ host to connect to
   port = The port number on which the TCQ host is listening
   wrappername = The name of the TCQ wrapper to push data into
   number_of_items = Amount of items to send every time.
   frequency = Number of seconds between two consecutive dispatches
```

Γίνεται έλεγχος του πλήθους των παραμέτρων που δόθηκαν κατά την εκκίνηση του προγράμματος και εάν αυτές είναι λιγότερες από 5, εμφανίζεται σχετικό μήνυμα στο τερματικό συμπεριλαμβανομένων των παραπάνω πληροφοριών και τερματίζεται η λειτουργία του προγράμματος.

Γίνεται έλεγχος στις παραμέτρους για την ύπαρξη της "-h" ή "--help". Στην περίπτωση που υπάρχει τουλάχιστον μία από αυτές τις λέξεις-κλειδιά, εμφανίζεται σχετικό μήνυμα με πληροφορίες και τερματίζεται η εκτέλεση του script.

Εάν η εκτέλεση του script περάσει από αυτούς τους δύο ελέγχους χωρίς να διακοπεί, αυτή συνεχίζεται με την αποθήκευση των τιμών των παραμέτρων σε συγκεκριμένες μεταβλητές:

$host = $ARGV[0]; (αποθήκευση της πρώτης παραμέτρου στη μεταβλητή $host), $port = $ARGV[1]; (αποθήκευση της δεύτερης παραμέτρου στην $port), $sourcename = $ARGV[2]; (αποθήκευση της τρίτης παραμέτρου στην $sourcename), $numitems= $ARGV[3]; (αποθήκευση της τέταρτης παραμέτρου στην $numitems) και τέλος $slot= $ARGV[4]; (αποθήκευση της πέμπτης παραμέτρου στην $slot).

Στη συνέχεια προετοιμάζεται η σύνδεση του script με τη συγκεκριμένη διεύθυνση host και το συγκεκριμένο port. Μετά δημιουργείται μια υποδοχή (socket) για τη σύνδεση με το συγκεκριμένο υπολογιστή και την συγκεκριμένη θέση. Με την εντολή connect()

γίνεται η επιθυμητή σύνδεση. Ακολουθεί η αποθήκευση στη μεταβλητή $namestr ειδικά διαμορφωμένης συμβολοσειράς με πληροφορίες σχετικά με το πλήθος των χαρακτήρων και την τιμή της τρίτης παραμέτρου που αναφέρεται στην οντότητα που θα δεχθεί τις πληροφορίες που αποστέλλουμε. Στη συνέχεια, οι πληροφορίες αυτές αποστέλλονται στον παραλήπτη της σύνδεσης (SOCK).

Μετά, η μεταβλητή starttime αποθηκεύει την τρέχουσα χρονική στιγμή ως το πλήθος των δευτερολέπτων που έχουν περάσει από την 01/01/1970 00:00:00 έως την στιγμή εκτέλεσης της συνάρτησης time.

Μετά ορίζεται η μεταβλητή $count και παίρνει την τιμή 0.

Ακολουθεί ο απαραίτητος βρόχος ο οποίος συντελεί στην αποστολή των γραμμών του ASCII αρχείου με τα δεδομένα του ρεύματος, μία μία στον επιθυμητό προορισμό. Ο STDIN είναι ένας τελεστής (operator) εισερχόμενων στο script γραμμών που μπορεί να προέρχονται είτε από πληκτρολόγηση από τον χρήστη είτε από συνδεδεμένο εξωτερικό αρχείο όπως στην περίπτωσή μας.

Μέσα στο βρόχο, νέα μεταβλητή με όνομα $str αποθηκεύει την τιμή της $_. Η $_ είναι η προκαθορισμένη μεταβλητή για είσοδο δεδομένων και αναζήτηση.

Ακολουθεί η αποστολή των πληροφοριών της πρώτης γραμμής του εξωτερικού αρχείου στη σύνδεση που έχει προκαθοριστεί (SOCK).

Ο μετρητής count αυξάνεται κατά 1. Ακολουθεί έλεγχος εάν η $count έχει τιμές πολλαπλάσιες του πλήθους των σειρών που επιθυμούμε να στέλνουμε κάθε φορά. Δηλαδή, στην περίπτωσή μας όπου θέλουμε 15 καταχωρήσεις-σειρές να αποστέλλονται κάθε φορά, ο έλεγχος αυτός ισχύει όταν ο μετρητής φτάσει στην τιμή 15 (οπότε η διαίρεση με 15 αφήνει υπόλοιπο μηδέν). Μετά από αυτόν τον έλεγχο (και τον κώδικα που εκτελείται στην περίπτωση που αυτός ισχύει), ο βρόχος τελειώνει και συνεπώς επανεκτελείται. Ο βρόχος εκτελείται για όσο ο τελεστής STDIN έχει πληροφορίες να αποστείλει.

Μετά λοιπόν από τις πρώτες 15 επαναλήψεις του βρόχου, ο $count έχει την τιμή 15 και εκτελείται πλέον ο σχετικός κώδικας. Μέσα λοιπόν στα σχετικά άγκιστρα υπάρχει ένας άλλος έλεγχος:

Εάν ο $count έχει τιμή πολλαπλάσια του πενταπλασίου του πλήθους των καταχωρήσεων που επιθυμούμε να στέλνουμε κάθε φορά (εάν δηλαδή έχει την τιμή 75, 150, 225 κ.ο.κ.), τότε τυπώνεται στο τερματικό η τρέχουσα σειρά-καταχώρηση του εξωτερικού ASCII αρχείου (στην περίπτωσή μας του points5_wgs84.log).

Μετά, καθορίζεται νέα μεταβλητή ($currtime) που λαμβάνει την τιμή της τρέχουσας χρονικής στιγμής (με ακρίβεια δευτερολέπτου).

Μετά καθορίζεται μία ακόμη μεταβλητή ($diff) ως η χρονική διαφορά της $starttime μείον την $currtime προσαυξημένη κατά το χρονικό διάστημα που έχει οριστεί με παράμετρο ότι θα υπάρχει ανάμεσα σε δύο διαδοχικές αποστολές των ομάδων των καταχωρήσεων. Με τον τρόπο αυτό μειώνεται στην πραγματική της τιμή η χρονική διαφορά μεταξύ δύο συνεχόμενων αποστολών καθώς υπάρχει και κάποιο χρονικό διάστημα που καταναλώνεται από το script για την εκτέλεση των εντολών ανάμεσα στην $starttime και την $currtime.

Ακολουθεί έλεγχος εάν η diff έχει θετική τιμή και εάν ισχύει (στην περίπτωσή μας αναμένεται να ισχύει σχεδόν πάντα καθώς είναι πραγματικά πολύ δύσκολο να χρειαστούν παραπάνω από 4 ή 5 δευτερόλεπτα για την αποστολή 15 καταχωρήσεων από το ASCII αρχείο στον wrapper), τότε η λειτουργία του script αναστέλλεται για όσο χρονικό διάστημα αναφέρει η $diff.

Τέλος, η $starttime παίρνει την καινούργια της τιμή ώστε να επανεκτελεστεί ο βρόχος.

Τελειώνει εδώ ο κώδικας που εκτελείται εάν ο $count έχει τιμή πολλαπλάσια του $numitems και ο βρόχος επανεκτελείται.

Όταν τελειώσει η ανάγνωση των πληροφοριών από το εξωτερικό ASCII αρχείο, κλείνει η σύνδεση με την PostgreSQL και τον wrapper αυτής και τερματίζεται η εκτέλεση του sliderate.pl.

Στο σημείο αυτό τελειώνει το πέμπτο κεφάλαιο της παρούσας εργασίας και ακολουθεί κεφάλαιο με θέμα την παρουσίαση της εφαρμογής σε λειτουργία.

Κεφάλαιο 6

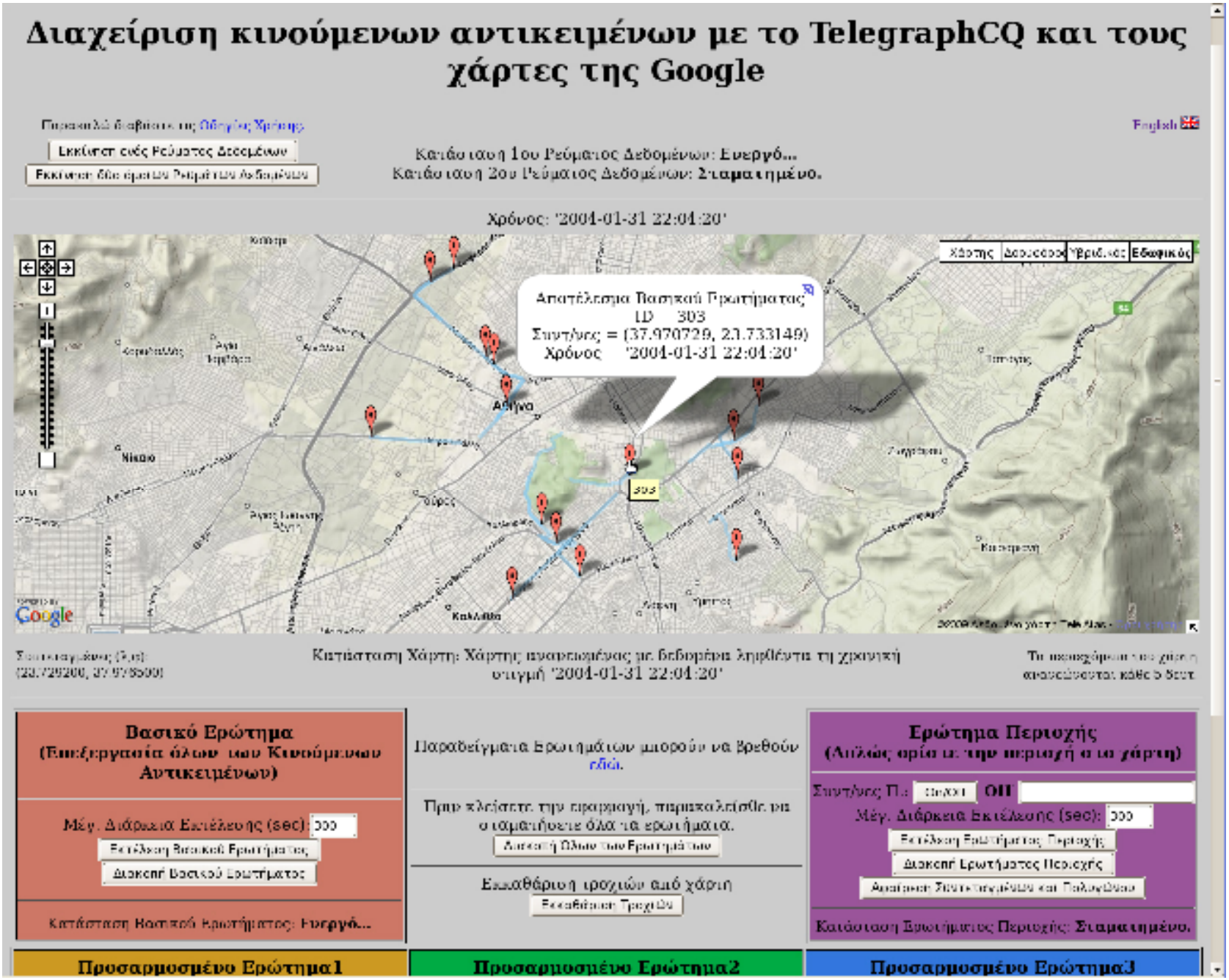
Η εφαρμογή σε λειτουργία

6.1 Γενικά – Εισαγωγή

Στο κεφάλαιο αυτό παραθέτουμε εικόνες από την εφαρμογή σε πλήρη λειτουργία με τις απαραίτητες πάντα επεξηγήσεις για το τι είναι αυτό που βλέπουμε στην κάθε εικόνα και τι ενέργειες έχει κάνει ο χρήστης για να εμφανιστεί το σχετικό αποτέλεσμα.

6.2 Πρώτο Ρεύμα Δεδομένων και Βασικό Ερώτημα Ενεργά

Παρακάτω παρατίθεται η εικόνα 6.1 και ακολουθούν σχετικές επεξηγήσεις:

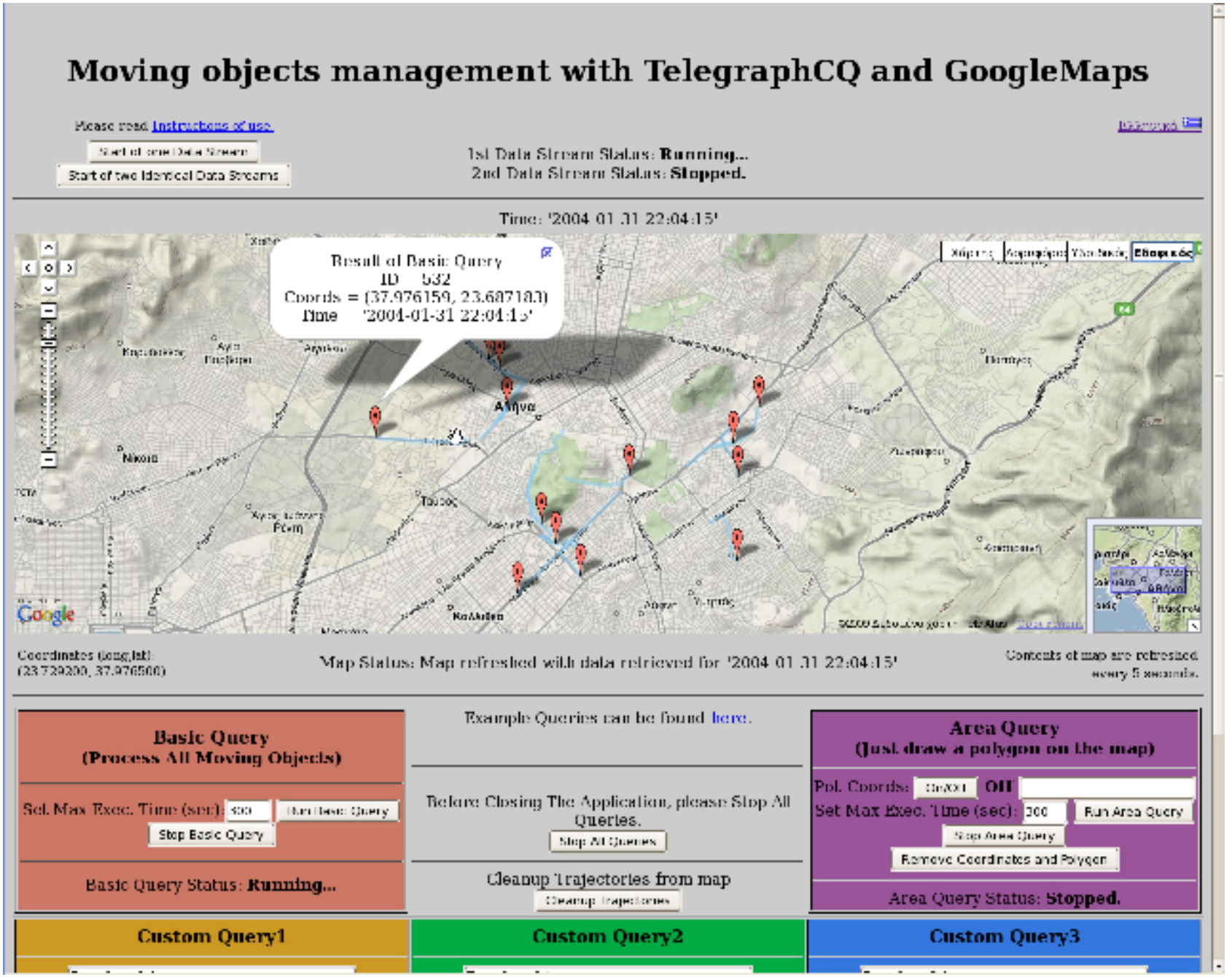


Εικόνα 6.1: Πρώτο Ρεύμα Δεδομένων και Βασικό Ερώτημα ενεργά (Ελληνική έκδοση)

Κατά τη στιγμή λήψης της εικόνας 6.1 ήταν ενεργό το Πρώτο Ρεύμα Δεδομένων (έχοντας πατήσει το πλαίσιο-κουμπί "Εκκίνηση Ενός Ρεύματος Δεδομένων") και το Βασικό Ερώτημα ήταν και αυτό ενεργό. Μόλις ξεκινά ένα οποιοδήποτε ερώτημα, αρχίζουν να δημιουργούνται από την JavaScript οι τροχιές των κινούμενων αντικειμένων από τις διαδοχικές θέσεις όπου αυτά εμφανίζονται. Οι γαλάζιες γραμμές είναι οι τροχιές των αντικειμένων. Οι απαντήσεις στο Βασικό Ερώτημα, όπως έχει προαναφερθεί, εμφανίζονται στο χάρτη με σημεία-σημάδια (markers) με ένα ανοιχτό κόκκινο χρώμα αντίστοιχο με το χρωματισμό του πλαισίου του Βασικού Ερωτήματος. Τα σημεία-σημάδια αντιπροσωπεύουν προφανώς τα κινούμενα αντικείμενα. Το Βασικό Ερώτημα επιστρέφει όλα τα κινούμενα αντικείμενα που διαχειρίζεται η εφαρμογή. Τη στιγμή λήψης της εικόνας 6.1 ο χρήστης έχει κάνει κλικ επάνω σε ένα από τα σημεία-σημάδια (κινούμενο αντικείμενο) και έχει εμφανιστεί σχετικό παράθυρο με πληροφορίες για το αντικείμενο αυτό.

Τα υπόλοιπα ερωτήματα είναι σταματημένα-ανενεργά.

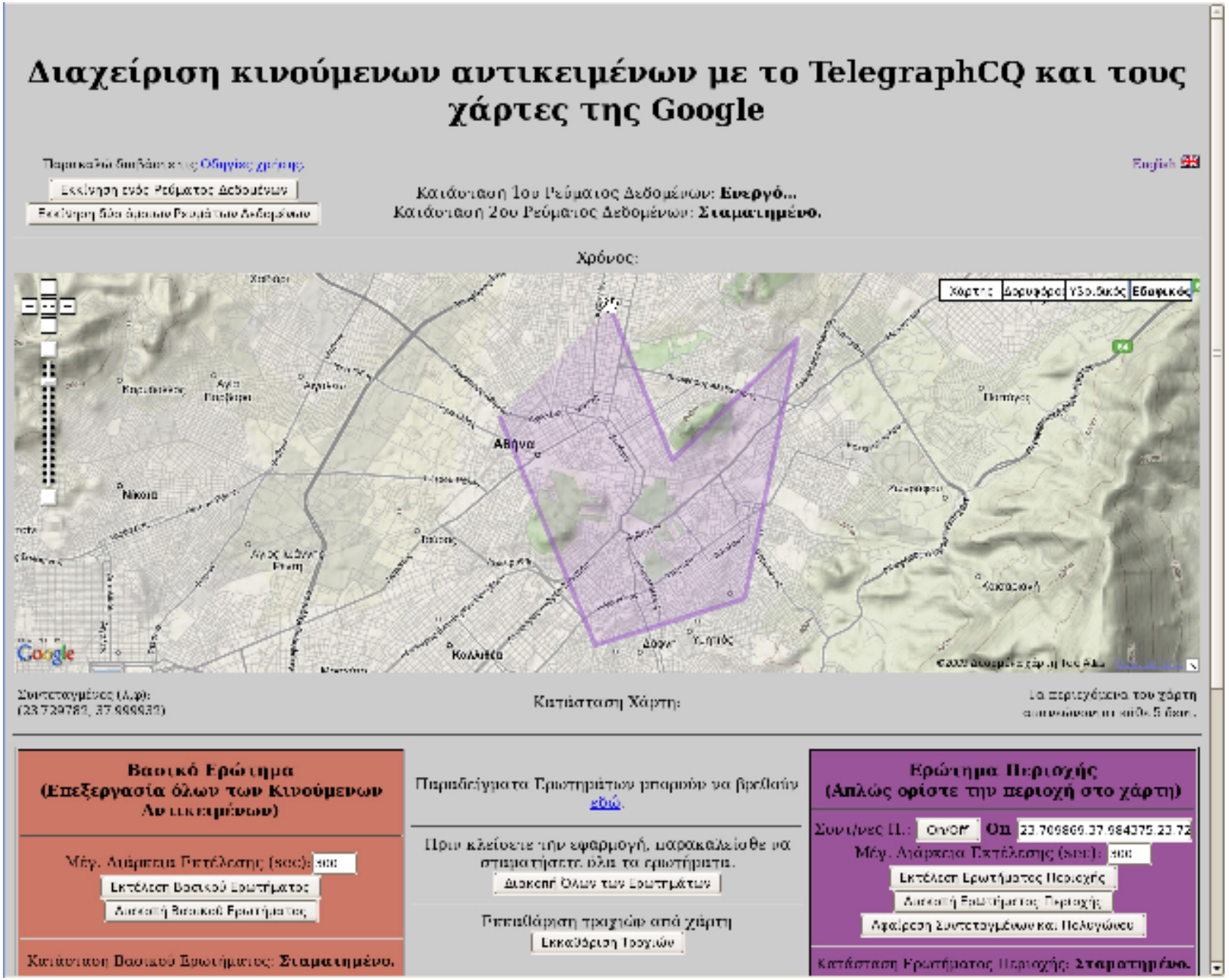
Η εικόνα 6.2, παρακάτω, δείχνει αντίστοιχη περίπτωση στην Αγγλική έκδοση της εφαρμογής:



Εικόνα 6.2: Πρώτο Ρεύμα Δεδομένων και Βασικό Ερώτημα ενεργά (Αγγλική έκδοση)

6.3 Σχεδίαση πολυγώνου για Ερώτημα Περιοχής

Παρακάτω παρατίθεται η σχετική εικόνα 6.3 και ακολουθούν επεξηγήσεις:



Εικόνα 6.3: Σχεδίαση πολυγώνου για Ερώτημα Περιοχής (Ελληνική έκδοση)

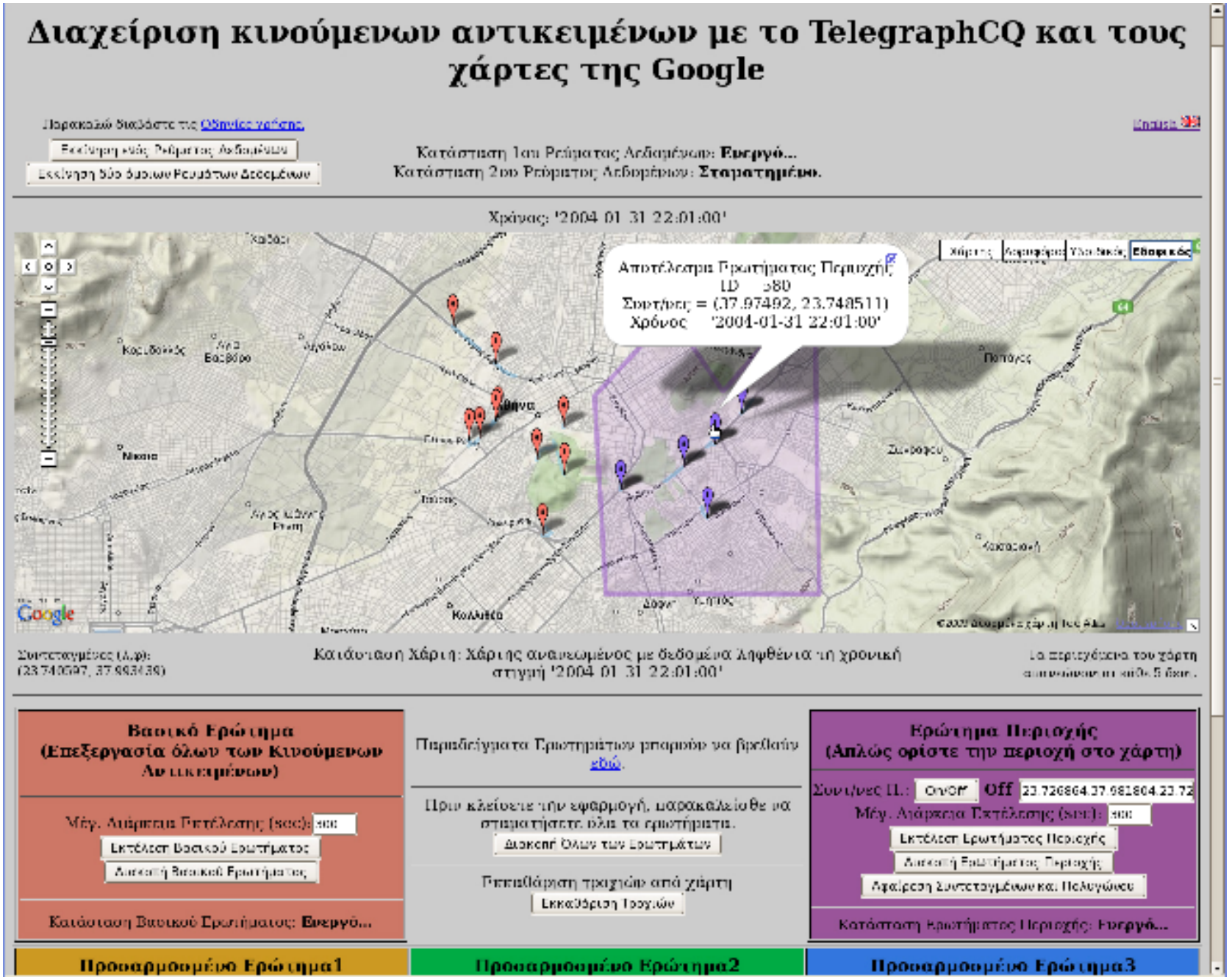
Στην εικόνα 6.3, ο χρήστης έχει ενεργοποιήσει πρώτα την δημιουργία πολυγώνου πατώντας μια φορά το κουμπί "On/Off" επιβεβαιώνοντας ότι αυτή είναι πλέον ενεργή από την εμφάνιση της κατάστασης δημιουργίας αυτού που από "Off" θα αλλάξει σε "On". Ακολούθως, ο χρήστης έχει κάνει κλικ σε διάφορα σημεία στο χάρτη δημιουργώντας το επιθυμητό πολύγωνο. Ειδικότερα, το πολύγωνο της εικόνας 6.3, τη στιγμή λήψη αυτής, αποτελείται από 6 κορυφές και πέντε πλευρές καθώς το πολύγωνο είναι ανοιχτό και ο χρήστης μπορεί να προσθέσει και άλλα σημεία. Οι συντεταγμένες των κορυφών έχουν καταχωρηθεί αυτόματα στο σχετικό πεδίο στο πλαίσιο του Ερωτήματος Περιοχής. Όταν ο χρήστης ζητήσει την εκτέλεση του ερωτήματος, το πολύγωνο θα κλείσει αυτόματα προσθέτοντας την τελευταία πλευρά που κάθε φορά θα λείπει.

Η δημιουργία πολυγώνου είναι ανεξάρτητη από το εάν είναι ενεργά ή όχι τα ρεύματα δεδομένων. Εάν δεν είναι ενεργό το πρώτο ρεύμα δεδομένων και ο χρήστης ζητήσει την εκτέλεση του Ερωτήματος Περιοχής, το ερώτημα δεν θα εκτελεστεί αλλά το πολύγωνο θα κλείσει ανεξαρτήτως της εκτέλεσης των ρευμάτων. Επίσης όταν ο χρήστης πατήσει το κουμπί εκτέλεσης του Ερωτήματος Περιοχής, η δημιουργία πολυγώνου απενεργοποιείται αυτόματα ώστε να μην προστεθεί εκ παραδρομής κάποιο επιπλέον ευθύγραμμο τμήμα στο δημιουργηθέν πολύγωνο.

Αντίστοιχη εικόνα από την Αγγλική έκδοση της εφαρμογής δεν θεωρείται σκόπιμο να ληφθεί καθώς δεν αλλάζει στο παραμικρό η παραπάνω διαδικασία και περιγραφή, παρά μόνο αλλάζουν τα εμφανιζόμενα κείμενα στην ιστοσελίδα.

6.4 Βασικό Ερώτημα και Ερώτημα Περιοχής Ενεργά

Παρακάτω παρατίθεται η σχετική για την περίπτωση αυτή εικόνα 6.4 και ακολουθούν αντίστοιχες επεξηγήσεις:



Εικόνα 6.4: Πρώτο Ρεύμα Δεδομένων, Βασικό Ερώτημα και Ερώτημα Περιοχής ενεργά (Ελληνική έκδοση)

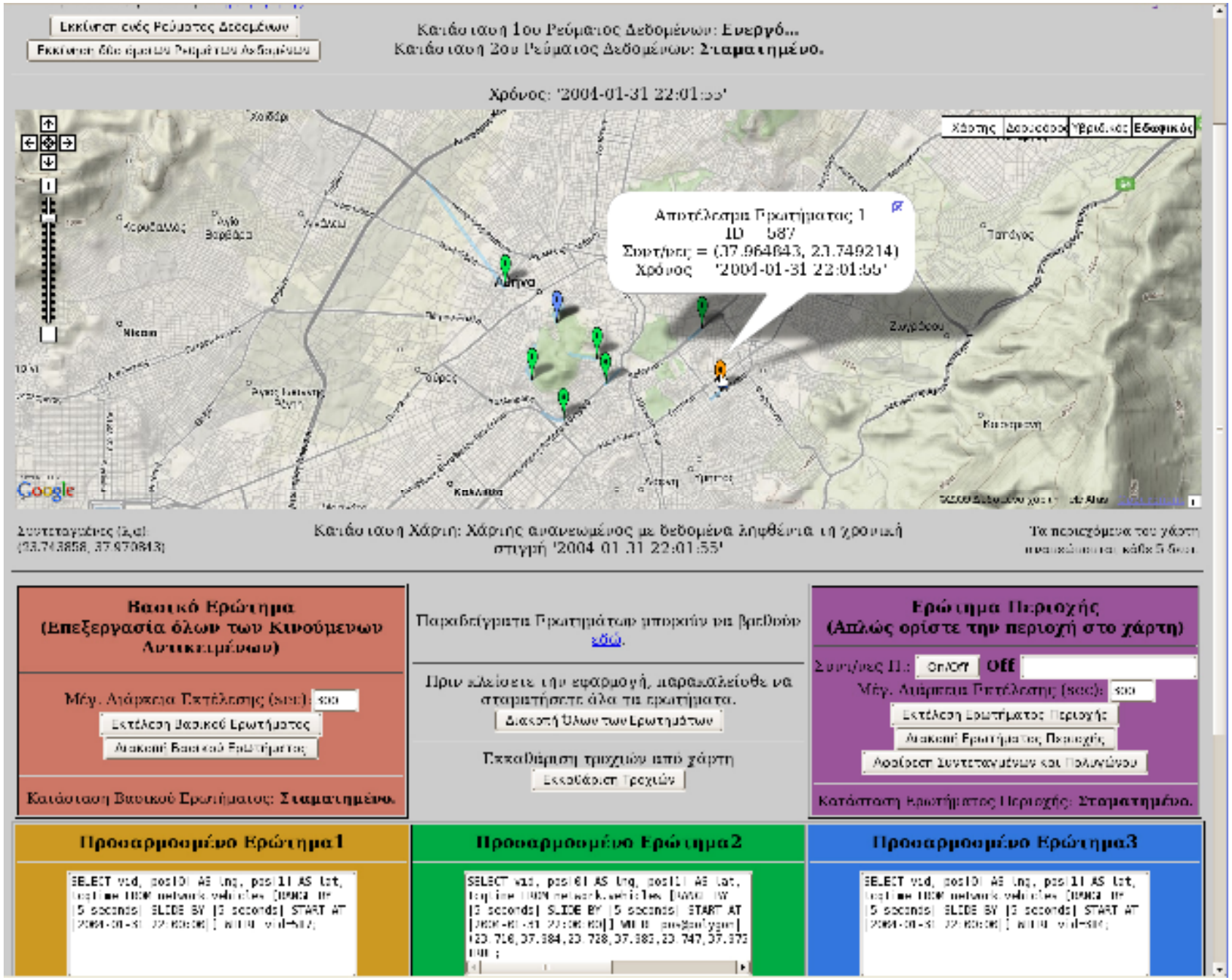
Στην εικόνα 6.4, ο χρήστης έχει δημιουργήσει το επιθυμητό πολύγωνο, έχει ενεργοποιήσει την εκτέλεση του πρώτου ρεύματος δεδομένων, του Βασικού Ερωτήματος και του Ερωτήματος Περιοχής. Επίσης, ο χρήστης έχει κάνει κλικ επάνω σε ένα από τα αποτελέσματα του Ερωτήματος Περιοχής και έχει εμφανιστεί παράθυρο με σχετικές πληροφορίες για το κινούμενο αντικείμενο.

Παρατηρήστε ότι αν και το Βασικό Ερώτημα επιστρέφει όλα τα κινούμενα αντικείμενα, μερικά από αυτά (5 στην παραπάνω περίπτωση) δεν εμφανίζονται καθώς αποτελούν απαντήσεις και στο Ερώτημα Περιοχής το οποίο έχει προγραμματιστεί να εμφανίζει από πάνω από το Βασικό Ερώτημα τα αποτελέσματά του. Εάν συνέβαινε το αντίστροφο, πολύ απλά, δεν θα εμφανιζόντουσαν αποτελέσματα του Ερωτήματος Περιοχής παρά μόνο στιγμιαία κατά την δημιουργία κάθε ομάδας αποτελεσμάτων.

Αντίστοιχη εικόνα από την Αγγλική έκδοση της εφαρμογής δεν θεωρείται σκόπιμο να ληφθεί καθώς θα είναι ένας συνδυασμός των εικόνων 6.2 και 6.4.

6.5 Το πρώτο Ρεύμα Δεδομένων και τα τρία Προσαρμοσμένα Ερωτήματα Ενεργά

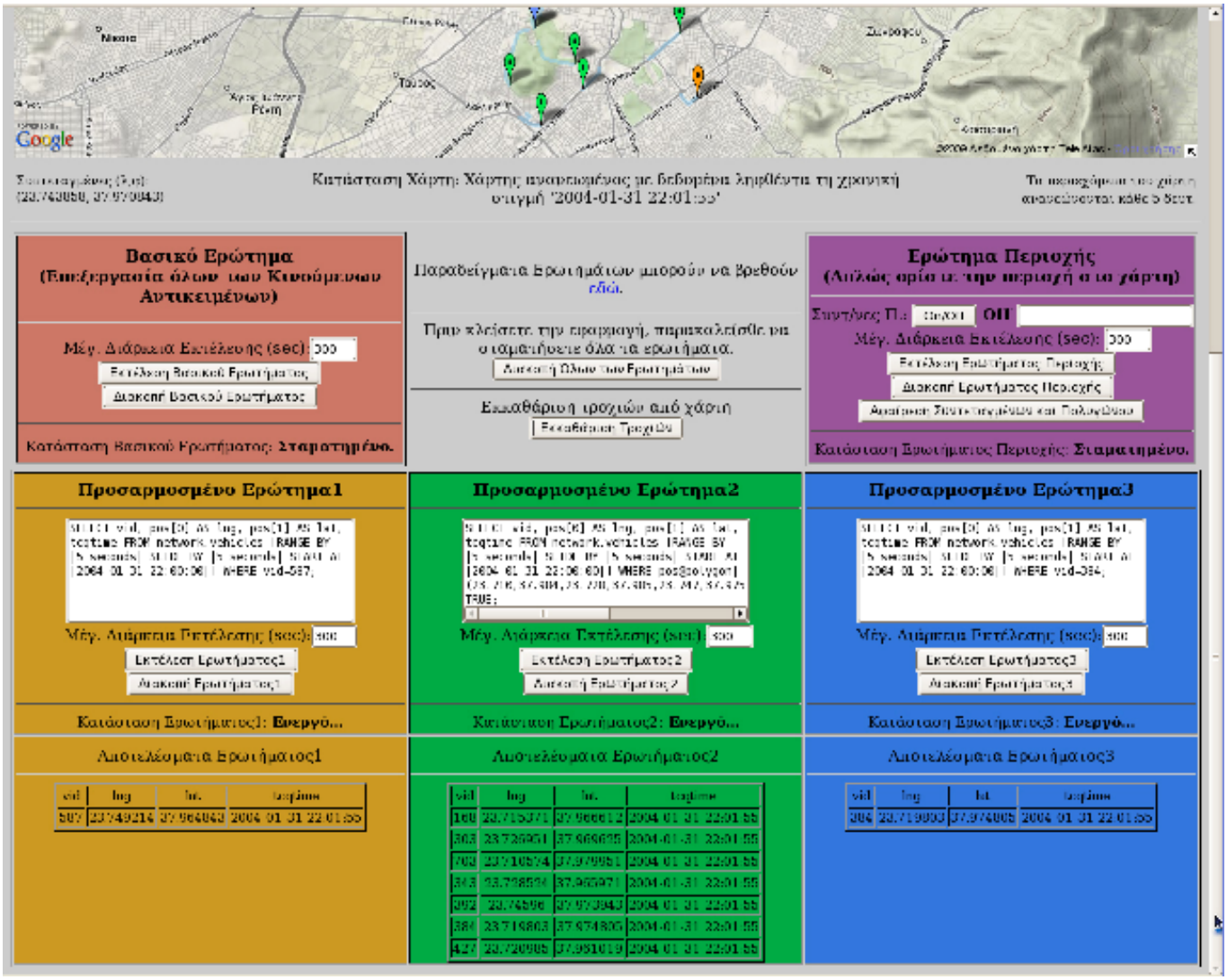
Παρακάτω παρατίθεται η σχετική για την περίπτωση αυτή εικόνα 6.5 και ακολουθούν αντίστοιχες επεξηγήσεις:



Εικόνα 6.5: Πρώτο Ρεύμα Δεδομένων και τρία Προσαρμοσμένα Ερωτήματα ενεργά – αποτελέσματα σε χάρτη

Στην εικόνα 6.5, ο χρήστης έχει ενεργοποιήσει το πρώτο Ρεύμα Δεδομένων και εκτελεί τα Προσαρμοσμένα Ερωτήματα 1,2 και 3 με τα Παραδείγματα 1, 2 και 3 από τη σχετική ιστοσελίδα στο κάθε ένα από αυτά αντίστοιχα.

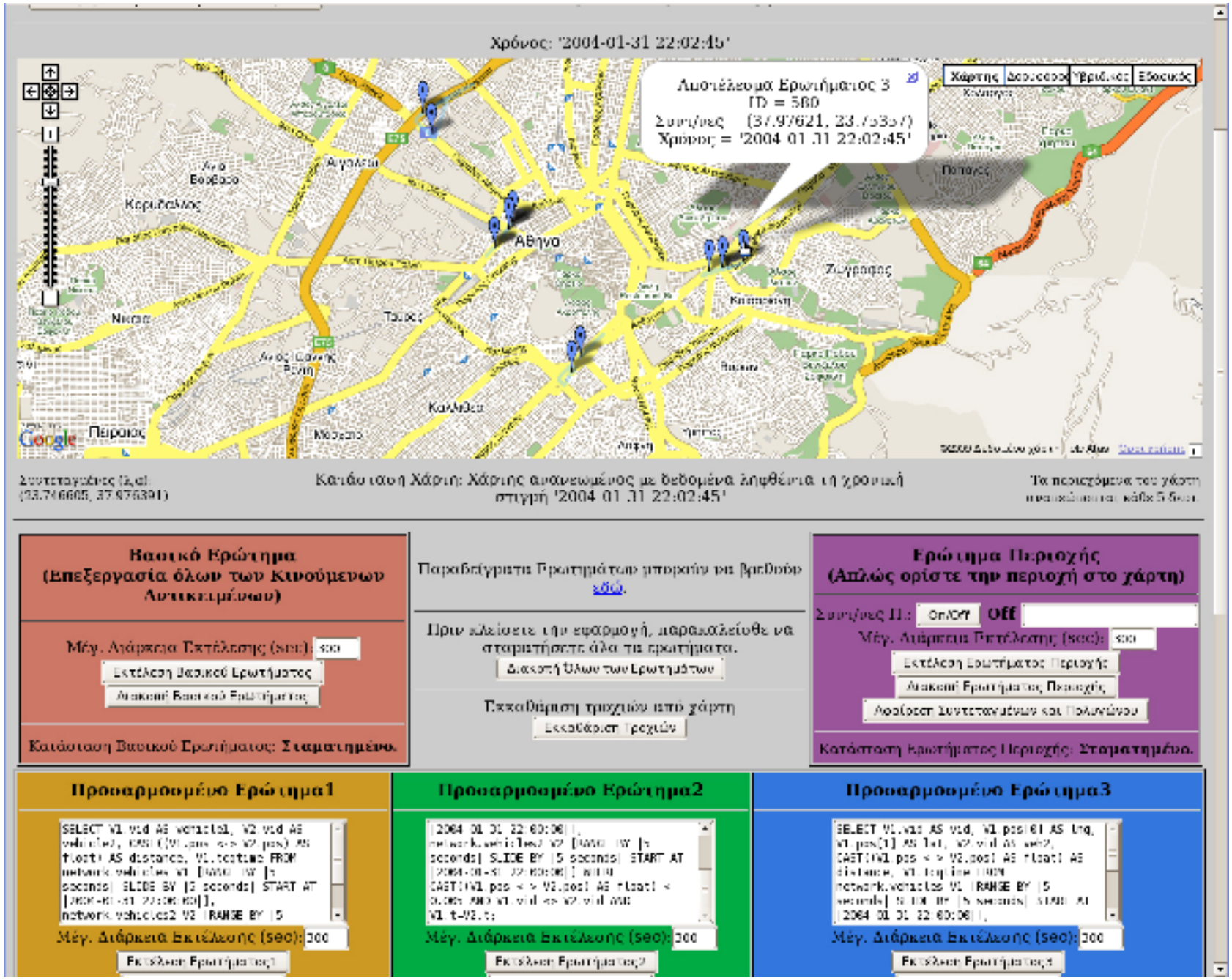
Τα αποτελέσματα εμφανίζονται και σε πίνακες κάτω από τα πλαίσια των προσαρμοσμένων ερωτημάτων όπως φαίνεται παρακάτω.



Εικόνα 6.6: Πρώτο Ρεύμα Δεδομένων και τρία Προσαρμοσμένα Ερωτήματα ενεργά – Αποτελέσματα σε πίνακες

6.6 Δύο όμοια Ρεύματα Δεδομένων και τα τρία Προσαρμοσμένα Ερωτήματα Ενεργά

Παρακάτω παρατίθεται η σχετική για την περίπτωση αυτή εικόνα 6.7 και ακολουθούν αντίστοιχες επεξηγήσεις:

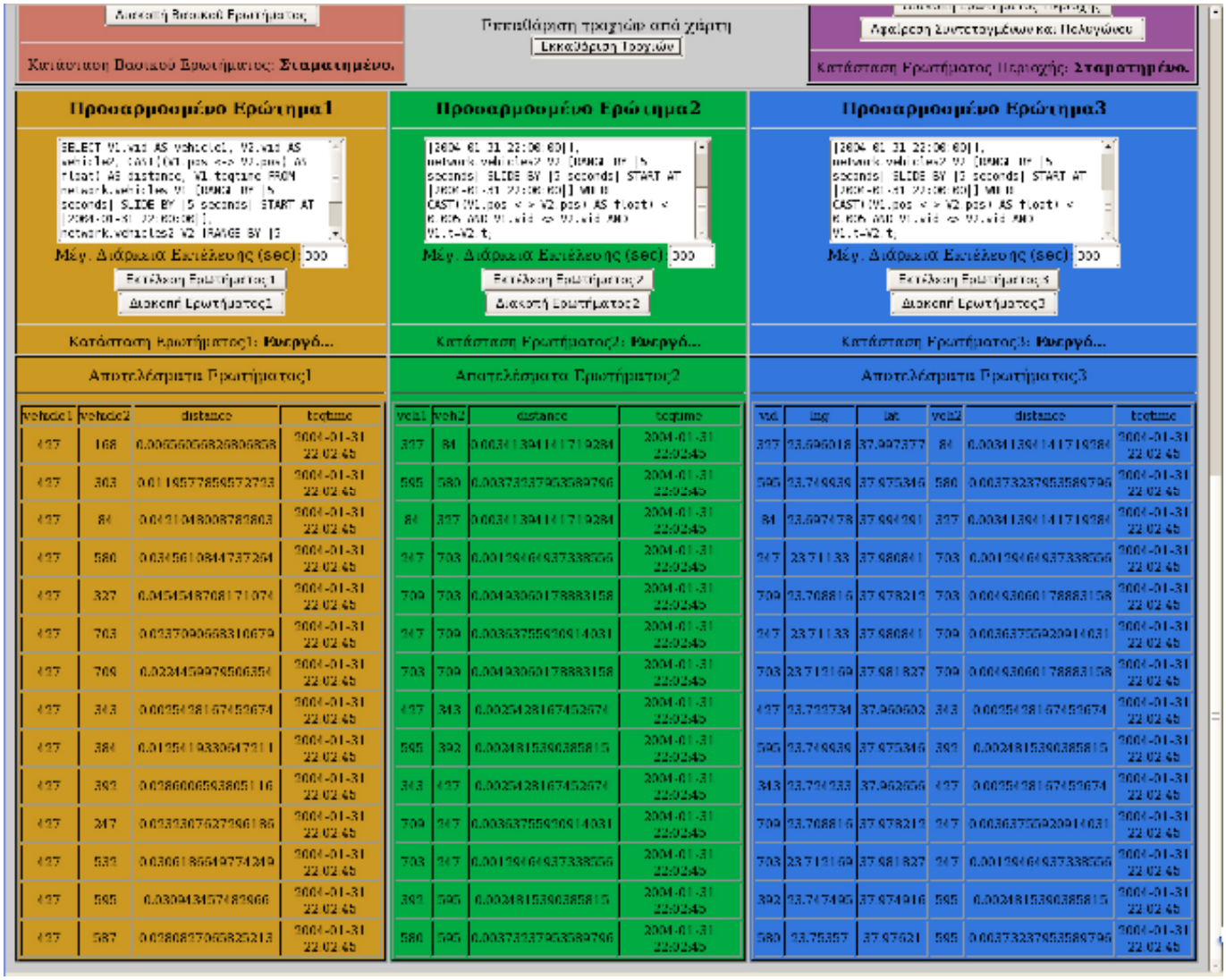


Εικόνα 6.7: Δύο όμοια Ρεύματα Δεδομένων και τρία Προσαρμοσμένα Ερωτήματα ενεργά – αποτελέσματα σε χάρτη

Στην εικόνα 6.7, ο χρήστης έχει ενεργοποιήσει και τα δύο Ρεύματα Δεδομένων και εκτελεί τα Προσαρμοσμένα Ερωτήματα 1,2 και 3 με τα Παραδείγματα 8, 9 και 10 από τη σχετική ιστοσελίδα στο κάθε ένα από αυτά αντίστοιχα. Επίσης, έχει επιλέξει την εμφάνιση του οδικού χάρτη της Google.

Τα Προσαρμοσμένα Ερωτήματα 1 και 2 δεν επιστρέφουν αποτελέσματα στο χάρτη (βλέπε περιγραφή Παραδειγμάτων 8 και 9 αντίστοιχα στην παράγραφο 5.4) και αυτά εμφανίζονται μόνο σε μορφή πίνακα στις σχετικές θέσεις στην ιστοσελίδα. Το Προσαρμοσμένο Ερώτημα 3 το οποίο είναι εδώ το Παράδειγμα 10, επιστρέφει αποτελέσματα και σε πίνακα και στο χάρτη. Έτσι, η εικόνα 6.7 δείχνει στο χάρτη μόνο τα αποτελέσματα του Πρ. Ερωτήματος 3.

Στην επόμενη εικόνα φαίνονται οι πίνακες με τα αποτελέσματα στα τρία ερωτήματα. Παρατηρούμε εκτός των άλλων ότι τα πλάτη των πλαισίων των τριών Ερωτημάτων προσαρμόζονται ανάλογα με τις απαιτήσεις σε χώρο του καθενός από τους πίνακες με τα αποτελέσματα.



Εικόνα 6.8: Δύο όμοια Ρεύματα Δεδομένων και τρία Προσαρμοσμένα Ερωτήματα ενεργά – αποτελέσματα σε πίνακες

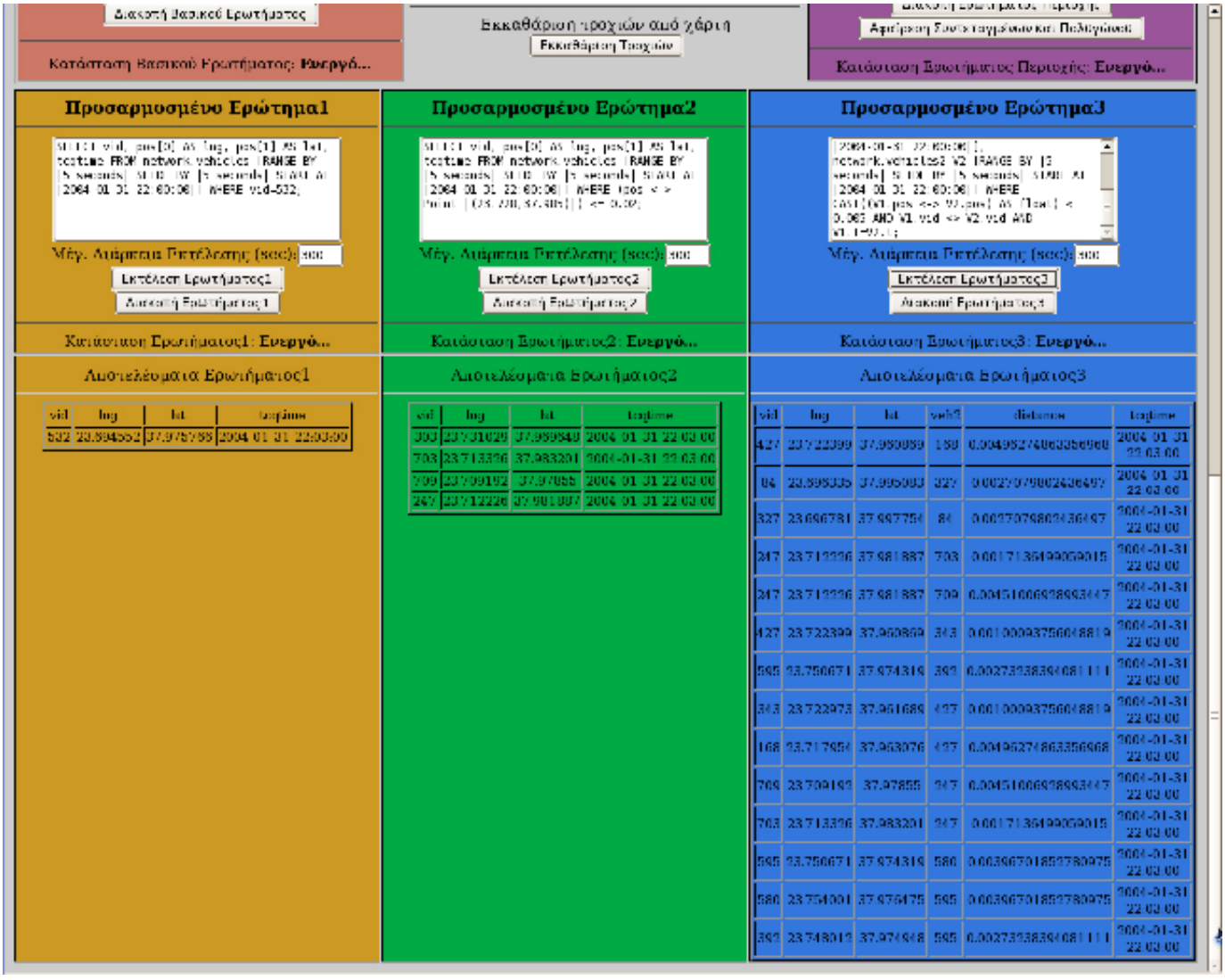
6.7 Δύο όμοια Ρεύματα Δεδομένων και Όλα τα Ερωτήματα Ενεργά

Παρακάτω παρατίθεται η σχετική για την περίπτωση αυτή εικόνα 6.9 και ακολουθούν αντίστοιχες επεξηγήσεις:

Εικόνα 6.9: Δύο όμοια Ρεύματα Δεδομένων και όλα τα ερωτήματα ενεργά – αποτελέσματα σε χάρτη (Ελληνική έκδοση)

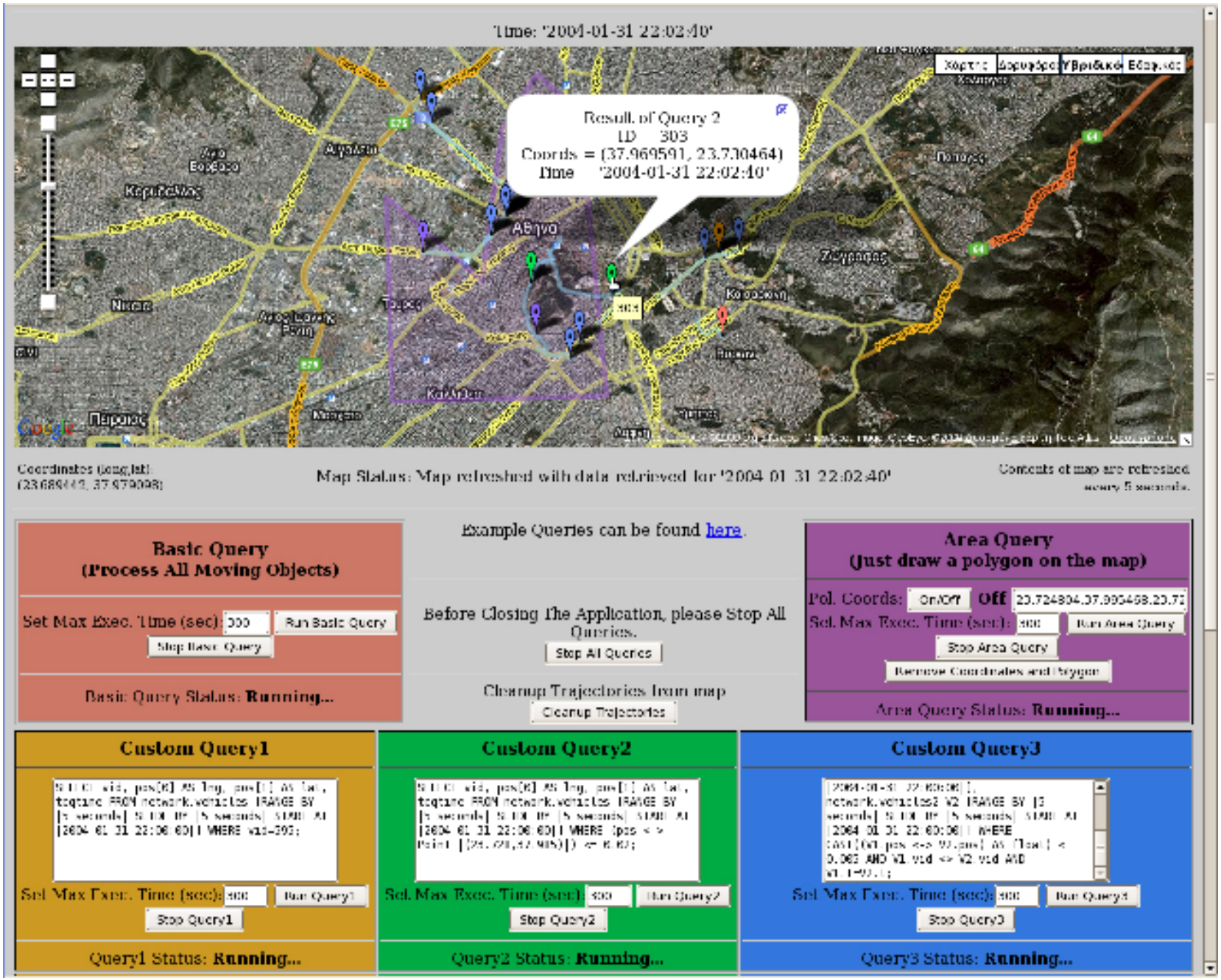
Στην εικόνα 6.9, ο χρήστης έχει ενεργοποιήσει και τα δύο Ρεύματα Δεδομένων και εκτελεί και τα πέντε ερωτήματα της εφαρμογής. Στα Προσαρμοσμένα Ερωτήματα έχει βάλει κατά σειρά το Παράδειγμα 1, το 6 και το 10. Έχει τροποποιήσει δε το Προσαρμοσμένο Ερώτημα 1 βάζοντας το id διαφορετικού οχήματος προς παρακολούθηση-διαχείριση. Επίσης, έχει επιλεγεί η εμφάνιση του υβριδικού χάρτη που αποτελεί έναν συνδυασμό εικόνων από δορυφόρο και οδικού χάρτη. Στα Προσαρμοσμένα Ερωτήματα έχουν επιλεγεί ερωτήματα που επιστρέφουν απαντήσεις επί του χάρτη, το δε Προσαρμοσμένο Ερώτημα 3 απαιτεί την εκτέλεση των δύο όμοιων Ρευμάτων Δεδομένων. Στο χάρτη λοιπόν, εμφανίζονται οι απαντήσεις και των πέντε ερωτημάτων κατά συγκεκριμένη σειρά. Κάτω από όλα εμφανίζονται οι απαντήσεις του Βασικού Ερωτήματος, αμέσως μετά εμφανίζονται οι απαντήσεις του Ερωτήματος Περιοχής και στη συνέχεια οι απαντήσεις των τριών Προσαρμοσμένων Ερωτημάτων κατ' αύξουσα σειρά.

Τα αποτελέσματα των τριών Προσαρμοσμένων Ερωτημάτων εμφανίζονται και σε πίνακες όπως δείχνει η εικόνα 6.10 παρακάτω.



Εικόνα 6.10: Δύο όμοια Ρεύματα Δεδομένων και όλα τα ερωτήματα ενεργά – αποτελέσματα σε πίνακες

Τέλος παρακάτω παρατίθεται αντίστοιχη εικόνα από την Αγγλική έκδοση της εφαρμογής (Εικόνα 6.11).



Εικόνα 6.11: Δύο όμοια Ρεύματα Δεδομένων και όλα τα ερωτήματα ενεργά – αποτελέσματα σε χάρτη (Αγγλική έκδοση)

Κεφάλαιο 7

Δυσκολίες – Συμπεράσματα – Μελλοντική εργασία

7.1 Δυσκολίες που προέκυψαν κατά τη δημιουργία της εφαρμογής

Η παρούσα εφαρμογή είναι πρωτότυπη και κυρίως λόγω των πολλών διαφορετικών εργαλείων που χρησιμοποιήθηκαν, κατά το στάδιο υλοποίησής της προέκυψαν αρκετές δυσκολίες και εμπόδια. Τα εμπόδια, τα προβλήματα και οι δυσκολίες, στο σύνολό τους αντιμετωπίστηκαν, άλλα με λιγότερη και άλλα με περισσότερη ενασχόληση και διερεύνηση.

Κατά το στάδιο λοιπόν της υλοποίησης της εφαρμογής παρουσιάστηκαν τα παρακάτω:

- Ο δημιουργός της εργασίας δεν είχε ασχοληθεί ποτέ στο παρελθόν με το λειτουργικό σύστημα Linux το οποίο χρειάστηκε να μάθει σε μεγάλο βαθμό. Ούτε και με κάποια γλώσσα προγραμματισμού.
- Υπήρξαν αρκετά προβλήματα κατά την εγκατάσταση του TelegraphCQ τα οποία ξεπεράστηκαν με ενασχόληση και σχετική διερεύνηση.
- Τα όποια προβλήματα με τον browser (Mozilla Firefox) λύθηκαν εγκαθιστώντας την τελευταία έκδοσή του.
- Υπήρξε πρόβλημα στην επικοινωνία μεταξύ PHP, Apache και PostgreSQL. Με την αντίστοιχη ενασχόληση και έρευνα λύθηκε το θέμα αυτό.
- Η επαναληπτική δημιουργία των XML επετεύχθη με σχετικό loop μέσα στο execQuery_getResults.php

- Η επαναληπτική ανάγνωση των δημιουργούμενων αρχείων XML επετεύχθη πάλι με σχετικό loop στον κώδικα της JavaScript.
- Υπήρξε μια δυσκολία στη σωστή δημιουργία των XML με τις απαντήσεις για τον παρακάτω λόγο. Το loop επαναληπτικής λήψης των αποτελεσμάτων από την PostgreSQL και το TelegraphCQ στο αρχείο PHP που εκτελεί τα ερωτήματα, επαναλαμβάνεται κάθε φορά που υπάρχει έστω ένα αποτέλεσμα από τη ΒΔ και το ΡΔ. Έτσι κάθε φορά, καθώς έρχονται 15 αποτελέσματα, το loop αυτό επαναλαμβάνεται 15 φορές με ελάχιστη χρονική διαφορά η μία από την άλλη εκτέλεση. Στη συνέχεια, το loop θα επαναληφθεί άλλη μία φορά καθώς το κριτήριο επανάληψής του είναι η πάροδος ή όχι του μισού χρόνου ανανέωσης των αποτελεσμάτων, κάτι που στην περίπτωσή μας είναι 5 δευτερόλεπτα και μπορεί πολύ εύκολα να αναπροσαρμοσθεί καθώς υπάρχει σχετική παράμετρος. Έτσι λοιπόν, ενώ τα αποτελέσματα είναι 15 (στο Βασικό Ερώτημα), αφού έχουμε 15 κινούμενα οχήματα, το loop θα εκτελεστεί μία ακόμη φορά, αφού το κριτήριο διακοπής του δεν θα πληρείται (θα έχουν περάσει λίγες στιγμές από την έναρξη του loop και όχι μερικά δευτερόλεπτα). Το πρόβλημα λοιπόν ήταν ότι στα XML υπήρχε πάντα μία ακόμα απάντηση από την επόμενη ομάδα εισερχόμενων δεδομένων κάτι που φαινόταν ειδικά από τη διαφορετική χρονική στιγμή αυτής της απάντησης. Για να μην αποθηκευτεί στο δημιουργούμενο XML αυτή η απάντηση που ανήκει στην επόμενη χρονική στιγμή, έγιναν σχετικές τροποποιήσεις και εφόσον υπάρχει απάντηση στο ερώτημα, αυτή αποθηκεύεται στο XML πριν τη λήψη της από τη Βάση Δεδομένων (αυτό μπορεί να ακούγεται παράλογο αλλά δεν είναι), με άλλα λόγια αποθηκεύεται στο XML κάθε φορά η προηγούμενη απάντηση η οποία έχει ήδη αποθηκευτεί προσωρινά σε σχετική γενική μεταβλητή του αρχείου PHP. Για να αποφευχθεί σχετικό σφάλμα κατά την πρώτη πρώτη εκτέλεση του loop, η γενική μεταβλητή παίρνει μία αρχική τιμή πριν την εκτέλεση του loop η οποία εάν ανιχνευθεί δεν εκτελείται η ομάδα εντολών που αποθηκεύουν την απάντηση στο ερώτημα στο αρχείο XML. Έτσι λοιπόν λύθηκε το πρόβλημα συγχρονισμού των απαντήσεων που υπήρχε. Η λύση αυτή μάλιστα είναι ανεξάρτητη από το πλήθος των απαντήσεων των ερωτημάτων το οποίο μπορεί να είναι διαφορετικό σε κάθε επανάληψη λήψης αποτελεσμάτων όπως στην περίπτωση του ερωτήματος περιοχής.
- Καθώς ο χρήστης μπορεί να θέσει δικά του ερωτήματα, μπορεί κάλλιστα να ζητά απαντήσεις χωρίς τις συντεταγμένες θέσης κινούμενων αντικειμένων όπως συμβαίνει με διάφορα παραδείγματα. Στην περίπτωση αυτή θα πρέπει η JavaScript να μην επιχειρεί να εμφανίσει σημάδια-σημεία (markers) επάνω στο χάρτη. Το πρόβλημα αυτό λύθηκε με σχετικό έλεγχο των τιμών συγκεκριμένων μεταβλητών της JavaScript πριν την εμφάνιση των αποτελεσμάτων επί χάρτη.
- Παρατηρήθηκε ότι σε κάποιες περιπτώσεις, ο πίνακας με τις απαντήσεις σε ένα ερώτημα, τύχαινε να εμφανιστεί στη θέση πίνακα απαντήσεων άλλου ερωτήματος είτε στιγμιαία, είτε περισσότερες φορές. Για την αποφυγή αυτού του σφάλματος αποθηκεύονται πλέον σε ξεχωριστές μεταβλητές οι διαφορετικοί πίνακες μέσα στον κώδικα JavaScript.
- Δυσκολία αντιμετωπίστηκε στην εύρεση τρόπου παύσης ενός ερωτήματος καθώς αυτά είναι συνεχή. Το πρόβλημα λύθηκε αρχικά με την εισαγωγή της έννοιας του μέγιστου χρόνου εκτέλεσης. Σε κάθε επανάληψη του βρόχου (loop) δημιουργίας του αρχείου XML με τις απαντήσεις μπήκε ένας έλεγχος εάν έχει παρέλθει ή όχι η μέγιστη διάρκεια εκτέλεσης. Το πρόβλημα είχε λυθεί εν μέρει. Στόχος ήταν να υπάρχει δυνατότητα παύσης οποιουδήποτε ερωτήματος όποτε το θελήσει ο χρήστης και όχι μόνο μετά από συγκεκριμένο χρόνο που θα έχει

ορίσει αυτός πριν την εκκίνηση του ερωτήματος (μπορεί π.χ. να άλλαξε γνώμη στο ενδιάμεσο). Έτσι λοιπόν, μετά από αρκετή ενασχόληση και διερεύνηση βρέθηκε τρόπος άμεσης διακοπής του κάθε ερωτήματος αρκεί να υπάρχει έστω και μία ακόμη απάντηση στο ερώτημα αφότου πατηθεί το σχετικό κουμπί διακοπής. Η διακοπή γίνεται με τον έλεγχο ύπαρξης ή όχι συγκεκριμένου εξωτερικού αρχείου (δαφτίστηκε αρχείο διακοπής). Εάν λοιπόν υπάρχει το αρχείο διακοπής, στην επόμενη επανάληψη του βρόχου δημιουργίας του αρχείου XML, η PHP θα το ανιχνεύσει και θα σταματήσει την εκτέλεση του συνεχούς ερωτήματος αλλά και του αρχείου execQuery_getResults.php. Αυτή η λύση κρίνεται ικανοποιητική αλλά στην περίπτωση που δεν υπάρξει άλλη απάντηση στο ερώτημα μετά το πάτημα του κουμπιού διακοπής, δε θα μπορέσει να σταματήσει το ερώτημα. Για τον λόγο αυτό, υπάρχει η ανά 24ώρο εκκαθάριση της εφαρμογής, αλλά και εκκαθάριση μικρής κλίμακας όταν ο χρήστης πατήσει το κουμπί διακοπής όλως των ερωτημάτων. Επίσης, το σχετικό αρχείο που εκτελείται όταν ο χρήστης ζητήσει τη διακοπή οποιουδήποτε ερωτήματος, μετά από μερικά δευτερόλεπτα σβήνει τα σχετικά αρχεία που δηλώνουν στη JavaScript ότι το ερώτημα εκτελείται (τα results#.xml και zQuery#IsRunning.xml). Έτσι, ο χρήστης βλέπει ότι το ερώτημα έχει σταματήσει και το σύστημα του επιτρέπει να εκτελέσει άλλο στη θέση του. Το αρχικό ερώτημα, εάν δεν σταματήσει με έναν από τους παραπάνω τρόπους, θα σταματήσει οπωσδήποτε στην επόμενη φορά που θα βρεθεί κάποιο αποτέλεσμα (αλλιώς θα σταματήσει στα σίγουρα στις 06:00 τα ξημερώματα της επόμενης μέρας με την εκκαθάριση).

- Σχετική δυσκολία υπήρξε για την εμφάνιση παραθύρου πληροφοριών όταν ο χρήστης κάνει κλικ επάνω σε ένα αποτέλεσμα (marker) στο χάρτη. Λύθηκε καθορίζοντας σχετική συνάρτηση δημιουργίας του και ακολουθώντας οδηγίες από διάφορες πηγές (κυρίως δικτυακούς τόπους με σχετικές πληροφορίες).
- Υπήρξε ένα θέμα με τον εντοπισμό, διαχωρισμό και αποθήκευση των χαρακτηριστικών που ζητά ένα ερώτημα, σε μεταβλητές της PHP. Οι ενέργειες αυτές είναι απαραίτητες εάν επιθυμούμε πραγματικά δυναμική λειτουργία της εφαρμογής και όχι απάντηση σε ερωτήματα με συγκεκριμένα πάντα χαρακτηριστικά. Έτσι, με αλλεπάλληλες αντικαταστάσεις συμβόλων και διαγραφή συμβολοσειρών μέσα στη συμβολοσειρά του ερωτήματος, φτάνουμε σε μία μεταβλητή που διατηρεί μόνο τα χαρακτηριστικά που έχουν ζητηθεί ως απαντήσεις στο ερώτημα. Από εκεί, με αντίστοιχη διαδικασία απομονώνονται σε ξεχωριστές μεταβλητές τα χαρακτηριστικά που έχουν ζητηθεί στο ερώτημα. Έτσι, καθίσταται εφικτή η απομόνωση των απαντήσεων του κάθε χαρακτηριστικού και η σωστή αποθήκευση αυτών στο αρχείο XML.
- Ένα άλλο θέμα που ανέκυψε κατά τους ελέγχους εκτέλεσης των αρχείων PHP από την κεντρική ιστοσελίδα της εφαρμογής, είναι ότι μόλις τελείωνε η εκτέλεση ενός αρχείου PHP, ο browser έφευγε από την ιστοσελίδα της εφαρμογής και έδειχνε τα αποτελέσματα εκτέλεσης του αρχείου PHP. Αυτό φυσικά δεν ήταν το επιθυμητό. Έτσι λοιπόν, μετά από σχετική διερεύνηση βρέθηκε λύση με τη χρήση του στοιχείου-ετικέτας <IFRAME> της HTML που δημιουργεί ένα πλαίσιο μέσα στην ιστοσελίδα όπου με τις κατάλληλες ρυθμίσεις εμφανίζονται τα αποτελέσματα εκτέλεσης των διαφόρων αρχείων PHP. Το IFRAME ρυθμίστηκε να έχει μηδενικές διαστάσεις (κάτι που επιτρέπουν οι παράμετροί του) και συνεπώς δεν εμφανίζεται καθόλου στην ιστοσελίδα.
- Στα πρώτα στάδια υλοποίησης της εφαρμογής, όταν σταματούσε την εκτέλεσή του ένα ερώτημα, τα τελευταία αποτελέσματα εξακολουθούσαν να εμφανίζονται

στο χάρτη. Το θέμα αυτό λύθηκε με κατάλληλες υποθέσεις (if) στον κώδικα JavaScript που αφαιρούν κάθε φορά τα αποτελέσματα πριν εμφανιστούν τα επόμενα και εάν αυτά δεν εμφανιστούν, τότε απλά τα τελευταία εξαφανίζονται στην επόμενη ανανέωση του χάρτη.

- Αντίστοιχα, για να εξαφανίζονται και οι τροχιές των ερωτημάτων, έπρεπε να τροποποιηθούν διάφορα τμήματα του κώδικα της JavaScript, κυρίως η συνάρτηση δημιουργίας τους και οι εντολές εμφάνισής τους επί του χάρτη.
- Μετά την αρχική καταχώρηση των ελέγχων από την JavaScript για το περιεχόμενο των διαφόρων πεδίων υποβολής πληροφοριών προς το execQuery_getResults.php, αποφάνηκε ότι ενώ τα πεδία εμφανίζουν μηνύματα λάθους, το αρχείο PHP εξακολουθούσε να εκτελείται. Έτσι, μπήκαν οι ίδιοι έλεγχοι και σε αυτό το αρχείο οπότε εάν τα υποβληθέντα δεδομένα περιέχουν λάθη, δε θα γίνει καν σύνδεση της PHP με την PostgreSQL, και το θέμα που είχε προκύψει λύθηκε.
- Το ερώτημα περιοχής δεν υπήρχε στα αρχικά στάδια υλοποίησης της εφαρμογής. Κρίθηκε ουσιώδης η προσθήκη του καθώς την κάνει πιο εύχρηστη και φιλική προς το χρήστη. Αρχικά ο χρήστης δεν δημιουργούσε πολύγωνο αλλά τεθλασμένη γραμμή. Η δημιουργία πολυγώνου βελτίωσε την εφαρμογή καθώς το εσωτερικό του εμφανίζεται σκιαγραφημένο. Πρόκληση αποτέλεσε η αυτόματη εισαγωγή των συντεταγμένων στο σχετικό πεδίο και μάλιστα με τον τρόπο που πρέπει να εισαχθούν στο ερώτημα προς εκτέλεση. Μετά από σχετική ενασχόληση το θέμα λύθηκε.
- Εκτός των παραπάνω, υπήρχε ένα θέμα με τις τροχιές των αντικειμένων. Κρίθηκε σκόπιμο να δίδεται η δυνατότητα στο χρήστη να τις καθαρίζει ανά πάσα στιγμή με το πάτημα ενός κουμπιού, καθώς μετά από κάποια ώρα ο χάρτης κυριολεκτικά γέμιζε γραμμές και χανόταν ένας από τους στόχους της εργασίας (φιλικότητα προς το χρήστη και σαφή και ευανάγνωστα αποτελέσματα στο χάρτη). Μετά από σχετική διερεύνηση το θέμα λύθηκε.
- Τέλος, η πιο σημαντική ίσως δυσκολία που έπρεπε να αντιμετωπιστεί ήταν η διαχείριση των ερωτημάτων που δεν επιστρέφουν αποτελέσματα. Η μέθοδος με την οποία λαμβάνουμε τα αποτελέσματα στα ερωτήματα από την PostgreSQL και το TelegraphCQ, είναι με τη χρήση κερσόρων (cursors). Το ερώτημα υποβάλλεται με τη δήλωση κέρσορα ως εξής: DECLARE [όνομα κέρσορα] CURSOR FOR [ερώτημα];. Τα δε αποτελέσματα λαμβάνονται με την εντολή FETCH ως εξής: FETCH FORWARD [πλήθος αποτελεσμάτων] IN [όνομα κέρσορα];. Η PHP λοιπόν, στην περίπτωσή μας, ζητά ένα-ένα τα αποτελέσματα στο κάθε συνεχές ερώτημα από την PostgreSQL και το TelegraphCQ. Καθώς τα ερωτήματα είναι συνεχή, η εντολή FETCH, έχει ρυθμιστεί έτσι ώστε εάν ο κέρσορας πάει σε θέση όπου δεν υπάρχουν ακόμη αποτελέσματα, να περιμένει αυτά να έρθουν. Το πρόβλημα είναι ότι θα περιμένει για πάντα εκτός και εάν διακοπεί με "βίαιο" τρόπο αυτή η αναμονή. Για παράδειγμα εάν εκτελούμε την εντολή FETCH μέσα από τερματικό, μπορούμε να πατήσουμε το συνδυασμό των πλήκτρων Ctrl + C που διακόπτει την τρέχουσα εργασία. Όταν όμως την πρόταση FETCH την υποβάλλει η PHP, τότε δεν υπάρχει αντίστοιχος τρόπος διακοπής της αναμονής. Επίσης, η εντολή pg_query() της PHP με την οποία εκτελούμε τις εντολές στην PostgreSQL, δεν έχει ούτε αυτή κάποιον τρόπο να παρακαμφθεί εάν δεν υπάρχουν αποτελέσματα για κάποιο χρονικό διάστημα. Αυτό συμβαίνει διότι τόσο η PostgreSQL όσο και η PHP δεν έχουν κατασκευασθεί για τη διαχείριση και σύνδεση αντίστοιχα με ρεύματα δεδομένων αλλά με βάσεις δεδομένων. Στην περίπτωση των βάσεων δεδομένων, η εντολή FETCH όταν φτάσει σε θέση (ανάγνωσης αποτελεσμάτων) όπου δεν

υπάρχουν αποτελέσματα επιστρέφει συγκεκριμένη τιμή και δεν περιμένει εσαεί την επόμενη απάντηση.
Το πρόβλημα λοιπόν θα λυνόταν εύκολα εάν υπήρχε μία προαιρετική παράμετρος μέγιστου χρόνου αναμονής για απάντηση είτε στην εντολή FETCH της PostgreSQL, είτε στην εντολή pg_query() της PHP.
Έτσι, για την όσο το δυνατόν καλύτερη λύση του προβλήματος, χρησιμοποιήθηκε η γλώσσα προγραμματισμού C++ σε συνδυασμό με εντολές του πυρήνα του λειτουργικού Linux. Με κατάλληλες προτάσεις, που έχουν αναλυθεί εκτενώς στις παραγράφους περιγραφής των προγραμμάτων C++, επιτυγχάνεται η αποθήκευση σε συγκεκριμένα αρχεία των ids των αντίστοιχων διεργασιών (processes) τα οποία, εάν χρειάζεται η διακοπή εκτέλεσης των ερωτημάτων, εισάγονται σε εντολές διακοπής διεργασιών που αναγνωρίζει το Linux και τερματίζουν τελικά τα επιθυμητά ερωτήματα. Πρόκειται για μία ικανοποιητική λύση του προβλήματος δεδομένου του χρόνου που ήταν διαθέσιμος για την επίλυσή του.

Θα μπορούσαν να αναφερθούν πολλές ακόμη τροποποιήσεις και μικροπροβλήματα που λύθηκαν αλλά τα πιο βασικά σημεία αναφέρθηκαν και περαιτέρω ανάλυση δεν κρίνεται σκόπιμη.

7.2 Συμπεράσματα

Η παρούσα εργασία είχε ως στόχο τη διαχείριση στόλου κινούμενων αντικειμένων με τη χρήση του συστήματος διαχείρισης ρευμάτων δεδομένων TelegraphCQ και απεικόνιση των αποτελεσμάτων σε χάρτες Google (Google Maps). Με τη λέξη διαχείριση εννοούμε την παρακολούθηση σε πραγματικό χρόνο συγκεκριμένων κινούμενων αντικειμένων αλλά και την επισκόπηση σχετικών με αυτά πληροφοριών. Η διαχείριση θα πρέπει να γίνεται σε ιστοσελίδα ώστε αυτή να είναι εφικτή από οποιονδήποτε υπολογιστή συνδεδεμένο με το διαδίκτυο οπουδήποτε στον κόσμο. Επιμέρους στόχοι ήταν η δυνατότητα εμφάνισης του συνόλου των κινούμενων αντικειμένων αλλά και τμημάτων αυτών ταυτόχρονα στους χάρτες Google της εφαρμογής. Άλλοι στόχοι ήταν η εμφάνιση των τροχιών των κινούμενων αντικειμένων για την καλύτερη διαχείριση αυτών και η δυνατότητα καθορισμού περιοχής και εμφάνιση των κινούμενων αντικειμένων που βρίσκονται μέσα σε αυτή. Για την καλύτερη διαχείριση των κινούμενων αντικειμένων, κρίθηκε σημαντικό να εμφανίζονται τα αποτελέσματα σε διάφορα ερωτήματα εκτός από τον χάρτη και σε πίνακες.

Οι παραπάνω στόχοι έχουν επιτευχθεί και η πρωτότυπη αυτή εφαρμογή έχει πολλά θετικά στοιχεία να επιδείξει. Έχει δοθεί ιδιαίτερη προσοχή εκτός των άλλων και στην εμφάνιση της ιστοσελίδας, ώστε να είναι όσο το δυνατόν πιο φιλική προς το χρήστη, με κύρια σημεία την ύπαρξη Ελληνικής και Αγγλικής έκδοσης, την εμφάνιση πληροφοριών σε διάφορα σημεία, όταν ο χρήστης σταματήσει τον δείκτη του ποντικιού επάνω τους, την ύπαρξη ολοκληρωμένων οδηγιών χρήσης και στις δύο γλώσσες, την εμφάνιση πληροφοριών σχετικά με τα κινούμενα αντικείμενα όταν ο χρήστης κάνει κλικ σε οποιοδήποτε από αυτά, την αυτόματη επιλογή του κειμένου κάθε πεδίου και πολλά άλλα.

Η συγκεκριμένη εφαρμογή θα μπορούσε να χρησιμοποιηθεί από μεταφορικές εταιρείες με σκοπό να γνωρίζουν ανά πάσα στιγμή πού βρίσκονται τα οχήματά τους και ως εκ τούτου να διαχειρίζονται καλύτερα έκτακτες καταστάσεις αλλά και το στόλο τους γενικότερα. Επίσης, οι εταιρίες ραδιοταξί θα μπορούσαν να επωφεληθούν από την εφαρμογή και να βλέπουν ανά πάσα στιγμή στο χάρτη πού είναι τα οχήματά τους, εάν είναι διαθέσιμα

για παραλαβή πελατών ή όχι και πλήθος άλλων πληροφοριών. Αντίστοιχα στρατιωτικές εφαρμογές θα επωφελούνταν από την παρούσα εφαρμογή και κυρίως από την άμεση εμφάνιση της πραγματικής θέσης των αντικειμένων που παρακολουθούν. Επίσης, τα ασθενοφόρα όλης της χώρας, θα μπορούσαν να εμφανίζονται στο χάρτη με διάφορες πιθανές καταστάσεις (επείγουσα μεταφορά ασθενών, επιστροφή στη βάση) και διαφόρων ειδών πληροφορίες, όπως τηλέφωνα οδηγών για άμεση επικοινωνία με τα νοσοκομεία, ταχύτητα κίνησης κλπ.

Οι χρήσεις αυτής της εφαρμογής είναι πραγματικά πάρα πολλές και ιδιαιτέρως χρήσιμες στη σημερινή κοινωνία.

Συμπερασματικά λοιπόν θα μπορούσε να πει κανείς ότι η παρούσα εφαρμογή ανοίγει το δρόμο για εύκολη και φιλική προς το χρήστη διαχείριση από οπουδήποτε στον κόσμο και σε πραγματικό χρόνο κινούμενων αντικειμένων με πληθώρα εφαρμογών.

7.3 Μελλοντική Εργασία

Καθώς πρόκειται για πρωτότυπη εφαρμογή, επιδέχεται πληθώρα επεκτάσεων οι οποίες θα μπορούσαν να διερευνηθούν μελλοντικά.

Θα μπορούσε πολύ εύκολα για παράδειγμα να ενσωματωθεί και ερώτημα απόστασης από σημείο. Ο χρήστης αφού θα είχε ενεργοποιήσει το συγκεκριμένο ερώτημα, θα μπορούσε να σημειώσει το σταθερό σημείο μέτρησης της απόστασης αλλά και την ίδια την απόσταση είτε γραφικά είτε πληκτρολογώντας σε σχετικά πεδία.

Οι αποστάσεις στην υλοποιημένη εφαρμογή μετρούνται σε μοίρες σε γεωγραφικό μήκος και πλάτος. Θα μπορούσε λοιπόν να ενσωματωθεί μετατροπέας μονάδων σε μέτρα (και ο χάρτης να εμφανίζει τις συντεταγμένες θέσεως στο ΕΓΣΑ '87) καθώς πρόκειται για μονάδα μέτρησης πολύ πιο οικεία από τις μοίρες οι οποίες μάλιστα σημαίνουν διαφορετική απόσταση στον άξονα Χ από ότι στον Υ.

Θα μπορούσε επίσης να επιχειρηθεί ο υπολογισμός της ταχύτητας (σε μέτρο και κατεύθυνση) των κινούμενων αντικειμένων με την JavaScript ως αποτέλεσμα της διαφοράς της πιο πρόσφατης θέσης από την προηγούμενη.

Χρήσιμη θα ήταν η ενσωμάτωση του id του κάθε κινούμενου αντικειμένου στο εικονίδιο με το οποίο εμφανίζεται στο χάρτη.

Για την ιδανική διαχείριση των ερωτημάτων που δεν επιστρέφουν αποτελέσματα, θα μπορούσε κανείς να ασχοληθεί εις βάθος με την ανάπτυξη του λογισμικού της PostgreSQL και να καταφέρει να τροποποιήσει την εντολή FETCH, στην ειδική έκδοση που χρησιμοποιείται από το TelegraphCQ, έτσι ώστε να δέχεται σε προαιρετική παράμετρο το μέγιστο χρόνο αναμονής για λήψη απάντησης και όταν αυτός παρέρχεται να επιστρέφει συγκεκριμένη τιμή.

Αντίστοιχα, θα μπορούσε κανείς να ασχοληθεί εις βάθος με τις εντολές και τις συναρτήσεις της PHP και να τροποποιήσει την εντολή pg_query() ώστε να δέχεται και αυτή μία προαιρετική παράμετρο σχετικά με το μέγιστο χρόνο αναμονής για απάντηση από την PostgreSQL και όταν αυτός παρέρχεται να επιστρέφει συγκεκριμένη τιμή.

Τέλος, μια σημαντική επέκταση θα ήταν η δυνατότητα εκτέλεσης ερωτημάτων με τροχιές, όπως η εμφάνιση οχημάτων των οποίων οι τροχιές διασταυρώθηκαν την προηγούμενη μισή ώρα ή αντίστοιχες ερωτήσεις.

Βιβλιογραφικές Αναφορές και Διαδικτυακοί Τόποι

Βιβλιογραφικές Αναφορές

[ABW03] A. Arasu, S. Babu, and J. Widom. CQL: A Language for Continuous Queries over Streams and Relations. Stanford University, 2003.

[ACC+03b] D.J. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. Aurora: a New Model and Architecture for Data Stream Management. VLDB Journal, 12(2):120-139, August 2003.

[Bal99] B. Ball. Sams Teach Yourself Linux in 24 Hours, Second Edition. Sams Publishing, Indianapolis, USA, 1999

[BAS+00] S. Bakken, A. Aulbach, E. Schmid, J. Winstead, L. Wilson, R. Lerdorf and Z. Suraski. PHP Manual. PHP Documentation Group, 2000.

[BBD+02] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and Issues in Data Stream Systems. In Proceedings of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'02), pp.1-16, Madison, Wisconsin, May 2002.

[Bro06a] F. Brokken. C++ Annotations Version 6.5.0, Computing Center, University of Groningen, Groningen, Netherlands, November 2006.

[Bro06b] M. Brown. Hacking Google® Maps and Google® Earth. Wiley Publishing Inc, Indianapolis, USA, 2006

[BW01] S. Babu and J.Widom. Continuous Queries over Data Streams. ACM SIGMOD Record, 30 (3): 109-120, September 2001.

[CCD+03] S. Chandrasekaran, O. Cooper, A. Deshpande, M.J. Franklin, J.M. Hellerstein, W. Hong, S. Krishnamurthy, S.R. Madden, V. Raman, F. Reiss, and M.A. Shah. TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. In Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR), Asilomar, California, January 2003.

[CJSS03] C. Cranor, T. Johnson, O. Spataschek, and V. Shkapenyuk. Gigascope: A Stream Database for Network Applications. In Proceedings of the 22nd ACM SIGMOD International Conference on Management of Data, pp. 647-651, San Diego, California, USA June 2003.

[CL07] R. Cadenhead and L. Lemay. Πλήρες Εγχειρίδιο της JAVA™ 6. Πέμπτη έκδοση. Μ. Γκιούρδας, Αθήνα. Απόδοση: Ε. Γκαγκάτσιου. Πρωτότυπο: Sams Teach Yourself Java™ 6 in 21 Days. Sams Publishing, Indianapolis, USA, 2007.

[Dav04] S. Davis. C++ For Dummies®, 5th Edition. Wiley Publishing Inc, Indianapolis, USA, 2004

[Dis01] S. Disbrow. JavaScript® Weekend Crash Course™. Hungry Minds, Inc, New York, USA, 2001.

[EGSV98] M. Erwig, R.H. Guting, M. M. Schneider, and M.Vazirgiannis. Abstract and Discrete Modeling of Spatio-Temporal DataTypes. In Proceedings of the 6th ACM Symposium on Geographic InformationSystems, Washington DC, pp.131-136, November 1998.

[GG98] V. Gaede, and O. Gunther. Multidimensional Access Methods. ACMComputing Surveys, 30 (2): 170-231, 1998.

[GO03] L. Golab, and M. Tamer Ozsu. Issues in Data Stream Management. ACM SIGMOD Record, 32(2):5-14, June 2003.

[Goo01] D. Goodman. JavaScript® Bible, Gold Edition. Hungry Minds, Inc, New York, USA, 2001.

[Gut94] R. H. Guting. An Introduction to Spatial Database Systems.VLDB Journal, Special Issue on Spatial Database Systems, 3 (4): 357-399,October 1994.

[Hol06] S. Holzner. Ajax For Dummies. Wiley Publishing Inc, Indianapolis, USA, 2006

[Laz02] Κ. Λάζος. Εισαγωγή στη C++ και στον Αντικειμενοστραφή Προγραμματισμό. Κ. Ε. Λάζος, Θεσσαλονίκη, 2002.

[MAHP03] M. F. Mokbel, W. G. Aref, S. E. Hambrusch, and S. Prabhakar.Towards Scalable Location-aware Services: Requirements and ResearchIssues. In Proceedings of the 11th ACM International Symposium on Advancesin Geographic Information Systems (GIS'03), pp. 110-117, New Orleans,Louisiana, USA, November 2003.

[Mel08] J. Meloni. Μάθετε PHP, MySQL και Apache Όλα σε ένα. Τέταρτη έκδοση. Μ. Γκούρδας, Αθήνα. Απόδοση: Γ. Σαμαράς. Πρωτότυπο: Sams Teach Yourself PHP, MySQL and Apache All in One, Fourth Edition. Sams Publishing, Indianapolis, USA, 2008

[Mon07] M. Moncur. Μάθετε την JavaScript σε 24 ώρες. Τέταρτη έκδοση. Μ. Γκιούρδας, Αθήνα. Απόδοση: Μ. Γκλαβά. Πρωτότυπο: Sams Teach Yourself JavaScript in 24 Hours. Sams Publishing, Indianapolis, USA, 2007.

[MS05] N. Matthew and R. Stones. Beginning Databases with PostgreSQL: From Novice to Professional, 2nd Edition. Springer-Verlag New York, Inc, 2005

[MWA+03] R. Motwani , J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein and R.Varma. Query Processing, Approximation, and Resource Management in a Data Stream Management System. In Proceedings of the 2003 Conference on Innovative Data Systems Research (CIDR), January 2003.

[Neg02] C. Negus. Red Hat Linux 7.2 Bible, Unlimited Edition. Hungry Minds, Inc, New York, USA, 2002.

[Pre01] R. Pressman. Software Engineering, A Practitioner's Approach, Fifth Edition. The McGraw-Hill Companies, Inc, New York, USA, 2001.

[PS04] K. Patroumpas and T. Sellis. Managing Trajectories of Moving Objects as Data Streams. In Proceedings of the 2nd Workshop on Spatio-Temporal Database Management (STDBM'04), Toronto, Canada, August 2004.

[Pur97] J. Purcell. Linux Complete Command Reference, First Edition. Red Hat Software, Inc, Indianapolis, USA, 1997.
[SCZ05] M. Stonebraker, U. Çetintemel, S. Zdonik. The 8 Requirements of Real-Time Stream Processing. ACM SIGMOD Record, Volume 34 , Issue 4, December 2005.
[SPF08] R. Schwartz, T. Phoenix, and B. Foy. Learning Perl, Fifth Edition. O'Reilly Media, July 2008.
[Ste00] A. Stevens. Οδηγός της C++ με παραδείγματα. Έκτη Αμερικανική Έκδοση. Ε. Γκαγκάτσιου (ΜΤΦ). Μ. Γκιούδρας, Αθήνα, 2000. Πρωτότυπο: Teach Yourself C++, Sixth Edition. IDG Books Worldwide Inc, Fostrecity USA, 2000.
[TA97] D. Taylor, J. Armstrong. Teach Yoursefl UNIX in 24 Hours, First Edition. Sams Publishing, Indianapolis, USA, 1997
[Tel05] M. Telles. C++ Timesaving Techniques™ FOR DUMmIES®. Wiley Publishing Inc, Indianapolis, USA, 2005
[The03] Y. Theodoridis. Ten Benchmark Database Queries for Location-based Services. The Computer Journal, 46(6), British Computer Society, February 2003.
[TSPM98] Y. Theodoridis, T. Sellis, A.N. Papadopoulos and Y. Manolopoulos. Specifications for efficient indexing in spatiotemporal databases. In Proceedings of 10th International Conference on Scientific and Statistical DatabaseManagement (SSDB 1998), pp. 123-132, Capri, Italy, 1998.
[Ven03] Ι. Βενιέρης. Σημειώσεις για τη γλώσσα HTML (HyperText Markup Language). Ε. Μ. Π. Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Ζωγράφου, Αθήνα, 2003.
[Woo01] C. Wootton. JavaScript Programmer's Reference. Wrox Press Ltd, Birmingham, UK, 2001.
[WXCJ98] O. Wolfson, B. Xu, S. Chamberlain, L. Jiang. Moving ObjectsDatabases: Issues and Solutions. In Proceedings of 10th InternationalConference on Scientific and Statistical Database Management (SSDB 1998),pp. 111-122, Capri, Italy, July 1998.
[Zan00] M. Zandstra. SAMS Teach Yourself PHP4 in 24 Hours. Sams Publishing, Indianapolis, USA, 2000

Διαδικτυακοί Τόποι

Σχετικά με το Linux

01. http://www.linuxquestions.org/
02. http://www.linuxforums.org/
03. http://www.oreillynet.com/linux/cmd/
04. http://www.linuxhelp.net/
05. http://webtools.live2support.com/linux/
06. http://www.redhat.com/
07. http://www.linux.gr/
08. http://www.linux.org/
09. http://www.linux.com/
10. http://el.wikipedia.org/wiki/Linux

Σχετικά με το TelegraphCQ

01. http://telegraph.cs.berkeley.edu/
02. http://telegraph.cs.berkeley.edu/telegraphcq/v0.2/
03. http://telegraph.cs.berkeley.edu/telegraphcq/v2.0/
04. http://telegraph.cs.berkeley.edu/telegraphcq/v2.1/
05. http://db.cs.berkeley.edu/papers/cidr03-tcq.pdf
06. http://www.cs.umd.edu/class/fall2002/cmsc818s/Lectures/TelegraphCQ.pdf
07. http://neilconway.org/talks/tcq.pdf

Σχετικά με την PostgreSQL

01. http://www.postgresql.org/
02. http://www.postgresql.gr/
03. http://en.wikipedia.org/wiki/PostgreSQL
04. http://www.pgsql.com/
05. http://www.planetpostgresql.org/
06. http://www.postgresqlforums.com/forums/
07. http://www.postgresql.org/docs/7.3/static/sql-commands.html
08. http://www.commandprompt.com/

Σχετικά με την HTML

01. http://www.w3schools.com/html/DEFAULT.asp
02. http://el.wikipedia.org/wiki/HTML
03. http://en.wikipedia.org/wiki/HTML
04. http://www.eeei.gr/odhgos/htmlex.htm
05. http://www.w3.org/TR/html401/
06. http://www.it.uom.gr/project/html2/main.html
07. http://www.it.uom.gr/project/htmlman/pages/index.htm

Σχετικά με την PHP

01. http://www.php.net/
02. http://el.wikipedia.org/wiki/PHP
03. http://en.wikipedia.org/wiki/PHP
04. http://www.w3schools.com/PHP/DEfaULT.asP
05. http://forums.devnetwork.net/
06. http://forums.devshed.com/php-development-5/
07. http://www.phpfreaks.com/
08. http://www.phpfreaks.com/forums/
09. http://www.phpbuilder.com/
10. http://www.forums.gr/forumdisplay.php?f=113
11. http://www.daniweb.com/forums/forum17.html

Σχετικά με την XML

01. http://www.w3.org/XML/
02. http://www.w3schools.com/xml/default.asp
03. http://en.wikipedia.org/wiki/XML
04. http://www.it.uom.gr/project/xml/Home%20Page.htm
05. http://www.xml.com/
06. http://www.xml.org/
07. http://www.w3.org/TR/REC-xml/
08. http://www.webdeveloper.com/forum/forumdisplay.php?f=5

Σχετικά με την JavaScript

01. http://www.w3schools.com/JS/default.asp
02. http://el.wikipedia.org/wiki/JavaScript
03. http://en.wikipedia.org/wiki/JavaScript
04. http://javascript.internet.com/
05. http://www.javascript.com/
06. http://www.it.uom.gr/project/Dhtml_Jscripts/jvscr.htm
07. http://www.javascriptkit.com/
08. http://www.webdeveloper.com/forum/forumdisplay.php?forumid=3&s
09. http://www.devshed.com/c/b/JavaScript/
10. http://forums.digitalpoint.com/forumdisplay.php?f=38
11. http://www.daniweb.com/forums/forum117.html

Σχετικά με τους χάρτες Google και το GoogleMaps API

01. http://maps.google.com/
02. http://code.google.com/intl/el/apis/maps/
03. http://maps.google.com/maps/ms?ie=UTF8&hl=el&msa=0&msid=100876167667258050915.0004408e39a6e62fb43a0
04. http://en.wikipedia.org/wiki/Google_Maps
05. http://econym.googlepages.com/index.htm
06. http://www.googlemapsforum.com/
07. http://groups.google.com/group/Google-Maps-API
08. http://www.google.com/support/forum/p/maps?hl=en
09. http://code.google.com/intl/el/apis/maps/documentation/reference.html
10. http://www.google.com/intl/el_ALL/help/terms_maps.html

Σχετικά με την C++

01. http://www.cplusplus.com/
02. http://el.wikipedia.org/wiki/C%2B%2B
03. http://en.wikipedia.org/wiki/C%2B%2B
04. http://www.it.uom.gr/project/cppmanual/cplman/index.htm
05. http://www.materials.uoc.gr/el/undergrad/courses/ΕΤΥ215/

06. http://www.computing.net/
07. http://www.codersource.net/
08. http://www.daniweb.com/forums/forum8.html

Σχετικά με την Perl

01. http://www.perl.org/
02. http://www.it.uom.gr/project/perl/win32perltut.html
03. http://el.wikipedia.org/wiki/Perl
04. http://en.wikipedia.org/wiki/Perl
05. http://www.perl.com/
06. http://www.daniweb.com/forums/forum112.html
07. http://www.perlmonks.org/
08. http://perldoc.perl.org/functions/
09. http://www.tutorialspoint.com/perl/
10. http://perl.active-venture.com/
11. http://perlguru.com/

Ορολογία

απεικόνιση	mapping
αποβολή φόρτου	load shedding
γλώσσα ερωταποκρίσεων	query language
γνώρισμα	attribute
δειγματοληψία	sampling
διαφορά	difference
ένθετο υποερώτημα	nested suquery
ένωση	union
επιλογή	selection
ερώτημα διαρκείας, συνεχές ερώτημα	continuous query
ερώτημα μονότονο	monotonic query
ερώτημα περιοχής	area query, range query
ερώτημα k-εγγύτερων γειτόνων	k-nearest neighbor query
ερώτημα συζευκτικό	conjunctive query
θέση	position, location
κάλυψη	coverage
κατηγόρημα	predicate
κινούμενο αντικείμενο	moving object
κλιμάκωση	scalability
κυματίδιο	wavelet
παράθυρο κυλιόμενο	sliding window
παράθυρο οροσήμου	landmark window
παράθυρο πλειάδων	count-based window
περίληψη	summary
πολλαπλή σύνδεση	multi-way join
προβολή	projection
προσέγγιση	approximation
προσχέδιο εκτέλεσης ερωτήματος	query execution plan
ρεύμα δεδομένων	data stream
συνάθροιση	aggregation
σύνδεση	join
σύνδεση τροχιών	trajectory join
σύνοψη	synopsis
συγχώνευση	merge
τελεστής	operator
ταξινόμηση	sorting
τομή	intersection
τροχιά	trajectory
υστέρηση	lag
υπηρεσίες εντοπισμού	location-based services
χρονόσημο	timestamp