ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΑΓΡΟΝΟΜΩΝ ΚΑΙ ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΓΕΩΠΛΗΡΟΦΟΡΙΚΗ»

ΔΙΑΧΕΙΡΙΣΗ ΔΥΝΑΜΙΚΩΝ ΜΕΤΑΒΟΛΩΝ ΣΕ ΧΩΡΙΚΑ ΔΕΔΟΜΕΝΑ ΜΕ ΧΡΗΣΗ ΔΙΑΔΙΚΤΥΑΚΩΝ ΥΠΗΡΕΣΙΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Νικολάου Κολιού

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Αθήνα, Δεκέμβριος 2008

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΑΓΡΟΝΟΜΩΝ ΚΑΙ ΤΟΠΟΓΡΑΦΩΝ ΜΗΧΑΝΙΚΩΝ
ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
«ΓΕΩΠΛΗΡΟΦΟΡΙΚΗ»

ΔΙΑΧΕΙΡΙΣΗ ΔΥΝΑΜΙΚΩΝ ΜΕΤΑΒΟΛΩΝ ΣΕ ΧΩΡΙΚΑ ΔΕΔΟΜΕΝΑ ΜΕ ΧΡΗΣΗ ΔΙΑΔΙΚΤΥΑΚΩΝ ΥΠΗΡΕΣΙΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Νικολάου Κολιού

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Αθήνα, Δεκέμβριος 2008

Στην μητέρα μου.

Επιτρέπεται η χρήση ή αναδημοσίευση με αναφορά της πηγής.
Επικοινωνία: n.kolios@gmail.com

Πρόλογος

Η παρούσα μεταπτυχιακή εργασία ξεκίνησε στις αρχές Νοεμβρίου 2007. Το γενικότερο πλαίσιο μέσα στο οποίο επρόκειτο να κινηθεί είχε να κάνει με την διερεύνηση των μηχανισμών τήρησης ιστορικού σε συναλλαγές χωρικών δεδομένων που διέπονται από το πρότυπο WFS-T του Open Geo-Spatial Consortium. Αρχικά έγινε βιβλιογραφική επισκόπηση γύρω από τις θεωρητικές έννοιες που περικλείουν τόσο γενικά τις συναλλαγές όσο και ειδικότερα τις συναλλαγές σε χωρικές βάσεις δεδομένων. Εξετάστηκαν τα πρότυπα WMS και WFS-T, καθώς και λογισμικά που έχουν αναπτυχθεί ειδικά για να υλοποιούν αυτές τις προδιαγραφές. Στην συνέχεια διερευνήθηκε η δημιουργία μίας πλατφόρμας λογισμικού που με την ενσωμάτωση υπαρχόντων επιμέρους λογισμικών θα αποτελούσε τοπικό περιβάλλον χρήστη, με σκοπό την διαδικτυακή πρόσβαση και επέμβαση, μέσω εξυπηρετητή, σε χωρικά δεδομένα που είναι αποθηκευμένα σε χωρική βάση δεδομένων. Το τοπικό αυτό περιβάλλον θα αναπτυσσόταν σε γλώσσα Java έτσι ώστε να είναι συμβατό με υπάρχουσες εργαλειοθήκες και λογισμικά που υλοποιούν την προδιαγραφή WFS-T.

Όμως, όσο η εφαρμογή προχωρούσε σε αυτή την κατεύθυνση, γινόταν περισσότερο ξεκάθαρο ότι η αντικειμενική δυσκολία της εξ 'αρχής ανάπτυξης της υπολογιστικής πλατφόρμας μετακινούσε το κέντρο βάρους της εργασίας από την διαχείριση των συναλλαγών σε χωρικά δεδομένα στο γενικότερο πεδίο του προγραμματισμού εφαρμογής σε γραφικό περιβάλλον. Σε αυτό το στάδιο αποφασίστηκε να μετακινηθεί η στόχευση της εργασίας στην κατεύθυνση της μελέτης των λειτουργιών τήρησης ιστορικού και διαχείρισης επεμβάσεων και υλοποίησης στο περιβάλλον μίας χωρικής βάσης δεδομένων όπου αυτό δεν ήταν δυνατό μέχρι τώρα.

Έτσι διερευνήθηκε η δομή των υπαρχόντων μοντέλων τήρησης ιστορικού και διαχείρισης εκδόσεων δεδομένων σε υπάρχοντα συστήματα χωρικών βάσεων δεδομένων. Από αυτή την αναζήτηση προκρίθηκε το μοντέλο που υλοποιείται με την χρήση επικουρικών πινάκων στους οποίους αποθηκεύονται στοιχεία σχετικά με τις επεμβάσεις. Εκτελώντας ερωτήματα σε αυτούς τους πίνακες είναι δυνατό να αναπαρασταθούν τα δεδομένα του αρχικού πίνακα σε συγκεκριμένες στιγμές του παρελθόντος. Προγραμματιστικά, αυτό αποφασίστηκε να υλοποιηθεί με την ανάπτυξη δομών ελέγχου (σκανδαλιστές, συναρτήσεις) που θα αναπτυχθούν με την χρήση της ενσωματωμένης διαδικαστικής γλώσσας PL/pgSQL. Στην συνέχεια αναπτύχθηκαν τα επιμέρους κομμάτια του συστήματος και καθώς και η βασική δομή των μηχανισμών τήρησης ιστορικού και επικυρώσεων των αλλαγών. Το τελευταίο στάδιο του τεχνικού σκέλους ολοκληρώθηκε με την εκτεταμένη δοκιμή του συστήματος για την αποθήκευση και επικύρωση διαφορετικών σεναρίων επεμβάσεων καθώς και την επιλεκτική επικύρωση μέρους αυτών των επεμβάσεων με βάση τα ιδιαίτερα χαρακτηριστικά τους. Η διαδικασία των δοκιμών παρείχε μία σειρά από χρήσιμες παρατηρήσεις που χρησίμευσαν στην διόρθωση σφαλμάτων και τελειοποίηση του συστήματος. Με τη συγγραφή του τόμου ολοκληρώθηκε και συνολικά η παρούσα μεταπτυχιακή εργασία.

Το αποτέλεσμα της συγγραφής της παρούσας εργασίας μπορεί να αξιολογηθεί σε τουλάχιστον δύο βασικά επίπεδα. Κατ' αρχήν το σύστημα που αναπτύχθηκε είναι πλήρως λειτουργικό και μπορεί να αποτελέσει τον πυρήνα ενός λογισμικού επέκτασης της χωρικής βάσης δεδομένων PostgreSQL/PostGIS (και των συστημάτων WFS-T που στηρίζονται σε αυτή) με λειτουργίες τήρησης ιστορικού και διαχείρισης των επεμβάσεων. Επιπλέον η ανάπτυξη του υπήρξε μία ιδιαίτερα εκπαιδευτική διαδικασία για τον γράφοντα τόσο σχετικά με τις τεχνικές γνώσεις του προγραμματισμού στην γλώσσα PL/pgSQL και την δομή συστημάτων WFS-T όσο και τις γενικότερες γνώσεις που έχουν να κάνουν με χωρικές βάσεις δεδομένων και την διαχείριση συναλλαγών σε αυτές.

Νιώθω τυχερός για την συμμετοχή μου σε αυτή την διαδικασία και γι' αυτό θα ήθελα να ευχαριστήσω των καθηγητή ΕΜΠ Τίμο Σελλή που μου έδωσε αυτή την ευκαιρία δείχνοντας μου την ανάλογη εμπιστοσύνη. Ακόμα θα ήθελα να ευχαριστήσω ιδιαίτερα τον υποψήφιο διδάκτορα Κώστα Πατρούμπα για την συνεχή και πολύτιμη υποστήριξη που μου παρείχε κατά την εκπόνησή της με την μορφή συμβουλών, διορθώσεων και ενθάρρυνσης. Τέλος θα ήθελα να ευχαριστήσω τους καθηγητές Μαρίνο Κάβουρα και Λύσανδρο Τσούλο για την συμμετοχή τους στην τριμελή επιτροπή που θα αξιολογήσει την εργασία. Παρ΄ όλη την σημαντική συμβολή των παραπάνω η ευθύνη για το τελικό αποτέλεσμα βαραίνει αποκλειστικά τον συγγραφέα.

Επειδή η παρούσα εργασία σηματοδοτεί το τέλος των σπουδών μου στο ΔΠΜΣ «Γεωπληροφορική» θα ήθελα να ευχαριστήσω θερμά όλους του παραπάνω για την συμμετοχή τους σε αυτό το μεταπτυχιακό πρόγραμμα. Με αυτήν, έδωσαν σε εμένα και άλλους νέους ανθρώπους την ευκαιρία να συμμετέχουμε σε ένα σπουδαίο μεταπτυχιακό πρόγραμμα και να αποκτήσουμε πολύτιμη παιδεία που θα μας επιτρέψει να κάνουμε τους εαυτούς μας καλύτερους όσο και - το πλέον αναγκαίο – να προσφέρουμε περισσότερα στο ευρύτερο κοινωνικό σύνολο.

Σύνοψη

Η κλιμακούμενη ανάγκη για αποτελεσματική διαχείριση μεγάλου όγκου χωρικών δεδομένων έχει οδηγήσει στην ανάπτυξη των συστημάτων διαχείρισης χωρικών βάσεων δεδομένων. Η πιθανότητα βλαβών, καθώς και η λειτουργία τους σε περιβάλλον όπου πολλοί χρήστες επεμβαίνουν στην βάση δεδομένων και ενδεχομένως στα ίδια στοιχεία δεδομένων οδήγησε στην ανάγκη για αποτελεσματική αποθήκευση και διαχείριση των επεμβάσεων δηλαδή στην ανάγκη για τήρηση ιστορικού.

Η τήρηση ιστορικού σε χωρικές βάσεις δεδομένων αφορά στην καταγραφή των επεμβάσεων που γίνονται (εισαγωγή, διαγραφή ή ενημέρωση δεδομένων), έτσι ώστε μέρος ή το σύνολο της ακολουθίας των ιστορικών συναλλαγών να μπορεί να ανακτηθεί ανά πάσα στιγμή.

Στο επίπεδο των διαδικτυακών υπηρεσιών σε χωρικές βάσεις δεδομένων, οι συναλλαγές διέπονται από το πρότυπο διεπαφών Web Feature Services-Transactional (WFS-T) που έχει αναπτυχθεί από το Open Geospatial Consortium και υποστηρίζει τους κανόνες υποβολής και ανταπόκρισης με χρήση του πρωτοκόλλου HTTP και της γλώσσας XML.

Μία λύση που έχει προταθεί για την τήρηση ιστορικού σε χωρικές βάσεις δεδομένων προβλέπει την δημιουργία επιπλέον πινάκων εντός της βάσης δεδομένων για την αποθήκευση των στοιχείων που εισάγονται, ενημερώνονται ή διαγράφονται. Με την χρήση αυτών των πινάκων ο διαχειριστής της βάσης δεδομένων μπορεί να επιλέξει ποιες από τις αλλαγές επιθυμεί να διατηρήσει και ποιες να αναιρέσει.

Στην παρούσα διπλωματική εργασία επιχειρείται η ανάπτυξη συστήματος τήρησης ιστορικού που να ακολουθεί αυτή την μεθοδολογία. Συγκεκριμένα διερευνώνται:

- Η δυνατότητα τήρησης ιστορικού των μεταβολών που επήλθαν στα στοιχεία μιας χωρικής βάσης δεδομένων με χρήση βοηθητικών πινάκων.
- Η δυνατότητα ελεγχόμενης επικύρωσης των συναλλαγών με κριτήρια το είδος τους (εισαγωγή, διαγραφή ή τροποποίηση) ή την χρονική στιγμή που συνέβησαν, εκμεταλλευόμενοι το ιστορικό που τηρήθηκε.

Για τον λόγο αυτό αναπτύχθηκε πλατφόρμα WFS-T η οποία αποτελείται από την βάση δεδομένων Postgres/PostGIS, τον διαδικτυακό εξυπηρετητή χωρικών δεδομένων GeoServer και το διαδικτυακό σύστημα γεωγραφικών πληροφοριών UDIG . Το σύστημα ενισχύθηκε με:

- Σκανδαλιστές και συναρτήσεις σε περιβάλλον PL/pgSQL που αυτόματα ενημερώνουν τους βοηθητικούς πίνακες της βάσης με σκοπό την τήρηση ιστορικού μεταβολών. Οι νέοι πίνακες περιέχουν χρονόσημο σε κάθε εγγραφή που δηλώνει την χρονική στιγμή επέμβασης στο γεωγραφικό στοιχείο.
- Συνάρτησης επικύρωσης PL/pgSQL των μεταβολών που επήλθαν στην βάση δεδομένων κατά την διάρκεια των συναλλαγών. Η συνάρτηση χρησιμοποιώντας το χρονόσημο που προσδιορίζει την χρονική στιγμή των επεμβάσεων που περιέχονται στους βοηθητικούς πίνακες και πραγματοποιεί

σειριακή ενσωμάτωση τους σε αντίγραφο του αρχικού πίνακα. Με αυτό τον τρόπο επιτρέπεται η επιλεκτική επικύρωση των συναλλαγών (με την χρήση παραμέτρων) ανάλογα με την φύση τους ή/και την χρονική στιγμή στην οποία συνέβησαν.

Εκτεταμένες δοκιμές έδειξαν ότι το σύστημα λειτουργεί με επιτυχία σε όλα τα επίπεδα. Συγκεκριμένα υπήρξε δυνατή η ανάκτηση χωρικών δεδομένων και επέμβαση (εισαγωγή, ενημέρωση, διαγραφή) σε αυτά στο γραφικό περιβάλλον χάρτη του λογισμικού UDIG. Τα δεδομένα διακινήθηκαν από τον εξυπηρετητή χωρικών δεδομένων GeoServer και ήταν αποθηκευμένα στην χωρική βάση δεδομένων PostgreSQL/PostGIS. Οι επεμβάσεις που έγιναν αποθηκεύθηκαν αυτόματα σε σύστημα βοηθητικών πινάκων. Ακόμη έγινε δυνατή η χρήση των περιεχομένων αυτών των πινάκων έτσι ώστε να επικυρωθεί επιλεκτικά μέρος των επεμβάσεων.

Λέξεις κλειδιά: Τήρηση ιστορικού, επικύρωση επεμβάσεων, PL/pgSQL, σκανδαλιστές, διαδικτυακές υπηρεσίες δεδομένων – συναλλαγών, χωρικές βάσεις δεδομένων.

Abstract

The increasing demand for effective management of a large volume of spatial data has led to the development of spatial database management systems. The potential of system failures as well as the ability to access or modify the same spatial features by multiple users have created the need for versioning mechanisms.

In spatial databases, versioning refers to recording modifications (i.e. insert, delete or update of data items) that occur to spatial entities across time in a database. In this way, all or some of the modifications can be retrieved when necessary.

In the context of web services in spatial databases, transactions are specified by the Web Feature Services-Transactional (WFS-T) interface specification, published by Open Geospatial Consortium. This standard describes the rules of requests and responses submitted via HTTP protocol and XML language.

A proposed solution for versioning of incremental changes in spatial features suggests the use of auxiliary tables inside the geodatabase for storing records that are being inserted, updated or deleted. Using these tables, the database administrator can selectively validate modifications based on their characteristics.

In this MSc. thesis, we attempt to develop a versioning mechanism according to the above methodology. Specifically we investigate:

- The possibility of registering modifications that happen to the records of a geodatabase by using auxiliary tables.
- The possibility of using the data included in these additional tables in the context of a selective validation of modifications.

Toward these goals we developed a WFS-T platform comprised of a spatially enabled database (Postgres/PostGIS), a spatial data server (GeoServer) and a network GIS (UDIG). The versioning mechanism was implemented with the addition of:

- Triggers with functions written in PL/pgSQL language, which automatically update the auxiliary tables with information about modifications. The new tables also include a timestamp attribute that specifies when each modification took place.
- Functions that validate modifications that occurred to the contents of the geodatabase. A timestamp attribute is stored in the auxiliary tables in order to serially merge modifications in a copy of the original table. Selective validation of modifications is possible based on their characteristics (insert, update or delete) or the time that they took place. Validation functions are also written in PL/pgSQL language.

This mechanism has been tested and proves to work effectively with several types of features. Spatial data were retrieved and modified in the graphical map environment of the UDIG network GIS software. The data were distributed by the spatial data server software GeoServer and stored in the

PostgreSQL/PostGIS geodatabase. The modifications were automatically stored in the additional tables. Finally, we used the contents of the additional tables in order validate all the modifications or selectively reconciliate part of them based on the time they happened or their kind (insert, update, delete).

Keywords: Versioning, validation of modifications, PL/pgSQL, triggers, web feature service – transactional (WFS-T), spatial databases.

Πίνακας περιεχομένων

Κεφάλαιο 1

Εισαγωγή

1.1 Σύγχρονες προκλήσεις στην διαχείριση χωρικών δεδομένων

Στην εποχή μας οι τεχνολογικές εφαρμογές που κάνουν χρήση γεωγραφικών πληροφοριών και χωρικών δεδομένων αναπτύσσονται συνεχώς και σταδιακά παίζουν όλο και περισσότερο ρόλο στην καθημερινότητα του σύγχρονου ανθρώπου. Τέτοια παραδείγματα αποτελούν οι εφαρμογές πλοήγησης, οι υπηρεσίες που βασίζονται στην θέση (location based services), η συλλογή, διαχείριση και διάθεση χαρτογραφικών δεδομένων και δορυφορικών εικόνων σε σχεδόν πραγματικό χρόνο, οι εφαρμογές παρακολούθησης της γήινης επιφάνειας, εικονικές υδρόγειοι (virtual globes) συνδυασμοί όλων των παραπάνω καθώς και πολλές ακόμα εφαρμογές. Αυτές οι εφαρμογές παράγονται από μία εκτεταμένη ανθρώπινη και τεχνολογική υποδομή και παράγει σε καθημερινή βάση σημαντικό όγκο χωρικών δεδομένων. Η ανάγκη για αποτελεσματική αποθήκευση και διαχείριση αυτών των δεδομένων οδήγησε στην ανάπτυξη των χωρικών βάσεων δεδομένων.

Χωρική βάση δεδομένων καλούμε μία βάση δεδομένων που έχει δημιουργηθεί έτσι ώστε να μπορεί να αποθηκεύει, να διαχειρίζεται και να εκτελεί ερωτήματα σε χωρικά δεδομένα. Οι χωρικές βάσεις δεδομένων -όπως και οι άλλες βάσεις δεδομένων- παρέχουν την δυνατότητα σε απομακρυσμένους χρήστες να προσπελαύνουν και να επεμβαίνουν στα δεδομένα τους μέσω διαδικτυακών υπηρεσιών. Οι χρήστες προβαίνουν σε επεμβάσεις που είναι οργανωμένες σε λογικές ενότητες που ονομάζουμε συναλλαγές. Συχνά οι συναλλαγές αυτές είναι χρονοβόρες με αποτέλεσμα να μην είναι δυνατό ή συμφέρον το κλείδωμα – δηλαδή ο περιορισμός επέμβασης στα πλαίσια μίας και μόνο συναλλαγής - των σχετικών στοιχείων δεδομένων κατά την διάρκειά τους. Κάτι τέτοιο σημαίνει ότι υπάρχει η πιθανότητα ταυτόχρονης επέμβασης εκ μέρους πολλαπλών χρηστών στο ίδιο στοιχείο δεδομένων (data item) και εγείρει ζητήματα συνέπειας (consistency) της βάσης δεδομένων. Έτσι προκύπτει ή ανάγκη για διαχείριση των πολλαπλών και πιθανά αντικρουόμενων επεμβάσεων εκ μέρους πολλών χρηστών. Η διαχείριση πολλαπλών επεμβάσεων σε μία βάση δεδομένων γίνεται με μηχανισμούς και μεθοδολογίες τήρησης ιστορικού. Η τήρηση ιστορικού σε βάσεις δεδομένων αφορά στην καταγραφή των

επεμβάσεων που γίνονται (εισαγωγή, διαγραφή ή ενημέρωση δεδομένων), έτσι ώστε το σύνολο ή μέρος της ακολουθίας των συναλλαγών (transactions) να μπορεί να ανακτηθεί ανά πάσα στιγμή.

Στο επίπεδο των διαδικτυακών υπηρεσιών (Web Services) σε χωρικές βάσεις δεδομένων, οι συναλλαγές διέπονται από το πρότυπο διεπαφών Web Feature Services-Transactional (WFS-T) που έχει οριστεί από το Open Geospatial Consortium [OGC+02]. Αυτό το πρότυπο ορίζει τους κανόνες υποβολής ερωτημάτων και ανταπόκρισης που αφορούν σε ανάκτηση των δεδομένων καθαυτών και όχι κάποιας αναπαράστασής τους. Η επικοινωνία γίνεται με χρήση του πρωτοκόλλου HTTP και της γλώσσας Geographical Markup Language (GML).

Μία λύση που έχει προταθεί για την τήρηση ιστορικού σε χωρικές βάσεις δεδομένων προβλέπει την δημιουργία επιπλέον πινάκων εντός της βάσης δεδομένων για την αποθήκευση των στοιχείων που εισάγονται, ενημερώνονται ή διαγράφονται. Με την χρήση αυτών των πινάκων ο διαχειριστής της βάσης δεδομένων μπορεί να επιλέξει ποιες από τις αλλαγές επιθυμεί να διατηρήσει και ποιες να αναιρέσει.

1.2 Αντικείμενο της εργασίας

Στην παρούσα διπλωματική εργασία επιχειρούμε να αναπτύξουμε ένα σύστημα διατήρησης ιστορικού και ελεγχόμενης επικύρωσης των ιστορικών μεταβολών. Σκοπός μας είναι:

- Να διερευνήσουμε τη δυνατότητα τήρησης ιστορικού των μεταβολών που επήλθαν στα στοιχεία μιας χωρικής βάσης δεδομένων με χρήση βοηθητικών πινάκων εντός της βάσης δεδομένων. Προκειμένου αυτό να γίνει δυνατό υλοποιούμε ένα τοπικό σύστημα WFS-T με στοιχεία για την Πελοπόννησο. Συγκεκριμένα δημιουργήσαμε χωρική βάση δεδομένων σε περιβάλλον PostgreSQL ενισχυμένη με τον σπόνδυλο χωρικών λειτουργιών PostGIS. Μέσω του διαδικτυακού εξυπηρετητή GeoServer επιτρέψαμε την πρόσβαση και επέμβαση σε αυτά τα δεδομένων με το διαδικτυακό σύστημα γεωγραφικών πληροφοριών UDIG. Ακόμα δημιουργούμε σκανδαλιστές (triggers) σε περιβάλλον PL/pgSQL που αυτόματα ενημερώνουν τους βοηθητικούς πίνακες της βάσης με σκοπό την τήρηση ιστορικού μεταβολών. Οι νέοι πίνακες περιέχουν χρονόσημο (timestamp) σε κάθε εγγραφή που δηλώνει την χρονική στιγμή επέμβασης στο γεωγραφικό στοιχείο.
- Να διερευνήσουμε τη δυνατότητα χρήσης των δεδομένων που περιέχονται σε αυτούς τους βοηθητικούς πίνακες στο πλαίσιο της ελεγχόμενης επικύρωσης των επεμβάσεων. Σε αυτό το πλαίσιο δημιουργήσαμε συνάρτηση επικύρωσης των επεμβάσεων που επήλθαν στην βάση δεδομένων κατά την διάρκεια των συναλλαγών. Η συνάρτηση πραγματοποιεί σειριακή ενσωμάτωση σε αντίγραφο του αρχικού πίνακα των επεμβάσεων έτσι όπως αυτές έχουν αποθηκευθεί στις εγγραφές των βοηθητικών πινάκων. Επιπλέον η συνάρτηση περιέχει παραμέτρους που αφορούν στα χαρακτηριστικά των μεταβολών που επικυρώνονται και επιτρέπει την επιλεκτική επικύρωση των συναλλαγών ανάλογα με την φύση τους (εισαγωγή, ενημέρωση, διαγραφή εγγραφών) ή/και την χρονική στιγμή στην οποία συνέβησαν.

Συνολικά, με την χρήση επιμέρους συστατικών στοιχείων αποτελούμενων από στοιχεία ελέγχου όπως βρόχοι, συναρτήσεις (functions) και σκανδαλιστές που δημιουργούνται με χρήση της διαδικαστικής γλώσσας PL/pgSQL επιτυγχάνεται η τήρηση ιστορικού των επεμβάσεων σε χωρική βάση δεδομένων καθώς και η ελεγχόμενη επικύρωση τους στον αρχικό πίνακα.

1.3 Δομή της εργασίας

Ο τόμος της παρούσας εργασίας έχει την ακόλουθη δομή: Στο κεφάλαιο 2 αναπτύσσονται οι βασικές έννοιες γύρω από τις οποίες αναπτύσσεται η παρούσα εργασία. Αρχικά συζητείται η έννοια των συναλλαγών ως ενοτήτων από ενέργειες που προσπελαύνουν και ενημερώνουν στοιχεία δεδομένων, οι ιδιότητες ACID τις οποίες πρέπει να έχουν οι συναλλαγές προκειμένου να μην επηρεάσουν την συνέπεια της βάσης δεδομένων, και οι διάφορες προσεγγίσεις σειριακότητας . Αναπτύσσονται οι βασικές προσεγγίσεις γύρω από τα σχήματα ελέγχου συγχρονικότητας, τους μηχανισμούς δηλαδή που εξασφαλίζουν σειριακότητα σε ταυτόχρονες συναλλαγές. Ακόμα αναπτύσσονται οι προδιαγραφές WMS, WFS/WFS-T που διέπουν τις τα αντίστοιχα πρωτόκολλα διαδικτυακών υπηρεσιών χωρικών δεδομένων.

Στο κεφάλαιο 3 εστιάζουμε στην τήρηση ιστορικού μεταβολών σε χωρικές βάσεις δεδομένων. Αρχικά αναπτύσσεται ο σκοπός και οι αρχές της τήρησης ιστορικού σε χωρικές βάσεις δεδομένων. Στην συνέχεια αφού συνοψίσουμε τις κυριότερες αρχιτεκτονικές τήρησης ιστορικού εμβαθύνουμε στο μοντέλο που εφαρμόζεται από την ESRI για την τήρηση ιστορικού σε χωρικές βάσεις δεδομένων. Το μοντέλο αυτό προϋποθέτει την ύπαρξη βοηθητικών πινάκων όπου αποθηκεύονται οι μεταβολές που γίνονται στην βασική εκδοχή των δεδομένων.

Το κεφάλαιο 4 αποτελείται από δύο βασικά μέρη. Στο πρώτο από αυτά (παράγραφος 4.1) αναπτύσσονται τα βασικά δομικά στοιχεία που χρησιμοποιήθηκαν στην ανάπτυξη του συστήματος τήρησης ιστορικού και ελεγχόμενης επικύρωσης των μεταβολών που έγινε στην παρούσα εργασία. Συγκεκριμένα περιγράφονται τα επιμέρους λογισμικά που αποτέλεσαν το σύστημα WFS-T. Πρόκειται για βάση δεδομένων PostgreSQL ενισχυμένη με τον σπόνδυλο χωρικών λειτουργιών PostGIS, τον διαδικτυακού εξυπηρετητή χωρικών δεδομένων GeoServer και το διαδικτυακό σύστημα γεωγραφικών πληροφοριών UDIG. Ακόμα εξηγείται ο ρόλος και τα χαρακτηριστικά της διαδικαστικής γλώσσας PL/pgSQL με την οποία υλοποιήθηκαν τα επιμέρους τμήματα της μεθοδολογίας τήρησης ιστορικού. Στο δεύτερο μέρος του κεφαλαίου (παράγραφοι 4.2 και 4.3) αναλύεται η δομή και λειτουργία του συστήματος τήρησης ιστορικού που αναπτύχθηκε ενώ επίσης δίνεται ένα εκτεταμένο παράδειγμα της λειτουργίας του.

Στο κεφάλαιο 5 αναπτύσσονται τα συμπεράσματα που εξάγονται από την παρούσα μεταπτυχιακή εργασία. Αποτιμάται η προσφορά της εργασίας στην συζήτηση σχετικά με το πρωτόκολλο διαδικτυακών υπηρεσιών WFS-T, το λογισμικό διαχείρισης βάσεων δεδομένων ανοικτού κώδικα και την σημασία των ενσωματωμένων διαδικαστικών γλωσσών σε χωρικές βάσεις δεδομένων. Επιπλέον επισημαίνονται τα πιθανά σημεία από τα οποία μπορεί να ξεκινήσει μια επερχόμενη προσπάθεια για περαιτέρω ανάπτυξη του εν λόγω συστήματος έτσι ώστε να γίνει πλήρως λειτουργικό.

Ο τόμος κλείνει με την ενότητα της βιβλιογραφίας, την ενότητα που περιλαμβάνει απόδοση στα ελληνικά των βασικών όρων που χρησιμοποιήθηκαν καθώς και το παράρτημα που περιλαμβάνει τα τμήματα πηγαίου κώδικα που συντάχθηκαν σε γλώσσα PL/pgSQL και επεξήγηση αυτών.

Κεφάλαιο 2

Βασικές έννοιες

Οι βάσεις δεδομένων μπορούν να αποθηκεύσουν, να διαχειριστούν και να εκτελούν ερωτήματα σε δεδομένα. Αυτές οι ενέργειες προσπέλασης και επέμβασης σε δεδομένα εκτελούνται οργανωμένες σε λογικές ενότητες που ονομάζουμε συναλλαγές. Συχνά, πολλαπλοί χρήστες επιδρούν σε μία βάση δεδομένων και ενδεχομένως στα ίδια στοιχεία δεδομένων. Αυτή η συνθήκη γεννά την ανάγκη για προστασία της λογικής συνέπειας των στοιχείων που περιέχονται στην βάση δεδομένων χωρίς μεγάλους συμβιβασμούς στην απόδοση του συστήματος και τις ανάγκες του σε υπολογιστικούς πόρους. Επιπλέον οι χρήστες δεν βρίσκονται κατ' ανάγκη στην ίδια φυσική τοποθεσία με την βάση δεδομένων και προκειμένου να τα προσπελάσουν και να εκτελέσουν συναλλαγές σε αυτά κάνουν χρήση διαδικτυακών υπηρεσιών. Στις βάσεις δεδομένων που περιέχουν χωρικά στοιχεία αυτό μπορεί να σημαίνει ότι οι χρήστες μπορούν να στέλνουν στην βάση δεδομένων ερωτήματα και οι βάσεις δεδομένων να τους απαντούν στέλνοντας τους απεικονίσεις αυτών των δεδομένων με την μορφή χαρτών. Το περιβάλλον τέτοιων συναλλαγών έχει οριστεί με προδιαγραφές όπως η WMS του Open Geospatial Consortium. Σε ένα βήμα παραπέρα οι χρήστες μπορούν όχι μόνο να κάνουν ερωτήσεις στην βάση δεδομένων, αλλά και να τροποποιούν τα χωρικά δεδομένα. Σε αυτή την περίπτωση οι βάσεις δεδομένων απαντούν όχι με αναπαραστάσεις των δεδομένων με την μορφή εικόνων-χαρτών αλλά στέλνοντας στους χρήστες αντίγραφα των σχετικών στοιχείων δεδομένων. Τέτοιου είδους συναλλαγές ορίζονται από το πρότυπο WFS-T του Open Geospatial Consortium. Στο κεφάλαιο που ακολουθεί θα συζητηθούν οι βασικές έννοιες των συναλλαγών, των σχημάτων ελέγχου συγχρονικότητας καθώς και τα πρότυπα WMS και WFS-T του Open Geospatial Consortium.

2.1 Συναλλαγές

Οι συναλλαγές είναι μονάδες εκτέλεσης προγραμμάτων που προσπελαύνουν και πιθανότατα ενημερώνουν δεδομένα. Η εκτέλεσή μίας συναλλαγής συχνά

προϋποθέτει την προσπέλαση ή ενημέρωση παραπάνω του ενός στοιχείων δεδομένων. Αν για οποιοδήποτε λόγο (πχ. βλάβη συστήματος) μία συναλλαγή διακοπεί πριν ολοκληρωθεί, και γι αυτό τον λόγο δεν ενημερώσει όλα τα διαφορετικά στοιχεία δεδομένων που περιλαμβάνει η εκτέλεσή της, τότε η βάση δεδομένων μπορεί να γίνει ασυνεπής (inconsistent). Για παράδειγμα μία συναλλαγή με την βάση δεδομένων μίας τράπεζας που έχει 50 ευρώ σε δύο λογαριασμούς Α, Β με 20 και 30 ευρώ αντίστοιχα, μπορεί να περιλαμβάνει την μεταφορά του ποσού 10 ευρώ από τον λογαριασμό Β στον λογαριασμό Α. Η συναλλαγή περιλαμβάνει τα εξής επιμέρους βήματα:

- Ανάγνωση του λογαριασμού Α
- Ανάγνωση του λογαριασμού Β
- Αφαίρεση του ποσού των 10 ευρώ από τον λογαριασμό Β
- Πρόσθεση των 10 ευρώ στον λογαριασμό Α

Αν για κάποιο λόγο η συναλλαγή διακοπεί μετά το τρίτο βήμα τότε και οι δύο λογαριασμοί θα περιέχουν 20 ευρώ και συνολικά η βάση δεδομένων θα περιέχει 40 ευρώ. Αυτό είναι λάθος γιατί η βάση δεδομένων της τράπεζας στην πραγματικότητα δεν έχει 40 αλλά 50 ευρώ. Σε αυτή την περίπτωση λέμε ότι η βάση δεδομένων της τράπεζας δεν αντικατοπτρίζει την πραγματικότητα ή με άλλα λόγια είναι ασυνεπής. Συνολικά λοιπόν η συγχρονική εκτέλεση και οι διάφορες βλάβες συστημάτων μπορούν να έχουν ως αποτέλεσμα η βάση να γίνει ασυνεπής. Για να μην συμβεί αυτό οι συναλλαγές πρέπει να έχουν τις ιδιότητες ACID: Ατομικότητα (Atomicity), Συνέπεια (Consistency), Απομόνωση (Isolation), Μονιμότητα (Durability) [EN04], [SKS04].

- Η Ατομικότητα εξασφαλίζει ότι είτε όλα τα αποτελέσματα της συναλλαγής αποτυπώνονται στην βάση δεδομένων ή κανένα δεν αποτυπώνεται. Μία αστοχία μπορεί να αφήσει την βάση δεδομένων σε μία κατάσταση όπου μία συναλλαγή έχει μόνο κατά ένα μέρος πραγματοποιηθεί.
- Η Συνέπεια εξασφαλίζει ότι αν η βάση δεδομένων είναι πριν την εκτέλεση της συναλλαγής είναι λογικά συνεπής θα είναι και το ίδιο μετά την εκτέλεση της συναλλαγής.
- Η Απομόνωση εξασφαλίζει ότι οι συγχρονικά πραγματοποιούμενες συναλλαγές είναι απομονωμένες μεταξύ τους έτσι ώστε η κάθε μία από αυτές να εκτελείται σαν να μην εκτελείται καμία άλλη ταυτόχρονα.
- Η Μονιμότητα εξασφαλίζει ότι όταν πια η συναλλαγή έχει πραγματοποιηθεί οι αλλαγές που έχει αυτή κάνει στα δεδομένα δεν χάνονται ακόμα και αν συμβεί αστοχία του συστήματος.

Στην πραγματικότητα, μία βάση δεδομένων περιλαμβάνει πολλά περισσότερα στοιχεία δεδομένων από αυτά που εμπλέκονται σε μία συναλλαγή. Για λόγους ταχύτητας και με δεδομένο ότι ο όγκος των συναλλαγών που πραγματοποιούνται σε μία βάση δεδομένων είναι μεγάλος, αλλά και για λόγους βέλτιστης αξιοποίησης των υπολογιστικών πόρων, οι συναλλαγές δεν εκτελούνται διαδοχικά (σειριακά) αλλά ταυτόχρονα (συγχρονικά). Η σειρά με την οποία εκτελούνται τα επιμέρους βήματα μίας σειράς διαφορετικών συναλλαγών σε μία βάση δεδομένων καλείται χρονοδιάγραμμα (schedule). Τα σειριακά χρονοδιαγράμματα είναι αδύνατο να αφήσουν την βάση δεδομένων σε στάδιο ασυνέπειας αν δεν μεσολαβήσει κάποια βλάβη συστήματος. Αντίθετα στα

συγχρονικά χρονοδιαγράμματα αυτό είναι πολύ πιο πιθανό να αφήσουν την βάση δεδομένων σε κατάσταση ασυνέπειας ακόμα και αν δεν μεσολαβήσει κάποια βλάβη συστήματος. Αυτό συμβαίνει γιατί τα ίδια στοιχεία δεδομένων μπορούν να προσπελαθούν ταυτόχρονα στο πλαίσιο περισσότερων από μία συναλλαγών.

Είναι συνεπώς απαραίτητο ένα σύστημα διαχείρισης βάσης δεδομένων να ελέγξει την αλληλεπίδραση ανάμεσα στις συναλλαγές. Εφ όσον μία συναλλαγή είναι μία ενότητα εντολών που διατηρούν την συνέπεια της βάσης δεδομένων, έτσι και μία σειρά συναλλαγών – που εκτελούνται σειριακά – θα διατηρήσουν ανάλογα την συνέπεια της βάσης δεδομένων. Ένα χρονοδιάγραμμα (δηλαδή ένας τρόπος συγχρονικής εκτέλεσης συναλλαγών) που εξασφαλίζει αυτή την κατάσταση για την βάση δεδομένων λέμε ότι εξασφαλίζει σειριακότητα (δηλαδή οι συναλλαγές εκτελούνται έτσι ώστε το αποτέλεσμα τους στην βάση δεδομένων να είναι το ίδιο με αυτό που θα προέκυπτε αν οι συναλλαγές δεν εκτελούνταν συγχρονικά αλλά σειριακά).

Υπάρχουν διάφορες προσεγγίσεις σειριακότητας ανάμεσα σε χρονοδιαγράμματα (schedules). Οι δύο βασικές αρχές είναι αυτή της σειριακότητας διαμάχης (conflict serialisability) και της σειριακότητας όψης (view serialisability).

Η σειριακότητα διαμάχης αναφέρεται στην ισότητα με ένα σειριακό χρονοδιάγραμμα (δηλαδή ένα χρονοδιάγραμμα όπου οι συναλλαγές συμβαίνουν διαδοχικά) έτσι ώστε τα δύο χρονοδιαγράμματα να έχουν τα ίδια σχετικά ζεύγη από αντικρουόμενες συναλλαγές και στην ίδια χρονολογική σειρά.

Η σειριακότητα όψης αναφέρεται στην ισότητα με ένα σειριακό χρονοδιάγραμμα, δηλαδή ένα χρονοδιάγραμμα όπου οι συναλλαγές συμβαίνουν διαδοχικά, έτσι ώστε τα δύο χρονοδιαγράμματα να προσπελαύνουν τα στοιχεία δεδομένων με την ίδια σειρά.

Τα συστήματα διαχείρισης βάσεων δεδομένων αντιμετωπίζουν αυτόν τον κίνδυνό με σχήματα ελέγχου συγχρονικότητας (concurrency control schemes) [GR03].

2.2 Έλεγχος συγχρονικότητας

Η σειριακότητα των προγραμμάτων που περιέχουν συγχρονικά εκτελούμενες συναλλαγές εξασφαλίζεται με μηχανισμούς που καλούνται σχήματα ελέγχου συγχρονικότητας (concurrency control schemes). Οι μηχανισμοί αυτοί προσδιορίζουν την σειρά με την οποία θα εκτελεστούν τα επιμέρους βήματα των συγχρονικά εκτελούμενων συναλλαγών έτσι ώστε τελικά να αποτελούν μέρος ενός προγράμματος με χαρακτηριστικά σειριακότητας. Ανάλογα με την δομή και τα χαρακτηριστικά τους μπορεί να εξασφαλίζουν σε διαφορετικό βαθμό σειριακότητα και προστασία από τα αδιέξοδα [EN04].

Τα πρωτόκολλα κλειδώματος είναι σύνολα κανόνων που προσδιορίζουν πότε μία συναλλαγή μπορεί να κλειδώνει και πότε να ξεκλειδώνει δεδομένα (πχ κελιά, γραμμές ή στήλες). Το πρωτόκολλο δύο φάσεων είναι ένα τέτοιο πρωτόκολλο που επιτρέπει σε μία συναλλαγή να κλειδώνει δεδομένα μόνο πριν αρχίσει να ξεκλειδώνει κάποια από αυτά. Το πρωτόκολλο αυτό εξασφαλίζει σειριακότητα αλλά όχι προστασία από αδιέξοδα (deadlocks). Όταν δεν έχουμε κάποια

πληροφορία σχετικά με την σειρά με την οποία πρέπει να προσπελάσουμε τα δεδομένα τότε αυτό το πρωτόκολλο παρέχει μία αναγκαία και επαρκή λύση.

Τα σχήματα διάταξης χρονοσήμων (timestamp ordering schemes) εξασφαλίζουν σειριακότητα προσδιορίζοντας την σειρά ανάμεσα σε ζεύγη συναλλαγών. Προσδιορίζουν ένα χρονόσημο για κάθε συναλλαγή και σύμφωνα με αυτήν οι συναλλαγές μπαίνουν σε μία σειρά φαινομενικής σειριακής εκτέλεσης. Κάθε φορά που αυτή η σειρά παραβιάζεται μία συναλλαγή αναιρείται (transaction rollback).

Τα σχήματα επικύρωσης (validation schemes) είναι κατάλληλα σε περιπτώσεις που η μεγάλη πλειοψηφία των συναλλαγών είναι συναλλαγές ανάγνωσης (read only transactions) και όχι συναλλαγές εγγραφής (write transactions). Οι συναλλαγές μπαίνουν σε μία σειρά φαινομενικής σειριακότητας όπως και πριν με την χρήση χρονοσήμων αλλά εδώ μια συναλλαγή δεν καθυστερείται ποτέ. Απλά για να εκτελεστεί πρέπει να περάσει με επιτυχία μία δοκιμή επικύρωσης (validation test). Αν δεν περάσει με επιτυχία αυτό το τεστ τότε η συναλλαγή αναιρείται στο αρχικό της στάδιο.

Υπάρχουν περιπτώσεις που θα ήταν σκόπιμο να ομαδοποιήσουμε διαφορετικά κομμάτια δεδομένων και να τα διαχειριστούμε σαν ένα ενιαίο τμήμα δεδομένων με αποτέλεσμα σε πολλαπλά επίπεδα (granularity). Σε αυτές τις περιπτώσεις έχουμε πολλά κομμάτια δεδομένων διαφορετικών μεγεθών και ορίζουμε μία ιεραρχία δεδομένων όπου τα μικρότερα τμήματα δεδομένων είναι φωλιασμένα σε μεγαλύτερα. Αυτή η ιεραρχία μπορεί να αναπαρασταθεί γραφικά σαν ένα δέντρο. Τα δεδομένα κλειδώνονται με σειρά από την ρίζα προς τα φύλλα. Τα δεδομένα απελευθερώνονται με σειρά από τα φύλλα προς την ρίζα. Αυτού του είδους τα πρωτόκολλά εξασφαλίζουν σειριακότητα αλλά όχι ασφάλεια από αδιέξοδα.

Τα πρωτόκολλα ελέγχου συγχρονικότητας πολλαπλών εκδοχών (multiversion) δημιουργούν μία νέα εκδοχή των δεδομένων κάθε φορά που κάποιος χρήστης αποφασίζει να επέμβει σε αυτά. Κάθε φορά που οι συναλλαγές πρέπει να διαβάσουν κάποια δεδομένα το σύστημα επιλέγει ποια έκδοση των δεδομένων θα διαβαστεί.

Τα διαφορετικά αυτά πρωτόκολλα δεν εξασφαλίζουν και προστασία από αδιέξοδα. Ένας τρόπος για να αποφύγουμε τα τελευταία είναι η χρήση προληπτικών αναιρέσεων. Για να ελέγξουμε αυτό τον μηχανισμό πρόληψης αποδίδουμε ένα μοναδικό χρονόσημο σε κάθε συναλλαγή. Αυτό το χρονόσημο χρησιμοποιείται για να αποφασιστεί αν κάποια συναλλαγή θα μπει σε αναμονή ή θα αναιρεθεί. Αν μία συναλλαγή αναιρεθεί τότε διατηρεί το παλιό χρονόσημο. Μία άλλη μέθοδος για να αντιμετωπιστούν τα αδιέξοδα είναι τα σχήματα ανίχνευσης και αντιμετώπισης των αδιεξόδων. Όταν ο αλγόριθμος αντιμετώπισης ανιχνεύσει ένα αδιέξοδο τότε το σύστημα αναιρεί μία ή περισσότερες συναλλαγές για το αποτρέψει.

Μία εντολή delete πραγματοποιείται μόνο εάν η συναλλαγή που την πραγματοποιεί έχει αποκλειστικό κλείδωμα στα δεδομένα που θα σβηστούν. Μία συναλλαγή που εισάγει νέα δεδομένα έχει αποκλειστικό κλείδωμα σε αυτά τα δεδομένα. Η εισαγωγή δεδομένων μπορεί κάποιες φορές να οδηγήσει στο φαινόμενο φαντασμάτων (phantom phenomenon) κατά το οποίο μία εισαγωγή δεδομένων έρχεται σε διαμάχη με ένα ερώτημα (query) ακόμα και αν οι δύο αυτέ

εντολές δεν προσπελαύνουν κάποιο κοινό κελί. Η τεχνική του κλειδώματος με δείκτες (index locking) λύνει αυτό το πρόβλημα επιβάλλοντας κλειδώματα σε δείκτες συνόλων δεδομένων (index buckets). Αυτά τα κλειδώματα δεδομένων εξασφαλίζουν ότι οι συναλλαγές θα έρθουν σε διαμάχη για πραγματικά και όχι phantom δεδομένα.

Ειδικές τεχνικές ελέγχου συγχρονικότητας μπορούν να εφαρμοστούν για ειδικές δομές δεδομένων όπως τα B+ δένδρα για να εξασφαλίσουν αυξημένη συγχρονικότητα. Αυτές οι τεχνικές επιτρέπουν μη σειριακή προσπέλαση του B+ δένδρου, αλλά εξασφαλίζουν ότι η δομή του δένδρου είναι σωστή και ότι η πρόσβαση στην βάση δεδομένων καθαυτή θα είναι σειριακή [GR03].

2.3 Το πρωτόκολλο Web Map Service

Με τον όρο Web map service εννοούμε μία διαδικτυακή υπηρεσία που παράγει χάρτες με γεωαναφερόμενα (georeferenced) δεδομένα (OGC, 2002). Οι χάρτες οπτικοποιούν τα δεδομένα, δεν αποτελούν τα δεδομένα καθαυτά. Οι προδιαγραφές του OGC προσδιορίζουν τρεις WMS λειτουργίες:

- Λειτουργία Get Capabilities επιστρέφει μεταδεδομένα σχετικά με τις υπηρεσίες, δηλαδή πληροφορίες σχετικά με το περιεχόμενο των υπηρεσιών και των τρόπο αίτησης τους.
- Λειτουργία Get Map, που επιστρέφει χάρτες με ακριβή χωρικά και γεωμετρικά χαρακτηριστικά.
- Λειτουργία Get Feature Info (προαιρετική) που επιστρέφει πληροφορίες σχετικά με συγκεκριμένα στοιχεία που απεικονίζονται σε έναν χάρτη.

Ο OGC προσδιορίζει ένα συντακτικό για URL's που καλεί κάθε μία από αυτές τις υπηρεσίες. Επιπλέον προσδιορίζεται μία γλώσσα XML για τα μεταδεδομένα που περιγράφουν τις δικτυακές υπηρεσίες.

Όταν ένας χρήστης ζητά από μία υπηρεσία WMS έναν χάρτη, τότε πρέπει να προσδιορίσει τα εξής:

- Ποια πληροφορία επιθυμεί να απεικονιστεί στον χάρτη (ένα ή παραπάνω θεματικά επίπεδα)
- Τον τρόπο (στυλ) απεικόνισης αυτών των χαρτών
- Το τμήμα της γήινης επιφάνειας που θα περιλαμβάνει ο χάρτης (bounding box)
- Το σύστημα προβολής ή συντεταγμένων
- Το είδος αρχείου εικόνας του χάρτη (πχ, jpeg, bmp, tiff, png)
- Τις διαστάσεις του χάρτη
- Την διαφάνεια και το χρώμα του φόντου

Όταν δύο ή περισσότεροι χάρτες έχουν το ίδιο περικλείον παραλληλόγραμμο, σύστημα αναφοράς και μέγεθος, τότε οι εικόνες τους μπορούν με ακρίβεια να υπερτεθούν έτσι ώστε να συνθέσουν ένα σύνθετο χάρτη. Η χρήση format εικόνας που επιτρέπουν διαφανή φόντο δίνει την δυνατότητα να είναι ορατά όλα τα επίπεδα. Επιπλέον τα επιμέρους επίπεδα ενός τέτοιου σύνθετου χάρτη μπορούν να προέρχονται από διαφορετικούς εξυπηρετητές (servers). Συνεπώς η προδιαγραφή του OGC επιτρέπει την δημιουργία ενός δικτύου από εξυπηρετητές που

προσφέρουν WMS, με την χρήση των οποίων οι χρήστες μπορούν να παράγουν σύνθετους χάρτες ανάλογα με τις ανάγκες τους.

Σε αντίθεση με τις κάθετα οργανωμένες ιστοσελίδες παροχής υπηρεσιών WMS, ένας πάροχος υπηρεσιών WMS δεν χρειάζεται να διακινεί μόνο την δική του συλλογή δεδομένων.

2.4 Το πρωτόκολλο Web Feature Service - Transactional

Στην προηγούμενη παράγραφο είδαμε πώς το OGC ορίζει το μοντέλο διεπαφών που υποστηρίζουν τους κανόνες υποβολής και ανταπόκρισης με την χρήση του Hypertext Transfer Protocol (Http) και της Geographical Markup Language (GML). Αυτό το πρότυπο επιτρέπει την ανάκτηση χαρτογραφικών δεδομένων που προέρχονται από εξυπηρετητές (συχνά πολλαπλούς) στο διαδίκτυο. Τα δεδομένα αυτά διακινούνται με την μορφή εικόνων-χαρτών. Σε ένα επόμενο βήμα το OGC θέσπισε τα πρότυπα Web Feature Service και Web Feature Service – Transactional. Αυτά τα πρότυπα ορίζουν τις προδιαγραφές με τις οποίες ένας χρήστης (client) μπορεί να περιγράφει και να αποστέλλει ερωτήματα (queries) και διεργασίες διαχείρισης δεδομένων (data manipulation operations) χρησιμοποιώντας πάλι το διαδίκτυο (πλατφόρμα HTTP) αλλά αυτή την φορά έτσι ώστε να του επιστρέφονται τα δεδομένα καθαυτά αντί για αναπαραστάσεις τους σε χάρτη όπως προηγουμένως. Οι διεργασίες μπορούν να περιλαμβάνουν τα εξής [OGC+02]:

- Επιλογή ή ερώτημα στοιχείων βασισμένη σε χωρικά και μη κριτήρια.
- Δημιουργία ενός νέου στοιχείου (feature instance)
- Διαγραφή ενός στοιχείου
- Ενημέρωση ενός στοιχείου

Η πρώτη από τις παραπάνω διεργασίες, αυτή δηλαδή που αφορά σε επιλογή ή ερώτημα στοιχείων βασισμένη σε χωρικά και μη κριτήρια, είναι υποχρεωτική δηλαδή πρέπει να υποστηρίζεται σε κάθε σύστημα WFS. Οι υπόλοιπες διεργασίες είναι προαιρετικές. Όταν ένα σύστημα WFS υποστηρίζει και τις προαιρετικές διεργασίες τότε λέμε ότι μπορεί να εκτελεί και συναλλαγές και ευθυγραμμίζεται με την προδιαγραφή WFS-T.

Οι Λειτουργίες WFS περιλαμβάνουν τις επιμέρους λειτουργίες: GetCapabilities (παροχή και λήψη πληροφόρησης για γεωγραφικές πληροφορίες που παρέχει ο εξυπηρετητής), DescribeFeatureType (πληροφόρηση για το σχήμα των γεωγραφικών δεδομένων) και GetFeature (αίτηση για παροχή γεωμ. οντοτήτων σε μορφή GML, λ.χ. όσες εμπίπτουν εντός δοθέντος ορθογωνίου).

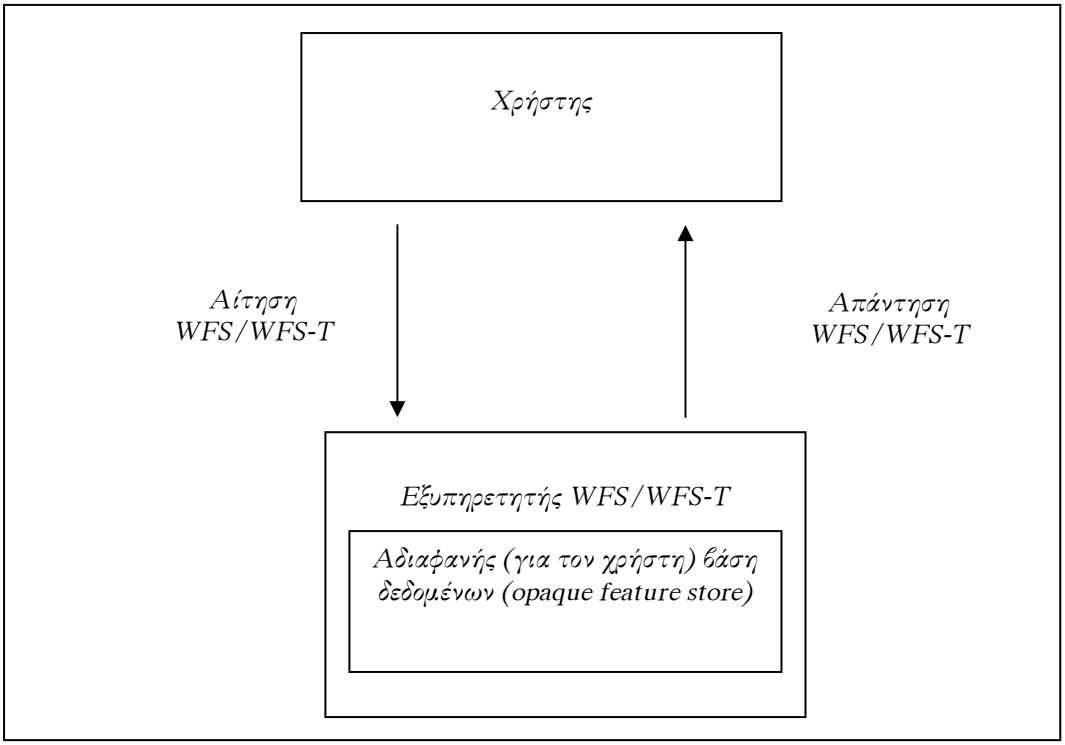
Μία αίτηση αποτελείται από την περιγραφή του ερωτήματος η του μετασχηματισμού των δεδομένων που θα εφαρμοστούν σε ένα ή περισσότερα στοιχεία. Η αίτηση δημιουργείται στο περιβάλλον του χρήστη και αποστέλλεται στον εξυπηρετητή μέσω πρωτοκόλλου HTTP είτε σαν εντολή URL (εφαρμόστηκε περισσότερο στις πρώτες εφαρμογές WFS/WFS-T) είτε σαν μήνυμα σε γλώσσα GML (Geography Markup Language). Η GML είναι παραλλαγή της γλώσσας XML που χρησιμοποιείται για την περιγραφή γεωγραφικών στοιχείων. Η γλώσσα αυτή, όπως περιγράφεται στις σχετικές προδιαγραφές του OGC, μπορεί να υποστηρίξει πέρα από απλούς τύπους

γεωμετρικών δεδομένων (σημεία, γραμμές, πολύγωνα) μέχρι και πιο σύνθετους τύπους (καμπύλες και τρισδιάστατες επιφάνειες) [GML+03]. Ο εξυπηρετητής γεωγραφικών στοιχείων διαβάζει και στην συνέχεια εκτελεί την αίτηση. Όταν η αίτηση εκτελεστεί, αποστέλλεται πίσω στον χρήστη η απάντηση που επιβεβαιώνει την επιτυχή εκτέλεση και περιλαμβάνει τα δεδομένα που ερωτήθηκαν, εισήχθησαν ή τροποποιήθηκαν. Οι πληροφορίες που αποτελούν την απάντηση και αποστέλλονται πίσω στον χρήστη είναι επίσης κωδικοποιημένες σε γλώσσα GML. Το πρότυπο WFS του OGC επιτρέπει σε έναν χρήστη να ανακτά γεωμετρικές και χωρικές πληροφορία από πολλαπλές υπηρεσίες WFS.

Σύμφωνα με το πρότυπο του OGC **[OGC+02]** σε ένα WFS πρέπει:

- Οι διεπαφές να ορίζονται σε κωδικοποίηση XML
- Να χρησιμοποιείται κωδικοποίηση GML για να εκφραστούν τα στοιχεία εντός των διεπαφών
- Η υπηρεσία πρέπει να μπορεί τουλάχιστον να παρουσιάσει γεωγραφικά στοιχεία σε κωδικοποίηση GML.
- Η βάση δεδομένων (datastore) δεν θα πρέπει να είναι ορατή στον χρήστη παρά μόνο μέσω της διεπαφής του WFS.
- Να χρησιμοποιείται ένα υποσύνολο εκφράσεων Xpath για τις ιδιότητες αναφοράς (referencing properties).

Το μοντέλο διεπαφής WFS που προσδιορίζεται από τον OGC σύμφωνα με τα παραπάνω κριτήρια είναι το εξής:



Σχήμα 2.1 . Ένα σύστημαWFS/WFS-T σύμφωνα με το πρότυπο OGC (Πηγή: **[OGC+02]**)

Κεφάλαιο 3

Τήρηση ιστορικού σε χωρικές βάσεις δεδομένων

Οι μηχανισμοί αποθήκευσης των αλλαγών που έγιναν σε διαφορετικές χρονικές στιγμές και στάδια εξέλιξης μίας βάσης δεδομένων καθώς και οι μηχανισμοί επιλεκτικής επικύρωσης αυτών των αλλαγών είναι παρόντες στα περισσότερα λογισμικά διαχείρισης χωρικών βάσεων δεδομένων. Οι προσεγγίσεις που έχουν ακολουθηθεί διαφέρουν ως προς τις δυνατότητες και την δομή των μηχανισμών που χρησιμοποιούν. Μία από τις πιο τεκμηριωμένες και εξελιγμένες προτάσεις για την τήρηση ιστορικού σε χωρικές βάσεις δεδομένων προβλέπει την δημιουργία επιπλέον πινάκων εντός της βάσης δεδομένων για την αποθήκευση των στοιχείων που εισάγονται, ενημερώνονται ή διαγράφονται. Με την χρήση αυτών των πινάκων ο διαχειριστής της βάσης δεδομένων μπορεί να επιλέξει ποιες από τις αλλαγές επιθυμεί να διατηρήσει και ποιες να αναιρέσει. Σε αυτό το κεφάλαιο αναλύουμε την χρησιμότητα και τον σκοπό των μηχανισμών τήρησης ιστορικού σε χωρικές βάσεις δεδομένων. Επιπλέον παραθέτουμε τα αποτελέσματα της σύντομης ανασκόπησης συστημάτων τήρησης ιστορικού και διαχείρισης εκδόσεων που παρατίθενται στην σχετική βιβλιογραφία. Στο τέλος περιγράφουμε τον μηχανισμό που περιλαμβάνεται σε λογισμικό χωρικής βάσης δεδομένων της ESRI.

3.1 Σκοπός και εφαρμογές τήρησης ιστορικού σε χωρικές βάσεις δεδομένων

Η συνθήκη της ταυτόχρονης προσπέλασης πολλών χρηστών σε μία χωρική βάση δεδομένων προϋποθέτει την τήρηση ιστορικού. Η τήρηση ιστορικού με την σειρά της εγείρει ζητήματα όπως αυτό της επικύρωσης των επεμβάσεων εκ μέρους των διαφορετικών χρηστών καθώς και άλλων εννοιών όπως αυτές της ανίχνευσης διαμάχης σε αλλαγές (conflict detection), ορισμού εκδόσεων των αντιγράφων των δεδομένων (posting versions), διαχείριση εκδόσεων (version administration) και ειδική επικύρωση περιπτώσεων (special reconcile cases). Οι τελευταίες από τις έννοιες που αναφέρθηκαν

δεν θα αποτελέσουν αντικείμενο εμβάθυνσης εδώ. Στην παρούσα εργασία θα εμβαθύνουμε στις βασικές έννοιες:

- των μηχανισμών της αποθήκευσης των ιστορικών αλλαγών (τήρησης ιστορικού)
- των μηχανισμών επικύρωσης αυτών των αλλαγών.

Ένα σύστημα διαχείρισης βάσεων δεδομένων πρέπει να παρέχει την δυνατότητα σε πολλούς χρήστες να επεμβαίνουν και να εισάγουν μεγάλο όγκο δεδομένων. Καθώς κάποιοι από αυτούς τους χρήστες μπορεί να επέμβουν ταυτόχρονα στο ίδιο στοιχείο δεδομένων το σύστημα διαχείρισης βάσεων δεδομένων θα πρέπει να παρέχει ένα περιβάλλον που να υποστηρίζει αυτή την ταυτόχρονη επέμβαση. Οι επεμβάσεις που κάνουν οι χρήστες της βάσεων δεδομένων είναι οργανωμένες σε λογικές ενότητες που καλούνται συναλλαγές (βλ. παράγραφο 2.1). Οι τελευταίες διακρίνονται σε σύντομες συναλλαγές (short transactions) και μακρές συναλλαγές (long transactions).

Οι σύντομες συναλλαγές τυπικά περιλαμβάνουν μικρές σε πλήθος επιμέρους τροποποιήσεων και χρονική διάρκεια αλλαγές γεγονός που επιτρέπει το κλείδωμα των δεδομένων ως μέσο διατήρησης της συνέπειας της βάσης δεδομένων.

Σε πολλές περιπτώσεις οι τροποποιήσεις σε γεωγραφικά δεδομένα περιλαμβάνουν ώρες, μέρες ή και μήνες για να ολοκληρωθούν. Τέτοιες εργασίες μπορούν να είναι η ενημέρωση ενός αρχείου κτηματολογίου ή η εισαγωγή ενός πολύπλοκού και εκτεταμένου τμήματος οδικού δικτύου. Αυτές οι μακρές συναλλαγές περιλαμβάνουν πολλές σύντομες συναλλαγές και επεμβάσεις από πολλαπλούς χρήστες. Η αντικειμενική αυτή συνθήκη δεν επιτρέπει το κλείδωμα των στοιχείων δεδομένων, την περιορισμό δηλαδή της δυνατότητας επέμβασης σε αυτά από έναν και μόνο χρήστη / συναλλαγή γιατί σε αυτή την περίπτωση η λειτουργία της βάσης δεδομένων θα καταστεί πολύ χρονοβόρα και η τροποποιήσεις στοιχείων ίσως να χρειάζεται να μπαίνουν σε καθεστώς αναμονής έως ότου ξεκλειδωθούν τα σχετικά δεδομένα. Σε αυτή την περίπτωση χρησιμοποιείται η τήρηση ιστορικού ως μέσο για την αποθήκευση στιγμιότυπων της βάσης δεδομένων σε διαφορετικές χρονικές στιγμές χωρίς όμως να δημιουργούνται πολλαπλά αντίγραφα των δεδομένων καθαυτών. Παρόλο που η απουσία κλειδώματος προφανώς οδηγεί σε διαμάχες ανάμεσα σε επιμέρους τροποποιήσεις, η τήρηση ιστορικού κάνει ευκολότερη την ανίχνευση και διαχείριση αυτών των διαφωνιών σε σχέση με τα πρωτόκολλα που κλειδώνουν επιμέρους στοιχεία δεδομένων.

Στην πραγματικότητα οι διαφωνίες ανάμεσα σε τροποποιήσεις δεδομένων είναι η εξαίρεση παρά ο κανόνας στις λειτουργίες μίας βάσης δεδομένων. Συνεπώς το υπολογιστικό έργο (overhead) που δαπανάται για την διαχείριση τους είναι μικρής σημασίας σε σχέση με τους περιορισμούς που επιβάλει το κλείδωμα δεδομένων ή το δυναμικό που πρέπει να δαπανηθεί για την δημιουργία τοπικών αντιγράφων.

Στην καθημερινή χρήση των περισσότερων χωρικών βάσεων δεδομένων (για παράδειγμα σε εταιρικές βάσεις δεδομένων ή σε αυτές που αναπτύχθηκαν για κάποιον δημόσιο οργανισμό) η ροή εργασίας προϋποθέτει την μετάβαση του αντικειμένου ή αντικειμένων που αναπαριστούν οι βάσεις δεδομένων από διαδοχικά στάδια όπως αυτό του σχεδιασμού, έγκρισης, υλοποίησης, ολοκλήρωσης κ.ο.κ. Σε αυτό το μοντέλο λειτουργίας οι χρήστες μπορούν να επιδρούν πάνω σε στοιχεία της βάσης δεδομένων. Αυτό όμως γίνεται χωρίς τα στοιχεία να κλειδώνονται και έτσι είναι δυνατή η ταυτόχρονη τροποποίησή τους από άλλους χρήστες που εργάζονται σε κάποιο άλλο στάδιο του έργου. Η λειτουργία της τήρησης ιστορικού σε μία χωρική βάση δεδομένων βασίζεται στην ύπαρξη μίας βασικής έκδοσης των δεδομένων που αναπαριστά τα δεδομένα που βρίσκονται αποθηκευμένα στην βάσης δεδομένων πριν από την έναρξη των τροποποιήσεων σε αυτά. Οι διαφορετικές εκδόσεις της βάσης δεδομένων που αντιστοιχούν σε διαφορετικά στάδια εξέλιξής της στην ουσία αποθηκεύονται σαν διαδοχικές ενότητες τροποποιήσεων που έχουν γίνει στην αρχική έκδοση των δεδομένων. Αν

αναπαραστήσουμε την εξέλιξη των δεδομένων της βάσης δεδομένων με ένα «δέντρο εκδόσεων» (version tree) όπου κάθε φύλλο αναπαριστά την κατάσταση των δεδομένων μετά από κάποια επέμβαση, τότε το αρχικό αντίγραφο βρίσκεται στην ρίζα αυτού του δέντρου και δεν έχει κάποιον γονέα. Όλες οι διαφορετικές εκδόσεις της βάσης δεδομένων προέρχονται τελικά από αυτό.

Έτσι, για παράδειγμα στο πρώτο στάδιο κάποιου έργου μία σειρά από αλλαγές στο αρχικό αντίγραφο μπορούν να αντανακλούν ένα προσχέδιο το οποίο ακόμα δεν έχει εγκριθεί. Ακόμα αυτό το προσχέδιο λόγω της προσωρινής φύσης του μπορεί να είναι ιδιωτικό (private) να είναι δηλαδή προσβάσιμο σε περιορισμένους χρήστες (πχ. μόνο στον δημιουργό του και τον διαχειριστή της βάσης δεδομένων). Όταν το σχέδιο αυτό γίνει δεκτό τότε μπορεί να δημιουργηθεί μία νέα έκδοση του που θα είναι προσβάσιμη σε περισσότερους χρήστες. Όμοια με νωρίτερα αλλαγές που θα γίνονται σε αυτό το τελευταίο αντίγραφο μπορεί να μην είναι ορατές στους υπόλοιπους χρήστες μέχρι και αυτές με την σειρά τους να γίνουν αποδεκτές. Σε ένα επόμενο στάδιο (πχ αυτό της κατασκευής του έργου) μία νέα έκδοση της βάσης δεδομένων μπορεί να δημιουργηθεί. Όμοια, στην αρχή της κατασκευής του υποθετικού έργου θα οριστεί μία νέα έκδοση των δεδομένων με τυχόν αλλαγές που προέκυψαν σε αυτό το στάδιο. Όταν το έργο πλέον ολοκληρωθεί μία νέα έκδοση θα καταγράψει όλες τις αλλαγές που έγιναν κατά την κατασκευή. Αυτή η τελευταία έκδοση μπορεί πλέον να ενσωματωθεί στο αρχικό αντίγραφο. Οι προηγούμενες εκδόσεις μπορούν πλέον να αρχειοθετηθούν ή να διαγραφούν. Συνολικά η τήρηση ιστορικού σε χωρικές βάσεις δεδομένων υποστηρίζει μία σειρά από λειτουργίες:

- Υποστηρίζει πολλαπλούς χρήστες που μπορούν να τροποποιούν τα ίδια στοιχεία δεδομένων συγχρονικά
- Επιτρέπει την συγχρονική επέμβαση πολλαπλών χρηστών χωρίς να δημιουργεί πολλαπλά αντίγραφα του συνόλου των δεδομένων
- Επιτρέπει την δημιουργία πολλαπλών δομών και εναλλακτικών διευθετήσεων της βάσης δεδομένων (alternative database designs) χωρίς να επηρεάζονται τα αρχικά / δημοσιευμένα (default/published) χωρικά δεδομένα.
- Υποστηρίζει την δυνατότητα ανάκτησης ιστορικών εκδόσεων των δεδομένων (historical data versions) στις οποίες μπορούν να υποβληθούν ερωτήματα έτσι ώστε να ανακτηθούν τα δεδομένα σε οποιαδήποτε χρονική στιγμή **[ESR+04]**.

3.2 Επισκόπηση συστημάτων τήρησης ιστορικού και διαχείρισης εκδόσεων δεδομένων

Πολλοί οργανισμοί αναπτύσσουν ή μελετούν την ανάπτυξη λογισμικού που να επιτρέπει την διατήρηση και διαχείριση ιστορικού ή διαφορετικών εκδόσεων δεδομένων σε χωρικές βάσεις δεδομένων. Σε αυτό το κεφάλαιο θα μελετήσουμε την πιο αναπτυγμένη και καλά τεκμηριωμένη πρόταση που αφορά σε λογισμικό της ESRI. Πριν από αυτό όμως θα συνοψίσουμε τα χαρακτηριστικά των βασικών εφαρμογών συστημάτων τήρησης ιστορικού που συναντήσαμε κατά την ανασκόπηση της σχετικής βιβλιογραφίας.

Σύμφωνα με την ιστοσελίδα του «The open planning project», φορέα που αναπτύσσει τον εξυπηρετητή χωρικών δεδομένων GeoServer, υπάρχει ένα σχέδιο ανάπτυξης λειτουργιών τήρησης ιστορικού για το συγκεκριμένο λογισμικό. Οι προδιαγραφές μέχρι στιγμής προβλέπουν την δυνατότητα χρήσης του συστήματος για αναίρεση συναλλαγών και έλεγχο επεμβάσεων εκ μέρους πολλαπλών χρηστών σε περιβάλλον συναλλαγών που διέπεται από το πρότυπο WFS-T. Η ροή εργασίας

προβλέπει την αποδοχή των αλλαγών που γίνονται από τους χρήστες από τον διαχειριστή της βάσης δεδομένων. Η εφαρμογή θα διατίθεται σε μορφή ομάδας επεκτάσεων του λογισμικού (plug-ins) που ανάμεσα σε άλλα θα αφορούν στον έλεγχο της ορθότητας των επεμβάσεων στις οποίες προβαίνουν οι χρήστες ή την ενημέρωση του διαχειριστή για τις αλλαγές με μήνυμα ηλεκτρονικού ταχυδρομείου. Η αρχιτεκτονική του συστήματος θα περιλαμβάνει εκδόσεις δεδομένων και επιπλέον πίνακες που θα ταξινομούν τις διαφορετικές εκδόσεις των δεδομένων και θα χρησιμοποιούν χρονόσημα για να προσδιορίσουν την στιγμή τελευταίας τροποποίησης κάθε έκδοσης. Οι χρήστες που εργάζονται σε μία έκδοση θα μπορούν να υποβάλλουν ερωτήματα σε αυτούς τους πίνακες προκειμένου να διερευνήσουν αν η έκδοση έχει τροποποιηθεί από κάποιον άλλο χρήστη μετά από την στιγμή που και σε αυτή την περίπτωση να ενημερώσουν την έκδοση που βλέπουν. Οι αλλαγές κάθε χρήστη επικυρώνονται όταν ο χρήστης αποθηκεύει την έκδοση του στον εξυπηρετητή ενώ παρέχεται η δυνατότητα οι αλλαγές να μπαίνουν σε ουρά αναμονής μέχρι να γίνουν αποδεκτές από τον διαχειριστή ή κάποιον άλλο χρήστη [GER08].

Ο οργανισμός Refractions που αναπτύσσει την επέκταση χωρικών λειτουργιών PostGIS χρησιμοποιεί το σύστημα διαχείρισης χωρικής βάσης δεδομένων PosgreSQL/PostGIS για να διαχειριστεί δεδομένα οδικού δικτύου (Digital Road Atlas - DRA) της περιοχής British Columbia στον Καναδά. Σε αυτή την βάση δεδομένων έχει αναπτυχθεί σύστημα τήρησης ιστορικού. Το σύστημα βασίζεται στην αρχή ότι οι εισαγωγές και τροποποιήσεις δεδομένων προσθέτουν νέες εγγραφές στην βάση δεδομένων και δεν τροποποιούν τις υπάρχουσες. Οι διαγραφές δεν αφαιρούν εγγραφές αλλά τροποποιούν την τιμή του δυαδικού γνωρίσματος «most_ recent» στην εγγραφή που διαγράφεται. Οι τροποποιήσεις προσθέτουν μία νέα εγγραφή με όλα τα χαρακτηριστικά που έχει το στοιχείο που ενημερώνεται αλλά η εγγραφή με την μορφή που είχε νωρίτερα διατηρείται στην βάση δεδομένων. Η κλήση παλαιότερων στιγμιότυπων της βάσης γίνεται με την χρήση γνωρίσματος που έχει την μορφή χρονοσήμου και προσδιορίζει την στιγμή που εισήχθη ή τροποποιήθηκε κάθε εγγραφή της βάσης δεδομένων. Η πλέον ενημερωμένη έκδοση των δεδομένων ανακτάται και ερωτάται με την χρήση του γνωρίσματος «most recent». Στιγμιότυπα των δεδομένων σε στιγμές του παρελθόντος ανακτώνται με την χρήση του γνωρίσματος χρονοσήμου. Αντίθετα με τις περισσότερες εφαρμογές τήρησης ιστορικού και παρόμοια με την εφαρμογή που αναπτύσσεται στην παρούσα εργασία το σύστημα DRA μπορεί να ανακτά στιγμιότυπα της χωρικής βάσης δεδομένων σε οποιαδήποτε στιγμή του παρελθόντος και όχι αποκλειστικά σε προκαθορισμένες χρονικές στιγμές. Το σύστημα έχει υλοποιηθεί με την χρήση λειτουργιών που γράφτηκαν στην γλώσσα προγραμματισμού C++ [DRA+03].

Ένα από τα πρώτα λογισμικά διαχείρισης χωρικών βάσεων δεδομένων που αναπτύχθηκαν ήταν το σύστημα Smallworld που αναπτύχθηκε από την General Electric για να υποστηρίξει (αρχικά) εφαρμογές πληροφοριών δικτύων ηλεκτροδότησης. Πρόκειται για την πρώτη εφαρμογή στην οποία αναγνωρίστηκε η εγγενής ανάγκη των χωρικών δεδομένων για εκτεταμένες στον χρόνο συναλλαγές (long transactions) που εν συνεχεία οδήγησε στην παραγωγή και διαχείριση πολλαπλών εκδόσεων των δεδομένων. Επιπλέον κατά την παραγωγή και χρήση αυτού του προγράμματος αναπτύχθηκε η δυνατότητα για την

παραχώρηση (check out) και επανακαταχώρηση (check in) δεδομένων από την χωρική βάση δεδομένων σε φορητά περιβάλλοντα χρήστη, δηλαδή σε φορητούς υπολογιστές που είχαν μαζί τους στο πεδίο οι τεχνικοί και μηχανικοί ηλεκτρικών δικτύων. Αυτά τα φορητά τοπικά συστήματα χρησιμοποιούνταν προκειμένου να ανακτώνται πληροφορίες καθώς και να ενημερώνονται τα δεδομένα σχετικά με τις διάφορες επεμβάσεις. Η λειτουργία του συστήματος βασίζεται στην δημιουργία πολλαπλών εκδόσεων των δεδομένων. Κάθε έκδοση δεδομένων είναι στην ουσία μία περιγραφή μεταβολών του συνόλου της βάσης δεδομένων (a delta of the entire database), δηλαδή αποτελείται από δεδομένα που περιγράφουν αλλαγές στα περιεχόμενα της βάσης δεδομένων. Οι εκδόσεις είναι αρχικά σχεδόν κενά αρχεία το μέγεθος των οποίων αυξάνεται μαζί με τον αριθμό επεμβάσεων που περιλαμβάνεται σε αυτές. Όταν μία μακρά συναλλαγή ξεκινά τότε δημιουργείται μία νέα έκδοση δεδομένων με σκοπό να ενσωματώσει όλες τις επεμβάσεις που θα συμβούν κατά την διάρκεια της συναλλαγής. Οι χρήστες που κάνουν τις συναλλαγές μπορούν να αποστείλουν τις αλλαγές που έχουν κάνει στο τοπικό περιβάλλον τους στην βάση δεδομένων (post) και να ενσωματώσουν στην έκδοση που επεξεργάζονται τις αλλαγές που ενδεχομένως έχουν εντωμεταξύ γίνει από άλλους χρήστες (merge). Στην περίπτωση αντιμαχόμενων επεμβάσεων αυτές μπορούν είτε να επιλύονται αυτόματα με την χρήση κανόνων (για παράδειγμα υπερισχύουν οι αλλαγές που βρίσκονται στις εκδόσεις γονείς) ή μπαίνουν σε καθεστώς αναμονής προκειμένου να επιλυθούν από κάποιον διαχειριστή [GEP+04].

Η εταιρία Oracle έχει αναπτύξει το σύστημα workspace manager για την αποτελεσματική διαχείριση διαφορετικών εκδόσεων δεδομένων στην χωρική βάση δεδομένων Oracle Spatial. Το μοντέλο δεδομένων που εφαρμόζει το εν λόγω λογισμικό διαχείρισης χωρικής βάσης δεδομένων αποθηκεύει και στοιχεία τοπολογίας. Αυτό υλοποιείται με την χρήση και τήρηση ιστορικού σε επιπλέον πίνακες εντός της βάσης δεδομένων. Οι πίνακες αυτοί αποθηκεύουν πληροφορίες σχετικά με τις τοπολογικές σχέσεις που συνδέουν τα χωρικά στοιχεία δεδομένων. Παράδειγμα τοπολογικών σχέσεων που υποστηρίζονται είναι οι σχέσεις «περιέχει» (contains), «εντός» (inside), «καλύπτει» (covers), «καλύπτεται» (covered by), «αγγίζει» (touch) και «καλύπτεται ενώ τα όρια τέμνονται» (overlap with boundaries intersecting). Οι τοπολογικές σχέσεις παραμένουν αμετάβλητες κατά τη διάρκεια μεταβολών στην γεωμετρία των δεδομένων ή το προβολικό σύστημα. Η σημασία τους αναδεικνύεται σε ένα περιβάλλον έντονων μεταβολών όπου ταυτόχρονα υπάρχει μεγάλη ανάγκη για ακεραιότητα και ορθότητα των δεδομένων. Επιπλέον αυτή η μέθοδος αποθήκευσης της τοπολογίας δίνει την δυνατότητα για σημαντική αύξηση στην ταχύτητα εκτέλεσης σχετικών ερωτημάτων. Το σημαντικό με το σύστημα τήρησης ιστορικού της Oracle είναι ότι παρέχεται η δυνατότητα τήρησης εκδόσεων δεδομένων που αφορούν εκτός των χωρικών δεδομένων καθαυτών, στις τοπολογικές σχέσεις ανάμεσα στα στοιχεία χωρικών δεδομένων. Η διαχείριση των εκδόσεων τοπολογίας γίνεται σε περιβάλλοντα εργασίας (workspaces) που στην ουσία είναι εικονικά περιβάλλοντα που απομονώνουν μία σειρά αλλαγών σε μία ή περισσότερες τοπολογίες, διατηρούν πληροφορίες σχετικά με το ιστορικό των αλλαγών και επιτρέπουν την δημιουργία διαφορετικών σεναρίων επέμβασης στα δεδομένα [ORA+07].

Η εφαρμογή Transaction Manager της GeoMedia ανήκει στις εφαρμογές που ακολουθούν το μοντέλο μακρών συναλλαγών. Συνεργάζεται με την χωρική βάση δεδομένων Oracle. Το λογισμικό παρέχει ένα ελεγχόμενο περιβάλλον για την πραγματοποίηση μακρών συναλλαγών. Τα τροποποιημένα η διαγραμμένα δεδομένα παραμένουν στην βάση δεδομένων προκειμένου να χρησιμοποιηθούν για την ανάκτηση στιγμιότυπων του παρελθόντος. Επιπλέον, η διεπαφή του προγράμματος επιτρέπει την εύκολη υποβολή ερωτημάτων που να απεικονίζουν εύγλωττα την διαχρονική εξέλιξη των δεδομένων. Οι συγχρονικές αλλαγές ρυθμίζονται από ένα σχήμα ελέγχου συγχρονικότητας το οποίο κατορθώνει να κλειδώνει τα στοιχεία ακριβώς πρίν από την τροποποίησή τους χωρίς να κλειδώνει άσκοπα άλλα δεδομένα που δεν πρόκειται να τροποποιηθούν αλλά βρίσκονται κοντά στην περιοχή επέμβασης [GET+04].

Η τεχνολογία ArcSDE της εταιρίας ESRI βασίζεται στην ίδια αρχή διαχείρισης των επεμβάσεων σε εκδόσεις δεδομένων. Οι εισηγμένες, διαγραμμένες ή τροποποιημένες εγγραφές αποθηκεύονται σε ένα ζεύγος πινάκων και οι πληροφορίες για αυτές τις επεμβάσεις σε ένα σύστημα πινάκων. Όλοι αυτοί οι πίνακες αποθηκεύονται εντός της βάσης δεδομένων. Η τροποποίηση των δεδομένων γίνεται σε στάδια κάθε ένα από τα οποία αφορά στις επεμβάσεις που κάνει ένας χρήστης σε μία έκδοση των δεδομένων. Όταν ένας χρήστης επιθυμεί να αποθηκεύσει τις επεμβάσεις που έχει κάνει τότε αυτές επικυρώνονται με τα περιεχόμενα της βάσης δεδομένων και προκύπτει μία νέα έκδοση των δεδομένων. Όπως και νωρίτερα οι διαμάχες αντιμετωπίζονται με κανόνες ή επιλύονται από τον διαχειριστή της βάσης δεδομένων. Αντίστοιχα παρέχεται η δυνατότητα για παραχώρηση και επανακαταχώρηση των δεδομένων σε τοπικά συστήματα [ESR+04]. Το μοντέλο τήρησης ιστορικού που εφαρμόζεται στο λογισμικό της ESRI αναπτύσσεται διεξοδικά στην συνέχεια του κεφαλαίου.

Στην επισκόπηση που προηγήθηκε διακρίνονται δύο γενικά μοντέλα τήρησης ιστορικού: Το μοντέλο μακρών συναλλαγών (long transactions model - GeoServer, Smallworld, Oracle Spatial Workspace Manager, Geomedia Transaction Manager, ESRI ArcSDE) βασίζεται στην λειτουργία «εκδόσεων» που αντιπροσωπεύουν τροποποιήσεις στην έκδοση «γονέα». Το μοντέλο της «ρηχής» τήρησης ιστορικού (shallow versioning model – PostGIS BCDRA) αποθηκεύει τις τροποποιήσεις σαν νέες εγγραφές. Στις εγγραφές δίνεται γνώρισμα χρονοσήμου που αντανακλά την χρονική στιγμή εισαγωγής ή τροποποίησης. Ερωτήματα που αφορούν το σύνολο των εγγραφών επιστρέφουν ιστορικά στιγμιότυπα της βάσης δεδομένων. Συγκριτικά μπορούμε να συνοψίσουμε τα εξής:

- Σαν μειονέκτημα του μοντέλου μακρών συναλλαγών καταγράφεται ή έλλειψη ευελιξίας που σχετίζεται με το γεγονός ότι πρέπει να αποτελεί μέρος της αρχικής σχεδίασης της βάσης δεδομένων. Δεν είναι δυνατόν γι' αυτό το μοντέλο να προστεθεί αργότερα σε μία βάση δεδομένων που δεν διαθέτει τον κατάλληλο σχεδιασμό. Αντίθετα, το μοντέλο «ρηχής» τήρησης ιστορικού είναι περισσότερο ευέλικτο και μπορεί να εφαρμοστεί στις περισσότερες «συμβατικές» βάσεις δεδομένων.
- Τα συστήματα που υπάγονται στο μοντέλο «ρηχής» τήρησης ιστορικού παρέχουν περισσότερες δυνατότητες ανάκτησης στιγμιότυπων της χωρικής βάσης δεδομένων. Αυτά τα συστήματα μπορούν να ανακτούν στιγμιότυπα της χωρικής βάσης δεδομένων σε οποιαδήποτε στιγμή του παρελθόντος και

όχι αποκλειστικά σε προκαθορισμένες χρονικές στιγμές. Αντίθετα τα συστήματα που υπάγονται στο μοντέλο μακρών συναλλαγών μπορούν να ανακτούν προκαθορισμένα στάδια της βάσης δεδομένων που αντιστοιχούν σε συγκεκριμένες χρονικές στιγμές του παρελθόντος.

- Από την άλλη πλευρά, τα συστήματα που υπάγονται στο μοντέλο μακρών συναλλαγών είναι ικανά να εκτελούν ερωτήματα και άλλες λειτουργίες δαπανώντας λιγότερο υπολογιστικό φόρτο. Για παράδειγμα λόγω του γεγονότος ότι στα συστήματα «ρηχής» τήρησης ιστορικού όλα τα δεδομένα αποθηκεύονται στο ίδιο σύνολο πινάκων εμφανίζεται η ανάγκη για δημιουργία μία σειράς από δυναμικά στοιχεία όπως όψεις (views) προκειμένου να αναπαρασταθούν βασικά στιγμιότυπα της βάσης δεδομένων που είναι απαραίτητα για την λειτουργία του συστήματος. Έτσι τα συστήματα τήρησης που βασίζονται σε αυτό το μοντέλο τείνουν να είναι πιο απαιτητικά και ακριβά σε υπολογιστικούς πόρους. Εξαιτίας της υποβόσκουσας πολυπλοκότητας που προϋποθέτουν τα συστήματα «ρηχής» τήρησης ιστορικού ο υπολογιστικός φόρτος αυξάνει δυσανάλογα με την σταδιακή αύξηση του αριθμού των συγχρονικών επεμβάσεων. Επιπλέον, διαχειριστικές εργασίες όπως ο καθαρισμός ή η συμπίεση δεδομένων είναι πιο πολύπλοκες σε τέτοια συστήματα.
- Οι πραγματικότητα των χωρικών βάσεων δεδομένων συχνά περιλαμβάνει αλλαγές στο σχήμα των δεδομένων. Τα συστήματα που βασίζονται στο μοντέλο μακρών συναλλαγών μπορούν να διαχειριστούν επεμβάσεις στο μοντέλο δεδομένων ανάμεσα σε διαφορετικές εκδόσεις δεδομένων. Αντίθετα, σε συστήματα «ρηχής» τήρησης ιστορικού κάτι τέτοιο δεν είναι δυνατό [GEP+04].

Συνολικά, το μοντέλο μακρών συναλλαγών μπορεί να αποδώσει καλύτερα σε περιβάλλον πολλαπλών χρηστών αφού είναι καταρχήν προσανατολισμένο στην διαχείριση χρονοβόρων συναλλαγών με ενδεχόμενες διαμάχες. Επιπλέον αυτό το μοντέλο μπορεί εγγενώς να διαχειρίζεται δεδομένα σε πολλαπλές εκδόσεις. Κάτι τέτοιο αποτελεί πλεονέκτημα καθώς οι διαφορετικοί οργανισμοί που αποθηκεύουν χωρικά δεδομένα συχνά επιδιώκουν την ταυτόχρονη τήρηση δεδομένων σε περισσότερες από μία εκδόσεις. Αντίθετα το μοντέλο «ρηχής» τήρησης ιστορικού είναι σχεδιασμένο με στόχο την αποθήκευση των ιστορικών επεμβάσεων καθαυτή και όχι την λειτουργία σε περιβάλλον ταυτόχρονων μακρών συναλλαγών και διαφορετικών εκδόσεων δεδομένων.

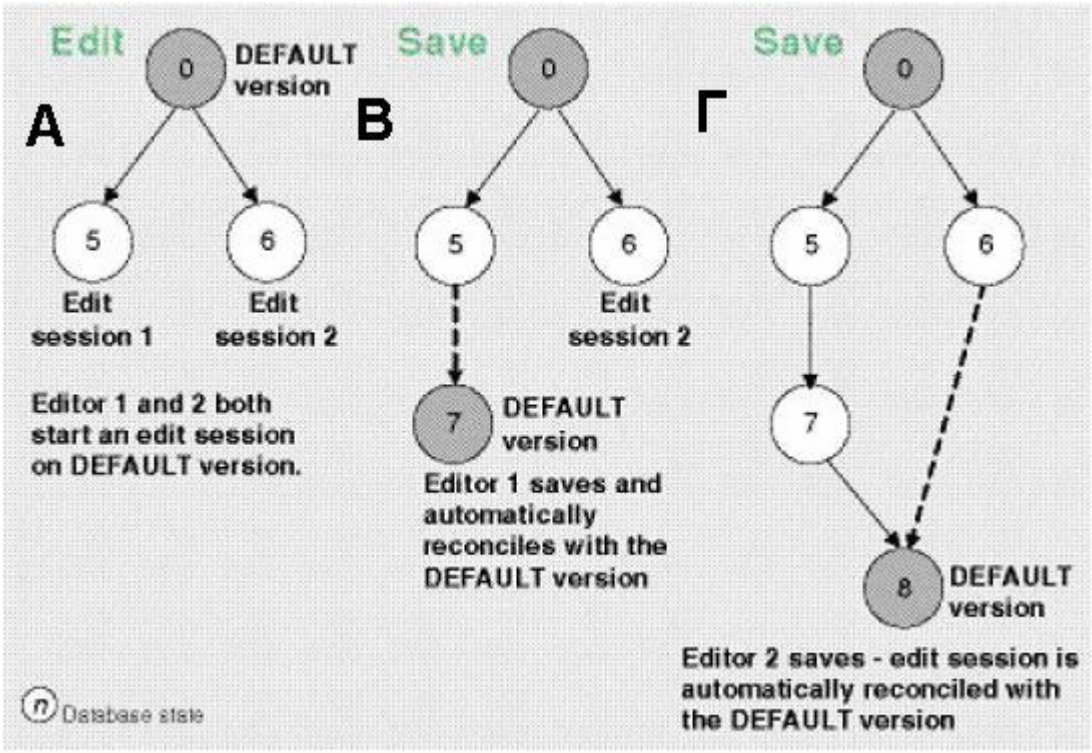
3.3 Τήρηση ιστορικού με την χρήση βοηθητικών πινάκων

Όπως αναφέρθηκε στην προηγούμενη παράγραφο το σύστημα τήρησης ιστορικού σε μία χωρική βάση δεδομένων δίνει την δυνατότητα για ανάκτηση των δεδομένων με την μορφή που είχαν σε διαφορετικές ιστορικές στιγμές στο παρελθόν. Η ESRI [ESR+04] έχει αναπτύξει ένα σύστημα τήρησης ιστορικού μεταβολών σε χωρική βάση δεδομένων το οποίο κάνει χρήση βοηθητικών πινάκων για την αποθήκευση των επεμβάσεων που γίνονται στα δεδομένα από τους διαφορετικούς χρήστες. Το σύστημα αυτό αντιμετωπίζει το παρελθόν της βάσης δεδομένων όχι σαν συνεχή ιστορικά περίοδο από την οποία μπορεί να ανακτηθεί οποιαδήποτε στιγμή αλλά το κατακερματίζει σε στάδια εξέλιξης της βάσης

δεδομένων κάποια από τα οποία έχουν συμβεί σε παράλληλες χρονικές στιγμές. Κάθε στάδιο της βάσης δεδομένων είναι δυνατό να αποτελείται από μία έκδοση τροποποίησης δηλαδή μία έκδοση των δεδομένων που δημιουργείται προκειμένου να επέμβει στα δεδομένα ένας χρήστης ή από μία βασική έκδοση (default version) δηλαδή μία έκδοση που προέρχεται από τα αρχικά δεδομένα επικυρωμένα με τις εκδόσεις τροποποίησης που έχουν ολοκληρωθεί.

Η επέμβαση σε αυτό το μοντέλο τήρησης ιστορικού συμβαίνει σε ενότητες επεμβάσεων (edit sessions) κάθε μία από τι οποίες περιλαμβάνει μία συναλλαγή και αριθμό τροποποιήσεων (edits) και αντιπροσωπεύει ένα στάδιο εξέλιξης της βάσης δεδομένων. Είναι δυνατό να πραγματοποιούνται παράλληλα πολλαπλές ενότητες τροποποιήσεων από διαφορετικούς χρήστες. Κάθε ενότητα τροποποίησης επεξεργάζεται τη δική της εσωτερική αναπαράσταση της δεδομένων που καλούμε έκδοση των δεδομένων. Οι ταυτόχρονες επεμβάσεις στα δεδομένα δεν είναι ορατές ανάμεσα στους χρήστες που τις πραγματοποιούν μέχρις ότου να ολοκληρωθούν οι επιμέρους ενότητες τροποποιήσεων. Όταν μία ενότητα τροποποιήσεων τελειώνει και οι αλλαγές που έγιναν κατά την διάρκειά της αποθηκευθούν, τότε τα περιεχόμενα της προσωρινής έκδοσης της χωρικής βάσης δεδομένων πάνω στα οποία έγιναν οι αλλαγές αυτόματα επικυρώνονται με την κύρια αναπαράσταση της έκδοσης. Το αποτέλεσμα αυτής της επικύρωσης είναι η νέα βασική έκδοση των δεδομένων. Τότε πλέον οι ταυτόχρονα συνδεδεμένοι χρήστες μπορούν εφόσον ανανεώσουν την σύνδεσή τους με την βάση δεδομένων να δουν τις επεμβάσεις που έχουν γίνει στο σύνολο των δεδομένων. Για τους χρήστες που θα ξεκινήσουν τώρα μία νέα ενότητα τροποποίησης η αντίστοιχη αναπαράσταση των δεδομένων θα έχει αρχικά την μορφή αυτής της βασικής έκδοσης.

Παράδειγμα: Στο σχήμα 3.1 Φαίνεται διαγραμματικά η συμπεριφορά διαχείρισης /επικύρωσης των διαφορετικών εκδόσεων των δεδομένων. Από αριστερά προς τα δεξιά διακρίνονται τα εξής διαδοχικά βήματα: Α) Δύο χρήστες δημιουργούν αντίστοιχες εκδόσεις των δεδομένων (5 και 6) οι οποίες προέρχονται από την βασική έκδοση των δεδομένων (0). Β) Ο χρήστης 1 αποθηκεύει τις επεμβάσεις του και έτσι το περιεχόμενο της έκδοσης των δεδομένων που έχει επεξεργαστεί επικυρώνεται με την αρχική έκδοση (έκδοση 0). Το αποτέλεσμα της επικύρωσης (έκδοση 7) είναι πλέον η νέα βασική έκδοση. Γ) Όταν και ο δεύτερος χρήστης αποθηκεύσει τις επεμβάσεις που έχει κάνει τότε αυτές δεν θα επικυρωθούν με την έκδοση γονέα (έκδοση 0) αλλά με την νέα βασική έκδοση των δεδομένων (έκδοση 7). Το αποτέλεσμα αυτής της επικύρωσης θα είναι η νέα βασική έκδοση των δεδομένων (έκδοση 8).



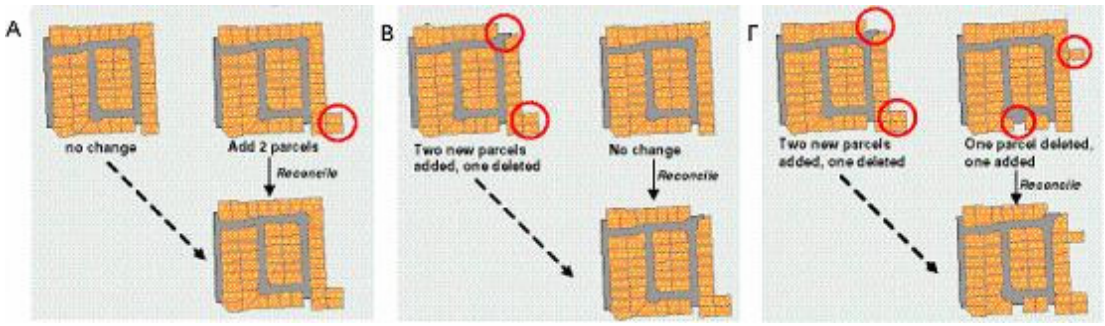
Σχήμα 3.1 Η συμπεριφορά διαχείρισης /επικύρωσης των διαφορετικών εκδόσεων των δεδομένων σε λογισμικό διαχείρισης χωρικής βάσης δεδομένων της ESRI (Πηγή: [ESR+04]).

Προκειμένου να επικυρωθούν οι αλλαγές που γίνονται σε μία έκδοση δεδομένων της βάσης δεδομένων που τροποποιήθηκε με την βασική έκδοση οι δύο εκδόσεις πρέπει να συμφιλιωθούν (reconciliate). Η επικύρωση διέπεται από κανόνες που προσδιορίζουν πως θα διαχειριστεί το σύστημα τις εκδόσεις που πρέπει να συμφιλιωθούν:

- Όταν η βασική έκδοση δεν έχει τροποποιηθεί τότε το αποτέλεσμα της συμφιλίωσης της με την έκδοση που τροποποιείται θα είναι το αποτέλεσμα της ενότητας επεμβάσεων.
- Όταν η έκδοση στόχος έχει τροποποιηθεί πριν από την διαδικασία συμφιλίωσης ενώ η έκδοση που βρίσκεται σε ενότητα επεμβάσεων δεν έχει τροποποιηθεί τότε προτεραιότητα έχουν οι αλλαγές που έγιναν στην έκδοση στόχο. Αυτές οι αλλαγές εκτός από προσθέσεις στοιχείων που δεν διαφωνούν με τα στοιχεία της έκδοσης επεμβάσεων μπορούν ακόμα να περιλαμβάνουν τροποποιήσεις στοιχείων που δεν τροποποιήθηκαν στην έκδοση τροποποίησης καθώς και διαγραφές στοιχείων που δεν διαγράφηκαν στην έκδοση τροποποίησης.
- Όταν τόσο η βασική έκδοση όσο και η έκδοση που βρίσκεται σε ενότητα επεμβάσεων έχουν τροποποιηθεί τότε το αποτέλεσμα της συμφιλίωσης θα είναι η συγχώνευση των αλλαγών που έχουν γίνει στις δύο αυτές εκδόσεις και δεν διαφωνούν. Οι αλλαγές που διαφωνούν επισημαίνονται και συμφιλιώνονται αυτόματα είτε χειροκίνητα. Αξίζει να αναφέρουμε ότι κατά την διάρκεια μίας ενότητας επεμβάσεων δεν επιτρέπονται μεταβολές στο σχήμα της έκδοσης στόχου.

Παράδειγμα: Στο σχήμα 3.2 φαίνεται διαγραμματικά η συμπεριφορά διαχείρισης διαμαχών ανάμεσα σε εκδόσεις των δεδομένων σε λογισμικό χωρικής βάσης δεδομένων ESRI. Από αριστερά προς τα δεξιά διακρίνονται οι εξής περιπτώσεις: Α) Όταν η έκδοση στόχος δεν έχει τροποποιηθεί Β) Όταν η έκδοση στόχος έχει τροποποιηθεί πριν από την διαδικασία συμφιλίωσης ενώ η έκδοση που βρίσκεται σε ενότητα επέμβασης δεν έχει τροποποιηθεί Γ) Όταν τόσο η έκδοση στόχος όσο και η έκδοση που βρίσκεται σε ενότητα επεμβάσεων έχουν τροποποιηθεί. Σε κάθε μία από τις τρεις περιπτώσεις που

απεικονίζονται εδώ: το πάνω αριστερά σχήμα αντιπροσωπεύει την έκδοση στόχο, το πάνω δεξιά την έκδοση που βρίσκεται σε ενότητα επεμβάσεων και το κάτω δεξιά το αποτέλεσμα της συμφιλίωσης.



Σχήμα 3.2 Η συμπεριφορά διαχείρισης διαμαχών ανάμεσα σε εκδόσεις των δεδομένων σε λογισμικό χωρικής βάσης δεδομένων της ESRI (Πηγή: **[ESR+04]**).

Δύο κατηγορίες διαμάχης στις επεμβάσεις μπορούν να σχετίζονται:

- Με διαφορετική τροποποίηση του ίδιου στοιχείου σε κάθε έκδοση
- Με την περίπτωση όπου το ίδιο στοιχείο που έχει τροποποιηθεί σε μία έκδοση έχει διαγραφεί σε μία άλλη.

Το λογισμικό επιδεικνύει ορισμένη συμπεριφορά (default behaviour) σε τέτοιου είδους διαμάχες επεμβάσεων:

Η αναπαράσταση ενός στοιχείου που βρίσκεται στην βασική έκδοση υπερισχύει αυτής που βρίσκεται στην έκδοση της ενότητας επεμβάσεων. Αν πολλαπλές ενότητες επεμβάσεων λειτουργούν ταυτόχρονα και διαπιστωθεί διαμάχη ανάμεσα σε στοιχεία τότε αυτή που αποθηκεύεται πρώτη γίνεται αναπαράσταση του συγκεκριμένου στοιχείου στην βασική έκδοση και έτσι υπερισχύει της αναπαράστασης του στοιχείου που τροποποιήθηκε στις άλλες εκδόσεις. Ο χρήστης προτρέπεται να επιθεωρεί τα στοιχεία όπου παρατηρούνται διαμάχες και να τις επιλύει [ESR+04].

Εσωτερικά το σύστημα τήρησης ιστορικού υλοποιείται με την δημιουργία πινάκων πλέον των πινάκων δεδομένων και των λοιπών πινάκων συστήματος. Πιο σημαντικοί από αυτούς είναι το ζεύγος των βοηθητικών πινάκων στο οποίο αποθηκεύονται οι εγγραφές που εισάγονται τροποποιούνται ή διαγράφονται. Συγκεκριμένα, ο πίνακας A<Registration_ID> (Adds table) αποθηκεύει πληροφορίες για τις εγγραφές που εισάγονται ή ενημερώνονται σε έναν πίνακα για τον οποίο έχει ενεργοποιηθεί η τήρηση ιστορικού. Πραγματοποιώντας ερωτήματα σε αυτόν τον πίνακα είναι δυνατό να ανακτήσουμε πληροφορίες σχετικά με το ποιες πληροφορίες έχουν εισαχθεί ή τροποποιηθεί σε κάποιο στάδιο της χωρικής βάσης δεδομένων. Ο πίνακας περιλαμβάνει όλα τα γνωρίσματα του σχετικού πίνακα για τον οποίο έχει ενεργοποιηθεί η τήρηση ιστορικού. Επιπρόσθετα περιλαμβάνει ένα μοναδικό για κάθε εγγραφή αύξοντα αριθμό και ένα γνώρισμα που προσδιορίζει το στάδιο κατά το οποίο εισήχθη ή τροποποιήθηκε η εγγραφή. Για παράδειγμα όταν εισάγεται μία νέα εγγραφή στον βασικό πίνακα εισάγεται στον συγκεκριμένο πίνακα μεταβολής μία νέα εγγραφή που περιλαμβάνει όλα τα γνωρίσματα της εγγραφής που εισήχθη καθώς και το στάδιο κατά το οποίο έγινε η εισαγωγή. Όταν μία εγγραφή τροποποιείται τότε εισάγεται μία νέα εγγραφή που περιλαμβάνει όλα τα γνωρίσματα του στοιχείου που τροποποιήθηκε, με το γνώρισμα (ή τα γνωρίσματα που τροποποιήθηκαν να έχουν την νέα τους τιμή.

Ο πίνακας D<Registration_ID> (Deletes table) αποθηκεύει πληροφορίες για τις εγγραφές που διαγράφονται ή ενημερώνονται σε έναν πίνακα για τον οποίο έχει ενεργοποιηθεί η τήρηση ιστορικού. Πραγματοποιώντας ερωτήματα σε αυτόν τον πίνακα είναι δυνατό να ανακτήσουμε πληροφορίες σχετικά με το ποιες πληροφορίες έχουν

διαγραφεί ή τροποποιηθεί σε κάποιο στάδιο της χωρικής βάσης δεδομένων. Ο πίνακας περιλαμβάνει όλα τα γνωρίσματα του σχετικού πίνακα για τον οποίο έχει ενεργοποιηθεί η τήρηση ιστορικού. Επιπρόσθετα περιλαμβάνει ένα μοναδικό για κάθε εγγραφή αύξοντα αριθμό και ένα γνώρισμα που προσδιορίζει το στάδιο κατά το οποίο διεγράφη ή τροποποιήθηκε η εγγραφή. Για παράδειγμα όταν μία υπάρχουσα εγγραφή από τον βασικό πίνακα, τότε εισάγεται στον συγκεκριμένο πίνακα μεταβολής μία νέα εγγραφή που περιλαμβάνει όλα τα γνωρίσματα της εγγραφής που διεγράφη καθώς και το στάδιο κατά το οποίο έγινε η διαγραφή. Όταν μία εγγραφή τροποποιείται τότε εισάγεται μία νέα εγγραφή που περιλαμβάνει όλα τα γνωρίσματα του στοιχείου που τροποποιήθηκε, με το γνώρισμα ή τα γνωρίσματα που τροποποιήθηκαν να έχουν την τιμή που είχαν πριν την επέμβαση.

Στην παρούσα μεταπτυχιακή εργασία αναπτύσσουμε μέρος ενός συστήματος τήρησης ιστορικού. Εστιάζουμε στα εξής επίπεδα:

- Αποθήκευση των αλλαγών που γίνονται στην βάση δεδομένων.
- Η δυνατότητα επιλεκτικής διαχείρισης (επικύρωσης) των αλλαγών ως προς:
    - Το είδος των αλλαγών
    - Την χρονική στιγμή στην οποία αυτές συνέβησαν.

Αντίστοιχα με την μεθοδολογία που περιγράφουμε παραπάνω και σαν μέρους του συστήματος τήρησης ιστορικού που αναπτύσσουμε χρησιμοποιείται σύστημα ζεύγους βοηθητικών πινάκων στους οποίους αποθηκεύονται οι τροποποιήσεις που γίνονται από τους χρήστες στα περιεχόμενα της χωρικής βάσης δεδομένων.

Κεφάλαιο 4

Υλοποίηση μηχανισμού τήρησης ιστορικού χωρικών μεταβολών

Στο κεφάλαιο που προηγήθηκε είδαμε πώς σε λογισμικά διαχείρισης χωρικών βάσεων δεδομένων επιστρατεύονται βοηθητικοί πίνακες προκειμένου αποθηκεύσουν αλλαγές που γίνονται στα δεδομένα με απώτερο στόχο την διαχείριση ταυτόχρονων επεμβάσεων στα ίδια στοιχεία δεδομένων και την μετέπειτα ανάκτηση σταδίων εξέλιξης της χωρικής βάσης δεδομένων. Στην παρούσα εργασία επεκτείνεται αυτή η μεθοδολογία ένα βήμα παραπέρα έτσι ώστε να παρέχεται η δυνατότητα επιλεκτικής επικύρωσης των επεμβάσεων με βάση την χρονική στιγμή στην οποία συνέβησαν. Αυτό θα επιτρέπει την ανάκτηση ενός στιγμιότυπου σε οποιαδήποτε χρονική στιγμή του παρελθόντος της βάσης δεδομένων και όχι μόνο σε προκαθορισμένα στάδια εξέλιξης της. Επιπλέον αναπτύσσεται η δυνατότητα για υποστήριξη επιλεκτικής επικύρωσης των μεταβολών ανάλογα με την φύση τους (εισαγωγή, διαγραφή ή επικύρωση). Στο κεφάλαιο που ακολουθεί αρχικά περιγράφονται τα βασικά συστατικά του τοπικού συστήματος WFS-T που αναπτύχθηκε στην παρούσα εφαρμογή. Αρχικά περιγράφεται το σύστημα διαχείρισης βάσεων δεδομένων PostgreSQL. Στην συνέχεια γίνεται μνεία για την επέκταση χωρικών λειτουργιών PostGIS που σε συνδυασμό με την PostgreSQL δημιουργούν ένα σύστημα διαχείρισης χωρικής βάσης δεδομένων. Έπειτα περιγράφονται τα χαρακτηριστικά της συμβατής με τα λογισμικά που προαναφέρθηκαν διαδικαστικής γλώσσας PL/pgSQL την οποία χρησιμοποιήσαμε για να αναπτύξουμε τα επιμέρους στοιχεία του συστήματος τήρησης ιστορικού και ελεγχόμενης επικύρωσης των επεμβάσεων. Ακόμα περιγράφονται το λογισμικό εξυπηρετητή χωρικών δεδομένων GeoServer που υλοποιεί την προδιαγραφή WFS-T του OGC καθώς και το δικτυακό ΣΓΠ UDIG. Στο δεύτερο μέρος του κεφαλαίου, αρχικά περιγράφεται η δομή του συστήματος τήρησης ιστορικού που αναπτύχθηκε καθώς και η λειτουργία των επιμέρους

τμημάτων. Στην συνέχεια δίνεται ένα πρακτικό παράδειγμα της λειτουργίας του συστήματος.

4.1 Συστατικά στοιχεία συστήματος

4.1.1 Το σύστημα διαχείρισης βάσεων δεδομένων PostgreSQL

Η PostgreSQL είναι ένα αντικειμενο-σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων (object-relational database management system - ORDBMS). Είναι ελεύθερο και ανοικτού κώδικα λογισμικό και δεν αναπτύσσεται απο μία μόνο εταιρία αλλά από μία παγκόσμια κοινότητα χρηστών εταιριών και ιδρυμάτων. Η PostgreSQL υποστηρίζει λειτουργίες όπως συναρτήσεις, δείκτες (B+-tree, hash, GiST and GiN), σκανδαλιστές, κανόνες, ένα ευρύ φάσμα από προκαθορισμένους και ορισμένους από τον χρήστη τύπους δεδομένων (data types) και αντικείμενα. Επιπλέον υποστηρίζονται λειτουργίες κληρονομικότητας χαρακτηριστικών πινάκων (inheritance), περιορισμοί, όψεις, συναλλαγές, λειτουργίες κρυπτογράφησης, αποθήκευσης μεγάλων αντικειμένων κ.α [PL/pg08].

Στην παρούσα μεταπτυχιακή εργασία κάνουμε εκτεταμένη χρήση των δυνατοτήτων συναρτήσεων και σκανδαλιστών που παρέχει η PostgreSQL. Οι συναρτήσεις αφορούν σε κομμάτια κώδικα που μπορούν να αποθηκευτούν και να εκτελεστούν από τον εξυπηρετητή.

Οι περιορισμένες δυνατότητες που παρέχει η γλώσσα SQL σε βασικά στοιχεία ελέγχου έχει οδηγήσει στην ανάπτυξη της δυνατότητας για ενσωμάτωση σε αυτήν κομματιών από ποιο σύνθετες γλώσσες. Τέτοιες είναι:

- Η διαδικαστική γλώσσα PL/pgSQL (βλ. παράγραφο 4.2.1) που είναι εξ' ορισμού ενσωματωμένη στην PostgreSQL και είναι αντίστοιχη της γλώσσας PL/SQL που χρησιμοποιείται στο εμπορικό σύστημα διαχείρισης βάσεων δεδομένων Oracle
- Οι γλώσσες (scripting languages) PL/Lua, PL/LOLCODE, PL/Perl, plPHP, PL/Python, PL/Ruby, PL/sh, PL/Tcl, PL/Scheme, PL/Java, PL/R
- Οι προγραμματιστικές γλώσσες C και C++.

Οι σκανδαλιστές είναι λειτουργίες που όταν ενεργοποιηθούν ανιχνεύουν την ικανοποίηση κάποιας ορισμένης από τον χρήστη συνθήκης που εκφράζεται με εντολή χειρισμού δεδομένων SQL. Όταν αυτή η συνθήκη ικανοποιηθεί τότε οι σκανδαλιστές με την σειρά τους ενεργοποιούν κάποια άλλη προκαθορισμένη λειτουργία. Για παράδειγμα στην εφαρμογή που αναπτύχθηκε στην παρούσα μεταπτυχιακή εργασία οι εντολές INSERT, UPDATE και DELETE ενεργοποιούν σκανδαλιστές που αποθηκεύουν πληροφορίες που αφορούν την κατάσταση των δεδομένων πριν και μετά από τις εντολές. Οι πληροφορίες αυτές ταξινομούνται σε βοηθητικούς πίνακες.

Η PostgreSQL ακόμα και χωρίς κάποια επέκταση υποστηρίζει κάποιες βασικές χωρικές λειτουργίες όπως ειδικούς τύπους δεδομένων για την αποθήκευση απλών γεωμετρικών οντοτήτων και μηχανισμούς δεικτοδότησης χωρικών δεδομένων όπως τετραδικά και R-δένδρα. Η δεικτοδότηση με R-δένδρα είναι μία από τις πιο αποτελεσματικές μεθόδους δεικτοδότησης σε χωρικά

δεδομένα. Επιπλέον υποστηρίζεται περιορισμένος αριθμός γεωμετρικών τελεστών (operators) και συναρτήσεων (functions) για την διατύπωση χωρικών ή συνδυασμένων ερωτημάτων (queries). Στους περιορισμούς που αντιμετωπίζει στον χειρισμό χωρικών δεδομένων θα πρέπει να συγκαταλεχθεί το ότι επιτρέπει μόνο δισδιάστατες γεωμετρικές οντότητες χωρίς δυνατότητα προσδιορισμού συστήματος αναφοράς.

Η PostgreSQL υποστηρίζει τη συνεργασία με πληθώρα επιπρόσθετων σπονδύλων λογισμικού (add-ons) κάθε ένα από τα οποία επιτρέπει την εκτέλεση συγκεκριμένων και εξειδικευμένων λειτουργιών. Το ποιο ευρύτερα διαδεδομένο από αυτά τα λογισμικά είναι η επέκταση γεωγραφικών λειτουργιών PostGIS.

4.1.2 Η επέκταση γεωγραφικών λειτουργιών PostGIS

Η PostGIS είναι επίσης λογισμικό ανοικτού κώδικα και ακολουθεί το πρότυπο OGC για τον ορισμό γεωγραφικών στοιχείων σε περιβάλλον SQL [POS08]. Συγκεκριμένα η PostGIS υποστηρίζει:

- Τους ακόλουθους τύπους στοιχείων:
  - Σημεία (points)
  - Γραμμές (linestrings)
  - Πολύγωνα (polygons)
  - Πολυσημεία (multipoints)
  - Πολυγραμμές (multilinestrings)
  - Πολύπολύγωνα (multipolygons)
- Συλλογές Γεωμετρικών στοιχείων (Geometrycollections)
- Χωρικά κατηγορήματα (spatial predicates) δηλαδή συνθήκες για τον προσδιορισμό των αλληλεπιδράσεων ανάμεσα σε γεωμετρικά στοιχεία με χρήση του 3x3 πίνακα Egenhofer
- Χωρικούς τελεστές (spatial operators) όπως area, distance, length, perimeter, χωρικές λειτουργίες union, difference, symmetric difference και buffers
- Χωρικούς δείκτες R-δένδρα και γενικευμένα δένδρα αναζήτησης (GIST)

Η αρχιτεκτονική της PostGIS στοχεύει στην ελαχιστοποίηση απαιτούμενων πόρων υπολογιστικής ισχύος και μνήμης. Η χρήση γεωμετρικών στοιχείων με χαμηλές απαιτήσεις φυσικής μνήμης επιτρέπει την διατήρηση μεγάλου όγκου δεδομένων από την φυσική μνήμη στην υπολογιστική μνήμη (RAM) με αποτέλεσμα την ταχύτερη εκτέλεση των ερωτημάτων **[POS08]**.

4.1.3 Διαδικαστική γλώσσα PLpgSQL

Η PL/pgSQL (Procedural Language/PostgreSQL Structured Query Language) είναι μία διαδικαστική γλώσσα που υποστηρίζεται από το λογισμικό σύστημα διαχείρισης βάσεων δεδομένων PostgreSQL. Η PL/pgSQL είναι η μόνη διαδικαστική γλώσσα εγκατεστημένη εξορισμού στο λογισμικό PostgreSQL αντίθετα με άλλες διαδικαστικές γλώσσες που είναι συμβατές μαζί του. Τέτοιες είναι οι: PL/Java, PL/Perl, plPHP, PL/Python, PL/R, PL/Ruby, PL/sh, και PL/Tcl.

Η διαδικαστική γλώσσα PL/pgSQL μοιάζει αρκετά με την αντίστοιχη γλώσσα PL/SQL που έχει αναπτυχθεί για το διαδεδομένο εμπορικό λογισμικό σύστημα διαχείρισης βάσεων δεδομένων Oracle. Σύμφωνα με τους δημιουργούς της [PL/pg08] η PL/pgSQL δημιουργήθηκε με σκοπό:

- Να επιτρέπει την δημιουργία συναρτήσεων και σκανδαλιστών,
- Να προσθέσει δομές ελέγχου στην γλώσσα SQL
- Να επιτρέπει πολύπλοκους υπολογισμούς
- Να επιτρέπει την κληρονομικότητα ορισμένων από τον χρήστη συναρτήσεων, τύπων και τελεστών
- Να μπορεί να οριστεί έτσι ώστε να διαχειρίζεται κάποιον δικτυακό εξυπηρετητή
- Να είναι εύχρηστη.

Οι συναρτήσεις που δημιουργούνται με την γλώσσα PL/pgSQL μπορούν να χρησιμοποιηθούν οπουδήποτε υπάρχει περιθώριο για ενσωματωμένες συναρτήσεις (built-in functions). Για παράδειγμα είναι δυνατή η δημιουργία σύνθετων υπολογιστικών συναρτήσεων υπό όρους (conditional computation functions) που αργότερα θα χρησιμοποιηθούν για τον ορισμό τελεστών (operators) ή σε ευρετήρια (index expressions).

Η PL/pgSQL είναι μία πλήρης γλώσσα προγραμματισμού, επιτρέπει πολύ περισσότερο διαδικαστικό έλεγχο από την γλώσσα SQL, και περιλαμβάνει την δυνατότητα για χρήση βρόχων και άλλων δομών ελέγχου. Οι λειτουργίες που δημιουργούνται στην γλώσσα PL/pgSQL μπορούν να κληθούν μέσω εντολών SQL ή να ενεργοποιηθούν μέσω σκανδαλιστών.

Τα πλεονεκτήματά της έναντι της SQL (της απλής και φορητής ανάμεσα σε συστήματα γλώσσας για την δόμηση ερωτημάτων που η PostgreSQL και οι περισσότερες άλλες σχεσιακές βάσεις δεδομένων χρησιμοποιούν) σχετίζονται με το ότι οι εντολές SQL πρέπει να εκτελούνται μεμονωμένα (μία κάθε φορά) από τον εξυπηρετητή της βάσης δεδομένων.

Αυτό σημαίνει πως η «εφαρμογή-χρήστης» (client application) πρέπει να στέλνει κάθε ερώτημα ξεχωριστά στον εξυπηρετητή, να περιμένει αυτό να επεξεργαστεί, να λάβει και να επεξεργαστεί με την σειρά της τα αποτελέσματα των ερωτημάτων, να κάνει υπολογισμούς και να στείλει τα περαιτέρω ερωτήματα στον εξυπηρετητή. Αυτή η συνέχεια λειτουργιών έχει ως αποτέλεσμα αυξημένο όγκο διαδικτυακής επικοινωνίας (network overhead) όταν η εφαρμογή-χρήστης είναι απομακρυσμένη από τον εξυπηρετητή.

Η γλώσσα PL/pgSQL επιτρέπει την ομαδοποίηση υπολογισμών και ερωτημάτων στον εξυπηρετητή και ως εκ τούτου παρέχει τις δυνατότητες μίας διαδικαστικής γλώσσας με την ευκολία της SQL αλλά με δυνατότητα ελαττωμένης επιβάρυνσης διαδικτυακής επικοινωνίας μεταξύ χρήστη και εξυπηρετητή (client/server communication overhead).

Συνολικά:

- Επιπλέον ερωτήσεις και αποκρίσεις μεταξύ χρήστη και εξυπηρετητή ελαχιστοποιούνται
- Ενδιάμεσα αποτελέσματα που ο χρήστης δεν χρειάζεται καθαυτά αλλά έχουν σημασία για επόμενα ερωτήματα δεν χρειάζεται να επεξεργαστούν από τον χρήστη και να μεταφερθούν ανάμεσα σε αυτόν και τον εξυπηρετητή

- Πολλαπλοί κύκλοι επεξεργασίας και υποβολής ερωτημάτων μπορούν να αποφευχθούν
- Τα παραπάνω συμβάλλουν στην δραματική βελτίωση της υπολογιστικής απόδοσης και ταχύτητας σε σχέση με ένα σύστημα που δεν αποθηκεύει συναρτήσεις (functions). Επιπλέον η γλώσσα PL/pgSQL μπορεί να συμπεριλάβει όλους τους τύπους δεδομένων, τελεστές και λειτουργίες της γλώσσας SQL [PL/pg08].

Στην παρούσα εφαρμογή και με σκοπό την δημιουργία επιμέρους τμημάτων του συστήματος διατήρησης ιστορικού σε χωρική βάση δεδομένων αξιοποιήθηκαν τα στοιχεία ελέγχου που προσφέρει η PL/pgSQL όπως οι σκανδαλιστές και οι συναρτήσεις. Έτσι συντάχθηκαν τμήματα κώδικα τα οποία μπορούν να αποθηκευτούν στο σύστημα διαχείρισης βάσεων δεδομένων και να καλούνται από τον διαχειριστή της βάσης δεδομένων ή να εκτελούνται από σκανδαλιστές που παρακολουθούν βασικές λειτουργίες του συστήματος διαχείρισης της χωρικής βάσης δεδομένων.

4.1.4 Λογισμικό δικτυακού εξυπηρετητή GeoServer

Ο GeoServer είναι ένα λογισμικό εξυπηρετητή ανοικτού κώδικα με αντικείμενο την δικτύωση και επικοινωνία γεωαναφερόμενων δεδομένων στο διαδίκτυο και υλοποίηση του προτύπου WFS του OGC [GEO08].

Δημιουργήθηκε από το «The Open Planning Project (TOPP)» με σκοπό να γίνει τμήμα διαλειτουργικής υπολογιστικής υποδομής για την επικοινωνία χωρικών πληροφοριών. Επιπλέον παρέχει την δυνατότητα σύνδεσης υπαρχόντων υποβάθρων με συστήματα απεικόνισης της γήινης επιφάνειας όπως το Google Earth.

Κατά κύριο λόγο ο GeoServer εφαρμόζει τις προδιαγραφές Web Feature Server 1.0 και 1.1, Web Map Server 1.1.1 και Web Coverage Server 1.0 του OGC. Σύμφωνα με αυτές ο GeoServer μπορεί να διανέμει γεωγραφικές πληροφορίες με την μορφή χαρτών/εικόνων (Web Map Server - WMS), ή σαν γεωγραφικά δεδομένα καθαυτά (Web Feature Server - WFS) ενώ σύμφωνα με το πρότυπο (Web Feature Server-Transactional – WFS-T) επιτρέπει στους χρήστες να ενημερώσουν, διαγράψουν και εισάγουν γεωγραφικά στοιχεία.

Ο GeoServer μπορεί να διανέμει γεωγραφικές πληροφορίες σε πληθώρα μορφών όπως KML, GML, Shapefile, GeoRSS, Portable Document Format, GeoJSON, JPEG, GIF, SVG, PNG κ.α. Εισάγει δεδομένα σε μορφές που περιλαμβάνουν την PostGIS, Oracle Spatial, ArcSDE, DB2, MySQL, Shapefiles, GeoTIFF, GTOPO30 κ.α.

Το λογισμικό εξυπηρετητή χωρικών δεδομένων GeoServer χρησιμοποιεί τις βιβλιοθήκες κώδικα GeoTools της γλώσσας Java για την επεξεργασία γεωγραφικών δεδομένων.

4.1.5 Λογισμικό δικτυακού ΣΓΠ Udig

Το UDIG (User friendly Desktop Internet Gis) είναι ένα λογισμικό συστήματος γεωγραφικών πληροφοριών (ΣΓΠ) που αναπτύσσεται από κοινότητα χρηστών και κυρίως από την εταιρία Refractions Research. Είναι ελεύθερο και

ανοικτού κώδικα λογισμικό και διανέμεται κάτω από τους όρους της άδειας LGPL. Βασίζεται στην προγραμματιστική πλατφόρμα Eclipse και είναι γραμμένο όπως και ο GeoServer σε γλώσσα Java. Το UDIG εκτελεί όλες τις βασικές λειτουργίες προσπέλασης διαχείρισης και επέμβασης σε γεωγραφικά δεδομένα. Παρέχεται η δυνατότητα μετακίνησης (pan), εστίασης (zoom), αλλαγής και τροποποίησης των συμβόλων, τροποποίησης γεωγραφικής προβολής, πρόσβασης στους πίνακες δεδομένων (attribute tables) αναζήτηση με κριτήρια που αφορούν στους πίνακες δεδομένων και δημιουργίας διάταξης χάρτη για εκτύπωση. Οι λειτουργίες τροποποίησης περιλαμβάνουν τροποποίηση γεωμετρίας, προσθήκη-αφαίρεση κόμβου, δημιουργία κενού σε πολυγωνικό επίπεδο και τροποποίηση δεδομένων σε πίνακα. Επιπλέον υποστηρίζονται λειτουργίες εισαγωγής (εισαγωγή σημείου, γραμμής, πολυγώνου, γέμισμα περιοχής-fill area, δημιουργία παραλληλογράμμου ή έλειψης) και διαγραφής σε γεωγραφικά και χωρικά δεδομένα. Τα δεδομένα που διαχειρίζεται μπορούν να είναι αποθηκευμένα τοπικά η να προσπελαύνονται μέσω δικτύου (από απομακρυσμένες πηγές δεδομένων όπως ArcSDE, DB2, αποθηκευμένα αρχεία τύπου Shapefiles, MapGraphic, MySQL, Oracle Spatial, PostGIS) είτε μέσω του διαδικτύου με επικοινωνία μέσω εξυπηρετητή που διακινεί τα χωρικά δεδομένα (πρωτόκολλα WMS, WFS, WFS-T) [UDI08]. Το λογισμικό UDIG υποστηρίζει την επικοινωνία με τον εξυπηρετητή χωρικών δεδομένων GeoServer σύμφωνα με το πρότυπο WFS-T του Open Geospatial Consortium. Στην παρούσα εργασία εστιάζουμε σε αυτή του την δυνατότητα για την ανάπτυξη σχετικής πλατφόρμας λογισμικών την οποία θα ενισχύσουμε με λειτουργία τήρησης ιστορικού μεταβολών και ελεγχόμενης επικύρωσης των επεμβάσεων.

4.2 Μεθοδολογία συστήματος τήρησης ιστορικού σε χωρική βάση δεδομένων

4.2.1 Δομή του συστήματος τήρησης ιστορικού

Το σύστημα τήρησης ιστορικού αποτελείται από τα εξής μη δυναμικά στοιχεία:

- Αντίγραφα επέμβασης του πίνακα της βάσης δεδομένων για τον οποίο ενεργοποιείται η δυνατότητα τήρησης ιστορικού. Για κάθε χρήστη που επεμβαίνει και τροποποιεί δεδομένα στον αρχικό πίνακα δημιουργείται ένα τέτοιο αντίγραφο. Έτσι το πλήθος αυτών των αντιγράφων ισούται με το πλήθος των χρηστών που επεμβαίνουν στον αρχικό πίνακα.
- Πίνακες χρονοσήμων όπου αποθηκεύονται τα χρονόσημα των αντιγράφων επέμβασης του αρχικού πίνακα. Σε κάθε αντίγραφο επέμβασης του αρχικού πίνακα αντιστοιχεί ένας πίνακας αποθήκευσης χρονοσήμων. Ο λόγος για τον οποίο τα χρονόσημα δεν διατηρούνται στα αντίγραφα επέμβασης αλλά σε διαφορετικούς πίνακες έχει να κάνει με την αδυναμία χειρισμού χρονοσήμων από την πλευρά του προγράμματος-χρήστη WFS-T που χρησιμοποιήσαμε στην παρούσα εφαρμογή.
- Δύο βοηθητικούς πίνακες ελέγχου ανά πίνακα για τον οποίο ενεργοποιείται η δυνατότητα τήρησης ιστορικού. Σε αυτούς τους πίνακες αποθηκεύονται οι

αλλαγές που γίνονται από τους χρήστες στα αντίγραφα του αρχικού πίνακα της βάσης δεδομένων. Οι πίνακες αυτοί είναι:

- Ο βοηθητικός πίνακας διαγραφών «_delete» στον οποίο αποθηκεύονται οι μεταβολές που έχουν να κάνουν με διαγραφές εγγραφών καθώς και στοιχεία που έχουν να κάνουν με τις τροποποιήσεις υπαρχόντων εγγραφών (αποθηκεύεται η εγγραφή στην μορφή που αυτή έχει πριν την επέμβαση).
- Ο βοηθητικός πίνακας εισαγωγών «_insert» στον οποίο αποθηκεύονται οι μεταβολές που έχουν να κάνουν με εισαγωγές νέων εγγραφών καθώς και στοιχεία που έχουν να κάνουν με τις τροποποιήσεις υπαρχόντων εγγραφών (αποθηκεύεται η εγγραφή στην μορφή που αυτή έχει μετά την επέμβαση).

- Σκανδαλιστή (trigger function) και συνάρτηση (function) που ενεργοποιείται από τον σκανδαλιστή. Ο σκανδαλιστής ενεργοποιείται με κάθε επέμβαση σε ένα από τα αντίγραφα του αρχικού πίνακα. Με την σειρά του ενεργοποιεί την συνάρτηση η οποία αποθηκεύει στους βοηθητικούς πίνακες στοιχεία σχετικά με τις αλλαγές που έχουν γίνει στα αντίγραφα του αρχικού πίνακα, το είδος τους και την χρονική στιγμή που αυτές συνέβησαν.
- Αντίγραφο επικύρωσης του αρχικού πίνακα. Πρόκειται για ένα αντίγραφο του αρχικού πίνακα στο οποίο επικυρώνονται μεταβολές που έχουν κάνει διαφορετικοί χρήστες στα πολλαπλά αντίστοιχα αντίγραφα επικύρωσης. Επιπλέον σε αυτό το αντίγραφο μπορεί να επικυρωθεί, εφόσον το επιλέξει ο διαχειριστής της βάσης δεδομένων, μέρος των αλλαγών που συνέβησαν και όχι το σύνολό τους.
- Συνάρτηση επικύρωσης που επικυρώνει συνολικά ή επιλεκτικά στο αντίγραφο επικύρωσης του αρχικού πίνακα τις αλλαγές που έχουν αποθηκευτεί στους δύο βοηθητικούς πίνακες.

4.2.3 Λειτουργία των επιμέρους τμημάτων του συστήματος

4.2.3.1 Αντίγραφα του αρχικού πίνακα και πίνακες αποθήκευσης των χρονοσήμων:

Για κάθε έναν από τους χρήστες που επεμβαίνουν στον αρχικό πίνακα της χωρικής βάσης δεδομένων για τον οποίο έχει ενεργοποιηθεί η τήρηση ιστορικού δημιουργείται ένα αντίγραφο επέμβασης. Ο χρήστης πλέον δεν βλέπει τον αρχικό πίνακα αλλά το αντίστοιχο αντίγραφο επέμβασης. Οι αλλαγές στις οποίες προβαίνει επηρεάζουν αυτό το αντίγραφο.

4.2.3.2 Βοηθητικοί πίνακες:

Για κάθε πίνακα στον οποίο ενεργοποιείται η τήρηση ιστορικού δημιουργούνται δύο βοηθητικοί πίνακες. Σε αυτούς τους πίνακες αποθηκεύονται οι τροποποιήσεις που γίνονται στα αντίγραφα του αρχικού πίνακα από το σύνολο των χρηστών. Στον ένα από τους δύο βοηθητικούς πίνακες αποθηκεύονται οι εισαγωγές εγγραφών και στοιχεία που αφορούν στην μορφή που έχουν οι εγγραφές μετά τις ενημερώσεις τους. Στον άλλο βοηθητικό πίνακα αποθηκεύονται οι διαγραφές εγγραφών και κάποια στοιχεία που αφορούν στην μορφή που είχαν οι

εγγραφές πριν από τις ενημερώσεις τους. Και στα δύο αυτά είδη πινάκων αποθηκεύεται πληροφορία που προσδιορίζει το είδος των επεμβάσεων (εισαγωγή, διαγραφή ή τροποποίηση) που αποθηκεύουν οι σχετικές εγγραφές.

4.2.3.3 Σκανδαλιστής (trigger function) και συνάρτηση (function) που ενεργοποιείται από τον σκανδαλιστή:

Σε κάθε ένα από τα αντίγραφα του αρχικού πίνακα δημιουργούμε έναν σκανδαλιστή ο οποίος ενεργοποιείται με κάθε επέμβαση (εισαγωγή, διαγραφή, ενημέρωση) στο αντίγραφο επέμβασης και ενεργοποιεί την συνάρτηση ενημέρωσης των βοηθητικών πινάκων. Η συνάρτηση διακρίνει τρεις υποπεριπτώσεις ανάλογα με τον τύπο επέμβασης στο αντίγραφο του αρχικού πίνακα:

- Αν η επέμβαση αφορά σε διαγραφή εγγραφής: Ο βοηθητικός πίνακας «_delete» ενημερώνεται με τα στοιχεία της εγγραφής που διαγράφεται, το χρονόσημο που εκτελείται η συνάρτηση, ενώ ταυτόχρονα αποθηκεύεται και ο κωδικός που προσδιορίζει ότι αυτή η εγγραφή του βοηθητικού πίνακα αφορά σε διαγραφή.
- Αν επέμβαση αφορά σε εισαγωγή στοιχείου: Ο βοηθητικός πίνακας «_insert» ενημερώνεται με όλα τα στοιχεία της εγγραφής που εισάγεται. Ταυτόχρονα ο πίνακας διατήρησης χρονοσήμου που αντιστοιχεί στο αντίγραφο επικύρωσης ενημερώνεται με το χρονόσημο της νέας εγγραφής. Επιπλέον αποθηκεύεται και ο κωδικός που προσδιορίζει ότι αυτή η εγγραφή του βοηθητικού πίνακα αφορά σε εισαγωγή.
- Αν επέμβαση αφορά σε ενημέρωση στοιχείου: Ο βοηθητικός πίνακας «_delete» ενημερώνεται με όλα τα στοιχεία της εγγραφής που τροποποιείται με την μορφή που αυτά τα στοιχεία είχαν πριν την ενημέρωση (συμπεριλαμβανομένου του χρονοσήμου που η εγγραφή είχε πριν την επέμβαση) το χρονόσημο που προσδιορίζει την χρονική στιγμή που έγινε η μεταβολή και ο κωδικός που προσδιορίζει ότι αυτή η εγγραφή του βοηθητικού πίνακα αφορά σε ενημέρωση. Ο βοηθητικός πίνακας «_insert» ενημερώνεται με όλα τα νέα στοιχεία της εγγραφής που ενημερώνεται. Επιπλέον αποθηκεύεται ο κωδικός 2 που προσδιορίζει ότι αυτή η εγγραφή του βοηθητικού πίνακα αφορά σε ενημέρωση εγγραφής. Ο πίνακας διατήρησης χρονοσήμου ενημερώνεται για το νέο χρονόσημο της εγγραφής που ενημερώθηκε.

4.2.3.4 Συνάρτηση επικύρωσης των αλλαγών που έχουν αποθηκευτεί στους δύο βοηθητικούς πίνακες:

Αφού πλέον οι αλλαγές έχουν αποθηκευτεί στους βοηθητικούς πίνακες ο διαχειριστής της βάσης δεδομένων μπορεί να επιλέξει ποιες από τις αλλαγές θα επικυρώσει. Η συνάρτηση επικύρωσης που αναπτύχθηκε στην παρούσα διπλωματική εργασία επιτρέπει στον διαχειριστή, εφόσον το επιθυμεί, να επικυρώσει επιλεκτικά μέρος των συναλλαγών με κριτήριο το είδος τους (αν δηλαδή πρόκειται για εισαγωγές εγγραφών, διαγραφές εγγραφών, η τροποποιήσεις εγγραφών) ή την χρονική στιγμή που συνέβησαν προσδιορίζοντας την χρονική περίοδο (χρονόσημα έναρξης και λήξης της χρονικής περιόδου) για την οποία επιθυμεί να επικυρωθούν οι συναλλαγές.

Ως εκ τούτου η συνάρτηση έχει τρεις παραμέτρους τις οποίες προσδιορίζει ο διαχειριστής της βάσης δεδομένων:

- Χρονική στιγμή έναρξης περιόδου συναλλαγών
- Χρονική στιγμή λήξης περιόδου συναλλαγών
- Είδος συναλλαγών που επιθυμείται να επικυρωθούν:
  0. Όλες οι συναλλαγές ανεξάρτητα από το είδος τους
  1. Συναλλαγές που αφορούν μόνο σε διαγραφές
  2. Συναλλαγές που αφορούν μόνο σε ενημερώσεις εγγραφών
  3. Συναλλαγές που αφορούν μόνο σε εισαγωγές εγγραφών.

Αρχικά η συνάρτηση επικύρωσης καλεί μία όψη (view) που αποτελεί την ένωση των δύο βοηθητικών πινάκων που περιέχουν τις αλλαγές που έχουν γίνει στους βοηθητικούς πίνακες. Αυτή η όψη περιέχει μόνο τις συναλλαγές που συνέβησαν μέσα στα χρονικά όρια τα οποία έχει προσδιορίσει ο διαχειριστής της βάσης δεδομένων. Αν ο διαχειριστής δεν προσδιορίσει χρονικά όρια τότε η όψη περιλαμβάνει όλες τις συναλλαγές. Η συνάρτηση διατρέχει τις εγγραφές που περιέχονται στην όψη (είναι στοιχισμένες ως προς την χρονική στιγμή που συνέβησαν) και τις διαχειρίζεται ως εξής ανάλογα με τα χαρακτηριστικά τους και τις παραμέτρους που έχει ορίσει ο διαχειριστής της βάσης δεδομένων:

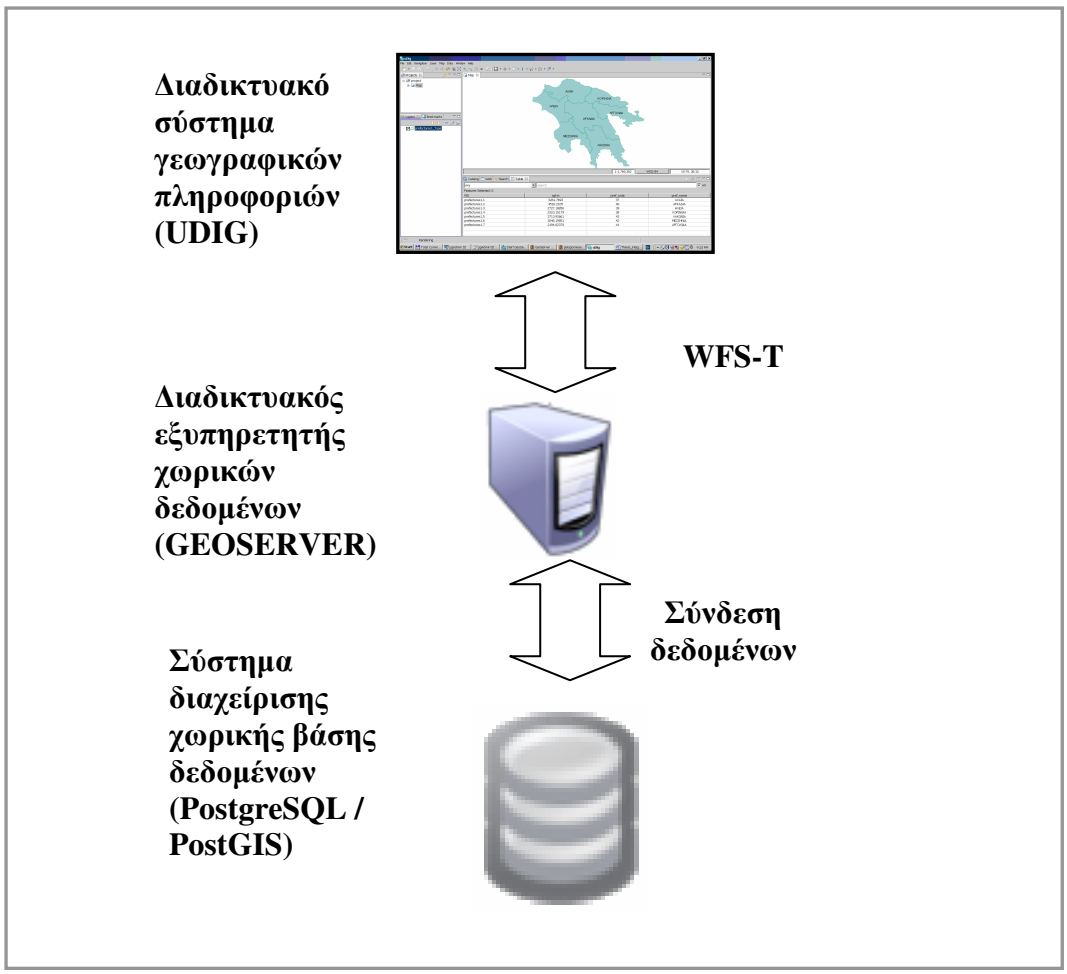
1. Αν ο διαχειριστής επιθυμεί να επικυρωθούν οι συναλλαγές που αφορούν σε διαγραφές στοιχείων τότε η συνάρτηση επικύρωσης διατρέχει την παραπάνω όψη και όταν εντοπίσει κάποια εγγραφή που αφορά σε διαγραφή στοιχείου τότε διαγράφεται από τον πίνακα επικύρωσης η αντίστοιχη εγγραφή.
2. Αν ο διαχειριστής επιθυμεί να επικυρωθούν οι συναλλαγές που αφορούν σε ενημερώσεις εγγραφών τότε η συνάρτηση επικύρωσης διατρέχει την παραπάνω όψη και όταν εντοπίσει κάποια εγγραφή που αφορά σε ενημέρωση στοιχείου, τότε αρχικά ελέγχεται αν η εγγραφή αφορά σε συναλλαγή στοιχείου που προϋπήρχε στον αρχικό πίνακα ή έχει εισαχθεί από κάποιο χρήστη μετά την τελευταία επικύρωση. Αυτό έχει σημασία γιατί ο διαχειριστής έχει επιλέξει να επικυρωθούν οι τροποποιήσεις υπαρχόντων και όχι νέων στοιχείων συνεπώς αν το στοιχείο είναι νέο δεν πρέπει να συνεχίσει να είναι μέρος της διαδικασίας επικύρωσης.
   - Αν λοιπόν το στοιχείο είναι νέο δεν επικυρώνεται παραπέρα.
   - Αν το στοιχείο δεν είναι νέο δηλαδή η επικύρωση αφορά σε προϋπάρχον στοιχείο τότε:
     i. Διαγράφεται από τον πίνακα επικύρωσης η εγγραφή που τροποποιήθηκε με την μορφή που αυτή είχε πριν την επέμβαση, δηλαδή διαγράφεται από τον πίνακα επικύρωσης η εγγραφή που έχει την ίδια τιμή πρωτεύοντος κλειδιού και χρονοσήμου με την αντίστοιχη εγγραφή του πίνακα «_delete».
     ii. Επανεισάγεται στον πίνακα επικύρωσης η εγγραφή που τροποποιήθηκε με όλα τα νέα της γνωρίσματα, δηλαδή όπως αυτή περιλαμβάνεται στον πίνακα «_insert».
3. Αν ο διαχειριστής επιθυμεί να επικυρωθούν οι συναλλαγές που αφορούν σε εισαγωγές στοιχείων τότε η συνάρτηση επικύρωσης διατρέχει την παραπάνω όψη. Όταν εντοπίσει κάποια εγγραφή που αφορά σε εισαγωγή

στοιχείου τότε εισάγεται στον πίνακα επικύρωσης μία νέα εγγραφή. Η τελευταία περιέχει τις τιμές της εγγραφής που εισήχθη στο σχετικό αντίγραφό επέμβασης όπως αυτή περιέχεται στον βοηθητικό πίνακα εισαγωγών.

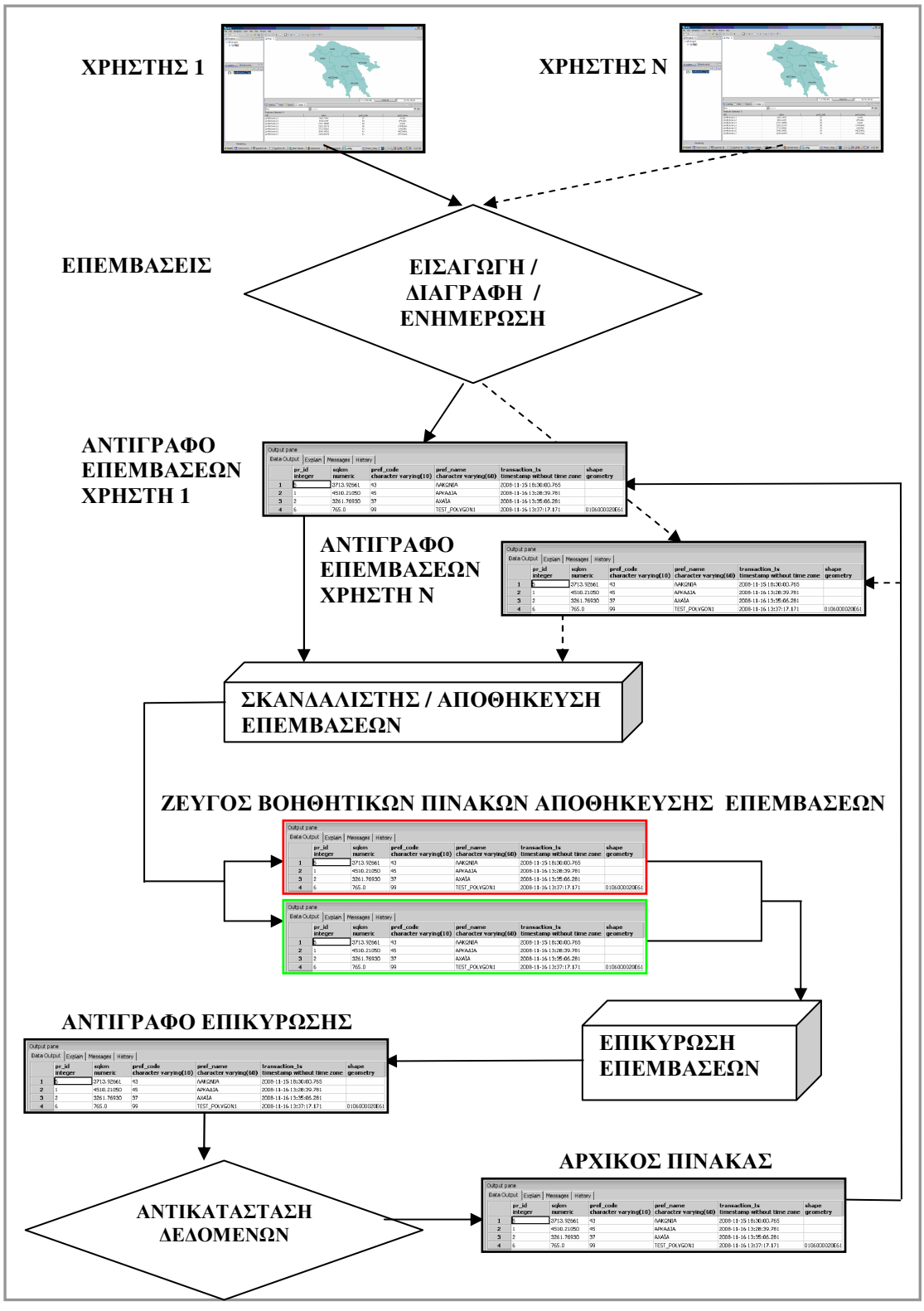
4. Αν ο διαχειριστής επιθυμεί να επικυρωθούν όλες οι συναλλαγές τότε η συνάρτηση επικύρωσης διατρέχει την παραπάνω όψη και κάθε στοιχείο αντιμετωπίζεται με τον τρόπο που περιγράφεται στα παραπάνω σχετικά βήματα.

Όταν πλέον η επικύρωση ολοκληρωθεί με την εκτέλεση της συνάρτησης επικύρωσης τότε ο αρχικός πίνακας μπορεί να αρχειοθετηθεί ή να διαγραφούν τα περιεχόμενά του και να εγγραφούν σε αυτόν τα περιεχόμενα του πίνακα επικύρωσης.

Η δομή και τρόπος λειτουργίας του συστήματος που αναπτύχθηκε στην παρούσα πτυχιακή εργασία αναπτύσσονται στις εικόνες 4.1 και 4.2. Συγκεκριμένα στην εικόνα 4.1 απεικονίζεται η αρχιτεκτονική του συστήματος σε τρία επίπεδα: Στο επίπεδο του χρήστη το σύστημα αποτελείται από το διαδικτυακό σύστημα γεωγραφικών πληροφοριών UDIG. Το τελευταίο συνδέεται με τον εξυπηρετητή χωρικών δεδομένων με επικοινωνία που διέπεται από το πρότυπο WFS-T του Open Geospatial Consortium. Ο εξυπηρετητής διαχειρίζεται δεδομένα που είναι αποθηκευμένα στην βάση χωρικών δεδομένων PostgreSQL / PostGIS. Στην εικόνα 4.2 απεικονίζεται η ροή εργασιών του συστήματος: Ένας ή περισσότεροι χρήστες επεμβαίνουν στα αντίγραφα επεμβάσεων του αρχικού πίνακα για τον οποίο ενεργοποιείται η τήρηση ιστορικού. Ο σκανδαλιστής αποθηκεύει στοιχεία σχετικά με αυτές τις επεμβάσεις στο ζεύγος των βοηθητικών πινάκων αποθήκευσης επεμβάσεων. Η συνάρτηση αποθήκευσης επεμβάσεων αντλεί στοιχεία που βρίσκονται αποθηκευμένα σε αυτούς τους πίνακες προκειμένου να επικυρώσει όλες τις αλλαγές ή επιλεκτικά ένα μέρος από αυτές στο αντίγραφο του αρχικού πίνακα που καλούμε αντίγραφο επικύρωσης. Αυτός ο πίνακας μπορεί, εφόσον η επικύρωση είναι οριστική, να αντικαταστήσει τον υπάρχοντα αρχικό πίνακα και να γίνει ο νέος αρχικός πίνακας.



Εικόνα 4.1 Απεικόνιση της αρχιτεκτονικής του συστήματος WFS-T που υλοποιήθηκε στην παρούσα εργασία.



Εικόνα 4.2 Απεικόνιση της ροής εργασιών εφαρμογής WFS-T που υλοποιήθηκε στην παρούσα εργασία.

4.3 Πρακτικό παράδειγμα τήρησης ιστορικού σε χωρική βάση δεδομένων

Στα πλαίσια της παρούσας μεταπτυχιακής εργασίας δημιουργήσαμε μία βάση δεδομένων σε περιβάλλον PostgreSQL/PostGIS με χωρικά δεδομένα που αφορούν στους νομούς της Ελλάδας. Σε αυτή την βάση δεδομένων επιτρέπουμε την προσπέλαση μέσω δικτυακού εξυπηρετητή GeoServer με το πρωτόκολλο WFS-T.

Στην βάση δεδομένων εισάγαμε πίνακα με πολυγωνικά γεωμετρικά στοιχεία που απεικονίζουν τους νομούς της Πελοποννήσου. Τον πίνακα αυτό ονομάσαμε prefectures_default. Ο πίνακας prefectures_default έχει τα ακόλουθα γνωρίσματα:

- pr_id, ακέραιος, πρωτεύον κλειδί
- sqkm αριθμός
- pref_code, αλφαριθμητικός (10)
- pref_name, αλφαριθμητικός (60)
- transaction_ts, χρονόσημο
- shape πεδίο, γεωμετρίας

Η εισαγωγή των στοιχείων έγινε με εντολές εισαγωγής SQL. Για παράδειγμα ο Νομός Αρκαδίας εισάγεται με την ακόλουθη εντολή SQL:

```
INSERT INTO prefectures_default (pr_id, sqkm,
pref_code, pref_name, transaction_ts, shape) VALUES
('1','4510.21050','40','ΑΡΚΑΔΙΑ', CURRENT_TIMESTAMP,
GeometryFromText ('MULTIPOLYGON (((22.3024475801268
37.8323199392195, ..............  22.3024475801268
37.8323199392195)))',4326) );
```

Ζητώντας από το σύστημα να μας επιστρέψει τα περιεχόμενα του πίνακα λαμβάνουμε το ακόλουθο

Output pane

Data Output | Explain | Messages | History

	pr_id integer	sqkm numeric	pref_code character var	pref_name character var	transaction_ts timestamp without time	shape geometry
1	1	3261.76930	37	ΑΧΑΪΑ	2008-12-08 23:07:23.484	
2	2	4510.21050	40	ΑΡΚΑΔΙΑ	2008-12-08 23:07:38.89	
3	3	2727.18856	39	ΗΛΕΙΑ	2008-12-08 23:07:51.39	0106000020E61
4	4	2323.15179	38	ΚΟΡΙΝΘΙΑ	2008-12-08 23:09:05.515	0106000020E61
5	5	3713.92661	43	ΛΑΚΩΝΙΑ	2008-12-08 23:09:28.765	
6	6	3048.19851	42	ΜΕΣΣΗΝΙΑ	2008-12-08 23:09:46.234	
7	7	2194.62379	41	ΑΡΓΟΛΙΔΑ	2008-12-08 23:09:46.234	

Πίνακας 4.1. Ο αρχικός πίνακας prefectures_default για τον οποίο ενεργοποιείται η τήρηση ιστορικού.

Ενεργοποιώντας την τήρηση ιστορικού για τον συγκεκριμένο πίνακα δημιουργούμε ένα ζεύγος βοηθητικών πινάκων στους οποίους θα αποθηκεύονται οι αλλαγές που θα γίνουν στα αντίγραφα επέμβασης του πίνακα prefectures_default. Οι πίνακες αυτοί ονομάζονται «prefectures_insert» και «prefectures_delete» αντίστοιχα και περιέχουν όλα τα γνωρίσματα του βασικού πίνακα, ένα επιπλέον γνώρισμα χρονοσήμου καθώς και ένα γνώρισμα που αποθηκεύει το είδος επέμβασης (εισαγωγή, διαγραφή ή τροποποίηση). Επιπλέον των βοηθητικών πινάκων, του σκανδαλιστή και της συνάρτησης που αποθηκεύει

σε αυτούς δεδομένα σαν μέρος της υποδομής τήρησης ιστορικού μεταβολών σε χωρική βάση δεδομένων δημιουργείται και μία συνάρτηση διαχείρισης των αλλαγών. Την συνάρτηση αυτή ονομάσαμε «συνάρτηση επικύρωσης των αλλαγών» και την καλεί ο διαχειριστής της βάσης δεδομένων κάθε φορά που θέλει να επικυρώσει συνολικά η μερικά τις αλλαγές που συνέβησαν στην βάση δεδομένων.

Έστω, τώρα, χρήστης1 που επιθυμεί να προσπελάσει και να τροποποιήσει τα χωρικά δεδομένα αυτού του πίνακα μέσω δικτυακού συστήματος γεωγραφικών πληροφοριών.

Ακολουθώντας την μεθοδολογία τήρησης ιστορικού που αναπτύχθηκε στην παρούσα μεταπτυχιακή εργασία δημιουργούμε ένα αντίγραφο του πίνακα prefectures_default το οποίο ονομάζουμε prefectures1. Το αντίγραφο αυτό πλέον βλέπει ο χρήστης1 και σε αυτό γίνονται οι αλλαγές που κάνει. Το αντίγραφο αυτό είναι πανομοιότυπο με τον πίνακα prefectures_default με την μόνη διαφορά ότι το πεδίο χρονόσημου δεν αποθηκεύεται πλέον στον ίδιο πίνακα αλλά σε έναν πίνακα που περιέχει αποκλειστικά τα χρονόσημα και συνδέεται με το αντίγραφο επέμβασης «prefectures1» μέσω του πρωτεύοντος κλειδιού «pr_id». Τον πίνακα που αποθηκεύει τα χρονόσημα ονομάζουμε «prefectures1_ts».

Οι πίνακες «prefectures1» και «prefectures1_ts» έχουν την ακόλουθη μορφή:

Output pane

Data Output | Explain | Messages | History

	pr_id integer	sqkm numeric	pref_code character var	pref_name character var	shape geometry
1	1	3261.76930	37	ΑΧΑΪΑ	
2	2	4510.21050	40	ΑΡΚΑΔΙΑ	
3	3	2727.18856	39	ΗΛΕΙΑ	0106000020E61
4	4	2323.15179	38	ΚΟΡΙΝΘΙΑ	0106000020E61
5	5	3713.92661	43	ΛΑΚΩΝΙΑ	
6	6	3048.19851	42	ΜΕΣΣΗΝΙΑ	
7	7	2194.62379	41	ΑΡΓΟΛΙΔΑ	

Πίνακας 4.2. Το αντίγραφο επέμβασης του αρχικού πίνακα prefectures1. Αυτός ο πίνακας αντιστοιχεί στον χρήστη1.

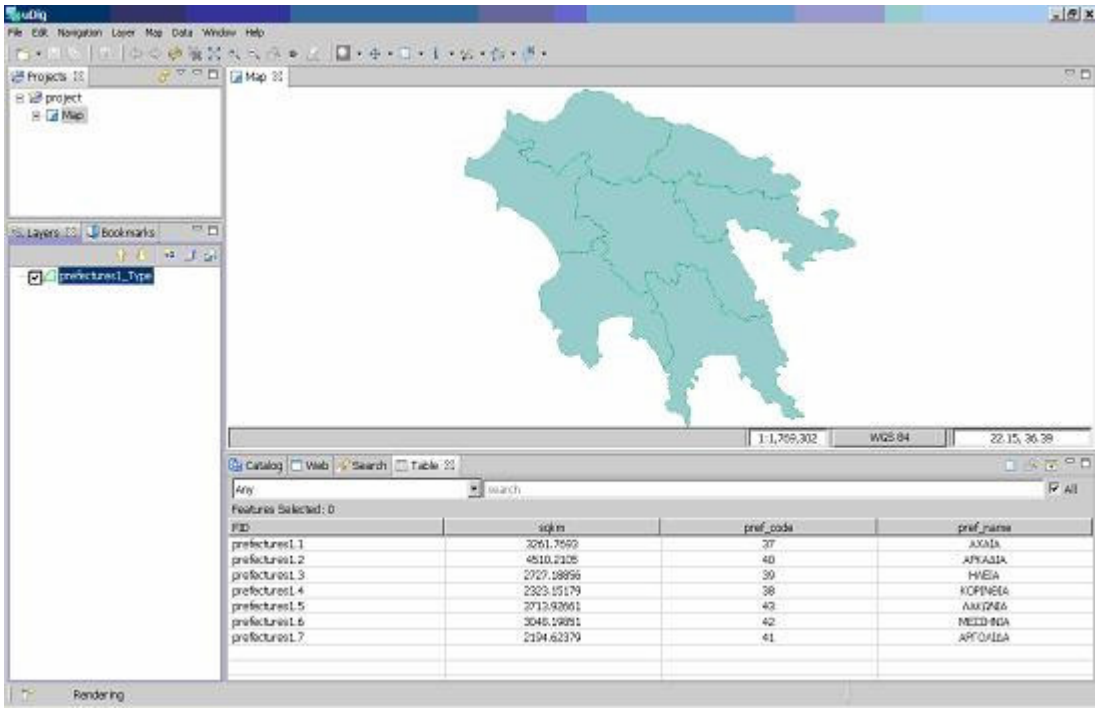
Output pane

Data Output | Explain | Messages | History

	pr_id integer	transaction_ts timestamp without time z
1	1	2008-12-08 23:07:23.484
2	2	2008-12-08 23:07:38.89
3	3	2008-12-08 23:07:51.39
4	4	2008-12-08 23:09:05.515
5	5	2008-12-08 23:09:28.765
6	6	2008-12-08 23:09:46.234
7	7	2008-12-08 23:09:46.234

Πίνακας 4.3. Ο πίνακας αποθήκευσης χρονοσήμου prefectures1_ts. Σε αυτό τον πίνακα αποθηκεύονται οι τιμές χρονοσήμου για τις εγγραφές του πίνακα prefectures1.

Μαζί με τους πίνακες «prefectures1» και «prefectures1_ts» δημιουργείται και ο σκανδαλιστής prefectures_insert_delete1 που γεμίζει τους βοηθητικούς πίνακες prefectures_insert και prefectures_delete ανάλογα με τις αλλαγές που κάνει ο χρήστης στο αντίγραφο επέμβασης που του αντιστοιχεί. Οι βοηθητικοί πίνακες σε αυτό το στάδιο είναι ακόμα κενοί.



Εικόνα 4.4 Απεικόνιση της γεωμετρίας του πίνακα prefectures1 πριν τις επεμβάσεις.

Ο χρήστης1 πραγματοποιεί τις παρακάτω αλλαγές στις ακόλουθες διαδοχικές χρονικές στιγμές:

1. Διαγράφει τον Νομό Κορινθίας (χρονική στιγμή: "2008-12-10 20:18:55.50")
2. Τροποποιεί ένα από τα γνωρίσματα του νομού Αρκαδίας (αλλάζει την τιμή στο γνώρισμα «pref_code» απο 40 σε 45, χρονική στιγμή: "2008-12-10 20:19:20.39")
3. Τροποποιεί την γεωμετρία του νομού Αχαΐας (επεκτείνει την ακτογραμμή στα βόρεια-βορειοδυτικά, εκεί όπου βρίσκεται η γέφυρα Ρίου-Αντιρρίου, την χρονική στιγμή: "2008-12-10 20:20:32.453")
4. Εισάγει μία νέα γεωμετρία που ονομάζει «ZAKYNTHOS» (χρονική στιγμή: "2008-12-10 20:22:15.171")
5. Διαγράφει τον Νομό Λακωνίας (χρονική στιγμή: "2008-12-10 20:23:50.828")
6. Τροποποιεί ένα από τα γνωρίσματα του νομού Ηλείας (γνώρισμα «pref_code» από 39 σε 43 την χρονική στιγμή: "2008-12-10 20:24:25.734")
7. Τροποποιεί την γεωμετρία του νομού Αχαΐας (επεκτείνει την ακτογραμμή στα νοτιοδυτικά την χρονική στιγμή: "2008-12-10 20:26:02.078")

8. Εισάγει μία νέα γεωμετρία που ονομάζει «KITHIRA» την χρονική στιγμή: "2008-12-10 20:27:34.734")

Οι πίνακες επέμβασης prefectures1 και αποθήκευσης χρονοσήμου prefectures1_ts που αντιστοιχούν στον χρήστη1 τώρα έχουν την ακόλουθη μορφή.

Output pane

Data Output | Explain | Messages | History

	pr_id integer	sqkm numeric	pref_code character var	pref_name character var	shape geometry
1	6	3048.19851	42	ΜΕΣΣΗΝΙΑ	
2	7	2194.62379	41	ΑΡΓΟΛΙΔΑ	
3	2	4510.21050	45	ΑΡΚΑΔΙΑ	
4	8	408.0	20	ZAKYNTHOS	0106000020E61
5	3	2727.18856	43	ΗΛΕΙΑ	0106000020E61
6	1	3261.76930	37	ΑΧΑΪΑ	
7	9	12.0	23	KYTHIRA	0106000020E61

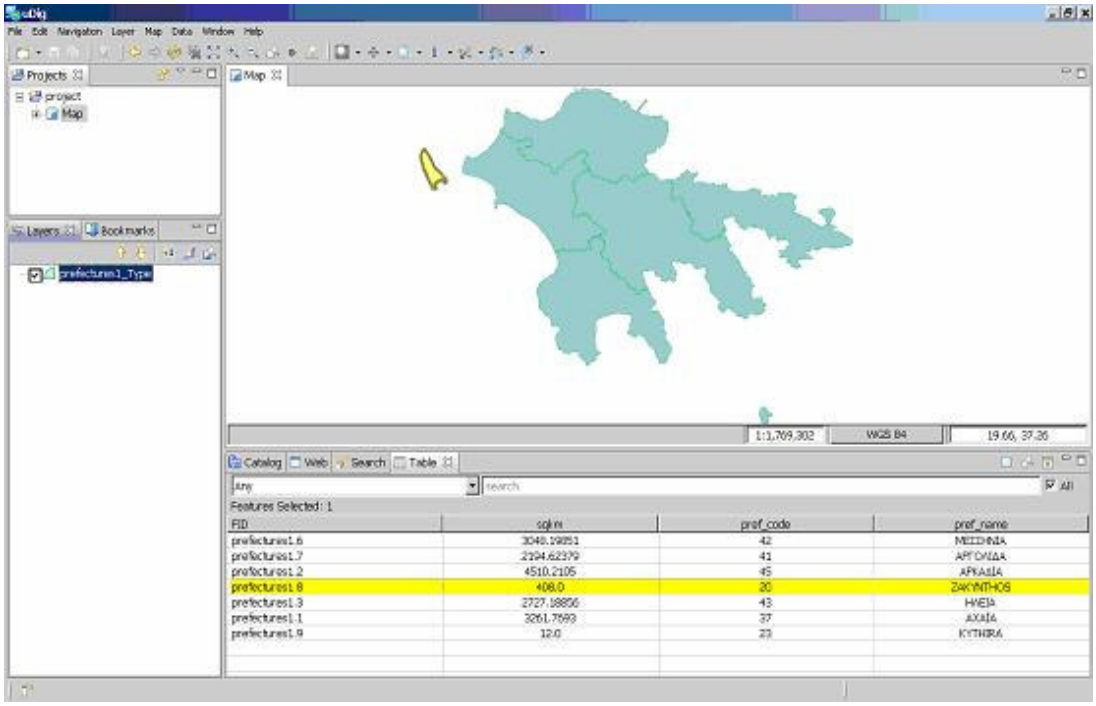
Πίνακας 4.4 Ο πίνακας prefectures1 μετά την ολοκλήρωση των επεμβάσεων σε αυτόν.

Output pane

Data Output | Explain | Messages | History

	pr_id integer	transaction_ts timestamp without time zone
1	6	2008-12-08 23:09:46.234
2	7	2008-12-08 23:09:46.234
3	2	2008-12-10 20:19:20.39
4	8	2008-12-10 20:22:15.171
5	3	2008-12-10 20:24:25.734
6	1	2008-12-10 20:26:02.078
7	9	2008-12-10 20:27:34.734

Πίνακας 4.4 Ο πίνακας prefectures1_ts μετά την ολοκλήρωση των επεμβάσεων στον πίνακα prefectures_1.



Εικόνα 4.5 Απεικόνιση της γεωμετρίας του πίνακα prefectures1 μετά την ολοκλήρωση των επεμβάσεων.

Οι αλλαγές αυτές έχουν αποθηκευτεί στον αντίστοιχο πίνακα επέμβασης (prefectures1) ενώ τα χρονόσημά τους έχουν αποθηκευτεί στον πίνακα διατήρησης χρονοσήμων (prefectures1_ts). Συνολικά όμως οι αλλαγές έχουν αποθηκευτεί στο ζεύγος βοηθητικών πινάκων αποθήκευσης αλλαγών το οποίο αντιστοιχεί στον αρχικό πίνακα (prefectures_insert, prefectures_delete). Οι βοηθητικοί πίνακες πλέον έχουν την ακόλουθη μορφή:

Output pane

Data Output | Explain | Messages | History

	pr_id integer	sqkm numeric	pref_ char	pref_name character v	record_ts timestamp without time	transaction_ts timestamp without time	trans char	shape geometry
1	2	4510.21050	45	ΑΡΚΑΔΙΑ	2008-12-10 20:19:20.39	2008-12-10 20:19:20.39	2	
2	1	3261.76930	37	ΑΧΑΪΑ	2008-12-10 20:20:32.453	2008-12-10 20:20:32.453	2	
3	8	408.0	20	ZAKYNTHOS	2008-12-10 20:22:15.171	2008-12-10 20:22:15.171	3	0106000020E61
4	3	2727.18856	43	ΗΛΕΙΑ	2008-12-10 20:24:25.734	2008-12-10 20:24:25.734	2	0106000020E61
5	1	3261.76930	37	ΑΧΑΪΑ	2008-12-10 20:26:02.078	2008-12-10 20:26:02.078	2	
6	9	12.0	23	KYTHIRA	2008-12-10 20:27:34.734	2008-12-10 20:27:34.734	3	0106000020E61

Πίνακας 4.5. Ο πίνακας prefectures_insert μετά την ολοκλήρωση των επεμβάσεων στον πίνακα prefectures_1.

Output pane

Data Output | Explain | Messages | History

	pr_i inte	sqkm numeric	pref char	pref_nam character	record_ts timestamp without tim	transaction_ts timestamp without tim	tran char	shape geometry
1	4	2323.15179	38	ΚΟΡΙΝΘΙΑ	2008-12-08 23:09:05.515	2008-12-10 20:18:55.50	1	0106000020E61
2	2	4510.21050	40	ΑΡΚΑΔΙΑ	2008-12-08 23:07:38.89	2008-12-10 20:19:20.39	2	
3	1	3261.76930	37	ΑΧΑΪΑ	2008-12-08 23:07:23.484	2008-12-10 20:20:32.453	2	
4	5	3713.92661	43	ΛΑΚΩΝΙΑ	2008-12-08 23:09:28.765	2008-12-10 20:23:50.828	1	
5	3	2727.18856	39	ΗΛΕΙΑ	2008-12-08 23:07:51.39	2008-12-10 20:24:25.734	2	0106000020E61
6	1	3261.76930	37	ΑΧΑΪΑ	2008-12-10 20:20:32.453	2008-12-10 20:26:02.078	2	

Πίνακας 4.6. Ο πίνακας prefectures_delete μετά την ολοκλήρωση των επεμβάσεων στον πίνακα prefectures_1.

Ο διαχειριστής της βάσης δεδομένων θα πρέπει τώρα να διαχειριστεί αυτές τις αλλαγές. Προκειμένου να γίνει αυτό θα χρησιμοποιηθεί η συνάρτηση επικύρωσης που επιτρέπει στον διαχειριστή της βάσης δεδομένων να επικυρώσει συνολικά τις αλλαγές που συνέβησαν η επιλεκτικά κάποιες από αυτές εισάγοντας παραμέτρους που αφορούν στην χρονική στιγμή που έγιναν οι αλλαγές ή στο είδος των αλλαγών.

Συγκεκριμένα ο διαχειριστής εισάγει τις ακόλουθες τρεις παραμέτρους στην συνάρτηση επικύρωσης:

- Έναρξη χρονικής περιόδου μέσα στην οποία συνέβησαν οι αλλαγές που επιθυμείται να διατηρηθούν (χρονική παράμετρος που εκφράζεται με χρονόσημο)
- Λήξη χρονικής περιόδου μέσα στην οποία συνέβησαν οι αλλαγές που επιθυμείται να διατηρηθούν (χρονική παράμετρος που εκφράζεται με χρονόσημο)
- Είδος αλλαγών που επιθυμείται να διατηρηθούν. Η παράμετρος λαμβάνει τις εξής τέσσερις τιμές:
  - 0: επιθυμείται να διατηρηθούν οι αλλαγές κάθε είδους
  - 1: επιθυμείται να διατηρηθούν οι αλλαγές που αφορούν σε διαγραφές στοιχείων
  - 2: επιθυμείται να διατηρηθούν οι αλλαγές που αφορούν σε τροποποιήσεις στοιχείων
  - 3: επιθυμείται να διατηρηθούν οι αλλαγές που αφορούν σε εισαγωγές στοιχείων

Για να γίνει πιο εύγλωττο το εύρος των δυνατοτήτων που προσφέρει στον διαχειριστή της βάσης δεδομένων η συνάρτηση επικύρωσης θα δώσουμε τα εξής παραδείγματα:

Παράδειγμα Α.

Έστω ότι ο διαχειριστής επιθυμεί να συμπεριλάβει στο αντίγραφο επικύρωσης όλες τις αλλαγές που έγιναν στην βάση δεδομένων. Σε αυτή την περίπτωση καλεί την συνάρτηση επικύρωσης ορίζοντας τις τρεις παραμέτρους επικύρωσης ως εξής:

- Έναρξη χρονικής περιόδου: NULL
- Λήξη χρονικής περιόδου: NULL
- Είδος αλλαγών: 0

Τότε η συνάρτηση επικύρωσης μεταφέρει στο αντίγραφο επικύρωσης όλες τις αλλαγές. Ο πίνακας επικύρωσης τότε θα έχει την ακόλουθη μορφή:

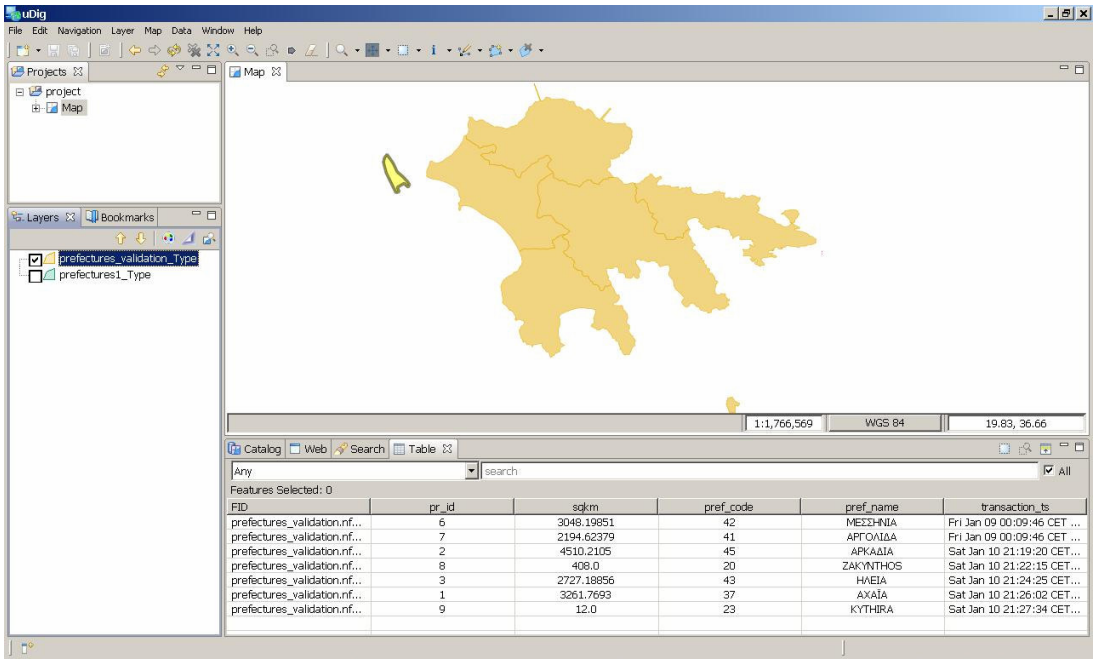
Output pane

Data Output | Explain | Messages | History

	pr_id integ	sqkm numeric	pref_ chara	pref_name character var	transaction_ts timestamp without tim	shape geometry
1	6	3048.19851	42	ΜΕΣΣΗΝΙΑ	2008-12-08 23:09:46.234	
2	7	2194.62379	41	ΑΡΓΟΛΙΔΑ	2008-12-08 23:09:46.234	
3	2	4510.21050	45	ΑΡΚΑΔΙΑ	2008-12-10 20:19:20.39	
4	8	408.0	20	ZAKYNTHOS	2008-12-10 20:22:15.171	0106000020E61
5	3	2727.18856	43	ΗΛΕΙΑ	2008-12-10 20:24:25.734	0106000020E61
6	1	3261.76930	37	ΑΧΑΪΑ	2008-12-10 20:26:02.078	
7	9	12.0	23	KYTHIRA	2008-12-10 20:27:34.734	0106000020E61

Πίνακας 4.7 Ο πίνακας επικύρωσης των μεταβολών μετά από επικύρωση με τιμές παραμέτρων: NULL-NULL-0.

Αν οπτικοποιήσουμε την πληροφορία στο περιβάλλον του ΣΓΠ τότε οι αλλαγές θα έχουν την ακόλουθη μορφή:



Εικόνα 4.6 Απεικόνιση της γεωμετρίας του πίνακα prefectures_validation μετά την επικύρωση των επεμβάσεων με τιμές παραμέτρων: NULL-NULL-0.

Παρατηρούμε τόσο στον πίνακα επικύρωσης καθεαυτό όσο και στο πεδίο γεωμετρίας που οπτικοποιήσαμε ότι έχουν ενσωματωθεί όλες οι αλλαγές είτε αυτές αφορούν σε εισαγωγές είτε σε διαγραφές είτε σε τροποποιήσεις γεωμετρικών ή άλλων γνωρισμάτων. Συγκεκριμένα:

- Διαγράφηκαν οι νομοί Κορινθίας και Λακωνίας
- Διατηρήθηκαν και οι δύο τροποποιήσεις μη γεωμετρικών γνωρισμάτων των νομών Αρκαδίας και Ηλείας
- Διατηρήθηκαν και οι δύο τροποποιήσεις γεωμετρικών γνωρισμάτων του νομού Αχαΐας

- Εισήχθησαν και οι δύο νέες γεωμετρίες «ZAKΥNTHOS» και «KΥTHIRA»

Παράδειγμα Β.

Έστω ότι ο διαχειριστής επιθυμεί να συμπεριλάβει στο αντίγραφο επικύρωσης μόνο τις αλλαγές που αφορούν σε εισαγωγές στοιχείων στην βάση δεδομένων άσχετα με την χρονική στιγμή στην οποία αυτές συνέβησαν. Σε αυτή την περίπτωση καλεί την συνάρτηση επικύρωσης ορίζοντας τις τρεις παραμέτρους επικύρωσης ως εξής:

- Έναρξη χρονικής περιόδου: NULL
- Λήξη χρονικής περιόδου: NULL
- Είδος αλλαγών: 3

Τότε η συνάρτηση επικύρωσης θα πρέπει να μεταφέρει στο αντίγραφο επικύρωσης μόνο τις εισαγωγές στοιχείων. Ο πίνακας επικύρωσης θα έχει την ακόλουθη μορφή:

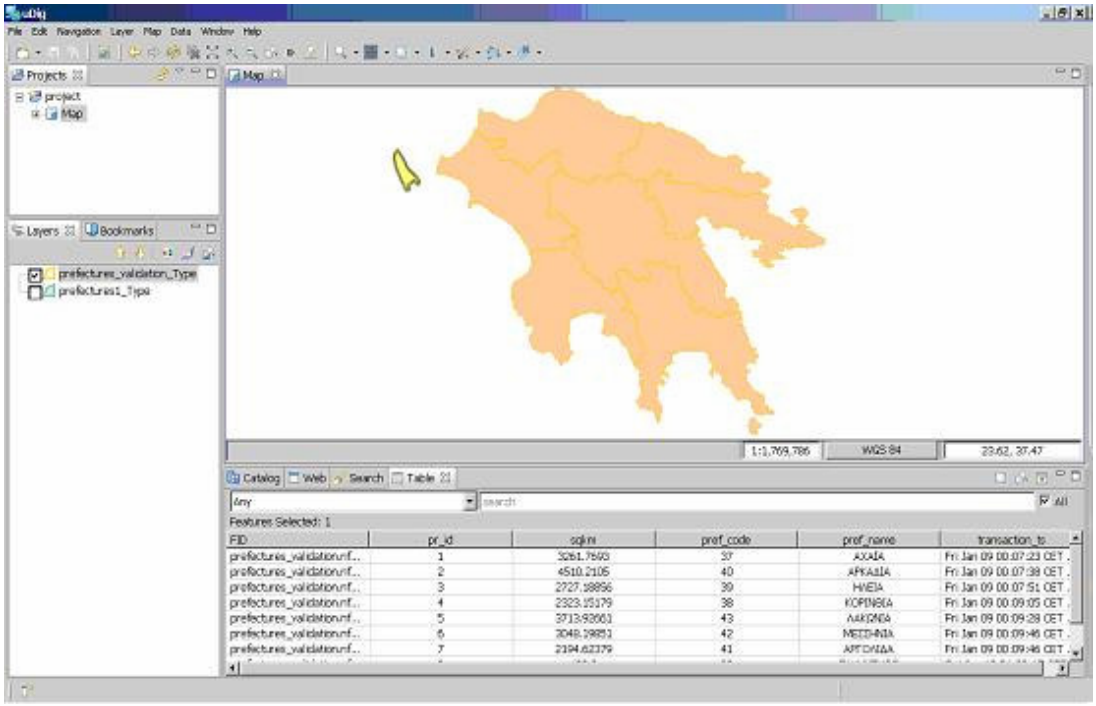
Output pane

Data Output | Explain | Messages | History

	pr_id integer	sqkm numeric	pref_code character var	pref_name character var	transaction_t timestamp wi	shape geometry
1	1	3261.76930	37	ΑΧΑΪΑ	2008-12-08 23:(	
2	2	4510.21050	40	ΑΡΚΑΔΙΑ	2008-12-08 23:(	
3	3	2727.18856	39	ΗΛΕΙΑ	2008-12-08 23:(	0106000020E61
4	4	2323.15179	38	ΚΟΡΙΝΘΙΑ	2008-12-08 23:(	0106000020E61
5	5	3713.92661	43	ΛΑΚΩΝΙΑ	2008-12-08 23:(	
6	6	3048.19851	42	ΜΕΣΣΗΝΙΑ	2008-12-08 23:(	
7	7	2194.62379	41	ΑΡΓΟΛΙΔΑ	2008-12-08 23:(	
8	8	408.0	20	ZAKYNTHOS	2008-12-10 20::	0106000020E61
9	9	12.0	23	KYTHIRA	2008-12-10 20::	0106000020E61

Πίνακας 4.8. Ο πίνακας επικύρωσης των μεταβολών μετά από επικύρωση με τιμές παραμέτρων NULL-NULL-3.

Αν οπτικοποιήσουμε την πληροφορία στο περιβάλλον του ΣΓΠ τότε οι αλλαγές θα έχουν την ακόλουθη μορφή:



Εικόνα 4.7 Απεικόνιση της γεωμετρίας του πίνακα prefectures_validation μετά την επικύρωση των επεμβάσεων με τιμές παραμέτρων: NULL-NULL-3.

Παρατηρούμε τόσο στον πίνακα επικύρωσης καθεαυτό όσο και στο πεδίο γεωμετρίας που οπτικοποιήσαμε ότι έχουν ενσωματωθεί μόνο οι εισαγωγές των δύο νέων γεωμετρίων «ZAKΥNTHOS» και «KΥTHIRA». Τα χρονόσημα των υπόλοιπων εγγραφών είναι τα αρχικά χρονόσημα που είχαν οι εγγραφές που την στιγμή που εισήχθησαν στην βάση δεδομένων.

Παράδειγμα Γ.

Έστω ότι ο διαχειριστής επιθυμεί να συμπεριλάβει στο αντίγραφο επικύρωσης τις αλλαγές που συνέβησαν από την χρονική στιγμή "2008-12-10 20:18:50.00" έως και την χρονική στιγμή "2008-12-10 20:22:20.000" ανεξαρτήτως είδους τροποποίησης. Σε αυτή την περίπτωση ο διαχειριστής της βάσης δεδομένων καλεί την συνάρτηση επικύρωσης ορίζοντας τις τρεις παραμέτρους επικύρωσης ως εξής:

- Έναρξη χρονικής περιόδου: "2008-12-10 20:18:50.00"
- Λήξη χρονικής περιόδου: "2008-12-10 20:22:20.000"
- Είδος αλλαγών: 0

Τότε η συνάρτηση επικύρωσης θα πρέπει να μεταφέρει στο αντίγραφο επικύρωσης μόνο τις εισαγωγές στοιχείων που έγιναν στο χρονικό συγκεκριμένο χρονικό διάστημα. Πράγματι, ο πίνακας επικύρωσης θα έχει την ακόλουθη μορφή:

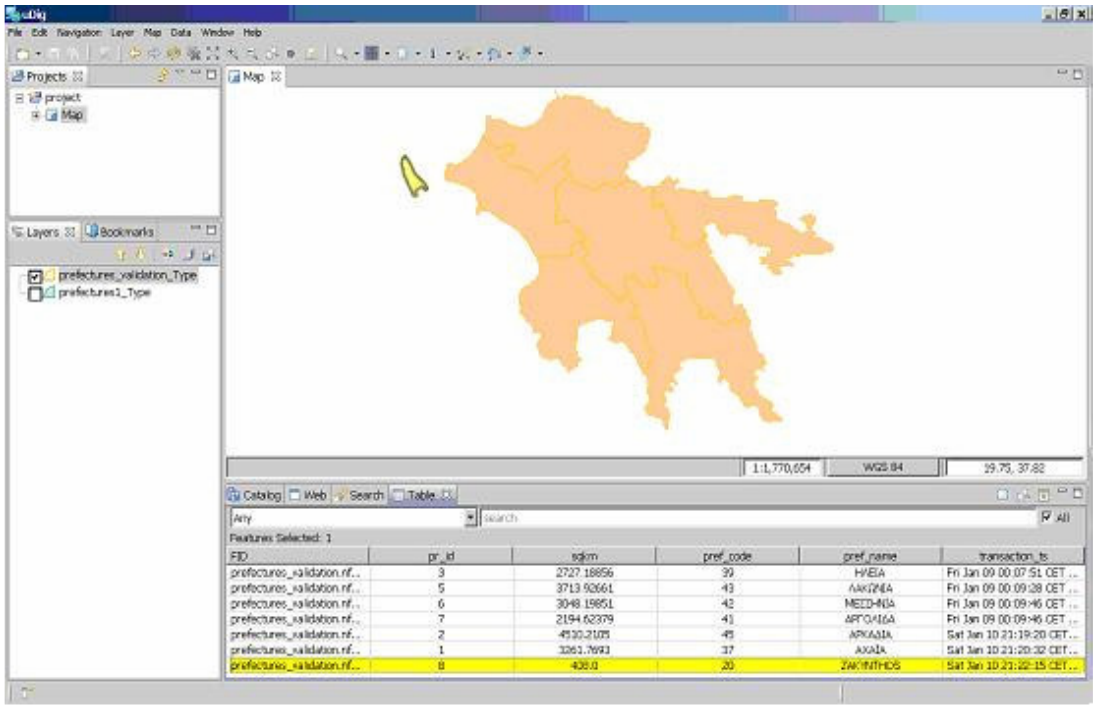
Output pane

Data Output | Explain | Messages | History

	pr_id integer	sqkm numeric	pref_code character var	pref_name character var	transaction_ts timestamp without time z	shape geometry
1	3	2727.18856	39	ΗΛΕΙΑ	2008-12-08 23:07:51.39	0106000020E61
2	5	3713.92661	43	ΛΑΚΩΝΙΑ	2008-12-08 23:09:28.765	
3	6	3048.19851	42	ΜΕΣΣΗΝΙΑ	2008-12-08 23:09:46.234	
4	7	2194.62379	41	ΑΡΓΟΛΙΔΑ	2008-12-08 23:09:46.234	
5	2	4510.21050	45	ΑΡΚΑΔΙΑ	2008-12-10 20:19:20.39	
6	1	3261.76930	37	ΑΧΑΪΑ	2008-12-10 20:20:32.453	
7	8	408.0	20	ZAKYNTHOS	2008-12-10 20:22:15.171	0106000020E61

Πίνακας 4.9. Ο πίνακας επικύρωσης των μεταβολών μετά από επικύρωση με τιμές παραμέτρων: "2008-12-10 20:18:50.00" - "2008-12-10 20:22:20.000" - 0.

Αν οπτικοποιήσουμε την πληροφορία στο περιβάλλον του ΣΓΠ τότε οι αλλαγές θα έχουν την ακόλουθη μορφή:



Εικόνα 4.8 Απεικόνιση της γεωμετρίας του πίνακα prefectures_validation μετά την επικύρωση των επεμβάσεων με τιμές παραμέτρων: "2008-12-10 20:18:50.00"- "2008-12-10 20:22:20.000" - 0.

Παρατηρούμε τόσο στον πίνακα επικύρωσης καθεαυτό όσο και στο πεδίο γεωμετρίας που οπτικοποιήσαμε ότι έχουν ενσωματωθεί οι εξής αλλαγές:

- Διαγράφηκε ο νομός Κορινθίας
- Διατηρήθηκε η τροποποίηση σε ένα από τα μη γεωμετρικά γνωρίσματα του νομού Αρκαδίας που έκανε ο χρήστης1
- Διατηρήθηκε μόνο η πρώτη τροποποίηση στο γεωμετρικό γνωρίσματα του νομού Αχαϊας που έκανε ο χρήστης1
- Διατηρήθηκε μόνο η εισαγωγή της νέας γεωμετρίας «ZAKYNTHOS»

Σύμφωνα με τις τιμές παραμέτρων που ορίσαμε, σε αυτό το παράδειγμα διατηρήθηκαν μόνο οι αλλαγές εκείνες που έγιναν στην χρονική περίοδο από "2008-12-10 20:18:50.00" έως "2008-12-10 20:22:20.000".

Κεφάλαιο 5

Συμπεράσματα

5.1 Επισκόπηση

Στην παρούσα εργασία είδαμε πώς οι τεχνολογικές εφαρμογές που κάνουν χρήση γεωγραφικών πληροφοριών και χωρικών δεδομένων αναπτύσσονται συνεχώς και δημιουργούν την ανάγκη για αποτελεσματική αποθήκευση και διαχείριση των δεδομένων που παράγουν και διαχειρίζονται. Αυτή η ανάγκη οδήγησε στην ανάπτυξη των χωρικών βάσεων δεδομένων. Οι τελευταίες αναπτύχθηκαν με στόχο την αποθήκευση, διαχείριση και να εκτέλεση ερωτημάτων σε χωρικά δεδομένα. Η δυνατότητα που προσφέρουν σε απομακρυσμένους χρήστες να προσπελαύνουν και να επεμβαίνουν στα δεδομένα τους μέσω διαδικτυακών υπηρεσιών, και η ανάγκη – για λόγους οικονομίας χρόνου και υπολογιστικών πόρων - για ταυτόχρονη επέμβαση σε αυτά τα δεδομένα εκ μέρους πολλαπλών χρηστών, δημιουργούν ζητήματα προστασίας της συνέπειας της βάσης δεδομένων.

Το πρότυπο διεπαφών Web Feature Services-Transactional (WFS-T) αποτελεί μία συστηματική προδιαγραφή που έχει προταθεί από το OGC και ορίζει την πρόσβαση και επέμβαση σε χωρικά δεδομένα σε περιβάλλον διαδικτυακών υπηρεσιών σε χωρικές βάσεις δεδομένων. Συγκεκριμένα οι κανόνες ορίζουν την υποβολή και ανταπόκριση ανάμεσα σε χρήστες και εξυπηρετητές με χρήση του πρωτοκόλλου HTTP και της γλώσσας XML.

Η τήρηση ιστορικού σε χωρικές βάσεις δεδομένων αφορά στην καταγραφή των επεμβάσεων που γίνονται στα δεδομένα έτσι ώστε να καθίσταται δυνατή η εκ των υστέρων ανάκτηση της ιστορικής ακολουθίας συναλλαγών.

Μία λύση που έχει προταθεί για την τήρηση ιστορικού σε χωρικές βάσεις δεδομένων προβλέπει την δημιουργία επιπλέον πινάκων εντός της βάσης δεδομένων για την αποθήκευση των στοιχείων που εισάγονται, ενημερώνονται ή διαγράφονται. Με την

χρήση αυτών των πινάκων ο διαχειριστής της βάσης δεδομένων μπορεί να επιλέξει ποιες από αυτές τις αλλαγές επιθυμεί να επικυρώσει στα αρχικά δεδομένα και ποιες όχι.

Στα προηγούμενα κεφάλαια διερευνήσαμε:

- Την δυνατότητα τήρησης ιστορικού των μεταβολών που επήλθαν στα στοιχεία μιας χωρικής βάσης δεδομένων με χρήση βοηθητικών πινάκων εντός της β.δ.
- Την δυνατότητα της χρήσης των δεδομένων που περιέχονται στους βοηθητικούς πίνακες στο πλαίσιο της ελεγχόμενης επικύρωσης μέρους των συναλλαγών.

Προκειμένου να γίνει αυτό υλοποιήσαμε ένα τοπικό σύστημα WFS-T αποτελούμενο από τα εξής στοιχεία:

- Λογισμικό συστήματος διαχείρισης βάσης δεδομένων PostgreSQL ενισχυμένο με τον σπόνδυλο χωρικών λειτουργιών PostGIS
- Δικτυακό εξυπηρετητή χωρικών δεδομένων GeoServer
- Δικτυακό Σύστημα Γεωγραφικών πληροφοριών UDIG

Στην βάση δεδομένων εισάγαμε στοιχεία σχετικά με τους νομούς της Πελοποννήσου. Υλοποιήσαμε υποδομή τήρησης ιστορικού από σκανδαλιστές και συναρτήσεις, σε περιβάλλον SQL/PLpgSQL, που αυτόματα ενημερώνουν τους βοηθητικούς πίνακες της βάσης με σκοπό την τήρηση ιστορικού μεταβολών. Επιπλέον υλοποιήσαμε συνάρτηση επικύρωσης που πραγματοποιεί σειριακή ενσωμάτωση σε αντίγραφο του αρχικού πίνακα των αλλαγών που περιέχονται στους βοηθητικούς πίνακες. Έτσι πετύχαμε την επιλεκτική επικύρωση των αλλαγών που έχουν αποθηκευθεί στους βοηθητικούς πίνακες με βάση την χρονική στιγμή που συνέβησαν ή/και το είδος τους (εισαγωγή διαγραφή τροποποίηση). Καταφέραμε να προσπελάσουμε και να τροποποιήσουμε με το λογισμικό δικτυακού ΓΣΠ UDIG γεωγραφικά δεδομένα που ήταν αποθηκευμένα στην χωρική βάση δεδομένων (PostgreSQL/PostGIS) δημοσιευμένα με το λογισμικό GeoServer. Το σύστημα τήρησης ιστορικού λειτούργησε με επιτυχία και έτσι όλες οι αλλαγές που κάναμε αποθηκεύθηκαν στο ζεύγος βοηθητικών πινάκων.

Επιπλέον εφαρμόστηκε με επιτυχία η επιλεκτική ενσωμάτωση των ιστορικών αλλαγών σε αντίγραφο του αρχικού πίνακα. Χρησιμοποιήθηκαν παράμετροι που προσδιόρισαν την ενσωμάτωση αλλαγών διαφορετικής φύσης και που συνέβησαν σε διαφορετικές χρονικές περιόδους.

5.2 Αποτίμηση

Κλείνοντας την παρούσα διπλωματική εργασία έχει σημασία να αποτιμήσουμε την συνεισφορά της στην έρευνα σχετικά με τις διαδικτυακές υπηρεσίες σε χωρικές βάσεις δεδομένων. Συνολικά μπορούμε να επισημάνουμε τα ακόλουθα:

- Οι διαδικτυακές υπηρεσίες χωρικών δεδομένων γίνονται ολοένα και πιο σημαντικές στην επικοινωνία χωρικών δεδομένων. Συνεχώς γίνονται διαθέσιμες νέες εφαρμογές, άλλοτε εξειδικευμένες και άλλοτε για το ευρύ κοινό, που κάνουν χρήση των πρωτοκόλλων WMS, WFS ή WFS-T. Ειδικότερα, το πρωτόκολλο WFS-T γίνεται σταδιακά πιο διαδεδομένο

παρέχοντας την δυνατότητα για προσπέλαση και επεξεργασία δεδομένων σε ένα σύστημα, όλες οι ενότητες του οποίου (σύστημα διαχείρισης χωρικής βάσης δεδομένων, λογισμικό ΣΓΠ) δεν βρίσκονται εγκατεστημένες στη ίδια τοπική μονάδα. Η παρούσα εργασία υλοποιεί την δυνατότητα τήρησης ιστορικού με ελεγχόμενη επικύρωση αλλαγών σε ένα περιβάλλον λογισμικών όπου μέχρι τώρα κάτι τέτοιο δεν ήταν διαθέσιμο και έτσι διευρύνει τον ορίζοντα δυνατοτήτων σε αυτό το πεδίο εφαρμογών.

- Αναδεικνύεται ο ρόλος των ενσωματωμένων διαδικαστικών γλωσσών σε χωρικές βάσεις δεδομένων. Με την χρήση αυτών των γλωσσών δημιουργούνται κομμάτια κώδικά τα οποία ενσωματώνονται σε γλώσσα SQL, και στην συνέχεια αποθηκεύονται και εκτελούνται από τον εξυπηρετητή. Το αποτέλεσμα αυτής της οργάνωσης λειτουργιών είναι η μείωση υπολογιστικής επιβάρυνσης χρήστη –εξυπηρετητή που έχει ως αποτέλεσμα την δραματική αύξηση ταχύτητας ερωτημάτων. Επιπλέον η δόμηση ενσωματωμένων τμημάτων κώδικά διαδικαστικών γλωσσών αυξάνει την δυνατότητα για χρήση στοιχείων ελέγχου που επιτρέπουν την αποτελεσματικότερη διαχείριση των δεδομένων και ευκολότερη ανάπτυξη λειτουργιών. Αυτό αποδεικνύεται στην παρούσα εργασία με την δημιουργία σειράς στοιχείων ελέγχου όπως σκανδαλιστές, συναρτήσεις, βρόχοι που όλα μαζί αποτελούν τα δομικά στοιχεία του συστήματος τήρησης ιστορικού και ελεγχόμενης επικύρωσης των αλλαγών.
- Η σημασία της δυνατότητας τήρησης ιστορικού σε σύστημα PostgreSQL/PostGIS. Η PostgreSQL είναι μία πολύ ισχυρή και αξιόπιστη βάση δεδομένων. Ενισχυμένη με τον σπόνδυλο χωρικών λειτουργιών PostGIS αποτελεί ένα αξιόπιστο σύστημα διαχείρισης χωρικής βάσης δεδομένων. Ταυτόχρονα πρόκειται για ένα συνδυασμό ελεύθερων και ανοικτού κώδικα λογισμικών. Αυτή η πρωτότυπη προσπάθεια ανάπτυξης υποδομής τήρησης ιστορικού σε αυτό το ελεύθερο / ανοιχτού κώδικα λογισμικό βάζει τα θεμέλια για την ανάπτυξη του τελειοποίηση του συστήματος έτσι ώστε να παρέχει αποτελέσματα που να είναι συγκρίσιμα με τα εμπορικά και κλειστού κώδικα λογισμικά. Επιπλέον το γεγονός καθαυτό ότι πρόκειται για λογισμικά ανοικτού κώδικα επιτρέπει την τροποποίηση ή ενσωμάτωση του κώδικά τους σε άλλες εφαρμογές. Κατά συνέπεια η μεθοδολογία που αναπτύξαμε εδώ μπορεί να ενσωματωθεί σε παραπλήσιες εφαρμογές υπάρχουσες ή μελλοντικές.

5.3 Μελλοντικές επεκτάσεις

Ολοκληρώνοντας την παρούσα μελέτη είναι σκόπιμο να γίνει αναφορά σε ορισμένα σημεία του συστήματος τα οποία δεν αναπτύχθηκαν περαιτέρω περιορισμένα από τον ορίζοντα της μελέτης. Τα σημεία αυτά μπορούν να αναπτυχθούν στο μέλλον:

5.3.1 Διεπαφή χρήστη / διαχειριστή: Το σύστημα που αναπτύχθηκε στην παρούσα εργασία δεν συμπεριέλαβε την δημιουργία κάποιας διεπαφής

χρήστη ή κάποιας εξειδικευμένης για την εφαρμογή διεπαφής για τον διαχειριστή. Τόσο ο χρήστης όσο και ο διαχειριστής αλληλεπιδρούν με το σύστημα μέσω του προγράμματος ΣΓΠ ο πρώτος και μέσω του SQL/PLpgSQL editor ο δεύτερος. Στην περίπτωση του χρήστη η δημιουργία εξειδικευμένης διεπαφής μπορεί να κάνει την χρήση πιο απλή. Μπορεί ακόμα να μην απαιτεί την τοπική εγκατάσταση κάποιου λογισμικού ΣΓΠ καθώς είναι δυνατό η διεπαφή να «τρέχει» μέσα σε κάποιο λογισμικό περιηγητή διαδικτύου. Επιπλέον μία διεπαφή χρήστη μπορεί να είναι προσανατολισμένη σε μία ειδική εφαρμογή του συστήματος WFS-T όπως για παράδειγμα διεπαφή για τον χρήστη που συλλέγει δεδομένα στο πεδίο (λ.χ. προκειμένου να λειτουργεί σε φορητές συσκευές, ή συσκευές που συνδέονται με σύστημα GPS και είναι προσαρμοσμένες σε αυτοκίνητο για τη ψηφιοποίηση οδικών δικτύων) είτε διεπαφή που είναι προσανατολισμένη σε ειδικό τύπο δεδομένων (λχ. μία διεπαφή σχεδιασμένη για την επέμβαση σε γραμμικά στοιχεία με σκοπό την χρήση σε διαχείριση δικτύων ηλεκτρισμού, επικοινωνιών ή ύδρευσης). Η διεπαφή μπορεί να επιτρέπει στον διαχειριστή να εκτελεί μία σειρά λειτουργιών χωρίς να συντάσσει εντολές αλλά επιλέγοντας παραμέτρους σε γραφικό περιβάλλον. Τέτοιες λειτουργίες είναι η κλήση της συνάρτησης επικύρωσης, η επιθεώρηση των βοηθητικών πινάκων, η ενεργοποίηση λειτουργίας τήρησης ιστορικού για κάποιον πίνακα κ.α.

5.3.2 Επέκταση των δυνατοτήτων επιλεκτικής επικύρωσης. Μέχρι στιγμής ο διαχειριστής μπορεί να επικυρώσει αλλαγές προσδιορίζοντας τα χρονικά περιθώρια (αρχή - τέλος) μέσα στα οποία έγιναν καθώς και το είδος των αλλαγών. Και οι δύο όμως αυτές κατηγορίες κριτηρίων δεν παρέχουν επαρκή ευελιξία. Όσο αφορά στη χρονική περίοδο μέσα στην οποία έγιναν οι αλλαγές αυτή πρέπει να είναι συνεχής. Δηλαδή δεν υπάρχει η δυνατότητα το χρονικό διάστημα μέσα στο οποίο συνέβησαν οι αλλαγές που θέλουμε να επικυρωθούν να αποτελείται από δύο η περισσότερα επιμέρους χρονικά διαστήματα. Επιπλέον η επιλογή του είδους των αλλαγών που επιθυμείται να επικυρωθούν είναι και αυτή σχετικά «άκαμπτη». Ο χρήστης μπορεί να επιλέξει να επικυρώσει μόνο τις εισαγωγές είτε μόνο τις διαγραφές είτε μόνο τις ενημερώσεις είτε όλες τις αλλαγές ανεξάρτητα από το είδος τους. Όμως κάποιος πιο σύνθετο συνδυασμός, όπως για παράδειγμα η επικύρωση διαγραφών και εισαγωγών αλλά όχι των ενημερώσεων ή η επικύρωση των διαγραφών και των ενημερώσεων αλλά όχι των εισαγωγών, δεν υποστηρίζεται ακόμη. Συνεπώς ένα σημαντικό βήμα περαιτέρω ανάπτυξης της βάσης δεδομένων θα αφορά στην παροχή περισσότερων δυνατοτήτων επιλεκτικής επικύρωσης.

5.3.3 Ενσωμάτωση της έννοιας του χρήστη στην λειτουργία της βάσης δεδομένων και της διαδικασίας επικύρωσης. Οι διαφορετικοί χρήστες που προσπελαύνουν και τροποποιούν τα χωρικά δεδομένα που είναι αποθηκευμένα στην βάση μπορούν να έχουν διαφορετικά δικαιώματα προσπέλασης των δεδομένων έτσι ώστε η λειτουργία της βάσης να ανταποκρίνεται σε πραγματικές ανάγκες ενός οργανισμού που χρειάζεται να αποθηκεύει και να προσπελαύνει χωρικά δεδομένα. Για παράδειγμα είναι δυνατό οι χρήστες να έχουν διαφορετικά προνόμια πρόσβασης που να τους

επιτρέπουν ή όχι να τροποποιούν τα δεδομένα. Αυτό δικαιολογείται από την ανάγκη για ασφάλεια των δεδομένων και της συνέπειας της βάσης δεδομένων. Επιπλέον είναι σκόπιμο κάποιες τροποποιήσεις να είναι ορατές σε ορισμένους μόνο χρήστες όταν για παράδειγμα αυτές οι αλλαγές βρίσκονται σε στάδιο σχεδιασμού και δεν έχουν γίνει ακόμα αποδεκτές. Αυτό προϋποθέτει την δυνατότητα δημιουργίας διαφορετικών όψεων των δεδομένων και την επικύρωση των μεταβολών σε προσωρινές όψεις. Κατά την επικύρωση αυτή θα επικυρώνονται επιλεκτικά οι μεταβολές με χρήση παραμέτρων που θα περιλαμβάνονται στις σχετικές για κάθε μεταβολή εγγραφές των βοηθητικών πινάκων μεταβολών και όπου θα περιέχονται πληροφορίες σχετικά με το χρήστη που έκανε την μεταβολή όσο και το επίπεδο πρόσβασης στα δεδομένα της μεταβολής. Αυτή η διευθέτηση ενδεχομένως να απαιτεί και έναν επιπλέον βοηθητικό πίνακα όπου θα αποθηκεύονται τα προνόμια των χρηστών. Τέλος η αποθήκευση στις εγγραφές των βοηθητικών πινάκων μεταβολών πληροφοριών σχετικά με την ταυτότητα του χρήστη που έχει δημιουργήσει τις αλλαγές στα δεδομένα μπορεί να χρησιμοποιηθεί και για την διευθέτηση σε περίπτωση διαμάχης μεταβολών. Μπορεί δηλαδή να οριστεί ότι σε περίπτωση διαμάχης δύο μεταβολών σχετικά με το ίδιο στοιχείο η μεταβολή ενός χρήστη θα υπερισχύει έναντι αυτής ενός άλλου.

5.3.4 Επεμβάσεις όχι σε αντίγραφα δεδομένων αλλά σε όψεις ή εκδόσεις δεδομένων. Σημαντική οικονομία αποθηκευτικού χώρου, μνήμης RAM και υπολογιστικών πόρων θα προκύψει από την υλοποίηση κατά την οποία οι χρήστες δεν θα «βλέπουν» αντίγραφα δεδομένων που θα δημιουργούνται για κάθε ένα από αυτούς αλλά προσωρινές όψεις, οι αλλαγές πάνω στις οποίες θα αποθηκεύονται αποκλειστικά και μόνο στους βοηθητικούς πίνακες. Ανάλογος τρόπος λειτουργίας εφαρμόζεται ήδη σε εμπορικά λογισμικά. Η λύση που έχουμε επιλέξει στο σύστημα που υλοποιήσαμε στην παρούσα εργασία προβλέπει ένα αντίγραφο του βασικού πίνακα δεδομένων ανά χρήστη. Με αυτό τον τρόπο είναι δυνατό να δημιουργηθούν πολλαπλά αντίγραφα του βασικού πίνακα δεδομένων (σε περίπτωση που μεγάλος αριθμός χρηστών προσπελαύνει ταυτόχρονα την χωρική βάση δεδομένων) γεγονός που θα έχει σαν αποτέλεσμα την εξάντληση των υπολογιστικών πόρων του συστήματος.

5.3.5 Διαχείριση διαμαχών. Ένα σημαντικό μέρος ενός συστήματος τήρησης ιστορικού έχει να κάνει με την αυτόματη διαχείριση περιπτώσεων όπου δύο ή παραπάνω χρήστες επιδρούν πάνω στο ίδιο στοιχείο με αποτέλεσμα να εμφανίζεται διαμάχη ανάμεσα σε αυτές τις μεταβολές. Εδώ χρειάζεται να οριστούν κάποιοι γενικοί κανόνες επικύρωσης που θα ορίζουν την συμπεριφορά του συστήματος σε περίπτωση διαμάχης. Μπορεί για παράδειγμα να οριστεί ότι από τις αντικρουόμενες μεταβολές θα διατηρείται η μεταγενέστερη, ή θα διατηρείται αυτή που έχει γίνει από χρήστη με ανώτερα προνόμια τροποποιήσεων. Επιπλέον αυτών των γενικών κανόνων χρειάζεται να υλοποιηθεί και η διεπαφή που θα ενημερώνει τον διαχειριστή της βάσης δεδομένων σχετικά με το ότι οι αλλαγές τις οποίες επιθυμεί να επικυρώσει περιέχουν διαμάχες και τελικά θα του ζητά να επιλέξει αν

επιθυμεί να προχωρήσει η επικύρωση σύμφωνα με τους γενικούς κανόνες του συστήματος διαχείρισης της χωρικής βάσης δεδομένων.

Βιβλιογραφία

[DRA+03] http://bcdra.refractions.net/geotec2003_dra1_paper.doc Graeme Leeming, Mark Sondheim, άρθρο σχετικά με εφαρμογή βάσης χωρικών δεδομένων με δεδομένα οδικού δικτύου από την περιοχή "British Columbia" του Καναδά ανεπτυγμένη σε περιβάλλον PostGIS. Refractions Research, 2003.

[EN04] R. Elmasri, and S.B. Navathe. Fundamentals of database systems, Addison-Wesley, 2004.

[ESR+04] Versioning: An ESRI Technical Paper, Chapter from forthcoming book: Inside a Geodatabase, ESRI, 2004.

[GEO08] **http://geoserver.org/display/GEOSDOC/Documentation** ιστοτόπος τεκμηρίωσης του λογισμικού GeoServer, προσπελάθηκε στις 05/10/2008.

[GEP+04]http://www.gepower.com/prod_serv/products/gis_software/en/downloads/GER4231_ManagingChange.pdf. Άρθρο σχετικά με το λογισμικό «Smallworld» της εταιρίας General Electric. General Electric, 2004.

[GER08]http://geoserver.org/display/GEOS/Versioning+WFS+Requirements Σύνοψη των προδιαγραφών για την επικείμενη ανάπτυξη επεκτάσεων τήρησης ιστορικού για το λογισμικό εξυπηρετητή χωρικών δεδομένων GeoServer. Προσπελάσθηκε στις 06/12/2008.

[GET+04]http://www.ambaoyasoc.com.ar/pdf/transaction%20manager.pdf. Product Sheet: GeoMedia Transaction Manager. Intergraph Corporation, 2004.

[GML+03] OGC GML, Geography Markup Language (GML) 3.0 Implementation Specification, OpenGIS Consortium, 2003.

[GMUW02] H. Garcia-Molina, J. D. Ullman, and J. Widom. Database Systems: the Complete Book, Prentice Hall, 2002.

[GR03] J. Gehrke, and R. Ramakrishnan. Database Management Systems, 3rd edition, McGraw-Hill, 2003.

[OGC+02] P. A. Vretanos, et al (eds) Web Feature Service Implementation Specification, Version: 1.0.0, Open GIS Consortium inc, 2002.

[ORA+07]http://www.oracle.com/technology/products/database/workspace_manager/pdf/twp_AppDev_Workspace_Manager_topology_11gR1.pdf, Bill Beauregard and Chuck Murray, Oracle Database 11g Workspace Manager Support for Oracle Spatial Topology Model, An Oracle White Paper, Oracle, June 2007.

[PL/pg08]http://www.postgresql.org/docs/current/static/plpgsql.html πίνακας περιεχομένων του ιστοτόπου τεκμηρίωσης της γλώσσας PL/pgSQL, Προσπελάσθηκε στις 05/10/2008.

[POS08] http://postgis.refractions.net/documentation/, ιστοτόπος τεκμηρίωσης του λογισμικού επέκτασης χωρικών λειτουργιών PostGIS, προσπελάθηκε στις 05/10/2008.

[POS08B]http://postgis.refractions.net/pipermail/postgis-users/2003-July/002726.html. Διάλογος σχετικά με τις δυνατότητες τήρησης ιστορικού στην επέκταση χωρικών λειτουργιών PostGIS στον ιστοτόπο ανάπτυξης του λογισμικού. Προσπελάσθηκε στις 06/12/2008.

[SKS04] A. Silberschatz, H. Korth, and S. Sudarshan. Database system concepts, 4th edition, McGraw-Hill, 2004.

[UDI08] http://udig.refractions.net/, ιστοτόπος του λογισμικού UDIG, προσπελάθηκε στις 05/10/2008.

Παράρτημα

Π.1 Τεκμηρίωση πηγαίου κώδικα

Στο παράρτημα που ακολουθεί παραθέτουμε διεξοδικά τον πηγαίο κώδικα που δημιουργήσαμε για την ανάπτυξη μεθοδολογίας συστήματος τήρησης ιστορικού σε χωρική βάση δεδομένων. Στις τρεις ενότητες που ακολουθούν παρατίθεται αντίστοιχα ο κώδικας για το πολυγωνικό, γραμμικό και σημειακό θεματικό επίπεδο. Στην πρώτη από αυτές ενότητες περιλαμβάνεται αναλυτική τεκμηρίωση ενώ στις δύο άλλες αυτό δεν συμβαίνει. Ο λόγος είναι πως και στις τρεις περιπτώσεις η δομή του κώδικα είναι η παραπλήσια και ως εκ τούτου η παράθεση επιπλέον τεκμηρίωσης δεν θα προσφέρει παρά ελάχιστα.

Π.1.1 Μεθοδολογία ανάπτυξης συστήματος τήρησης ιστορικού σε πολυγωνικό θεματικό επίπεδο και επεξήγηση

Π1.1.1 Δημιουργία βάσης δεδομένων

```
CREATE DATABASE plpgsql_for_loop_1_13_PREFS
WITH
ENCODING='UTF8'
OWNER = postgres
TEMPLATE= template_postgis
TABLESPACE = pg_default;
```

Δημιουργούμε βάση δεδομένων ενισχυμένη με χωρικές λειτουργίες PostGIS και συμβολοσειρά UTF8.

Π1.1.2 Δημιουργία βασικού πίνακα και εισαγωγή δεδομένων

```
CREATE TABLE public.prefectures_default
       (pr_id Integer PRIMARY KEY,
       sqkm numeric,
       pref_code varchar(10),
       pref_name varchar(60),
       transaction_ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP);

SELECT AddGeometryColumn('public','prefectures_default','shape','4326',
'MULTIPOLYGON',2);

--Εισάγω τα στοιχεία των νόμων της Πελοποννήσου από το αρχείο
--nomoi_pelloponnese.sql
```

Προσδιορίζουμε τον βασικό πίνακα. Τον ονομάζουμε prefectures_default. Ο πίνακας έχει τα ακόλουθα γνωρίσματα τα οποία επιλέξαμε για να περιγράψουμε βασικές ιδιότητες της χωρικής ενότητας «Νομός»:

- Αύξων αριθμός (pr_id) πρωτεύων κλειδί
- Επιφάνεια (sqkm) αλφαριθμητική μεταβλητή
- Κωδικός νομού (pref_code) αλφαριθμητική μεταβλητή
- Όνομα Νομού (pref_name) αλφαριθμητική μεταβλητή
- Χρονική στιγμή συναλλαγής (transaction_ts) χρονόσημο
- Γεωμετρία (shape) πολυγωνικό γεωμετρικό γνώρισμα με προβολή στο WGS84.

Αφού δομήσουμε τον βασικό πίνακα εισάγουμε σε αυτόν δεδομένα σχετικά με τρεις νομούς της Πελοποννήσου.

Π1.1.3 Αντίγραφα του αρχικού πίνακα και πίνακες αποθήκευσης των χρονοσήμων:

```
CREATE TABLE public.prefectures1
       (pr_id Integer PRIMARY KEY,
       sqkm numeric,
       pref_code varchar(10),
       pref_name varchar(60));

SELECT
AddGeometryColumn('public','prefectures1','shape','4326','MULTIPOLYGON',2);

CREATE TABLE public.prefectures1_ts
       (pr_id Integer PRIMARY KEY,
       transaction_ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP);

INSERT INTO prefectures1
SELECT pr_id, sqkm, pref_code, pref_name, shape
FROM prefectures_default;

INSERT INTO prefectures1_ts
SELECT pr_id, transaction_ts
FROM prefectures_default;
```

Για κάθε έναν από τους χρήστες που επεμβαίνουν στον πίνακα της χωρικής βάσης δεδομένων για τον οποίο έχει ενεργοποιηθεί η τήρηση ιστορικού δημιουργείται ένα αντίγραφο του πίνακα. Το αντίγραφο περιέχει όλα τα γνωρίσματα του αρχικού πίνακα εκτός από το χρονόσημο. Το χρονόσημο για τα αντίγραφα αποθηκεύεται

στους πίνακες αποθήκευσης χρονοσήμων. Τα αντίγραφα του αρχικού πίνακα ονομάζονται παραπλήσια με τον αρχικό πίνακα και η ονομασία τους περιέχει τον αύξοντα αριθμό του χρήστη. Στο παράδειγμά μας ο αρχικός πίνακας ονομάζεται prefectures_default και τα αντίγραφά του prefectures1, prefectures2, prefectures3 κ.ο.κ. Τα αντίγραφα του αρχικού πίνακα περιέχουν ένα γνώρισμα πρωτεύον κλειδί όπως και ο αρχικός πίνακας. Οι πίνακες τήρησης χρονοσήμου έχουν μόνο δύο γνωρίσματα, α) ένα πρωτεύον κλειδί που συνδέει κάθε εγγραφή τους με μία εγγραφή του αντίστοιχου αντιγράφου β) γνώρισμα χρονοσήμου που αντιστοιχεί στο χρονόσημο της αντίστοιχης εγγραφής του αντιγράφου. Το χρονόσημο του αντιγράφου αποθηκεύει πληροφορία σχετικά με την χρονική στιγμή στην οποία δημιουργήθηκε η τροποποιήθηκε η σχετική εγγραφή. Οι πίνακες αποθήκευσης χρονοσήμων ονομάζονται σύμφωνα με τους πίνακες αντίγραφα στους οποίου αντιστοιχούν με την επίθεση του προσδιοριστικού «_ts» που δηλώνει ότι η λειτουργία τους έχει να κάνει με την τήρηση χρονοσήμου. Στο παράδειγμά μας, στον πίνακα-αντίγραφο prefectures1 αντιστοιχεί ο πίνακας διατήρησης χρονοσήμου prefectures1_ts, στον πίνακα prefectures2 αντιστοιχεί ο πίνακας διατήρησης χρονοσήμου prefectures2_ts κ.ο.κ.

Π1.1.4 Βοηθητικοί πίνακες

```
CREATE TABLE public.prefectures_insert
   (pr_id Integer,
      sqkm numeric,
      pref_code varchar(10),
      pref_name varchar(60),
    Record_ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    Transaction_ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    Transaction_type varchar (10),
      UNIQUE (pr_id, transaction_ts)
);

SELECT
AddGeometryColumn('public','prefectures_insert','shape','4326','MULTIPOLYGON
',2);


CREATE TABLE public.prefectures_delete
   (pr_id Integer,
      sqkm numeric,
      pref_code varchar(10),
      pref_name varchar(60),
     Record_ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
     Transaction_ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
     Transaction_type varchar (10),
       UNIQUE (pr_id, transaction_ts)
);

SELECT
AddGeometryColumn('public','prefectures_delete','shape','4326','MULTIPOLYGON
',2);
```

Για κάθε πίνακα στον οποίο ενεργοποιείται η τήρηση ιστορικού δημιουργούνται δύο βοηθητικοί πίνακες. Σε αυτούς τους πίνακες αποθηκεύονται οι τροποποιήσεις που γίνονται στα αντίγραφα του αρχικού πίνακα από το σύνολο των χρηστών. Στον ένα από τους δύο βοηθητικούς πίνακες αποθηκεύονται οι εισαγωγές

εγγραφών και στοιχεία που αφορούν στην μορφή που έχουν οι εγγραφές μετά τις ενημερώσεις τους. Τον πίνακα αυτό ονομάζουμε επιθέτοντας με το «_insert» και στο παράδειγμά μας θα ονομάσουμε «prefectures_insert». Στον άλλο βοηθητικό πίνακα αποθηκεύονται οι διαγραφές εγγραφών και κάποια στοιχεία που αφορούν στην μορφή που είχαν οι εγγραφές πριν από τις ενημερώσεις τους. Τον πίνακα αυτό ονομάζουμε επιθέτοντας με το «_delete» και στο παράδειγμά μας θα ονομάσουμε «prefectures_delete». Οι βοηθητικοί πίνακες περιέχουν δύο στοιχεία επιπλέον αυτών που περιείχε ο αρχικός πίνακας. Το ένα από αυτά ονομάζεται record_ts και εξυπηρετεί στο να προσδιορίζει στις περιπτώσεις διαγραφών και ενημερώσεων το χρονόσημο της εγγραφής που καταργήθηκε ή τροποποιήθηκε. Το άλλο επιπλέον γνώρισμα ονομάζεται transaction type και προσδιορίζει το είδος της επέμβασης (εισαγωγή, διαγραφή ή τροποποίηση) που αποθηκεύει η εγγραφή. Ο τρόπος με τον οποίο εγγράφονται στοιχεία στους βοηθητικούς πίνακες αναπτύσσεται αναλυτικά στην επόμενη ενότητα.

Π1.1.5 Σκανδαλιστής (trigger function) και συνάρτηση (function) που ενεργοποιείται από τον σκανδαλιστή.

```
CREATE OR REPLACE FUNCTION insert_delete_prefectures1() RETURNS trigger AS '
DECLARE
       OLD_TS TIMESTAMP;
       NEW_TS TIMESTAMP;
BEGIN

       NEW_TS:= CURRENT_TIMESTAMP;

       IF tg_op = ''DELETE'' THEN
              OLD_TS:=(SELECT transaction_ts
              FROM public.prefectures1_ts
              WHERE pr_id=old.pr_id);
              INSERT INTO prefectures_delete(pr_id, sqkm, pref_code,
              pref_name, Record_ts, Transaction_ts, Transaction_type, shape)
              VALUES (old.pr_id, old.sqkm, old.pref_code, old.pref_name,
              OLD_TS, NEW_TS, 1, old.shape);

              DELETE FROM public.prefectures1_ts WHERE pr_id = old.pr_id;
              RETURN old;
        END IF;

       IF tg_op = ''INSERT'' THEN

              INSERT INTO prefectures_insert(pr_id, sqkm, pref_code,
              pref_name, Record_ts, Transaction_ts, Transaction_type, shape)
              VALUES (new.pr_id, new.sqkm, new.pref_code, new.pref_name,
              NEW_TS, NEW_TS, 3, new.shape);
              INSERT INTO public.prefectures1_ts VALUES (new.pr_id, NEW_TS);
              RETURN new;
              END IF;

       IF tg_op = ''UPDATE'' THEN

              BEGIN
              OLD_TS:=(SELECT transaction_ts
              FROM public.prefectures1_ts
              WHERE pr_id=old.pr_id);

              INSERT INTO prefectures_delete(pr_id, sqkm, pref_code,
              pref_name, Record_ts, Transaction_ts, Transaction_type, shape)
```

```
            VALUES (old.pr_id, old.sqkm, old.pref_code, old.pref_name,
            OLD_TS, NEW_TS, 2, old.shape);
            INSERT INTO prefectures_insert(pr_id, sqkm, pref_code,
            pref_name, Record_ts, Transaction_ts, Transaction_type, shape)
            VALUES (new.pr_id, new.sqkm, new.pref_code, new.pref_name,
            NEW_TS, NEW_TS, 2, new.shape);

            UPDATE public.prefectures1_ts SET transaction_ts = NEW_TS WHERE
            pr_id = new.pr_id;

            RETURN old;
      END;
 END IF;

RETURN new;
END
'LANGUAGE plpgsql;

CREATE TRIGGER insert_delete_prefectures1 AFTER INSERT OR DELETE OR UPDATE
ON prefectures1
FOR EACH ROW EXECUTE PROCEDURE insert_delete_prefectures1();
```

Σε κάθε ένα από τα αντίγραφα του αρχικού πίνακα δημιουργούμε έναν σκανδαλιστή ο οποίος ενεργοποιείται με κάθε επέμβαση (εισαγωγή, διαγραφή, ενημέρωση) στο αντίγραφο και ενεργοποιεί την συνάρτηση ενημέρωσης των βοηθητικών πινάκων. Η τελευταία ονομάζεται κατά τον αρχικό πίνακα με το πρόθεμα «insert_delete_». Στο παράδειγμά μας η συνάρτηση ενημέρωσης των βοηθητικών πινάκων ονομάζεται insert_delete_prefectures. Η συνάρτηση λειτουργεί με τον ακόλουθο τρόπο:
Αρχικά ορίζει δύο μεταβλητές:

OLD_TS
NEW_TS

Και οι δύο αυτές μεταβλητές έχουν μορφή χρονοσήμου. Η δεύτερη από αυτές τις μεταβλητές ορίζεται ως το χρονόσημο της παρούσας χρονικής στιγμής που εκτελείται η συνάρτηση. Η πρώτη ορίζεται κατά περίπτωση αργότερα. Στην συνέχεια η συνάρτηση διακρίνει τρεις υποπεριπτώσεις ανάλογα με τον τύπο επέμβασης στο αντίγραφο του αρχικού πίνακα:

- Αν η επέμβαση αφορά σε διαγραφή εγγραφής:
    - Η μεταβλητή OLD_TS ορίζεται ως το χρονόσημο από τον πίνακα διατήρησης χρονοσήμου που αφορά στην εγγραφή που διαγράφεται.
    - Μέσω της παραμέτρου συστήματος «.old» της βάσης δεδομένων ο βοηθητικός πίνακας «_delete» ενημερώνεται με όλα τα στοιχεία της εγγραφής που διαγράφεται εκτός από το χρονόσημό του αποθηκεύεται στο γνώρισμα record_ts (στο γνώρισμα transaction_ts αποθηκεύεται η (παρούσα) χρονική στιγμή που εκτελείται η συνάρτηση) και στο γνώρισμα transaction_type αποθηκεύεται ο κωδικός 1 που προσδιορίζει ότι αυτή η εγγραφή του βοηθητικού πίνακα αφορά σε διαγραφή.
    - Διαγράφεται από τον πίνακα διατήρησης χρονοσήμου η εγγραφή που αφορά στην εγγραφή του αντιγράφου πίνακα που καταργείται με την διαγραφή.

- Αν επέμβαση αφορά σε εισαγωγή στοιχείου:

- Μέσω της παραμέτρου συστήματος «.new» της βάσης δεδομένων ο βοηθητικός πίνακας «_insert» ενημερώνεται με όλα τα στοιχεία της εγγραφής που εισάγεται. Σαν χρονόσημό του αποθηκεύεται μέσω της μεταβλητής NEW_TS η (παρούσα) χρονική στιγμή που εκτελείται η συνάρτηση χρονική τόσο στο γνώρισμα transaction_ts όσο και στο γνώρισμα record_ts. Στο γνώρισμα transaction_type αποθηκεύεται ο κωδικός 3 που προσδιορίζει ότι αυτή η εγγραφή του βοηθητικού πίνακα αφορά σε εισαγωγή στοιχείου.
- Εισάγεται στον πίνακα διατήρησης χρονοσήμου μία νέα εγγραφή που περιέχει την τιμή του γνωρίσματος «πρωτεύον κλειδί» της εγγραφής που μόλις εισήχθη και την τιμή της μεταβλητής NEW_TS που προσδιορίζει την χρονική στιγμή που εκτελείται η συνάρτηση.

- Αν επέμβαση αφορά σε ενημέρωση στοιχείου:
  - Η μεταβλητή «OLD_TS» ορίζεται ως η εγγραφή χρονοσήμου από τον πίνακα διατήρησης χρονοσήμου που αφορά στην εγγραφή που ενημερώνεται. Με άλλα λόγια ως OLD_TS ορίζεται ως η τιμή χρονοσήμου (μεταβλητή transaction_ts) της εγγραφής που ενημερώνεται.
  - Μέσω της παραμέτρου συστήματος «.old» της βάσης δεδομένων ο βοηθητικός πίνακας «_delete» ενημερώνεται με όλα τα στοιχεία της εγγραφής με την μορφή που αυτά τα στοιχεία είχαν πριν την ενημέρωση. Το χρονόσημό του αποθηκεύεται στο γνώρισμα record_ts μέσω της μεταβλητής OLD_TS (στο γνώρισμα transaction_ts αποθηκεύεται η (παρούσα) χρονική στιγμή που εκτελείται η συνάρτηση) και στο γνώρισμα transaction_type αποθηκεύεται ο κωδικός 2 που προσδιορίζει ότι αυτή η εγγραφή του βοηθητικού πίνακα αφορά σε ενημέρωση.
  - Μέσω της παραμέτρου συστήματος «.new» της βάσης δεδομένων ο βοηθητικός πίνακας «_insert» ενημερώνεται με όλα τα νέα στοιχεία της εγγραφής που ενημερώνεται. Σαν χρονόσημο αποθηκεύεται μέσω της μεταβλητής NEW_TS η (παρούσα) χρονική στιγμή που εκτελείται η συνάρτηση τόσο στο γνώρισμα transaction_ts όσο και στο γνώρισμα record_ts. Στο γνώρισμα transaction_type αποθηκεύεται ο κωδικός 2 που προσδιορίζει ότι αυτή η εγγραφή του βοηθητικού πίνακα αφορά σε ενημέρωση εγγραφής.
  - Μέσω της μέσω της μεταβλητής NEW_TS ενημερώνεται ο πίνακας διατήρησης χρονοσήμου για το νέο χρονόσημο της εγγραφής που ενημερώνεται.

Π1.1.6 Συνάρτηση επικύρωσης των αλλαγών που έχουν αποθηκευτεί στους δύο βοηθητικούς πίνακες:

```
CREATE OR REPLACE VIEW prefectures_insert_delete
AS
```

```
(SELECT * FROM prefectures_INSERT) UNION ALL (SELECT * FROM
prefectures_DELETE)
ORDER BY record_ts, Transaction_type ASC;

CREATE TABLE prefectures_log
(prefectures_id INTEGER,
transaction_ts TIMESTAMP,
note_msg VARCHAR(200));


CREATE OR REPLACE FUNCTION prefectures_reconciliate_transactions (start_ts
TIMESTAMP, end_ts TIMESTAMP, trans_type INTEGER) RETURNS INTEGER AS $$

DECLARE
  target RECORD;
  cur_start TIMESTAMP;
  cur_end TIMESTAMP;
  cur_type INTEGER;
  curr_count integer;

BEGIN
       cur_start := start_ts;
       cur_end := end_ts;
       cur_type := trans_type;

       EXECUTE 'DELETE FROM prefectures_log;';
       EXECUTE 'DROP TABLE prefectures_validation;';
       EXECUTE 'CREATE TABLE prefectures_validation AS SELECT * FROM
       prefectures_default;';

       IF cur_start IS NULL
              THEN cur_start := (SELECT MIN(transaction_ts)
              FROM prefectures_insert_delete);
       END IF;

       IF cur_end IS NULL
              THEN cur_end := (SELECT MAX(transaction_ts)
              FROM prefectures_insert_delete);
       END IF;

       FOR
       target IN     (SELECT *
                     FROM prefectures_insert_delete
                     WHERE transaction_ts>= cur_start
                     AND transaction_ts <= cur_end
              ORDER BY transaction_ts, transaction_type, record_ts asc)
       LOOP

              IF target.transaction_type = 1 AND cur_type IN (0, 1) THEN

              BEGIN
                     EXECUTE 'DELETE FROM prefectures_validation'
                     || ' WHERE '
                     || quote_literal(target.pr_id)
                     || ' = prefectures_validation.pr_id ; ' ;

                     EXECUTE 'INSERT INTO prefectures_log VALUES ('
                     || quote_literal(target.pr_id)
                     || ','
                     || quote_literal(target.record_ts)
                     || ','
                     || quote_literal(target.transaction_ts)
                     || ');' ;
              END;

              ELSIF target.Transaction_type = 2 AND cur_type IN (0, 2) THEN

              BEGIN
```

```
curr_count := (SELECT COUNT(*) FROM prefectures_default
WHERE pr_id = target.pr_id);

IF target.transaction_ts != target.record_ts THEN

       EXECUTE 'DELETE FROM prefectures_validation'
       || ' WHERE '
       || quote_literal(target.pr_id)
       || ' = prefectures_validation.pr_id '
       || ' AND '
       || quote_literal(target.record_ts)
       || '= prefectures_validation.transaction_ts ;';

       ELSIF curr_count = 1 THEN

       EXECUTE 'INSERT INTO prefectures_validation
       VALUES('
       ||quote_literal(target.pr_id)
       ||','
       ||quote_literal(target.sqkm)
       ||','
       ||quote_literal(target.pref_code)
       ||','
       ||quote_literal(target.pref_name)
       ||','
       ||quote_literal(target.Transaction_ts)
       ||','
       ||quote_literal(target.shape)
       ||');';

       END IF;

       EXECUTE 'INSERT INTO prefectures_log VALUES ('
       || quote_literal(target.pr_id)
       || ','
       || quote_literal(target.record_ts)
       || ','
       || quote_literal(target.transaction_ts)
       || ');' ;
END;

ELSIF target.Transaction_type = 3 AND cur_type IN (0, 3)
THEN
       BEGIN

       EXECUTE'INSERT INTO prefectures_validation
       VALUES('
       ||quote_literal(target.pr_id)
       ||','
       ||quote_literal(target.sqkm)
       ||','
       ||quote_literal(target.pref_code)
       ||','
       ||quote_literal(target.pref_name)
       ||','
       ||quote_literal(target.Transaction_ts)
       ||','
       ||quote_literal(target.shape)
       ||');';

       EXECUTE 'INSERT INTO prefectures_log VALUES ('
       || quote_literal(target.pr_id)
       || ','
       || quote_literal(target.record_ts)
       || ','
       || quote_literal(target.transaction_ts)
       || ');' ;
END;
```

```
            END IF;
        END LOOP;
        RETURN 1;
END;
$$ LANGUAGE plpgsql;
```

Αφού πλέον οι αλλαγές έχουν αποθηκευτεί στους βοηθητικούς πίνακες ο διαχειριστής της βάσης δεδομένων μπορεί να επιλέξει ποιες από τις αλλαγές θα επικυρώσει. Η συνάρτηση επικύρωσης που αναπτύχθηκε στην παρούσα διπλωματική εργασία επιτρέπει στον διαχειριστή, εφόσον το επιθυμεί, να επικυρώσει επιλεκτικά μέρος των συναλλαγών με κριτήριο το είδος τους (αν δηλαδή πρόκειται για εισαγωγές εγγραφών, διαγραφές εγγραφών, η τροποποιήσεις εγγραφών) ή την χρονική στιγμή που συνέβησαν προσδιορίζοντας την χρονική περίοδο (χρονική στιγμή έναρξης και χρονική στιγμή λήξης) για την οποία επιθυμεί να επικυρωθούν οι συναλλαγές. Ως εκ τούτου η συνάρτηση έχει τρεις παραμέτρους τις οποίες προσδιορίζει ο διαχειριστής της βάσης δεδομένων:

- Χρονική στιγμή έναρξης περιόδου συναλλαγών
- Χρονική στιγμή λήξης περιόδου συναλλαγών
- Είδος συναλλαγών που επιθυμείται να επικυρωθούν (παίρνει τις ακόλουθες τιμές από 0 έως 3):
    - 0: Όλες οι συναλλαγές ανεξάρτητα από το είδος τους
    - 1: Συναλλαγές που αφορούν μόνο σε διαγραφές
    - 2: Συναλλαγές που αφορούν μόνο σε ενημερώσεις εγγραφών
    - 3: Συναλλαγές που αφορούν μόνο σε εισαγωγές εγγραφών

Η συνάρτηση εκτελεί τα ακόλουθα βήματα:

Αρχικά η συνάρτηση ορίζει πέντε μεταβλητές. Αυτές αφορούν στις τρεις παραμέτρους της συνάρτησης που περιγράφονται παραπάνω, σε μία μεταβλητή ελέγχου (curr_count) και μία μεταβλητή με μορφή εγγραφής (target) οι οποίες θα αναπτυχθούν στην συνέχεια. Έπειτα καθαρίζονται περιεχόμενα του πίνακα "_log" καθώς και αυτά του πίνακα επικύρωσης "_validation" που έχουν μείνει σε αυτούς τους πίνακες από προηγούμενες εκτελέσεις της συνάρτησης και μεταφέρονται τα περιεχόμενα του αρχικού πίνακα στον πίνακα επικύρωσης. Σε αυτό το στάδιο ο πίνακας επικύρωσης είναι ένα αντίγραφο του αρχικού πίνακα. Στην συνέχεια η συνάρτηση εξετάζει τις τιμές των χρονικών παραμέτρων της (αρχική και τελική χρονική στιγμή) και αν αυτές είναι μηδενικές (Null) τότε τους δίνει τιμές ίσες με τα πραγματικά χρονικά όρια της συνάρτησης (ελάχιστο μέγιστο).

Στην συνέχεια η συνάρτηση επικύρωσης καλεί μία όψη (view) που αποτελεί την ένωση των δύο βοηθητικών πινάκων στοιχισμένα κατά αύξουσα τιμή των γνωρισμάτων «transaction_ts», «transaction_type», «record_ts». Αυτή η όψη επιπλέον περιέχει τα χρονικά όρια τα οποία έχει προσδιορίσει ο διαχειριστής της συνάρτησης ως εκείνα μέσα στα οποία πραγματοποιήθηκαν οι συναλλαγές που επιθυμεί να επικυρωθούν. Σε αυτήν την όψη λοιπόν ορίζεται η μεταβλητή «target» που έχει την μορφή εγγραφής. Πρόκειται δηλαδή για την ακολουθία τιμών για διαφορετικά γνωρίσματα σε κάθε γραμμή της όψης. Η μεταβλητή target παίρνει διαδοχικά τις τιμές όλων των εγγραφών της όψης. Για αυτή την μεταβλητή καλείται η λειτουργία FOR LOOP. Με άλλα λόγια η λειτουργία FOR LOOP «διαβάζει» διαδοχικά τις εγγραφές της όψης με την σειρά που αυτές είναι στοιχισμένες και τις διαχειρίζεται με προκαθορισμένο τρόπο:

- Αν η τιμή του γνωρίσματος transaction_type της target είναι 1 (δηλαδή η εγγραφή αφορά σε διαγραφή στοιχείου) και η παράμετρος είδους συναλλαγών που ο διαχειριστής επιθυμεί να επικυρωθούν έχει οριστεί ως 0 ή 1 (δηλαδή επιθυμεί να επικυρωθούν είτε όλες οι συναλλαγές είτε μόνο οι διαγραφές) τότε:
    - Διαγράφεται από τον πίνακα επικύρωσης η εγγραφή που έχει την ίδια τιμή πρωτεύοντος κλειδιού με την μεταβλητή target
    - Εισάγονται στον πίνακα «_log» οι τιμές πρωτεύοντος κλειδιού, record_ts (χρονόσημο της εγγραφής που διαγράφηκε κατά την συναλλαγή) transaction_ts (χρονόσημο που προσδιορίζει την χρονική στιγμή της διαγραφής) της μεταβλητής target. Ο πίνακας _log δεν έχει λειτουργικό ρόλο στην τήρηση ιστορικού αλλά βοηθά να ελέγχουμε την λειτουργία της συνάρτησης.

- Αν η τιμή του γνωρίσματος transaction_type της target είναι 2 (δηλαδή η εγγραφή αφορά σε ενημέρωση στοιχείου) και η παράμετρος είδους συναλλαγών που ο διαχειριστής επιθυμεί να επικυρωθούν έχει οριστεί ως 0 ή 2 (επιθυμεί να επικυρωθούν είτε όλες οι συναλλαγές είτε μόνο οι ενημερώσεις εγγραφών) τότε:
    - Αρχικά δίνεται τιμή στην μεταβλητή curr_count. Αυτό έχει σαν σκοπό να ελεγχθεί το αν η εγγραφή που τροποποιήθηκε κατά την συναλλαγή προϋπήρχε στον αρχικό πίνακα (ή έχει εισαχθεί από κάποιο χρήστη πριν από προηγούμενη επικύρωση) ή αν η εγγραφή που τροποποιήθηκε εισήχθη από κάποιο χρήστη μετά την τελευταία επικύρωση, αν δηλαδή αποτελεί μέρος των συναλλαγών διαχειρίζεται η συνάρτηση επικύρωσης. Ο έλεγχος γίνεται διερευνώντας αν η τιμή του πρωτεύοντος κλειδιού της μεταβλητής target υπάρχει στον αρχικό πίνακα. Αν αυτή η τιμή δεν υπάρχει στον αρχικό πίνακα τότε η μεταβλητή curr_start παίρνει τιμή 0 ενώ αν υπάρχει η μεταβλητή παίρνει τιμή 1. Το αν η εγγραφή που τροποποιείται κατά την συγκεκριμένη συναλλαγή είναι νέα η προϋπάρχουσα εγγραφή έχει σημασία γιατί ο διαχειριστής έχει επιλέξει να επικυρωθούν οι τροποποιήσεις υπαρχόντων και όχι νέων στοιχείων.
    - Αφού δοθεί τιμή στην μεταβλητή curr_start η συνάρτηση επικύρωσης προχωρά ως εξής:
        - Αν η εγγραφή target αφορά στο σκέλος διαγραφής της ενημέρωσης (αυτό πιστοποιείται από την συνθήκη transaction_ts!=record_ts) τότε διαγράφεται από τον πίνακα επικύρωσης η εγγραφή για την οποία ισχύει ότι: α) έχει την ίδια τιμή πρωτεύοντος κλειδιού με την μεταβλητή target β) ή τιμή του χρονοσήμου της ισούται με την τιμή record_ts της μεταβλητής target.
        - Αν η εγγραφή target αφορά στο σκέλος εισαγωγής μίας ενημέρωσης (αυτό πιστοποιείται από την συνθήκη transaction_ts=record_ts) και η εγγραφή αφορά σε τροποποίηση στοιχείου το οποίο προϋπάρχει στον αρχικό

πίνακα (αυτό πιστοποιείται από την τιμή 1 της μεταβλητής curr_count) τότε εισάγεται στον πίνακα επικύρωσης μία εγγραφή που περιέχει τις τιμές της μεταβλητής target για όλα τα γνωρίσματα της εκτός από το record_ts και transaction_type (ο πίνακας επικύρωσης δεν διαθέτει αυτά τα γνωρίσματα).

- Εισάγονται στον πίνακα «_log» οι τιμές πρωτεύοντος κλειδιού, record_ts (χρονόσημο της εγγραφής που τροποποιήθηκε κατά την συναλλαγή) transaction_ts (χρονόσημο που προσδιορίζει την χρονική στιγμή της επέμβασης) της μεταβλητής target.

- Αν η τιμή του γνωρίσματος transaction_type της target είναι 3 (δηλαδή η εγγραφή αφορά σε εισαγωγή στοιχείου) και η παράμετρος είδους συναλλαγών που ο διαχειριστής επιθυμεί να επικυρωθούν έχει οριστεί ως 0 ή 3 (δηλαδή επιθυμεί να επικυρωθούν είτε όλες οι συναλλαγές είτε μόνο εκείνες που αφορούν σε εισαγωγές) τότε:
  - Εισάγεται στον πίνακα επικύρωσης μία εγγραφή που περιέχει τις τιμές της μεταβλητής target για όλα τα γνωρίσματά της εκτός από το record_ts και transaction_type.
  - Εεισάγονται στον πίνακα «_log» οι τιμές πρωτεύοντος κλειδιού, record_ts (χρονόσημο της εγγραφής που εισήχθη κατά την συναλλαγή) transaction_ts (χρονόσημο που προσδιορίζει την χρονική στιγμή της επέμβασης) της μεταβλητής target.

Μετά τον τελευταίο έλεγχο τερματίζεται η λειτουργία «FOR LOOP» και έπειτα η συνάρτηση επικύρωσης.

Π1.1.8 Κλήση της συνάρτησης επικύρωσης, προσδιορισμός των σχετικών παραμέτρων και ερώτηση του πίνακα που περιέχει τα αποτελέσματα.

```
Select prefectures_reconciliate_transactions (Null, Null, 2);
Select * from prefectures_validation
order by transaction_ts asc;
```

Ο διαχειριστής της βάσης δεδομένων καλεί την συνάρτηση επικύρωσης ορίζοντας τις σχετικές παραμέτρους επικύρωσης (για παράδειγμα Null, Null, 2) μέσα σε παρένθεση.

Π1.1.9 Ενσωμάτωση των περιεχομένων του αντιγράφου επικύρωσης στον βασικό πίνακα

```
DELETE FROM prefectures_default;
INSERT INTO prefectures_default
SELECT pr_id, sqkm, pref_code, pref_name, shape
FROM prefectures_validation;
```

Όταν πλέον η επικύρωση ολοκληρωθεί με την εκτέλεση της συνάρτησης επικύρωσης τότε ο αρχικός πίνακας μπορεί να αρχειοθετηθεί ή να διαγραφούν τα

περιεχόμενά του και να εγγραφούν σε αυτόν τα περιεχόμενα του πίνακα επικύρωσης.

Π.1.2 Μεθοδολογία ανάπτυξης συστήματος τήρησης ιστορικού σε γραμμικό θεματικό επίπεδο

```
CREATE DATABASE plpgsql_for_loop_1_13_ROADS
WITH
ENCODING='UTF8'
OWNER = nikos
TEMPLATE= template_postgis
TABLESPACE = pg_default;


CREATE TABLE roads_default
(roads_id INTEGER PRIMARY KEY,
length numeric,
road_type int4,
natlcode varchar(12),
intlcode varchar(12),
name varchar(60),
toll int4,
transaction_ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP);

SELECT
AddGeometryColumn('public','roads_default','shape','4326','MULTILINESTRING',
2);

INSERT INTO roads_default
(roads_id,length,road_type,natlcode,intlcode,name,toll,transaction_ts,shape)
VALUES ('16','22765.1878','3','39','E961','TRIPOLI -
SPARTA','0',CURRENT_TIMESTAMP,GeometryFromText('MULTILINESTRING
((22.4388369277982 37.1535211251709,….22.4284403334525
37.3256205192025))',4326) );

INSERT INTO roads_default
(roads_id,length,road_type,natlcode,intlcode,name,toll,transaction_ts,shape)
VALUES ('20','26601.7137','3','33','N/a','TRIPOLI -
MYLOI','0',CURRENT_TIMESTAMP,GeometryFromText('MULTILINESTRING
((21.6883318138736 37.9858277645037,……,21.7275374544813
38.1838262670496))',4326) );

INSERT INTO roads_default
(roads_id,length,road_type,natlcode,intlcode,name,toll,transaction_ts,shape)
VALUES ('21','22979.1554','3','33','N/a','TRIPOLI -
KALAMATA','0',CURRENT_TIMESTAMP, GeometryFromText('MULTILINESTRING
((21.6569277704258 37.847228845649,…,21.8081328133808
37.8611266878998))',4326) );

CREATE TABLE public.roads1
(roads_id INTEGER PRIMARY KEY,
length numeric,
road_type int4,
natlcode varchar(12),
intlcode varchar(12),
name varchar(60),
toll int4);

SELECT
AddGeometryColumn('public','roads1','shape','4326','MULTILINESTRING',2);
```

```
CREATE TABLE public.roads1_ts
(roads_id INTEGER PRIMARY KEY,
transaction_ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP);


INSERT INTO roads1
SELECT roads_id, length, road_type, natlcode, intlcode, name, toll, shape
FROM roads_default;

SELECT *
FROM roads1;

INSERT INTO roads1_ts
SELECT roads_id, transaction_ts
FROM roads_default;

SELECT *
FROM roads1_ts;

CREATE TABLE public.roads_insert
(roads_id INTEGER,
length numeric,
road_type int4,
natlcode varchar(12),
intlcode varchar(12),
name varchar(60),
toll int4,
record_ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
transaction_ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
transaction_type varchar (10),
UNIQUE (roads_id, transaction_ts)
);

SELECT
AddGeometryColumn('public','roads_insert','shape','4326','MULTILINESTRING',2
);

CREATE TABLE public.roads_delete
(roads_id INTEGER,
length numeric,
road_type int4,
natlcode varchar(12),
intlcode varchar(12),
name varchar(60),
toll int4,
record_ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
transaction_ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
transaction_type varchar (10),
UNIQUE (roads_id, transaction_ts)
);

SELECT
AddGeometryColumn('public','roads_delete','shape','4326','MULTILINESTRING',2
);

CREATE OR REPLACE FUNCTION insert_delete_roads() RETURNS trigger AS '
       DECLARE
       OLD_TS TIMESTAMP;
       NEW_TS TIMESTAMP;
       BEGIN
       NEW_TS:= CURRENT_TIMESTAMP;

 IF tg_op = ''DELETE'' THEN

  OLD_TS:=    (SELECT transaction_ts
              FROM public.roads1_ts
              WHERE roads_id=old.roads_id);
```

```
      INSERT INTO roads_delete(roads_id, length, road_type, natlcode,
      intlcode, name, toll, record_ts, transaction_ts, transaction_type,
      shape)
      VALUES(old.roads_id, old.length, old.road_type, old.natlcode,
      old.intlcode, old.name, old.toll, OLD_TS, NEW_TS, 1, old.shape);
      DELETE FROM public.roads1_ts WHERE roads_id = old.roads_id;
      RETURN old;
 END IF;

 IF tg_op = ''INSERT'' THEN

     INSERT INTO roads_insert(roads_id, length, road_type, natlcode,
     intlcode, name, toll, record_ts, transaction_ts, transaction_type,
     shape)
     VALUES (new.roads_id, new.length, new.road_type, new.natlcode,
     new.intlcode, new.name, new.toll, NEW_TS, NEW_TS, 3, new.shape);
   INSERT INTO public.roads1_ts VALUES (new.roads_id, NEW_TS);
   RETURN new;
 END IF;

 IF tg_op = ''UPDATE'' THEN
  BEGIN
      OLD_TS:=(SELECT transaction_ts
      FROM public.roads1_ts
      WHERE roads_id=old.roads_id);

      INSERT INTO roads_delete(roads_id, length, road_type, natlcode,
      intlcode, name, toll, record_ts, transaction_ts, transaction_type,
      shape)
      VALUES(old.roads_id, old.length, old.road_type, old.natlcode,
      old.intlcode, old.name, old.toll, OLD_TS, NEW_TS, 2, old.shape);

      INSERT INTO roads_insert(roads_id, length, road_type, natlcode,
      intlcode, name, toll, record_ts, transaction_ts, transaction_type,
      shape)
      VALUES (new.roads_id, new.length, new.road_type, new.natlcode,
      new.intlcode, new.name, new.toll, NEW_TS, NEW_TS, 2, new.shape);

      UPDATE public.roads1_ts SET transaction_ts = NEW_TS
      WHERE roads_id = new.roads_id;
   RETURN old;
  END;
 END IF;
 RETURN new;
END
'LANGUAGE plpgsql;

CREATE TRIGGER insert_delete_roads AFTER INSERT OR DELETE OR UPDATE ON
roads1
FOR EACH ROW EXECUTE PROCEDURE insert_delete_roads();


CREATE OR REPLACE VIEW roads_insert_delete AS
       (SELECT * FROM roads_insert) UNION ALL (SELECT * FROM roads_delete)
       ORDER BY record_ts, Transaction_type ASC;

CREATE TABLE public.roads_validation(
roads_id INTEGER PRIMARY KEY,
length numeric,
road_type int4,
natlcode varchar(12),
intlcode varchar(12),
name varchar(60),
toll int4,
transaction_ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP);
```

```
SELECT
AddGeometryColumn('public','roads_validation','shape','4326','MULTILINESTRIN
G',2);

INSERT INTO roads_validation
SELECT *
FROM roads_default;

CREATE TABLE roads_log
(roads_id INTEGER,
transaction_ts TIMESTAMP,
note_msg VARCHAR(200));

CREATE OR REPLACE FUNCTION roads_reconciliate_transactions (start_ts
TIMESTAMP, end_ts TIMESTAMP, trans_type INTEGER) RETURNS INTEGER AS $$

       DECLARE
         target RECORD;
         cur_start TIMESTAMP;
         cur_end TIMESTAMP;
         cur_type INTEGER;
         curr_count integer;

BEGIN
       cur_start := start_ts;
       cur_end := end_ts;
       cur_type := trans_type;

       EXECUTE 'DELETE FROM roads_log;';
       EXECUTE 'DROP TABLE roads_validation;';
       EXECUTE 'CREATE TABLE roads_validation AS SELECT * FROM
       roads_default;';

       IF cur_start IS NULL
              THEN cur_start := (SELECT MIN(transaction_ts)
              FROM roads_insert_delete);
       END IF;

       IF cur_end IS NULL
              THEN cur_end := (SELECT MAX(transaction_ts)
              FROM roads_insert_delete);
       END IF;

       FOR target IN       (SELECT *
                           FROM roads_insert_delete
                           WHERE transaction_ts>= cur_start
                           AND transaction_ts <= cur_end
                           ORDER BY transaction_ts, transaction_type asc)
       LOOP

       IF target.Transaction_type = 1 AND cur_type IN (0, 1) THEN

              BEGIN
                     EXECUTE 'DELETE FROM roads_validation'
                                  || ' WHERE '
                                  || quote_literal(target.roads_id)
                                  || ' = roads_validation.roads_id ' ;

              EXECUTE 'INSERT INTO roads_log VALUES ('
                                  || quote_literal(target.roads_id)
                                  || ','
                                  || quote_literal(target.record_ts)
                                  || ','
                                  || quote_literal(target.transaction_ts)
                                  || ');' ;
              END;

       ELSIF target.transaction_type = 2 AND cur_type IN (0, 2) THEN
```

```
BEGIN
              curr_count :=       (SELECT COUNT(*) FROM roads_default
                                  WHERE roads_id = target.roads_id);

              IF target.transaction_ts != target.record_ts THEN

              EXECUTE 'DELETE FROM roads_validation'
                     || ' WHERE '
                     || quote_literal(target.roads_id)
                     || ' = roads_validation.roads_id '
                     || ' AND '
                     || quote_literal(target.record_ts)
                     || '= roads_validation.transaction_ts ;';

              ELSIF curr_count = 1 THEN

              EXECUTE 'INSERT INTO roads_validation VALUES ('
                     || quote_literal(target.roads_id)
                     || ','
                     || quote_literal(target.length)
                     || ','
                     || quote_literal(target.road_type)
                     || ','
                     || quote_literal(target.natlcode)
                     || ','
                     || quote_literal(target.intlcode)
                     || ','
                     || quote_literal(target.name)
                     || ','
                     || quote_literal(target.toll)
                     || ','
                     || quote_literal(target.transaction_ts)
                     || ','
                     || quote_literal(target.shape)
                     || ');';

                    END IF;

              EXECUTE 'INSERT INTO roads_log VALUES ('
              || quote_literal(target.roads_id)
              || ','
              || quote_literal(target.record_ts)
              || ','
              || quote_literal(target.transaction_ts)
              || ');' ;
              END;

 ELSIF target.Transaction_type = 3 AND cur_type IN (0, 3) THEN
       BEGIN

       EXECUTE 'INSERT INTO roads_validation VALUES ('
              || quote_literal(target.roads_id)
              || ','
              || quote_literal(target.length)
              || ','
              || quote_literal(target.road_type)
              || ','
              || quote_literal(target.natlcode)
              || ','
              || quote_literal(target.intlcode)
              || ','
              || quote_literal(target.name)
              || ','
              || quote_literal(target.toll)
              || ','
              || quote_literal(target.transaction_ts)
              || ','
```

```
                    || quote_literal(target.shape)
                    || ');';

        EXECUTE 'INSERT INTO roads_log VALUES ('
             || quote_literal(target.roads_id)
             || ','
             || quote_literal(target.record_ts)
             || ','
             || quote_literal(target.transaction_ts)
             || ');' ;

                    END;
             END IF;
       END LOOP;
       RETURN 1;
END;
$$ LANGUAGE plpgsql;

Select roads_reconciliate_transactions (Null, Null, 2);
Select * from roads_validation
order by transaction_ts asc;
```

Π.1.3 Μεθοδολογία ανάπτυξης συστήματος τήρησης ιστορικού σε σημειακό θεματικό επίπεδο

```
CREATE DATABASE plpgsql_for_loop_1_13_POIS
WITH
ENCODING='UTF8'
OWNER = nikos
TEMPLATE= template_postgis
TABLESPACE = pg_default;

CREATE TABLE public.pois_default(
     pois_id INTEGER PRIMARY KEY,
     pois_categ INTEGER,
     pois_name varchar(50),
     transaction_ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);

SELECT AddGeometryColumn('public','pois_default','shape','4326','POINT',2);

INSERT INTO public.pois_default VALUES (1, 4110,'Hotel Tegea',
CURRENT_TIMESTAMP,GeomFromText('POINT(22.29346 37.59231)', 4326));

INSERT INTO public.pois_default VALUES (2, 5110, 'Riza Public Hospital',
CURRENT_TIMESTAMP, GeomFromText('POINT(22.28505 37.51318)', 4326));

INSERT INTO public.pois_default VALUES (3, 6110, 'Stadio Police Station',
CURRENT_TIMESTAMP, GeomFromText('POINT(22.29356 37.59241)', 4326));

CREATE TABLE public.pois1(
     pois_id INTEGER PRIMARY KEY,
     pois_categ INTEGER,
     pois_name varchar(50)
);

SELECT AddGeometryColumn('public','pois1','shape','4326','POINT',2);

CREATE TABLE public.pois1_ts(
     pois_id INTEGER PRIMARY KEY,
     transaction_ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);
```

```
INSERT INTO pois1
SELECT pois_id, pois_categ, pois_name, shape
FROM POIS_DEFAULT;

SELECT *
FROM POIS1;

INSERT INTO pois1_ts
SELECT pois_id, transaction_ts
FROM POIS_DEFAULT;

SELECT *
FROM POIS1_TS;

CREATE TABLE public.pois_insert (
     pois_id INTEGER,
     pois_categ INTEGER,
     pois_name varchar(50),
     Record_ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
     Transaction_ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
     Transaction_type varchar (10),
        UNIQUE (pois_id, transaction_ts)
);

SELECT AddGeometryColumn('public','pois_insert','shape','4326','POINT',2);


CREATE TABLE public.pois_delete (
     pois_id INTEGER,
     pois_categ INTEGER,
     pois_name varchar(50),
     Record_ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
     Transaction_ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
     Transaction_type varchar (10),
        UNIQUE (pois_id, transaction_ts)
);

SELECT AddGeometryColumn('public','pois_delete','shape','4326','POINT',2);


CREATE OR REPLACE FUNCTION insert_delete_pois() RETURNS trigger AS '
       DECLARE
              OLD_TS TIMESTAMP;
              NEW_TS TIMESTAMP;
       BEGIN
       NEW_TS:= CURRENT_TIMESTAMP;

IF tg_op = ''DELETE'' THEN
       OLD_TS:=     (SELECT transaction_ts
                    FROM public.pois1_ts
                    WHERE pois_id=old.pois_id);

       INSERT INTO pois_delete(pois_id, pois_categ, pois_name, record_ts,
       transaction_ts, Transaction_type, shape)
       VALUES (old.pois_id, old.pois_categ, old.pois_name, OLD_TS, NEW_TS, 1,
       old.shape);

       DELETE FROM public.pois1_ts WHERE pois_id = old.pois_id;
       RETURN old;
 END IF;

IF tg_op = ''INSERT'' THEN
INSERT INTO pois_insert(pois_id, pois_categ, pois_name, record_ts,
transaction_ts, Transaction_type, shape)
VALUES (new.pois_id, new.pois_categ, new.pois_name, NEW_TS, NEW_TS, 3,
new.shape);
   INSERT INTO public.pois1_ts VALUES (new.pois_id, NEW_TS);
```

```
 RETURN new;
 END IF;

IF tg_op = ''UPDATE'' THEN
       BEGIN
              OLD_TS:=     (SELECT transaction_ts
                           FROM public.pois1_ts
                           WHERE pois_id=old.pois_id);

              INSERT INTO pois_delete(pois_id, pois_categ, pois_name,
              record_ts, transaction_ts, Transaction_type, shape)
              VALUES (old.pois_id, old.pois_categ, old.pois_name, OLD_TS,
              NEW_TS, 2, old.shape);

              INSERT INTO pois_insert(pois_id, pois_categ, pois_name,
              record_ts, transaction_ts, Transaction_type, shape)
              VALUES (new.pois_id, new.pois_categ, new.pois_name, NEW_TS,
              NEW_TS,2, new.shape);

              UPDATE public.pois1_ts SET transaction_ts = NEW_TS WHERE
              pois_id = new.pois_id;
       RETURN old;
       END;
 END IF;
 RETURN new;
END
'LANGUAGE plpgsql;

CREATE TRIGGER insert_delete_pois AFTER INSERT OR DELETE OR UPDATE ON pois1
FOR EACH ROW EXECUTE PROCEDURE insert_delete_pois();


CREATE OR REPLACE VIEW pois_insert_delete
       AS
       (SELECT * FROM POIS_INSERT) UNION ALL (SELECT * FROM POIS_DELETE)
       ORDER BY record_ts, Transaction_type ASC;

CREATE TABLE public.pois_validation(
     pois_id INTEGER PRIMARY KEY,
     pois_categ INTEGER,
     pois_name varchar(50),
     transaction_ts TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
);

SELECT
AddGeometryColumn('public','pois_validation','shape','4326','POINT',2);

INSERT INTO pois_validation
SELECT *
FROM POIS_DEFAULT;

CREATE TABLE pois_log
(pois_id INTEGER,
transaction_ts TIMESTAMP,
note_msg VARCHAR(200));


CREATE OR REPLACE FUNCTION pois_reconciliate_transactions (start_ts
TIMESTAMP, end_ts TIMESTAMP, trans_type INTEGER) RETURNS INTEGER AS $$

       DECLARE
              target RECORD;
              cur_start TIMESTAMP;
              cur_end TIMESTAMP;
              cur_type INTEGER;
              curr_count integer;

       BEGIN
```

```
        cur_start := start_ts;
        cur_end := end_ts;
        cur_type := trans_type;

        EXECUTE 'DELETE FROM pois_log;';
        EXECUTE 'DROP TABLE pois_validation;';
        EXECUTE 'CREATE TABLE pois_validation AS SELECT * FROM
        pois_default;';

IF cur_start IS NULL THEN
        cur_start := (SELECT MIN(record_ts)
                      FROM pois_insert_delete);
END IF;

IF cur_end IS NULL THEN
        cur_end :=    (SELECT MAX(record_ts)
                      FROM pois_insert_delete);
END IF;


FOR target IN         (SELECT *
                      FROM pois_insert_delete
                      WHERE transaction_ts>= cur_start
                      AND transaction_ts <= cur_end
                      ORDER BY transaction_ts, transaction_type asc)
LOOP

        IF target.Transaction_type = '1' AND cur_type IN (0, 1) THEN

                BEGIN
                        EXECUTE 'DELETE FROM pois_validation'
                                    ||' WHERE '
                                    || quote_literal(target.pois_id)
                                    || ' = pois_validation.pois_id ';


                        EXECUTE 'INSERT INTO pois_log VALUES ('
                               || quote_literal(target.pois_id)
                               || ','
                               || quote_literal(target.record_ts)
                               || ','
                               || quote_literal(target.transaction_ts)
                               || ');' ;
                        END;

        ELSIF target.Transaction_type = '2' AND cur_type IN (0, 2) THEN

                BEGIN
                        curr_count :=        (SELECT COUNT(*) FROM
                                             pois_default
                                             WHERE pois_id =
                                             target.pois_id);

                IF target.transaction_ts != target.record_ts THEN

                        EXECUTE 'DELETE FROM pois_validation'
                        || ' WHERE '
                        || quote_literal(target.pois_id)
                        || ' = pois_validation.pois_id '
                        || ' AND '
                        || quote_literal(target.record_ts)
                        || '= pois_validation.transaction_ts ;';

                ELSIF curr_count = 1 THEN

                        EXECUTE 'INSERT INTO pois_validation VALUES ('
                        || quote_literal(target.pois_id)
                        || ','
```

```
                        || quote_literal(target.pois_categ)
                        || ','
                        || quote_literal(target.pois_name)
                        || ','
                        || quote_literal(target.transaction_ts )
                        || ','
                        || quote_literal(target.shape)
                        || ');';

                        END IF;

                EXECUTE 'INSERT INTO pois_log VALUES ('
                || quote_literal(target.pois_id)
                || ','
                || quote_literal(target.record_ts)
                || ','
                || quote_literal(target.transaction_ts)
                || ');' ;
                        END;

        ELSIF target.Transaction_type = '3' AND cur_type IN (0, 3) THEN
                BEGIN

                EXECUTE 'INSERT INTO pois_validation VALUES ('
                        || quote_literal(target.pois_id)
                        || ','
                        || quote_literal(target.pois_categ)
                        || ','
                        || quote_literal(target.pois_name)
                        || ','
                        || quote_literal(target.transaction_ts )
                        || ','
                        || quote_literal(target.shape)
                        || ');';

                EXECUTE 'INSERT INTO pois_log VALUES ('
                        || quote_literal(target.pois_id)
                        || ','
                        || quote_literal(target.record_ts)
                        || ','
                        || quote_literal(target.transaction_ts)
                        || ');' ;

                        END;
                END IF;

                END LOOP;
        RETURN 1;
END;
$$ LANGUAGE plpgsql;


Select pois_reconciliate_transactions ('2008-10-11 14:20:00.000', '2008-10-
11 14:30:00.000', '1');
Select * from pois_validation
order by transaction_ts asc;
```

Π2. Ορολογία

Αδιέξοδο	Deadlock
Αναίρεση συναλλαγής	Transaction rollback
Αντοχή / ανθεκτικότητα	Durability
Απομόνωση	Isolation
Ατομικότητα	Atomicity
Γεωαναφερόμενα δεδομένα	Georeferenced data
Γράφος προτεραιότητας	Precedence graph
Διαγραφή	Delete
Διαχειριστής κλειδωμάτων	Lock manager
Διένεξη / διαμάχη	Conflict
Δικτυακές υπηρεσίες δεδομένων	Web Feature Service (WFS)
Δικτυακές υπηρεσίες δεδομένων με δυνατότητα συναλλαγών	Web Feature Service - Transactional (WFS-T)
Δικτυακές υπηρεσίες χαρτών	Web Map Service (WMS)
Δοκιμή επικύρωσης	Validation test
Εγγραφή δεδομένων	Data Record
Εισαγωγή	Insert
Έκδοση της βάσης δεδομένων	Database version
Έλεγχος συγχρονικότητας	Concurrency control
Ενημέρωση / τροποποίηση	Update
Εξυπηρετητής	Servers
Επικύρωση	Validation
Εφαρμογή χρήστης	Client application
Κλείδωμα δύο φάσεων	Two-phase locking
Κλείδωμα ευρετηρίου	Index locking
Κλείδωμα με δείκτες	Index locking
Μακρές συναλλαγές	Long transactions
Πρωτόκολλα κλειδώματος	Locking protocols
Πρωτόκολλο πολλαπλών εκδόσεων	Multiversion protocol
Σειριακότητα όψης	View serialisability
Σειριακότητας διαμάχης	Conflict serialisability
Σκανδαλιστής	Trigger
Συγχρονική εκτέλεση συναλλαγών	Concurrent transaction execution
Συναλλαγές ανάγνωσης	Read only transactions
Συναλλαγές εγγραφής	Write transactions
Συνάρτηση	Function
Συνέπεια	Consistency
Συνέπεια βάσης δεδομένων	Database consistency
Σύντομες συναλλαγές	Short transactions
Σχήμα διάταξης χρονοσήμων	Timestamp ordering scheme
Σχήμα ελέγχου ανάκτησης	Recovery control scheme

Σχήμα ελέγχου συγχρονικότητας	Concurrency control scheme
Σχήμα επικύρωσης	Validation scheme
Σχήματα ελέγχου συγχρονικότητας	Concurrency control schemes
Υπολογιστικό έργο/φόρτος	Overhead
Φαινόμενο φάντασμα / φαντασμάτων	Phantom phenomenon
Φόρτος επικοινωνίας χρήστη-εξυπηρετητή	Client/server communication overhead
Χρονοδιάγραμμα	Schedule
Χρονόσημο	Timestamp