ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟΥΠΟΛΗ ΙΩΑΝΝΙΝΩΝ, ΙΩΑΝΝΙΝΑ

ΔΙΠΛ-2010-01 14 Ιουλίου 2010

HECATE

an SQL Schema diff viewer

Ιωάννης Σκουλής

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ: Π. Βασιλειάδης

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Πρόλογος

Κατά την Ελληνική Μυθολογία η Εκάτη φέρεται ως μια χθόνια θεότητα που συνδέεται με την μαγεία και τα σταυροδρόμια. Στον ελληνικό κόσμο η κλασική μορφή της Εκάτης στέκει αυστηρή και παράξενη, ανάγλυφη πάνω σε ένα τρίγωνο, με τα πρόσωπά της στραμμένα σε τρεις κατευθύνσεις. Οι Έλληνες προσπάθησαν να απαλλαγούν από την αυστηρότητα αυτών των αγαλμάτων διασπώντας την τρισυπόστατη θεότητα σε τρεις παρθένες χορεύτριες.[1] Ο συχνότερος μύθος που αφορά την Εκάτη αναφέρεται στην μετάβαση: Είναι η προστάτιδα των θυρών, εποπτεύει την γέννηση και το θάνατο και οδηγεί όσους τολμούν να διασχίσουν τους δύο κόσμους.[2] Έτσι λοιπόν μας υπενθυμίζει την σημασία της αλλαγής, βοηθώντας μας να απελευθερωθούμε από το παρελθόν, ιδίως εκείνα τα πράγματα και τις καταστάσεις που παρεμποδίζουν την ανάπτυξή μας, και να δεχθούμε την αλλαγή και την μετάβαση. Μερικές φορές μας ζητεί να αφήσουμε ό,τι θεωρούμε οικείο και ασφαλές και να ταξιδέψουμε στα σκοτεινά μονοπάτια της ψυχής.[3]

14 Ιουλίου 2010
Ιωάννης Σκουλής

[1] http://el.wikipedia.org/wiki/Εκάτη
[2] http://hekate.timerift.net/roles.htm
[3] http://www.goddessgift.com/goddess-myths/greek_goddess_hecate.htm

Περίληψη

Η τεχνολογία των Βάσεων Δεδομένων (ΒΔ) αποτελεί τον πυρήνα των σύγχρονων πληροφοριακών συστημάτων και δη των διαδικτυακών πληροφοριακών συστημάτων. Σε συνδυασμό με την ταχύτατη ανάπτυξη λογισμικού, που αποτελεί τις περισσότερες φορές λογισμικό ανοιχτού κώδικα που συντηρείται από κοινότητες, βρισκόμαστε αντιμέτωποι με το δύσκολο πρόβλημα διαχείρισης των αλλαγών που επιφέρει μια τέτοια εξέλιξη. Στην εργασία αυτή παρουσιάζεται η Εκάτη, ένα εργαλείο για την οπτικοποίησης των αλλαγών μεταξύ δύο εκδόσεων ενός σχήματος μιας ΒΔ. Στην συνέχεια χρησιμοποιούμε το εργαλείο αυτό για να μελετήσουμε την εξελικτική πορεία της ΒΔ την Βικιπαίδειας. Η Βικιπαίδεια αποτελεί ένα χαρακτηριστικό παράδειγμα ενός μεγάλου διαδικτυακού πληροφοριακού συστήματος που χρησιμοποιεί το MediaWiki, ένα λογισμικό ανοιχτού κώδικα.

Λέξεις Κλειδιά: Βάσεις Δεδομένων, οπτικοποίηση, diff

Abstract

Databases are the core of modern information systems and web information systems in particular. In addition to the fast software development, which in many cases is open-source software maintained by communities, we face the difficult task of managing the changes that result from evolving. In this thesis we introduce Hecate, a tool for visualizing the changes between two schema versions of a database. We then use this tool to study the evolution of the database of Wikipedia. Wikipedia is a well-known example of a large web information system built using the open-source software MediaWiki.

Keywords: Data Bases, schema, diff, visualization

Περιεχόμενα

1 Εισαγωγή

Κάθε Πληροφοριακό Σύστημα (ΠΣ) είναι αντικείμενο μιας συνεχής εξελικτικής διαδικασίας με σκοπό την προσαρμογή του σε πολλούς παράγοντες όπως η αλλαγή των απαιτήσεων, καινούριες λειτουργίες, συμμόρφωση με νέους κανονισμούς, ενσωμάτωση με άλλα συστήματα καθώς και μεταβολή σε θέματα ασφάλειας και προστασίας προσκοπικών δεδομένων. Η Βάση Δεδομένων (ΒΔ) ενός συστήματος, καθότι αποτελεί τον πυρήνα αυτού, είναι το ποιο κρίσιμο τμήμα όσον αφορά την εξέλιξη.

Στον κύκλο ζωής μιας ΒΔ, το πρόβλημα της εξέλιξη του σχήματός της αρχίζει να γίνεται ορατό από τα πρώτα στάδια της σχεδίασης. Η πολυπλοκότητα των ΒΔ και η συντήρηση λογισμικού είναι ανάλογα με το μέγεθος και την πολυπλοκότητα του συστήματος. Ξεκινώντας από ενδοεταιρικά συστήματα, που συνήθως αναπτύσσονται και συντηρούνται από σχετικά μικρές και σταθερές ομάδες προγραμματιστών, σε συστήματα συντηρούμενα και αναπτυσσόμενα από κοινότητες ή/και μεμονωμένα άτομα, η ανάγκη για μία καλά οργανωμένη εξέλιξη γίνεται αναγκαία.

Στην συνέχεια θα αναλύσουμε το αντικείμενο αυτής της εργασίας καθώς και ορισμένους σχετικούς όρους και έννοιες. Τέλος παρουσιάζεται περιληπτικά το περιεχόμενο των ακόλουθων κεφαλαίων.

1.1 Ορισμός του Προβλήματος

Η διαφορά μεταξύ της εξέλιξης, της τροποποίησης και του ελέγχου εκδόσεων των σχημάτων των ΒΔ είναι συχνά αντικείμενο σύγχυσης καθώς αυτοί οι όροι έχουν χρησιμοποιηθεί με διαφορετικές έννοιες. Για την καλύτερη κατανόηση αυτής της εργασίας δίνονται οι παρακάτω ορισμοί. [JCE+94, RCR94]

Τροποποίηση Σχήματος (Schema Modification) - Τροποποίηση Σχήματος έχουμε όταν ένα σύστημα εφοδιασμένο με μία ή περισσότερες ΒΔ επιτρέπει την αλλαγή του σχήματος αυτών.

Εξέλιξη Σχήματος (Schema Evolution) - Εξέλιξη Σχήματος έχουμε όταν ένα σύστημα εφοδιασμένο με μία ή περισσότερες ΒΔ πραγματοποιεί μια Τροποποίηση Σχήματος χωρίς απώλεια υπάρχουσας πληροφορίας.

Έλεγχος Εκδόσεων Σχήματος (Schema Versioning) - Έλεγχος Εκδόσεων Σχήματος έχουμε όταν ένα σύστημα εφοδιασμένο με μία ή περισσότερες ΒΔ επιτρέπει την πρόσβαση όλων των δεδομένων, τόσο εκ των υστέρων όσο και προοπτικά, μέσω διεπαφών εκδόσεων ορισμένων από το χρήστη.

Κάποια επιμέρους στοιχεία είναι καλό να διευκρινιστούν όσον αφορά του ανωτέρω ορισμούς.

1. Η εξέλιξη του σχήματος δεν υπονοεί πλήρη ιστορική υποστήριξη για το σχήμα· μόνο την αλλαγή της δομής του σχήματος χωρίς απώλεια δεδομένων. Σε αντίθεση ο έλεγχος εκδόσεων του σχήματος, ακόμη και στην ποιο απλή του μορφή, απαιτεί την διατήρηση ενός ιστορικού όσον αφορά τις αλλαγές ώστε να διατηρείται η δομή του σχήματος παλαιότερων αλλαγών.

2. Η σημαντική διαφορά μεταξύ της εξέλιξης και των εκδόσεων είναι η δυνατότητα για τους χρήστες να εντοπίζουν και να διατηρούν σταθερά σημεία κατά την μεταβολή για μελλοντική αναφορά. Η εξέλιξη του σχήματος δεν απαιτεί ειδικό μηχανισμό για την περιήγηση σε παλιούς ορισμούς του σχήματος.

3. Αλλαγές στο σχήμα δεν επιφέρουν πάντα καινούριες εκδόσεις. Συνήθως οι αλλαγές στο σχήμα είναι πολύ πιο μικρές από΄ τη η διαφορά μεταξύ δύο εκδόσεων.

Εμείς θα ασχοληθούμε κυρίως με τους δύο πρώτους ορισμούς, δηλαδή την Τροποποίηση και Εξέλιξη Σχημάτων ΒΔ. Ας δούμε όμως πρώτα ορισμένα θέματα που αφορούν την Εξέλιξη Σχημάτων.

1.1.1 Εξέλιξη Σχημάτων ΒΔ

Τα συστήματα διαχείρισης σχεσιακών Βάσεων Δεδομένων (ΣΔΣΒΔ) είναι αυτή τη στιγμή τα ποιο διαδεδομένα. Τα ΣΔΣΒΔ βασίζονται σε ένα σχεσιακό μοντέλο δεδομένων και μια ΒΔ σε ένα ΣΔΣΒΔ λέγετε Σχεσιακή ΒΔ΄ και περιέχει σχέσεις (πίνακες) και πεδία (γνωρίσματα ή στήλες). Μία λίστα αλλαγών σε ένα σχεσιακό σχήμα είναι η εξής:

1. Προσθήκη καινούριας σχέσης
2. Αλλαγή ονόματος μιας σχέσης
3. Διαγραφή μιας σχέσης
4. Προσθήκη γνωρίσματος σε μία σχέση
5. Αλλαγή ονόματος ενός γνωρίσματος
6. Αλλαγή τύπου ενός γνωρίσματος
7. Διαγραφή

Ένα από τα πιο σημαντικά θέματα στην ανάπτυξη λογισμικού είναι η προσπάθεια επίτευξης σωστών αλλαγών. Αλλαγές σε σχήματα αποτελούν μία σημαντική κατηγορία των παραπάνω αλλαγών. Υπάρχουν πολλές αιτίες αυτών των αλλαγών, για παράδειγμα:

- Δεν υπάρχει γνώση, ή υπάρχει αδυναμία προσδιορισμού εκ των προτέρων όλων των επιθυμητών χαρακτηριστικών και λειτουργιών μιας μεγάλης εφαρμογής. Μόνο έπειτα από εκτεταμένη χρήσης του συστήματος μπορούν να προσδιοριστούν ικανοποιητικά οι ανάγκες και οι απαιτήσεις του.

- Ο ¨κόσμος' των εφαρμογών μεταβάλετε συνεχώς. Επομένως, για να παραμένει ένα σύστημα βιώσιμο, απαιτούνται συνεχείς βελτιώσεις και προσθήκες λειτουργιών ώστε να συμβαδίζει με αυτές τις αλλαγές.

- Συχνά οι αλλαγές γίνονται σε τέτοια κλίμακα που απαιτείται ολικός επανασχεδιασμός του συστήματος. Αυτό φέρει ως αποτέλεσμα την αλλαγή των απαιτήσεων ορισμένων ήδη εγκατεστημένων υποσυστημάτων.

Συνεπώς, συνεχείς αλλαγές στα σχήματα είναι απαραίτητες ώστε να ικανοποιούνται οι απαιτήσεις του συστήματος ανά πάσα στιγμή.

1.1.2 Επιπτώσεις της Εξέλιξης Σχημάτων

Όσον αφορά την αλλαγή, οι ακόλουθες αρχές πρέπει να τηρούνται:

- Η αλλαγή πρέπει να γίνετε με την ελάχιστη δυνατή απώλεια δεδομένων και την ελάχιστη δυνατή επιρροή άλλον συστατικών του συστήματος.Πρέπει να περιορίζετε η διάδοσης περιττών αλλαγών.

- Όλες οι συνέπειες της αλλαγής πρέπει να αντιμετωπιστούν. Πρέπει να εξασφαλιστεί η διάδοση των απαραίτητων αλλαγών.

Ο τρόπος με τον οποίο αντιμετωπίζονται οι αλλαγές στα σχήματα απαιτούν, τις περισσότερες φορές, δύσκολες και άκριβές΄ μετατροπές όπως τερματισμός ή επανεκκίνηση του συστήματος, προγραμματιστικό κόστος, απαιτήσεις σε πόρους συστήματος, κτλ.

Τα αποτελέσματα των αλλαγών στα σχήματα χωρίζονται σε τρεις κατηγορίες.

1. Επιρροές σε άλλα τμήματα του ίδιου σχήματος

2. Επιρροές σε εξωτερικά δεδομένα

3. Επιρροές στον τρόπο λειτουργίας των εφαρμογών που τα χρησιμοποιούν.

Η Εξέλιξη Σχήματος, όπως την ορίσαμε στην ενότητα 1.1 εξασφαλίζει μόνο την ακεραιότητα όσων αφορά τις επιρροές των δύο πρώτον κατηγοριών. Εκεί επικεντρώνεται και το μεγαλύτερο μέρος της βιβλιογραφίας. Ελάχιστη προσοχή έχει δοθεί στην έρευνα που αφορά την τρίτη κατηγορία. Εμείς δεν θα ασχοληθούμε τόσο με τι επιπτώσεις της εξέλιξης αλλά με τον τρόπο με τον οποίο έγινε αυτή η εξέλιξη και ποιες είναι οι άμεσες αλλαγές που έγιναν.

1.2 Αντικείμενο της Πτυχιακής

Παρατηρώντας τη σπουδαιότητα της εξέλιξης των ΒΔ στα ΠΣ, κρίνεται άμεση η ανάγκη δημιουργίας εργαλείων για την διαχείριση των όποιον αλλαγών γίνονται στα σχήματα των ΒΔ. Υπάρχουν ήδη κάποια εργαλεία για τον έλεγχο εκδόσεων των σχημάτων μιας ΒΔ καθώς και ιδιαίτερες γλώσσες για την πραγματοποίηση αλλαγών, με τα οποία θα ασχοληθούμε στο επόμενο κεφάλαιο.

Αντικείμενο αυτής της εργασίας είναι η ανάπτυξη ενός εργαλείου οπτικοποίησης των αλλαγών στο σχεσιακό σχήμα μιας ΒΔ, καθώς και η λήψη ορισμένων μετρικών για την καλύτερη κατανόηση των αλλαγών.

1.3 Οργάνωση του τόμου

Περιγραφή Θέματος Σ΄ αυτό το κεφάλαιο θα δούμε ορισμένα εργαλεία που έχουν υλοποιηθεί για να εξυπηρετήσουν το γενικότερο πρόβλημα σχεδίασης, υλοποίησης και συντήρησης, καθώς και το πρόβλημα εξέλιξης σχημάτων ΒΔ. Ακόμη θα αναφερθούμε σε τεχνολογίες πάνω στις οποίες βασίζετε το εργαλείο που θα υλοποιήσουμε. Τέλος θα περιγράψουμε εκτενώς το στόχο αλλά και τα κενά που προσπαθούμε να καλύψουμε με την ανάπτυξη αυτού του εργαλείου.

Ανάλυση και Σχεδίαση Το κεφάλαιο αυτό αφορά στην ανάλυση της αρχιτεκτονικής του εργαλείου αλλά και στα τεχνικά χαρακτηριστικά που το πλαισιώνουν. Θα αναλύσουμε τα βασικά μέρη της αρχιτεκτονικής του συστήματος αλλά και όλες τις τεχνικές λεπτομέρειες.

Έλεγχος και Μετρήσεις Στο κεφάλαιο αυτό θα επικεντρωθούμε στην περιγραφή του περιβάλλοντος υλοποίησης και ελέγχου του συστήματος. Στην συνέχεια εξετάζουμε τα αποτελέσματα ορισμένων πειραμάτων που εκτελέσαμε βασισμένα στο σχήμα του MediaWiki. Ολοκληρώνοντας θα αναλύσουμε τα συμπεράσματα των πειραμάτων.

Επίλογος Στο τελευταίο αυτό κεφάλαιο κάνουμε μία αναδρομή στα όσα αναφέρθηκαν σε αυτή την εργασία και συζητούμε τις πιθανές μελλοντικές επεκτάσεις του συστήματος.

2 Περιγραφή Θέματος

Στο κεφάλαιο αυτό περιγράφεται το σχετικό υπόβαθρο για τα ζητήματα της ανάλυσης εξέλιξης και οπτικοποίησης των σχημάτων και μεταβολών των σχημάτων βάσεων δεδομένων. Αρχικά αναφέρουμε ερευνητικές εργασίες που άπτονται του θέματος, εστιάζοντας στις πλέον σχετικές με την δικιά μας. Κατόπιν, παρουσιάζουμε τις τεχνολογίες πάνω στις οποίες βασιζόμαστε για την ανάπτυξη και υλοποίηση του συστήματός μας.

2.1 Σχετικές εργασίες

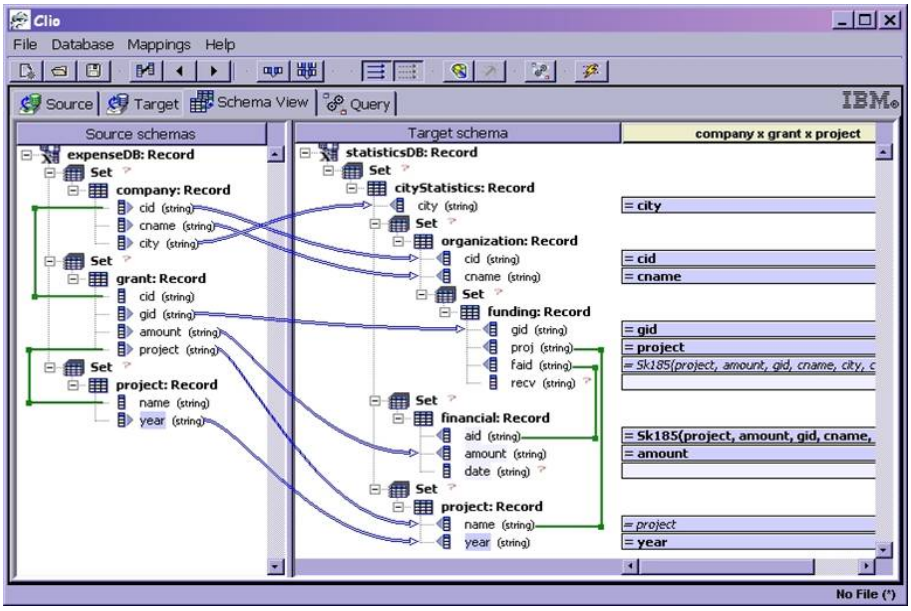
Η εξέλιξη των σχημάτων των σχεσιακών και μη βάσεων δεδομένων υπήρξε αντικείμενο έρευνας από πολλούς στο παρελθόν.[Sjø93, Rod95, ALP91, WZZM05b, WZZM05a, MS90]. Υπάρχει επίσης σχετική δουλειά όσον αφορά την αναγνώριση και αντιμετώπιση των προβλημάτων της εξέλιξης [Rod95]. Στην συνέχεια θα δούμε περιληπτικά τρία συστήματα που αναπτύχθηκαν για την αντιμετώπιση ατών των προβλημάτων, Την Clio [PHV+02, HMHY01], τον Hecateus [PKVV08, PVSV07] και το PRISM [CMTZ08, CMZ08b].

2.1.1 Clio

Η Clio είναι ένα γραφικό ημι-αυτόματο εργαλείο για την αντιστοίχηση και μετατροπή σχημάτων που αναπτύχθηκε από την IBM. Ο χρήστης μπορεί να φορτώσει ένα αρχικό και ένα τελικό σχήμα στο πρόγραμμα και να συνδέσει, με την βοήθεια της διεπαφής, αντικείμενα από το αρχικό στο τελικό σχήμα. Αυτές οι συνδέσεις (*αντιστοιχίες τιμών* όπως αποκαλούνται) εκφράζουν πώς οι τιμές ενός ή περισσοτέρων αντικειμένων του πηγαίου σχήματος συνδυάζονται και μετατρέπονται σε αντικείμενα του παραγόμενου σχήματος. Για παράδειγμα το παραγόμενο γνώρισμα μισθός μπορεί να προσδιοριστεί ως μία συνάρτηση μεταξύ τον τιμών εργατοώρες και ωριαία αντιμισθία. Δεδομένων λοιπών αυτών των συνδέσεων, η Clio μπορεί να παράγει μία πιθανή αντιστοίχηση η οποία εκφράζετε ως όψεις SQL που επιτρέπουν στις εφαρμογές να επεξεργαστούν τα δεδομένα χρησιμοποιώντας μια ενδιάμεση μηχανή ερωτήσεων.

Τα βασικά χαρακτηριστικά του συστήματος αυτού συνοψίζονται στα εξής:

- XML σε XML mapping (XML views on Web data, ...)
- Σχεσιακό σε XML mapping (Web-publishing of legacy data,...)
- XML σε Σχεσιακό mapping (XML shredding into RDBMS, ...)
- Σχεσιακό σε Σχεσιακό (Data warehousing, ...)
- Πλήρης χρήση των πηγαίων και παραγόμενων περιορισμών δεδομένων
- Ανακάλυψη περιορισμών δεδομένων
- Δυναμική επεξεργασία των δεδομένων που εισάγει ο χρήστης



Σχήμα 1: Clio

2.1.2 Hecateus

Ο Hecateus είναι ένα εργαλείο για την αναπαράσταση σχημάτων ΒΔ καθώς και των όψεων και ερωτήσεων που συνοδεύουν αυτά τα σχήματα ως κατευθυνόμενων γράφων. Επίσης δίνει την δυνατότητα στους χρήστες να εισάγουν υποθετικές μεταβολές και να ελέγχουν πιθανές επιρροές σε όλο τον γράφο. Ακόμη παρέχει την δυνατότητα ορισμού κανόνων έτσι ώστε να διατηρείται η συντακτική και σημασιολογική ορθότητα της όλης δομής.

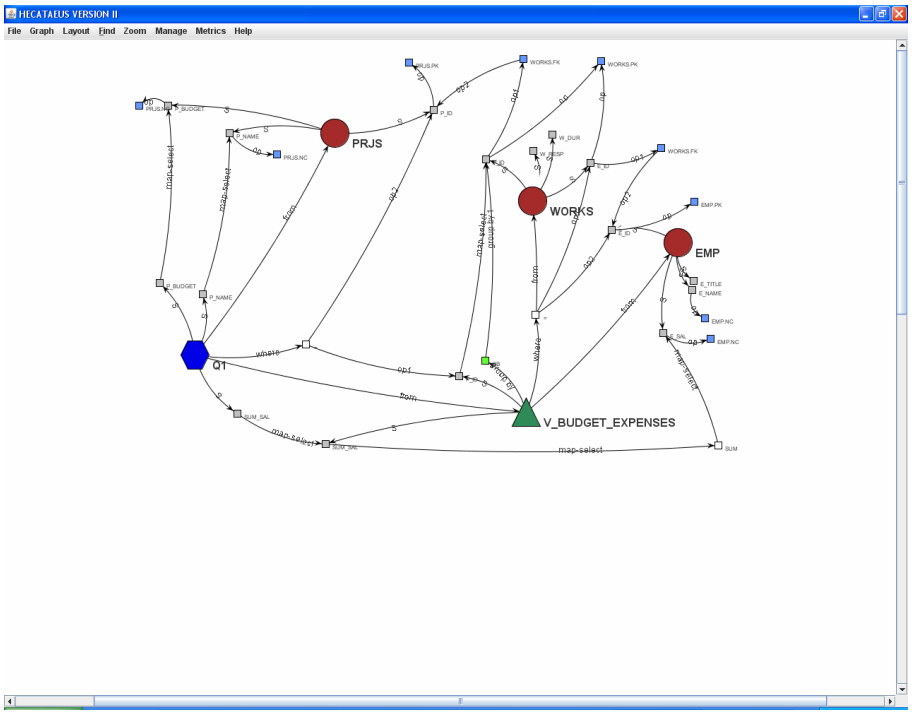
Παρέχει, λοιπόν, ένα μηχανισμό ανάλυσης αλλαγών στις ρυθμίσεις και τα στοιχεία που αποτελούν μια ΒΔ. Βάση αυτής της υλοποίησης αποτελεί ένα μοντέλο γράφων που αναπαριστά συγκεντρωτικά ερωτήσεις, όψεις, σχέσεις και σημαντικά στοιχεία που περιέχονται στις σχέσεις αυτές όπως γνωρίσματα και συνθήκες. Πέραν όμως από την απλή αναπαράσταση των σημασιολογικών στοιχείων μιας ΒΔ. Ακόμη, προσφέρει ένα μηχανισμό προσθήκης κανόνων στον γράφο που ορίζουν την πολιτική που πρέπει να ακολουθηθεί σε περίπτωση αλλαγών. Μπορεί, για παράδειγμα, ένας κόμβος του γράφου οριστεί να μην επιτρέπει διαγραφές στα γνωρίσματά του. Τέλος ο Hecateus διαθέτει την δυνατότης υπολογισμού σημαντικών μετρικών που χαρακτηρίζουν την σημασία ή την ευαισθησία κάθε κόμβου.

2.1.3 PRISM

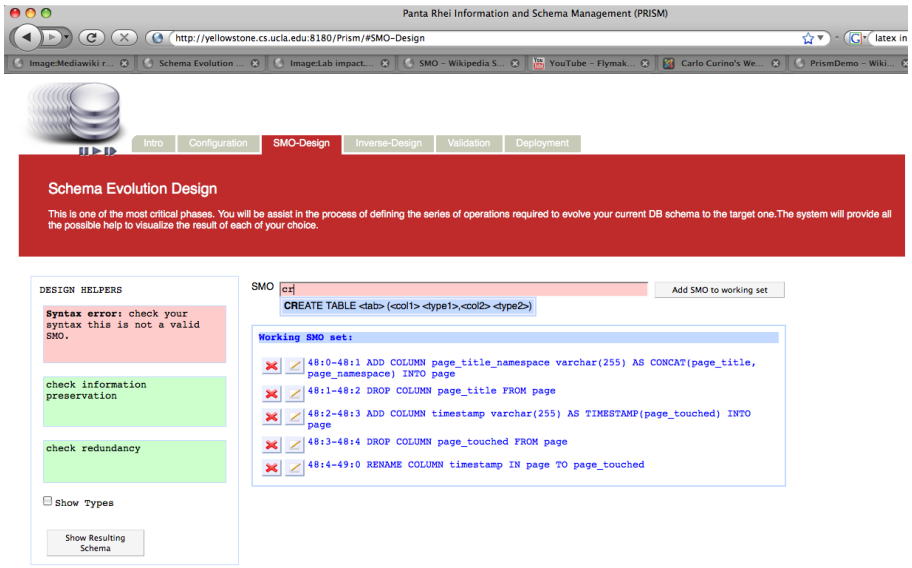
Το PRISM είναι ένα σύστημα που προσπαθεί να αντιμετωπίσει προβλήματα πρόβλεψης και εκτίμησης προτεινόμενων αλλαγών σε σχήματα ΒΔ και ανασύνταξης ερωτήσεων που κάνουν εφαρμογές στη ΒΔ καθώς και ανασύνταξη των ίδιων των εφαρμογών.

Προσφέρει τα εξής:

i Μία γλώσσα αλλαγών (Schema Modification Operations or SMO) για την έκφραση



Σχήμα 2: Hecateus



Σχήμα 3: PRISM Workbench

σύνθετων αλλαγών στα σχήματα ΒΔ.

ii Ένα εργαλείο που προσφέρει στους διαχειριστές ΒΔ να μπορούν γρήγορα να εκτιμήσουν το αποτέλεσμα αυτών τον αλλαγών.

iii Μεταγλώττιση παλιών ερωτημάτων ώστε να δουλεύουν με την καινούρια έκδοση του σχήματος.

iv αυτόματο data migration.

v πλήρης διατήρηση των αντίστροφων αλλαγών για υποστήριξη της ανάποδης διαδικασίας.

Μαζί με το σύστημα PRIMA [MCD+08] και το Historical Metadata Manager (HMM) [CMZ08a] βοήθησαν στην ανάλυση του σχήματος του λογισμικού MediaWiki[4] που αποτελεί την βάση του Pantha Rei Schema Evolution Benchmark [CMTZ08]. Πάνω από 170 εκδώσεις σχημάτων, μετά από 4.5 χρόνια ανάπτυξης, συγκρίθηκαν και μελετήθηκαν.

2.2 Στόχος

Τα ανωτέρω συστήματα, όπως αναφέραμε και πριν προσπαθούν να λύσουν προβλήματα που αφορούν στη εξέλιξη των σχημάτων ΒΔ. Κανένα όμως από αυτά δεν οπτικοποιεί τις άμεσες και εκ των υστέρων αλλαγές. Η Clio δημιουργεί ναι διαχειρίζεστε αλλαγές. Ο Hecateus οπτικοποιεί σχήματα και εξετάζει τις επιρροές της αλλαγής. Τέλος, το PRISM εξασφαλίζει ομοιόμορφη μεταβολή. Τι συμβαίνει όμως με την αλλαγή αυτή καθεαυτή.

Σκοπός μας λοιπόν είναι η δημιουργία ενός εργαλείου που δεδομένων δύο διαφορετικών εκδόσεων της ίδιας βάσης δεδομένων θα εντοπίζει, θα οπτικοποιεί και θα μετρά την αλλαγή που υπέστη το σχήμα. Έτσι ο διαχειριστής της βάσης θα έχει την δυνατότητα να βλέπει με έναν εύκολο και γρήγορο τρόπο τι αλλαγές έχουν γίνει μέσω μίας απλής και όμορφης διεπαφής.

2.3 Σχετικές Τεχνολογίες

Σ′ αυτή την ενότητα θα αναλύσουμε δύο βασικές τεχνολογίες πάνω στις οποίες θα βασιστούμε για την υλοποίηση του συστήματος μας. Η πρώτη αφορά στην γλώσσα ορισμού της δομής του σχήματος μιας σχεσιακής βάσης δεδομένων και η δεύτερη περιγράφει έναν αλγόριθμο για την ανίχνευση διαφορών στις διαφορετικές εκδόσεις την ΒΔ.

2.3.1 Data Definition Language

Η DDL είναι μία γλώσσα για τον ορισμό της δομής των δεδομένων. Αρχικά αναφερόταν σε ένα υποσύνολο της SQL αλλά πλέον χρησιμοποιείται γενικότερα για κάθε μορφή γλώσσας που περιγράφει δεδομένα ή πληροφοριακές δομές, όπως τα XML σχήματα. Υπάρχουν τέσσερις βασικές κατηγορίες εντολών που αποτελούν την DDL.

[4]Available at: http://www.mediawiki.org

CREATE Χρησιμοποιείται για την δημιουργία νέας βάσης, σχέσης, ευρετηρίου ή όψης. Δημιουργεί ένα αντικείμενο σε ένα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων (ΣΣΔ-ΒΔ). Οι τύποι των αντικειμένων που μπορούν να δημιουργηθούν εξαρτάται από το ποιο ΣΣΔΒΔ χρησιμοποιείται. Τα περισσότερα υποστηρίζουν την δημιουργία σχέσεων, ευρετηρίων χρηστών και βάσεων. Κάποια ΣΣΔΒΔ (όπως πχ. PostgreSQL) επιτρέπουν εντολές δημιουργίας μέσα σε δοσοληψίες οπότε σε περίπτωση αποτυχίας της δοσοληψίας επαναφέρουν την βάση στην αρχική της μορφή.

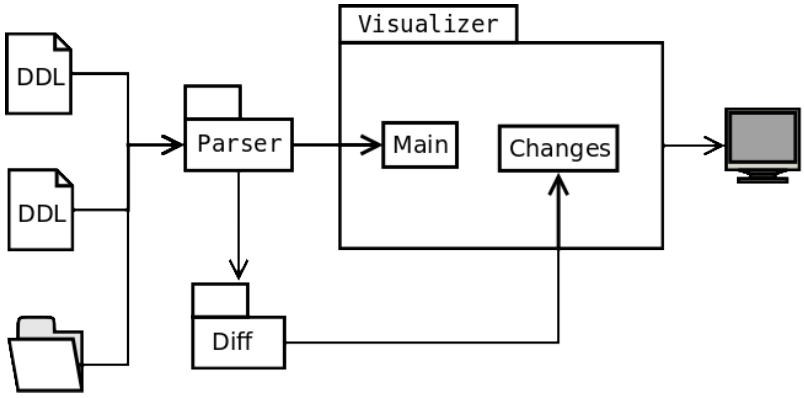
DROP Η αντίθετη λειτουργία των εντολών δημιουργίας. Οι εντολές αυτές διαγράφουν αντικείμενα από τα ΣΣΔΒΔ. Ομοίως τα ήδη των αντικειμένων εξαρτώνται από το σύστημα και σε μερικά συστήματα μπορούν να αναιρεθούν κατά την αποτυχία δοσοληψιών.

ALTER Χρησιμοποιείται για την αλλαγή των χαρακτηριστικών ενός αντικειμένου μέσα σε ένα ΣΔΔΒΔ Και πάλι τα είδη των αντικειμένων εξαρτώνται από τη ΣΣΔΒΔ.

Τέλος, άλλα είδη εντολών ασχολούνται με τον ορισμό αναφορικών σχέσεων ακεραιότητας, οι οποίες συνήθως υλοποιούνται υπό την μορφή πρωτεύον και ξένων κλειδιών σε κάποια γνωρίσματα μίας σχέσης. Αυτές οι εντολές μπορούν να συμπεριληφθούν σε εντολές δημιουργίας ή μεταβολής.

2.3.2 Sort Merge Join

Ο Sort-Merge Join αλγόριθμος (γνωστός και ως Merge-Join) είναι ένα παράδειγμα αλγορίθμου συνένωσης και χρησιμοποιείται στην υλοποίηση ΣΣΔΒΔ. Το βασικό πρόβλημα του αλγορίθμου είναι η εύρεση, για κάθε διακριτή τιμή του προς συνένωση γνωρίσματος, μία ομάδα από πλειάδες σε κάθε σχέση που έχουν αυτή την τιμή. Η βασική ιδέα του αλγορίθμου είναι να ταξινομήσει πρώτα τις σχέσεις ως το προς συνένωση γνώρισμα έτσι ώστε η γραμμική αναζήτηση στις δύο σχέσεις να εντοπίζει τα κοινά στοιχεία μαζί. Στην πράξη το ποιο χρονοβόρο και άκριβό΄ κομμάτι είναι αυτό της ταξινόμησης των δύο εισόδων. Αυτό επιτυγχάνετε συνήθως με την χρήση εξωτερικής ταξινόμησης ή εκμεταλλεύεται μία ήδη υπάρχουσα ταξινόμηση σε μία ή και τις δυο σχέσεις.



Σχήμα 4: Η αρχιτεκτονική του συστήματος

3 Ανάλυση του συστήματος

Σε αυτό το κεφάλαιο θα ασχοληθούμε με την ανάλυση της Εκάτης, του συστήματος που αναπτύξαμε στα πλαίσια αυτής την πτυχιακής εργασίας. Θα περιγράψουμε τα συστατικά της αρχιτεκτονικής του συστήματος καθώς και τον τρόπο αλληλεπίδρασης και εξάρτησης αυτών. Στην συνέχεια Θα εξετάσουμε αναλυτικά τα API που υλοποιήσαμε σε κάθε υποσύστημα της αρχιτεκτονικής.

3.1 Hecate: an SQL Schema Diff Viewer

Στα πλαίσια λοιπόν της εξέλιξης των σχημάτων ΒΔ που αναφέραμε στο πρώτο κεφάλαιο, παρουσιάζουμε την Εκάτη. Ένα εργαλείο για την οπτικοποίηση των αλλαγών μεταξύ δύο εκδόσεων ενός σχήματος. Ο χρήστης ορίζει τα αρχεία που περιέχουν του ορισμούς του σχήματος σε DDL. Στην συνέχεια το σύστημα επεξεργάζεται τα εισαγόμενα δεδομένα και αναπαριστά τις δύο εκδώσεις του σχήματος με δύο δέντρα επισημαίνοντας τα στοιχεία που μεταβλήθηκαν με ανάλογο χρώμα. Ο χρήστης έχει επίσης την δυνατότητα να δει ορισμένες μετρικές όσον αφορά το μέγεθος της αλλαγής. Τέλος η Εκάτη διαθέτει την δυνατότητα να επεξεργάζεται επαναληπτικά περισσότερες από δύο εκδώσεις προσφέροντας ορισμένες μετρικές για την πορεία των αλλαγών σε όλες τις εκδώσεις.

Η αρχιτεκτονική του συστήματος της Εκάτης αποτελείται από τρία βασικά συστατικά: τον *Parser*, τον αλγόριθμο του *Diff* και τον *Visualiser*.

- Ο Parser διατρέχει τα DDL αρχεία που δίνονται σαν είσοδος αναλύοντας κάθε εντολή. Στην συνέχεια αντιστοιχεί τα στοιχεία του σχήματος της βάσης που ορίζουν αυτά τα αρχεία σε μία συγκεκριμένη δομή κλάσεων που υλοποιήσαμε και θα αναλύσουμε παρακάτω.

- Ο Diff αλγόριθμος είναι υπεύθυνος για την επεξεργασία των αντικειμένων που δημιούργησε ο Parser και την ανίχνευση αλλαγών μεταξύ δύο σχημάτων. Αποτελεί μία διασκευασμένη μορφή του Sort-Merge Join αλγορίθμου που είδαμε στην ενότητα

```
CREATE TABLE employees (
        id              INTEGER     NOT NULL,
        first_name      CHAR(50)    NULL,
        last_name       CHAR(75)    NOT NULL,
        dateofbirth     DATE        NULL
       PRIMARY KEY(id)
);
```

Σχήμα 5: Η γλώσσα DDL

2.3.2. Οι αλλαγές μαρκάρονται στα αντικείμενα τα οποία στην συνέχει θα σχεδιαστούν στο κεντρικό παράθυρο.

- Τέλος, ο Visualizer αναλαμβάνει να σχεδιάσει τα μέρη του σχήματος σε μορφή δέντρου. Οι κόμβοι και τα φύλα του δέντρου που έχουν εντοπιστεί να φέρουν αλλαγές ή εισήχθησαν/αφαιρέθηκαν μεταξύ των εκδόσεων χρωματίζονται με αντίστοιχα χρώματα.

Στην συνέχεια αυτού του κεφαλαίου θα αναλύσουμε τα βασικά μέρη της αρχιτεκτονικής που μόλις αναφέραμε. Να πούμε επίσης σε αυτό το σημείο ότι το σύστημα υλοποιήθηκε σε γλώσσα προγραμματισμού Java για τις έτοιμες και εύκολα προσβάσιμες δομές που προσφέρει.

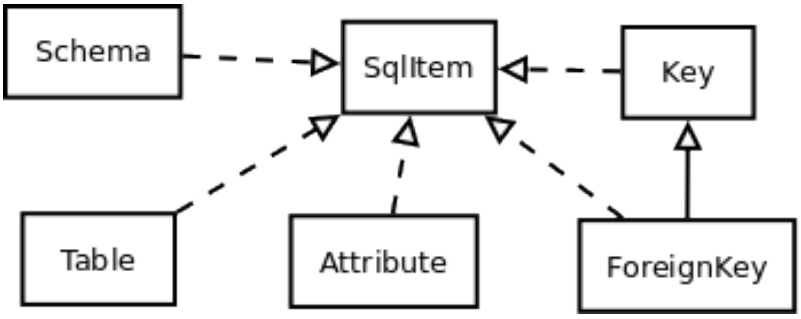
3.1.1 Parsing/Mapping

Αυτό το τμήμα του συστήματος αναλαμβάνει την λεκτική και συντακτική ανάλυση των αρχείων DDL και την δημιουργία κατάλληλων αντικειμένων από αυτά. Στα πλαίσια της ανάλυσης αυτής δημιουργήσαμε μία απλή DDL γλώσσα. Μπορείτε να ανατρέξετε στο παράρτημα για τα χαρακτηριστικά και τον ορισμό αυτής της γλωσσάς. Υπάρχουν αρκετές έτοιμες υλοποιήσεις που καλύπτουν όμως όλη την SQL και ήταν αρκετά σύνθετες. Θεωρήσαμε περιττή την χρήση τόσο σύνθετων γλωσσών και γι αυτό υλοποιήσαμε την δική μας.

Η γλώσσα αυτή υποστηρίζει μόνο CREATE εντολές για σχήματα σχέσεις και ευρετήρια βάσεων δεδομένων. Ακόμη υποστηρίζονται κλειδιά και ξένα κλειδιά για γνωρίσματα σχέσεων καθώς και παράμετροι για τις σχέσεις την βάσης που ορίζουν κάποια λειτουργικότητα στο σύστημα διαχείρισης ΒΔ. Υποστηρίζονται επίσης σχόλια σε δύο μορφές. Χρησιμοποιώντας // και – στην αρχή μίας γραμμής για τον σχολιασμό σε όλη την γραμμή και /* για τη έναρξη και */ για τον τερματισμό ενός σχολίου. Ένα παράδειγμα φαίνεται στο Σχήμα 5. Ολόκληρη η γραμματική της γλώσσας είναι διαθέσιμη σε BNF μορφή στο παράρτημα Α′

Η προγραμματιστική υλοποίηση αυτής της γλώσσας έγινε με την χρήση του ANTLR (ANother Tool for Language Recognition)[5]. Το ANTLR είναι ένα εργαλείο που παρέχει ένα περιβάλλον για την κατασκευή μεταφραστών και μεταγλωττιστών από μία γραμματική

[5] Διαθέσιμο εδώ: http://www.antlr.org



Σχήμα 6: SQL κλάσεις

περιγραφή σε μία ποικιλία διαφορετικών γλωσσών προγραμματισμού. Γράφοντας την γραμματική της γλώσσας σε μια μορφή που ορίζει το ANTLR, παράγεται, στην επιθυμητή γλώσσα προγραμματισμού, ένα πρόγραμμα που μπορεί να ελέγξει αν ένα κομμάτι κώδικα ανήκει σε αυτή την γλώσσα. Ακόμη μας δίνει την δυνατότητα να εισάγουμε κώδικα ανάμεσα στους κανόνες της γραμματικής και να επεξεργαστούμε ορισμένα δεδομένα καθώς το παραγόμενο πρόγραμμα του ANTLR διατρέχει τα αρχεία της γλώσσας.

Μεταφέροντας λοιπόν την γραμματική της DLL γλώσσας που κατασκευάσαμε σε μορφή κατανοητή από το ANTLR καταφέραμε να δημιουργήσουμε τον parser του συστήματός μας. Εισάγοντας και τον κατάλληλο κώδικα στα μέρη της γραμματικής κατασκευάζουμε μία ομάδα αντικειμένων από μία σειρά κλάσεων που αντιπροσωπεύουν την δομή του σχήματος της βάσης που θέλουμε να επεξεργαστούμε. Μία γενική μορφή αυτών τον κλάσεων φτιάνεται στο Σχήμα 6. Αναλυτικά διαγράμματα για τα πεδία και τις μεθόδους που υλοποιούνται σε κάθε κλάση μπορείτε να βρείτε στο παράρτημα Β΄

Η διεπαφή SqlItem ορίζει κάποιες γενικές μεθόδους για όλα τα αντικείμενα που αποτελούν στοιχεία του σχήματος και υλοποιούν αυτή την διεπαφή. Σημαντικότερες από αυτές είναι οι getMode και setMode. Όλα τα SQL αντικείμενα διαθέτουν ένα πεδίο που αναφέρει την κατάσταση, όσον αφορά την αλλαγή, στην οποία βρίσκονται. Στην συνέχεια, όταν μιλήσουμε για τον αλγόριθμο diff, θα εξετάσουμε τις πιθανές τιμές αυτών των πεδίων. Αυτές οι δύο μέθοδοι διαχειρίζονται αυτές τις τιμές.

Αντικείμενα της κλάσης Attribute αντιστοιχίζονται σε γνωρίσματα των σχέσεων του σχήματος. Διαθέτουν πεδία για τον ορισμό του ονόματος, τύπου, προ επιλεγμένης τιμής, αν αποτελούν ή όχι κλειδί της σχέσης στην οποία ανήκουν, αν επιτρέπεται να πάρουν null τιμές και ένα πεδίο για τον ορισμό της κατάστασης που αναφέραμε προηγουμένως. Πέραν από τις μεθόδους της διεπαφής SqlItem που υλοποιούν διαθέτουν getters και setters για την διαχείρηση των τιμών τους.

Η κλάση Key ορίζει την δομή των αντικειμένων που αντιστοιχίζονται στα κλειδιά των σχέσεων του σχήματος. Διαθέτουν μία ταξινομημένη δομή (TreeMap), που περιέχει αντικείμενα τύπου Attribute τα οποία αποτελούν το κλειδί. Την σημαντικότητα της χρήσης ταξινομημένων δομών για την αποθήκευση τιμών θα την εξετάσουμε στο επόμενη ενότητα. Η κλάση ForeignKey κληρονομεί όλα τα χαρακτηριστικά της κλάσης

Key και αντιστοιχίζεται στα ξένα κλειδιά της σχέσης. Διαθέτει μία επιπλέον τιμή που περιέχει την σχέση στην οποία είναι ξένο κλειδί το κλειδί αυτό.

Η κλάση Table αντιπροσωπεύει τα σχήματα της σχέσης. Τα σημαντικότερα πεδία αυτής της κλάσης είναι τα arrts, pKey, και fKey. Το πεδίο attrs είναι μία ταξινομημένη δομή που περιέχει τα γνωρίσματα της σχέσης, δηλαδή αντικείμενα τύπου Attribute. Τα πεδία pKey και fKey είναι τα πρωτεύοντα και ξένα κλειδιά της σχέσης. Και πάλι υλοποιεί μεθόδους της κλάσης SqlItem καθώς και setters και getters για την διαχείριση των πεδίων.

Τέλος, η κλάση Schema είναι η αντιστοίχηση του όλου σχήματος. Διαθέτει ένα TreeMap με αντικείμενα τύπου Table. Ένα αντικείμενο αυτής της κλάσης κατασκευάζεστε για κάθε σχήμα. Ομοίως με τις προηγούμενες κλάσεις, υλοποιούνται και οι μέθοδοι.

Κατά την διαδικασία του parsing, λοιπόν, κατασκευάζονται αντικείμενα των ανωτέρω κλάσεων τα οποία στην συνέχεια περνούν στο επόμενο βασικό τμήμα του συστήματος, τον αλγόριθμο diff.

3.1.2 Diff

Σε αυτή την φάση έχουν κατασκευαστεί τα αντικείμενα που αναπαριστούν το σχήμα της ΒΔ και επεξεργάζονται για τυχών αλλαγές. Όπως αναφέραμε και προηγουμένως ο βασικός αλγόριθμος που χρησιμοποιούμε αποτελεί μία παραλλαγή του του Sort-Merge Join, όπως φαίνεται στο Σχήμα 7. Η σωστή λειτουργία του αλγορίθμου αυτού απαιτεί τα δεδομένα να είναι ταξινομημένα. Για αυτό τον λόγο τα αντικείμενα στα οποία αντιστοιχίζονται τα στοιχεία του σχήματος βρίσκονται σε TreeMap δομές. Οι δομές αυτές αποτελούν ένα είδος κόκκινο-μαύρων δέντρων. Κατά τον ορισμό τους απαιτούν τον τύπο των αντικειμένων ως προς αποθήκευση στη δομή (στην περίπτωσή μας Attributes και Tables) και τον τύπο των κλειδιών πάνω στα οποία θα ταξινομηθούν τα αντικείμενα. Για τις τιμές των κλειδιών χρησιμοποιήθηκαν τα ονόματα των αντικειμένων. Έτσι λοιπόν η ταξινόμηση και ο έλεγχος αλλαγών γίνετε με βάση αυτά τα ονόματα.

Ο αλγόριθμος διατρέχει τα αντικείμενα και τα μαρκάρει ανάλογα με την κατάσταση την οποία έχουν:

'm' ή matched για τα στοιχεία τα οποία βρέθηκαν ότι είναι αμετάβλητα.

'i' ή inserted για στοιχεία τα οποία εισήχθησαν. Αυτός ο χαρακτηρισμός υπάρχει, προφανώς, μόνο για στοιχεία της μεταγενέστερης έκδοσης του σχήματος.

'd' ή deleted για στοιχεία τα οποία διαγράφηκαν. Ομοίως μόνο για στοιχεία του προγενέστερου σχήματος.

'u' ή updated για στοιχεία που κάποιο περιεχόμενο στοιχείο ή πεδίο τους μεταβλήθηκε. Για παράδειγμα ένας πίνακας που έκανε match προηγουμένως εμφανίζετε να έχει ένα καινούριο γνώρισμα ή ο τύπος ενός γνωρίσματος μεταβλήθηκε ή το κλειδί μιας σχέσης άλλαξε.

```
var left, right;
left = left.getNext();
right = right.getNext();
while (left != null && right != null) {
        if (left.key == right.key) {
                // match
                left = left.getNext();
                right = right.getNext();
        }
        else if (left.key > right.key) {
                // insertion
                right = right.getNext();
        }
        else if (left.key < right.key) {
                // deletion
                left = left.getNext();
        }
}
```

Σχήμα 7: Ο αλγόριθμος Diff

Ξεκινώντας με έναν δείκτη στην αρχή κάθε TreeMap κάθε σχήματος, που περιέχει τις σχέσεις, ελέγχουμε την ομοιότητα των κλειδιών. Αν τα κλειδιά βρεθούν ίδια τότε οι δυο πίνακες σημειώνονται ως όμοιοι. Σε περίπτωση που το πρώτο κλειδί είναι μεγαλύτερο λεξικογραφικά του δεύτερου τότε αυτό σημαίνει ότι στο στοιχείο του δεύτερου δεν υπάρχει στο πρώτο και άρα έχουμε εισαγωγή. Αντίστοιχα, αν το κλειδί του πρώτου είναι μικρότερο του δεύτερου τότε έχουμε διαγραφή αφού στο στοιχείο του πρώτου δεν υπάρχει πλέον στο δεύτερο. Στις περιπτώσεις της εισαγωγή και διαγραφής σημειώνονται και όλα τα γνωρίσματα της σχέσης με ανάλογο τρόπο. Στην πρώτη περίπτωση όμως, πρέπει να ελέγξουμε και τα γνωρίσματα της σχέσης. Με παρόμοιο τρόπο κοιτάμε ποιο κλειδί είναι λεξικογραφικά μεγαλύτερο ή μικρότερο. Σε περίπτωση μεταβολής μαρκάρουμε αντίστοιχα το γνώρισμα, άλλα και τις σχέσεις που το περιέχουν ως updated. Ακόμη, στο ενδεχόμενο ίδιων κλειδιών για τα γνωρίσματα ελέγχουμε τον τύπο γνωρίσματος και αν ή όχι είναι κλειδί της σχέσης επισημαίνοντας την μεταβολή στο γνώρισμα και την σχέση που το περιέχει.

3.1.3 Visualisation

Μετά από τον έλεγχο των αντικειμένων για αλλαγές τα δεδομένα παρουσιάζονται στο κεντρικό παράθυρο του συστήματος. Στην αριστερή πλευρά του παραθύρου έχουμε μία πρώτη έκδοση του σχήματος και δεξιά μία επόμενη έκδοση. Τα στοιχεία του σχήματος παρουσιάζονται ως κόμβοι και φύλα δύο δέντρων, ένα για κάθε έκδοση του σχήματος. Σε πρώτη φάση δίνονται οι σχέσεις κάθε σχήματος συμπτυγμένες. Αναπτύσσοντας μία σχέση βλέπουμε τα γνωρίσματα αυτής της σχέσης, καθώς και ορισμένα χαρακτηριστικά αυτών, με υπο-

γράμμιση στα γνωρίσματα που αποτελούν κλειδί της σχέσης. Τρία χρώματα είναι πιθανόν να συναντήσουμε σε μία απεικόνιση:

Κόκκινο Αυτό το χρώμα επισημαίνει τη διαγραφή κάποιου στοιχείου. Εμφανίζεται μόνο στο αριστερό δέντρο.

Πράσινο Αυτό το χρώμα επισημαίνει την εισαγωγή κάποιου στοιχείου. Εμφανίζεται μόνο στο δεξί δέντρο.

Κίτρινο Αυτό το χρώμα επισημαίνει τη μεταβολή κάποιου στοιχείου. Αυτό μπορεί να σημαίνει την μεταβολή του τύπου ενός γνωρίσματος, την μεταβολή του κλειδιού μιας σχέσης ή την εισαγωγή ή διαγραφή ενός γνωρίσματος από μία σχέση. Εμφανίζεται και στα δύο δέντρα.

3.2 Τεχνικές Λεπτομέρειες

Όπως αναφέραμε και προηγουμένως, το όλο σύστημα υλοποιήθηκε σε Java[6] και η παραγωγή του εκτελέσιμου κώδικα έγινε με την έκδοση Java(TM) SE Runtime Environment (build 1.6.0_20-b02) Η έκδοση του ANTLR που χρησιμοποιήθηκε για την δημιουργία του parser ήταν η 3.2. Πέραν των τυπικών πακέτων που προσφέρει το προγραμματιστικό περιβάλλον της Java χρησιμοποιήθηκε το πακέτο org-netbeans-swing-outline από το NetBeans[7] για την υλοποίηση του δέντρου-πίνακα της εφαρμογής.

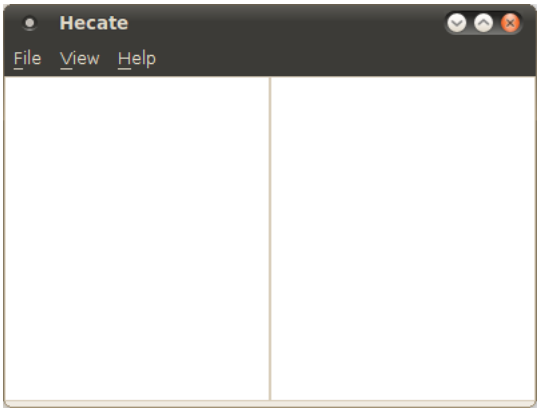
Το τελικό εκτελέσιμο παράχθηκε και έτρεξε τις μετρήσεις που θα δούμε στο επόμενο κεφάλαιο σε ένα GNU/Linux φορητό 64-bit σύστημα με επεξεργαστή συχνότητας 1.8Ghz εφοδιασμένο με 2GB μνήμη RAM. Το σύστημα είναι πλήρως ανεξάρτητο πλατφόρμας και ελέγχθηκε η λειτουργία του σε 32-bit και 64-bit GNU/Linux και Windows συστήματα.

3.3 Τυπική Χρήση

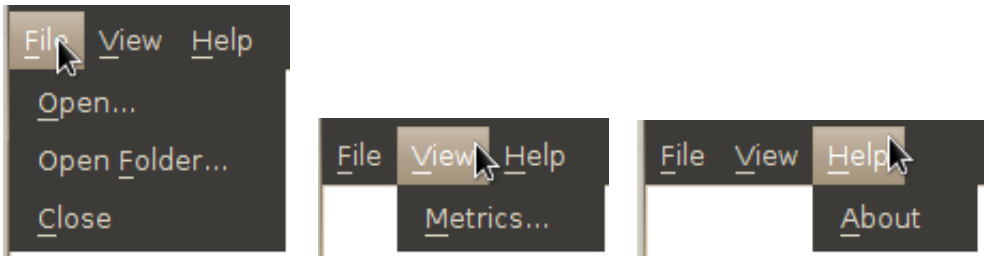
Τρέχοντας για πρώτη φορά την Εκάτη βρισκόμαστε απέναντι στο κεντρικό παράθυρο της εφαρμογής όπως φαίνεται στο σχήμα 8. Το παράθυρο αυτό χωρίζετε σε δύο μέρη, ένα για κάθε έκδοση. Η αναπαράσταση του αρχικού σχήματος γίνετε στο αριστερό μέρος του παραθύρου ενώ η αναπαράσταση του τελικού στο δεξί.

Στην μπάρα του μενού διακινούμενε τρία μενού όπως φαίνονται στο σχήμα 9. Το μενού File (σχήμα 9(α΄)) μας δίνει την δυνατότητα ανοίγματος δύο αρχείων για την προβολή των διαφορών τους, άνοιγμα ενός φακέλου που περιέχει διαφορετικές εκδώσεις του ίδιου σχήματος ταξινομημένες χρονολογικά από την παλαιότερη στην νεότερη και την επιλογή τερματισμού. Πατώντας Open File ... βλέπομε το παράθυρο του σχήματος 10 στο οποίο μπορούμε να δώσουμε στα πεδία την όλη την διεύθυνση των αρχείων προς επεξεργασία. Μπορούμε εναλλακτικά να πατήσουμε τα κουμπιά με τις τρεις τελείς δεξιά για την επιλογή αρχείου από ένα ειδικό παράθυρο που μας επιτρέπει την περιήγηση στο σύστημα αρχείων. Παρόμοια εμφάνιση έχει και το παράθυρο για την επιλογή φακέλου πατώντας το μενού Open Folder ... . Το μενού Close τερματίζει την Εκάτη.

[6] http://java.sun.com/javase/downloads/index.jsp
[7] http://netbeans.org/

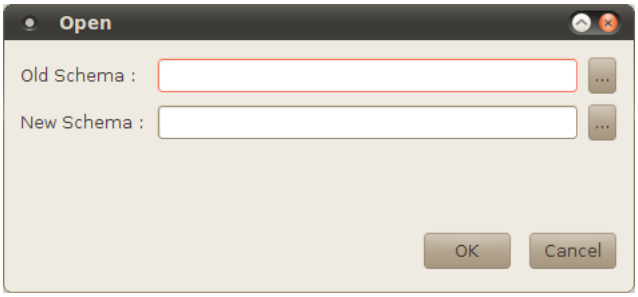


Σχήμα 8: Κεντρικό Παράθυρο



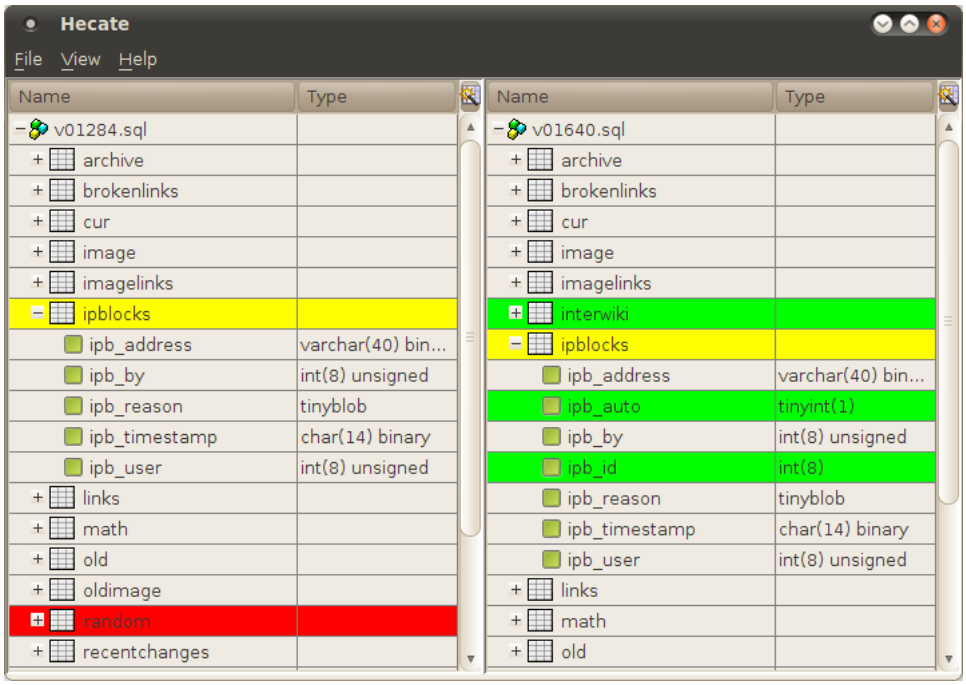
(α′) File (β′) View (γ′) Help

Σχήμα 9: Μενού



Σχήμα 10: Επιλογή αρχείων

Φορτώνοντας, λοιπόν δύο αρχεία η Εκάτη επεξεργάζεται τα δεδομένα και οπτικοποιεί τα δύο σχήματα. Ένα παράδειγμα φαίνεται στο σχήμα 11. Πατώντας τα + αριστερά των σχέσεων βλέπουμε τα γνωρίσματα της κάθε σχέσης. Έχοντας δύο σχήματα στο παράθυρο, μπορούμε να επιλέξουμε το μενού Metrics και η Εκάτη θα μας δώσει σε ένα καινούριο παράθυρο ορισμένες μετρικές που αφορούν στις αλλαγές των δύο σχημάτων.



Σχήμα 11: Η Εκάτη επί το έργον

Τέλος το μενού About δημιουργεί ένα παράθυρο με ορισμένα στοιχεία για την Εκάτη και τον δημιουργό της.

4 Έλεγχος και Μετρήσεις

Σε αυτό το κεφάλαιο θα χρησιμοποιήσουμε την Εκάτη για να μελετήσουμε τις διαφορετικές εκδώσεις του σχήματος της βάσης της γνωστής σε όλους μας Βικιπαίδειας. Η Βικιπαίδεια αποτελεί ένα χαρακτηριστικό παράδειγμα μίας ταχύτατα αναπτυσσόμενης διαδικτυακής εφαρμογής. Η πλατφόρμα στην οποία στηρίζετε η Βικιπαίδεια είναι το MediaWiki, ένα ανοιχτού κώδικα εργαλείο που αρχικά δημιουργήθηκε σαν ο πυρήνας της Βικιπαίδειας αλλά πλέον χρησιμοποιείται από περισσότερα από 30000 wiki[8].

4.1 MediaWiki

Το MediaWiki είναι ένα ελεύθερο server-based λογισμικό υπό την Γενική Άδεια Δημόσιας Χρήσης GNU (GNU General Public License (GPL))[9]. Είναι σχεδιασμένο να λειτουργεί σε μεγάλες φάρμες από εξυπηρετητές για ιστοσελίδες που δέχονται εκατομμύρια επισκέψεις ημερησίως. Είναι ένα πολύ ισχυρό λογισμικό με πληθώρα χαρακτηριστικών που χρησιμοποιεί PHP για την επεξεργασία και παρουσίαση των δεδομένων που αποθυκεύονται σε μία MySQL βάση δεδομένων.

4.1.1 Δομή Σχήματος του MediaWiki

Η ΒΔ, στην τελευταία έκδοση από αυτές που μελετήσαμε[10] αποτελείται από 34 σχέσεις με συνολικά 242 γνωρίσματα. Σ′ αυτή την ΒΔ αποθηκεύεται ολόκληρο το περιεχόμενο της σελίδας, περίπου 700GB για την Wikipedia. Οι σχέσεις αυτές μπορούν να ομαδοποιηθούν στις εξής κατηγορίες.

Άρθρα και διαχείριση περιεχομένου (6): `page`, `revision`, `text`, `image`, `user_netwalk`, `math`

Ιστορικό και διαχείριση αρχείου (4): `archive`, `filearchive`, `oldimages`, `looging`

Σύνδεσμοι και δομή ιστοσελίδας (9): `categorylinks`, `externallinks`, `imagelinks`, `interwiki`, `langlinks`, `pagelinks`, `redirect`, `templatelinks`, `trackbacks`

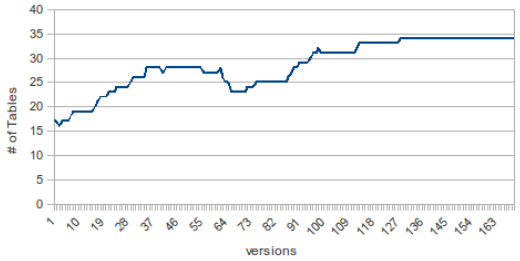
Χρήστες και δικαιώματα (5): `user`, `user_group`, `ipblocks`, `watchlist`, `page_restrictions`

Απόδοση και κρυφή μνήμη (7): `objectivecache`, `querycache`, `querycache_info`, `job`, `querycachetwo`, `transcache`, `searchindex`

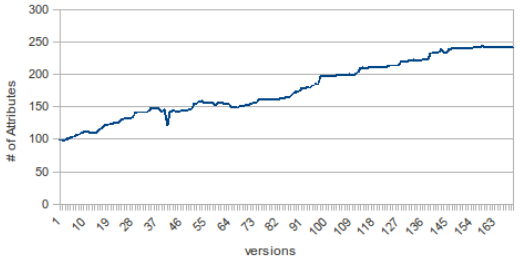
[8] http://s23.org/wikistats

[9] http://el.wikipedia.org/wiki/GNU_General_Public_License

[10] Όλο το dataset που χρησιμοποιήσαμε είναι διαθέσιμο εδώ: http://yellowstone.cs.ucla.edu/schema-evolution/documents/mediawiki-schema.tar.gz



(α′) Σχέσεις ανά έκδοση



(β′) Γνωρίσματα ανά έκδοση

Σχήμα 12: Μεγέθη εκδόσεων

Στατιστικά και ειδικά χαρακτηριστικά υποστήριξης (3): `recentchanges`, `hitcounter`, `site_stats`

4.2 Συμπεράσματα

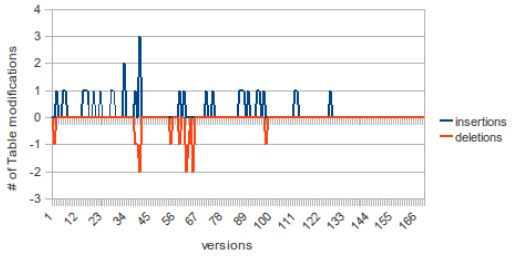
Χρησιμοποιήσαμε λοιπόν την Εκάτη για την μέτρηση των αλλαγών του σχήματος της βάσης του WikiMedia. Τοποθετήσαμε τα αρχεία με τις διαφορετικές εκδώσεις σε ένα φάκελο δίνοντας τους ονόματα κατά αύξουσα σειρά αντίστοιχη της χρονολογικής τους σειράς. Η Εκάτη παρήγαγε ορισμένα αποτελέσματα τα οποία θα συζητήσουμε στις ακόλουθες ενότητες.

4.2.1 Συνολικά μεγέθη εκδόσεων

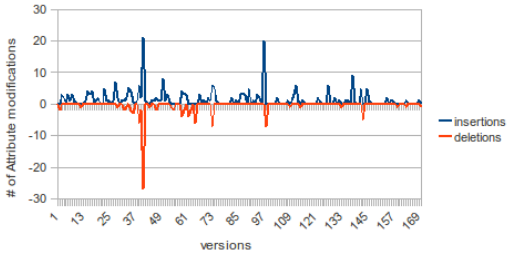
Στα σχήματα 12(α′) και 12(β′) βλέπουμε το μέγεθος του σχήματος της ΒΔ όσον αφορά τις σχέσεις και τα γνωρίσματα αντίστοιχα. Παρατηρούμε μια γενική αύξηση του μεγέθους του σχήματος με τις σχέσεις από 17 να γίνονται 34 (100% αύξηση) και τα γνωρίσματα από 100 να αυξάνονται σε 242 (σημειώνοντας 142% αύξηση). Παρατηρούμε διαφορετικούς ρυθμούς αύξησης που σχετίζονται, πιθανότατα, με τις περιόδους που προηγούνται ή ακολουθούν τα επίσημα releases του όλου συστήματος. Η αύξηση αυτή αφορά τρεις βασικές κατηγορίες: προσθήκη επιπλέον χαρακτηριστικών στο σχήμα, βελτίωση της απόδοσης του σχήματος και διατήρηση καινούριων στοιχείων που αφορούν το ιστορικό.

4.2.2 Μέτρηση των αλλαγών

Στα σχήματα 13(α′) και 13(α′) βλέπουμε με μπλε χρώμα τις εισαγωγές και με κόκκινο τις διαγραφές στις σχέσεις και τα συνολικά γνωρίσματα αντίστοιχα ανά τις εκδώσεις του σχήματος. Στο σχήμα 14(α′) έχουμε συνολικά τις προσθέσεις και αφαιρέσεις σχέσεων και γνωρισμάτων στο σχήμα. Παρατηρούμε μία έντονη δραστηριότητα από την 41 στην 42 έκδοση όπου έχουμε μία αλλαγή στην στρατηγική αποθήκευσης των εκδόσεων ενός άρθρου. Επίσης, σημαντική αλλαγή έχουμε και από την έκδοση 97 στην 98 όπου έχουμε την προσθήκη της σχέσης `τλφιλεαρςηιε`, μία από τις μεγαλύτερες σχέσεις (περιέχει 20 γνωρίσματα), που σηματοδοτεί έναν καινούριο τρόπο αποθήκευσης παλιών εκδόσεων άρθρων. Στο σχήμα 14(β′) έχουμε τις αλλαγές του σχήματος 14(α′) με την προσθήκη αλλαγών σε κλειδιά και αλλαγών τύπων

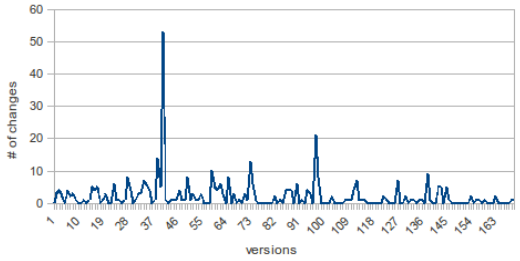


(α′) Μεταβολές σχέσεων

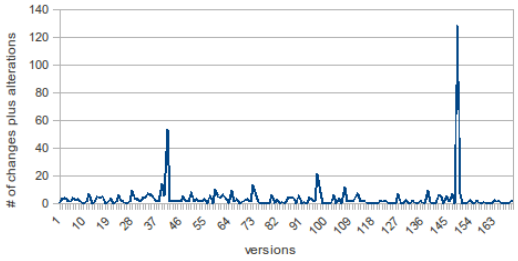


(β′) Μεταβολές Γνωρισμάτων

Σχήμα 13: Μεταβολές



(α′) Εισαγωγές/Διαγραφές



(β′) Εισαγωγές/Διαγραφές και αλλαγές στα γνωρίσματα

Σχήμα 14: Συνολικές αλλαγές

των γνωρισμάτων. Παρατηρούμε μία πολύ μεγάλη αλλαγή από την έκδοση 149 στην 150 όπου έχουμε μεταβολή πάρα πολλών, αν όχι όλων, των γνωρισμάτων από char σε binary πιθανότατα για καλύτερη υποστήριξη από το ΣΔΒΔ MySQL.

Να πούμε σε αυτό το σημείο ότι τα αποτελέσματα των μετρήσεων που κάναμε είναι πολύ κοντά σε παρόμοια αποτελέσματα που έχουμε από την βιβλιογραφία [CMTZ08]. Ορισμένες αποκλίσεις οφείλονται κυρίως στο γεγονός ότι ο parser του συστήματος που υλοποιήσαμε είναι αρκετά ευέλικτος σε συντακτικά λάθη. Έτσι έχουμε μια γενική εικόνα του τι ήθελε ο προγραμματιστής να υλοποιήσει χωρίς να μας απασχολούν μικρές αστοχίες στην σύνταξη των σχημάτων. Τέλος, να σημειώσουμε ότι μετρήσεις όσων αφορά τα κλειδιά και τους τύπους των δεδομένων δεν είχαμε απο την βιβλιογραφία.

5 Επίλογος

Ολοκληρώνοντας την παρουσίαση της Εκάτης, θα κάνουμε μία μικρή σύνοψη των όσων είδαμε σε αυτή την εργασία επισημαίνοντας τα σημεία που θεωρούμε πιο σημαντικά. Στην συνέχεια θα αναφέρουμε ορισμένες μελλοντικές επεκτάσεις του συστήματός μας και σημεία που ενδέχεται να χρειαστούν περαιτέρω μελέτη και ανάλυση.

5.1 Σύνοψη

Στόχος αυτής της εργασίας ήταν η υλοποίηση ενός συστήματος για την οπτικοποίηση αλλαγών μεταξύ δύο διαφορετικών εκδόσεων ενός σχήματος μίας σχεσιακής ΒΔ στα πλαίσια της εξέλιξης σχημάτων. Διευκρινίσαμε κάποιους ορισμούς που αφορούν αυτό το ερευνητικό πεδίο και είδαμε πιθανές επιρροές της εξέλιξης σχημάτων. Αναφερθήκαμε σε σχετικά εργαλεία για αντιστοίχηση (Clio), προεργασία και οπτικοποίηση εξέλιξης (Hecateus) και υλοποίηση της εξέλιξης (PRISM). Δημιουργήσαμε, λοιπόν με επιτυχία, στα πλαίσια του αρχικού στόχου, την Εκάτη και είδαμε τα βασικά συστατικά που απαρτίζουν το σύστημά της. Εξετάσαμε ορισμένα τεχνικά ζητήματα και δείξαμε μία τυπική συνεδρία χρήσης της Εκάτης. Τέλος χρησιμοποιήσαμε το σύστημα για την μέτρηση της εξέλιξης του σχήματος του MediaWiki και παρουσιάσαμε τα δεδομένα επαληθεύοντας την βιβλιογραφία και εμπλουτίζοντας την.

5.2 Μελλοντικές Επεκτάσεις

Η Εκάτη, ως ένα αρκετά πρώιμο εργαλείο, θα μπορούσε να επεκταθεί σε διάφορους τομείς.

- Θα μπορούσε να δοθεί η δυνατότητα στο σύστημα να παίρνει τον ορισμό του σχήματος όχι μόνο από αρχεία DDL αλλά και από τα metadata μίας ήδη ανεπτυγμένης βάσης σε ένα ΣΔΒΔ με την χρήση του jdbc API που προσφέρει η Java.

- Μία άλλη πιθανή επέκταση θα μπορούσε να είναι η παραγωγή ενός DDL αρχείου με ένα σύνολο από εντολές που αν εφαρμοστούν στο σχήμα της αρχικής έκδοσης θα μας δώσουν το σχήμα της μεταγενέστερης έκδοσης.

- Μία αδυναμία του συστήματος αυτή την στιγμή είναι ότι σε περίπτωση που κάποιο στοιχείο αλλάξει όνομα τότε αναγνωρίζετε ως μία διαγραφή και μία εισαγωγή με το καινούριο όνομα. Θα μπορούσε λοιπόν αν εντοπίσει δύο στοιχεία που τα υπο-στοιχεία τους είναι ίδια και τα δύο ονόματα σχετίζονται να θεωρεί ότι υπήρξε μετονομασία.

- Στα ίδια πλαίσια με την προηγούμενη επέκταση θα μπορούσε να υποστηρίζει ποιο σύνθετες πράξεις μεταξύ πινάκων όπως merges και splits.

Αναφορές

[ALP91] J. Andany, M. Léonard, and C. Palisser. Management of Schema Evolution in Databases. In *Proceedings of the 17th International Conferance on Very Large Data Bases*, pages 161–170, 1991.

[CMTZ08] C. Curino, H. J. Moon, L. Tanca, and C. Zaniolo. Schema Evolution in Wikipedia: toward a Web Information System Benchmark. In *Proceedings of ICEIS 2008*. Citeseer, 2008.

[CMZ08a] C. Curino, H. J. Moon, and C. Zaniolo. Managing the History of Metadata in support for DB Archiving and Schema Evolution. In I.-Y. Song et al., editors, *Proceedings of Workshop on Evolution and Change in Data Management*. Springer-Verlag, 2008.

[CMZ08b] C. A. Curino, H. J. Moon, and C. Zaniolo. Graceful Database Schema Evolution: the PRISM Workbench. *Proceedings of the VLDB Endowment*, 1:761–772, 2008.

[HMHY01] M. A. Hernández, R. J. Miller, L. M. Haas, and L. Yan. Clio: A Semi-Automatic Tool for Schema Mapping. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, 2001.

[JCE+94] C. S. Jensen, J. Clifford, R. Elmasri, S. K. Gadia, P. J. Hayes, and S. Jajodia. A Consensus Glossary of Temporal Database Concepts. *Sigmod Record*, 23(1):52–64, 1994.

[MCD+08] H. J. Moon, C. A. Curino, A. Deutch, C. Y. Hou, and C. Zaniolo. Managing and Querying Transaction-time Databases under Schema Evolution. *Proceedings of the VLDB Endowment*, 1:882–895, 2008.

[MS90] E. McKenzie and R. Snodgrass. Schema evolution and the relational algebra. *Information Systems*, 15(2):207–232, 1990.

[PHV+02] L. Popay, M. A. Hernándezy, Y. Velegrakisz, R. J. Millerz, F. Naumanny, and H. Hoy. Mapping XML and relational schemas with Clio. In *Proceedings of the International Conference on Data Engineering*. Citeseer, 2002.

[PKVV08] G. Papastefanatos, K. Kyzirakos, P. Vassiliadis, and Y. Vassiliou. Hecataeus: A Framework for Representing SQL Constructs as Graphs. In *Proceedings of 10th International Workshop on Exploring Modeling Methods for Systems Analysis and Design-EMMSAD*, 2008.

[PVSV07] G. Papastefanatos, P. Vassiliadis, A. Simitsis, and Y. Vassiliou. What-if Analysis for Data Warehouse Evolution. *Data Warehousing and Knowledge Discovery*, pages 23–33, 2007.

[RCR94] J. F. Roddick, N. Craske, and T. Richards. A taxonomy for schema versioning based on the relational and entity relationship models. *Entity-Relationship Approach—ER'93*, pages 137–148, 1994.

[Rod95] F. J. Roddick. A Survey of Schema Versioning Issues for Database Systems. *Issues for Database Systems*, 37(7):383–393, 1995.

[Sjø93] D. Sjøberg. Quantifying schema evolution. *Information and Software Technology*, 35(1):35–44, 1993.

[WZZM05a] F. Wang, C. Zaniolo, X. Zhou, and H. J. Moon. Managing Multiversion Documents & Historical Databases: a Unified Solution on XML. In *Proceedings of Web and Databases 2005*, pages 151–153, 2005.

[WZZM05b] F. Wang, C. Zaniolo, X. Zhou, and H. J. Moon. Version Management and Historical Queries in Digital Libraries. In *Proceedings of the 12th International Symposium on Temporal Representation and Reasoning*, pages 207–209. IEEE Computer Society, 2005.

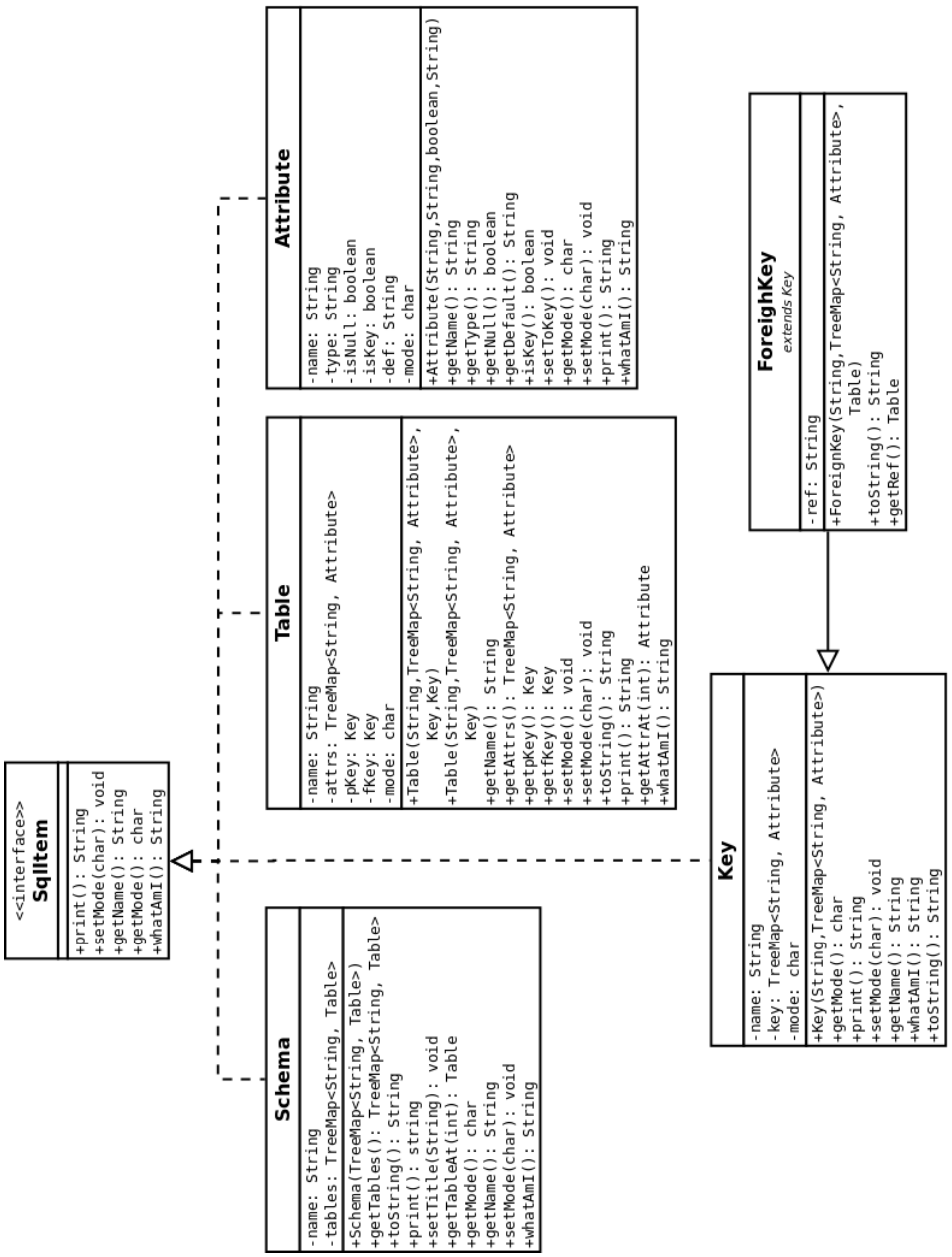
Α΄ BNF της DDL γλώσσας που υλοποιήσαμε

```
<start> ::= <drop> <create> <namespace> ;

<namespace> ::= 'use' <name> ';' ;

<drop> ::=
'drop' 'table' [ 'if' 'exists' ] <nameList> ';' ;

<create> ::=
'create' <schema> | <table> | <index> ';' ;

<schema> ::=
'schema' [ 'if' 'not' 'exists' ] <name> [ <parameter> ] ;

<table> ::=
'table' [ 'if' 'not' 'exists' ] <name>
'(' <definition> ')' [ <parameter> ] ;

<definition>::=
<column> | <constraint>  | <index>
{ ',' <column> | <constraint> | <index> } ;

<column>::= <name> <type> { <option> } ;

<constraint> ::= <key> ;


<index> ::= <p_index> | <fulltext> ;

<p_index> ::=
[ 'unique' | 'primary' ] 'index' [ <name> ]
[ 'on' <name> ] '(' <nameList> ')' ;

<fulltext> ::=
'fulltext' <name> '(' <nameList> ')' ;

<key> ::= <p_key> | <f_key> ;

<p_key>::=
[ 'unique' | 'primary' ] 'key' [ <name> ]
[ '(' <nameList> ')' ] ;
```

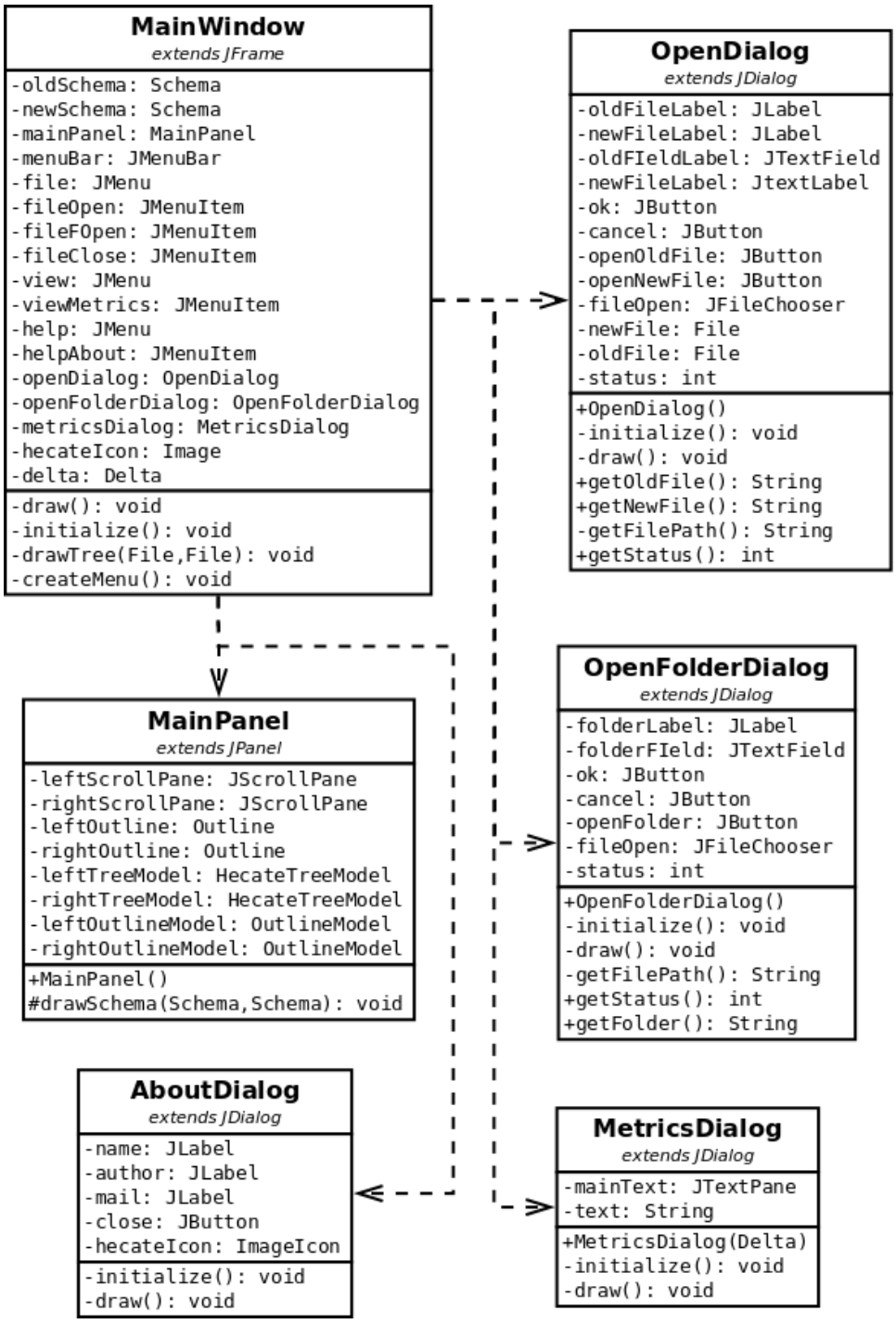
```
<f_key>::=
'foreign' ' key' [ <name> ] [ '(' <nameList> ')' ] <reference>

<option> ::=
<key> | <reference> | <null> | <incr> | <default> ;

<null> ::= [ 'not' ] 'null' ;

<incr> ::= 'auto_increment' ;

<deafult> ::= 'default' <value> | 'null' ;

<reference> ::=
'references' [ <name> ] [ '(' <nameList> ')' ] { referenceOptions } ;

<referenceOptions> ::= <on_delete> | <on_update> ;

<on_delete> ::=
'on' 'delete' 'cascade' | 'restrict' | <no_action> | <set> );

<on_update> ::= 'on' 'update' 'cascade' | <set> ;

<no_action> ::= 'no' 'action' ;

<set> ::= 'set' 'default' | 'null' ;

<order> ::= 'asc' | 'dec' ;

<parameter> ::= <asign> | <char_set>  ;

<asign> ::= <name> '=' <value> { [ ',' ] <name> '=' <value> } ;

<char_set> ::= 'default' 'character' 'set' [ '=' ] <value> ;

<type> ::= <main_type> | <enum> | <binary>;

<main_type> ::=
<name> [ '(' <integer> [ ',' <integer> ] ] [ 'unsigned' | 'binary' ];

<emun> ::= 'enum' '(' <nameList> ')' ;

<binary> ::= 'binary' '(' <integer> ')' ;
```

```
<nameList> ::=
<name> [ '(' <value> ')' ] [ <order> ]
{ [ ',' ] <name> [ '(' <value> ')' ] order? } ;

<value> ::= <name> | <integer> ;

<name> ::= <m_id> | <m_def> ;

<m_id> ::= <id> { '.' <id> } ;

<m_def> ::= <def> { '.' <def> } ;

<id> ::= { <char> }

<def> ::= ''' | '`' <id> ''' | '`' ;

<char>::=
'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'g' |
'k' | 'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' |
'u' | 'v' | 'w' | 'x' | 'y' | 'z' ;

<integer>::=
'0' | '1' | '2' | '3' | '4' |
'5' |'6' | '7' | '8' | '9' |
```

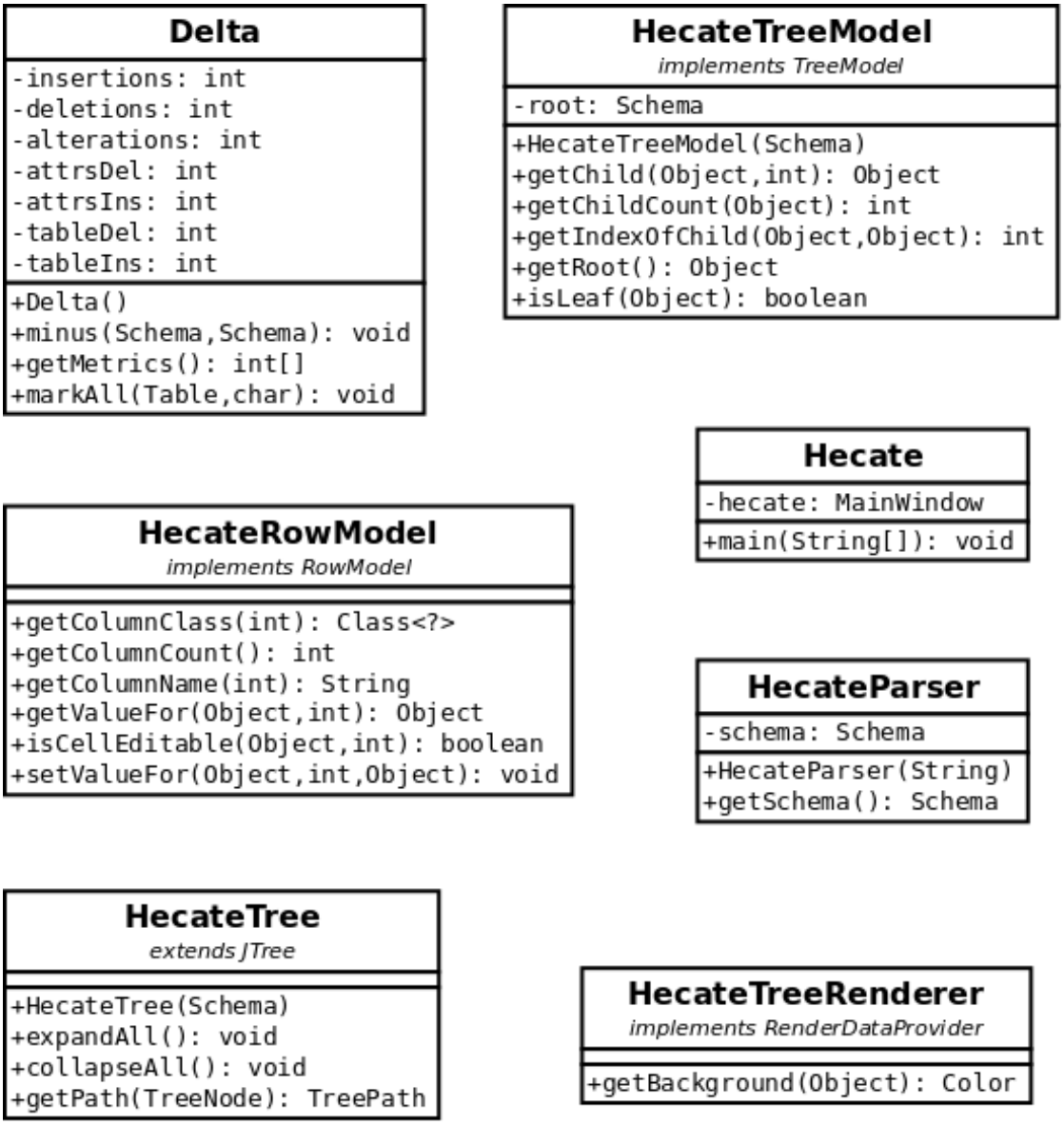
B′ UML – Class Diagrams



Σχήμα 15: Package sql



Σχήμα 16: Package swing



Σχήμα 17: Miscellaneous classes