ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

IntegraHEALTH 1.0: Εφαρμογή Σημασιολογικής Ολοκλήρωσης και Ομόσπονδης Αναζήτησης σε Ετερογενή Δεδομένα Υγείας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νικόλαος Α. Χριστοδούλου

Επιβλέπων : Νικόλαος Μήτρου
Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2012

Η σελίδα αυτή είναι σκόπιμα λευκή.

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

IntegraHEALTH 1.0: Εφαρμογή Σημασιολογικής Ολοκλήρωσης και Ομόσπονδης Αναζήτησης σε Ετερογενή Δεδομένα Υγείας

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νικόλαος Α. Χριστοδούλου

Επιβλέπων : Νικόλαος Μήτρου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 28/03/2012

.....	.....	.....
Νικόλαος Μήτρου	Ευστάθιος Συκάς	Μιχαήλ Θεολόγου
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Μάρτιος 2012

.....
Νικόλαος Α. Χριστοδούλου
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη της εφαρμογής ολοκλήρωσης ετερογενών δεδομένων υγείας IntegraHEALTH 1.0. Η εφαρμογή αυτή λαμβάνει τα αποθηκευμένα δεδομένα ενός συνόλου ιατρών και τα μετατρέπει από τις αρχικές μορφές τους (σχεσιακή βάση δεδομένων, αρχεία ιατρικών προτύπων DICOM και HL7) σε μια ενιαία μορφή, ώστε να τους υποβάλλονται με ευκολία ερωτήματα.

Η εφαρμογή είναι γραμμένη σε γλώσσα Java και χρησιμοποιεί τεχνολογίες Σημασιολογικού Ιστού μέσω του προγραμματιστικού πλαισίου Jena. Τα δεδομένα μετατρέπονται σε προτάσεις του μοντέλου δεδομένων RDF, ενώ χρησιμοποιούνται στοιχεία από τις γλώσσες Σημασιολογικού Ιστού RDFS και OWL. Η μετατροπή γίνεται μέσω αντιστοιχίσεων των δεδομένων σε όρους λεξιλογίων που ονομάζονται οντολογίες. Τη διαδικασία αναλαμβάνουν κλάσεις Java και το ανεξάρτητο εργαλείο D2RQ.

Τα RDF δεδομένα υφίστανται διαδικασίες συλλογιστικής και αποθηκεύονται σε ειδικά διαμορφωμένες βάσεις (triple stores), οι οποίες δημοσιεύονται μέσω του Joseki RDF Server σε τελικά σημεία SPARQL. Ο χρήστης υποβάλλει ομόσπονδα ερωτήματα σε γλώσσα SPARQL τα οποία απευθύνονται σε όλα τα διαθέσιμα σημεία μέσω του ανεξάρτητου εργαλείου FedX, το οποίο απαντάει με τέτοιο τρόπο ώστε να δίνεται στο χρήστη η αίσθηση της ύπαρξης μίας μοναδικής δεξαμενής δεδομένων.

Λέξεις – κλειδιά: Ολοκλήρωση δεδομένων, Σημασιολογικός Ιστός, σχεσιακή βάση δεδομένων, DICOM, HL7, RDF, RDFS, OWL, οντολογία, SPARQL, triple store, συλλογιστική, ομόσπονδο ερώτημα.

Abstract

The aim of this diploma thesis is to develop the heterogeneous healthcare data integration application IntegraHEALTH 1.0. This application receives the stored data from a set of doctors and transforms them from their original formats (relational database, DICOM and HL7 file formats) into a global one, in order for queries to be easily submitted to them.

The application is developed using the Java programming language as well as Semantic Web technologies through the Jena programming framework. The data are transformed in RDF model statements, and elements from the Semantic Web languages RDFS and OWL are used. The transformation is done through matching the data against the terms of vocabularies called ontologies. That procedure is being carried out by Java classes and the independent tool D2RQ.

The RDF data undergo reasoning procedures and are stored in specially formatted databases called triple stores, which are being published through the Joseki RDF Server in SPARQL endpoints. The user submits federated queries using the SPARQL language which are directed towards every available endpoint using the independent tool FedX, which responds in such a way that the user is given the illusion of the existence of only one global data repository.

Keywords: Data integration, Semantic Web, relational database, DICOM, HL7, RDF, RDFS, OWL, ontology, SPARQL, triple store, reasoning, federated query.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1:
ΕΙΣΑΓΩΓΗ

Η παρούσα διπλωματική εργασία ασχολείται με το θέμα της ολοκλήρωσης δεδομένων (data integration). Μια ευρέως διαδεδομένη λύση σε αυτό το πρόβλημα είναι ο μετασχηματισμός δεδομένων που είναι μορφολογικώς ανόμοια ή/και βρίσκονται τοποθετημένα σε διάφορες ανεξάρτητες πηγές, σε μια ενιαία μορφή, ώστε ο χρήστης να αντιλαμβάνεται την ύπαρξη μίας μοναδικής πηγής δεδομένων. Απώτερος στόχος είναι η εύκολη αναζήτηση πληροφοριών, χωρίς να είναι απαραίτητη η γνώση ούτε της αρχικής μορφής τους, ούτε της κατανομής τους (ποια δεδομένα βρίσκονται πού).

Μερικά από τα εργαλεία που χρησιμοποιούνται για την υλοποίηση της ολοκλήρωσης δεδομένων χρησιμοποιούν τεχνολογίες Σημασιολογικού Ιστού (Semantic Web), με απώτερο σκοπό το μετασχηματισμό δεδομένων σε μορφές αναγνώσιμες από τους υπολογιστές (machine-readable), ώστε να είναι κατανοητό από αυτούς και το «νόημα» τους. Έτσι, μπορούν να υποβάλλονται ερωτήματα (queries) μέσω διαδικτύου σε απομακρυσμένες πηγές δεδομένων, τα οποία θα μπορούν να είναι και «έξυπνα», καθώς θα απαντώνται βάσει όχι μόνο του περιεχομένου αλλά και του νοήματος (meaning) των δεδομένων. Η λύση που προτείνεται σε αυτή την εργασία περιλαμβάνει την μετατροπή των υπό ολοκλήρωση δεδομένων σε μορφή RDF και ομόσπονδη αναζήτηση σε αυτά μέσω της γλώσσας ερωτημάτων SPARQL.

Ως αντικείμενο για την ανάπτυξη σχετικής εφαρμογής επιλέχθηκε το πεδίο της ιατροφαρμακευτικής περίθαλψης (healthcare). Στα πλαίσια της επιλογής αυτής, δημιουργήθηκε η εφαρμογή IntegraHEALTH 1.0, στόχος της οποίας είναι η ολοκλήρωση δεδομένων που αφορούν σε ασθενείς, περιλαμβάνοντας τόσο τα δημογραφικά τους στοιχεία (ονοματεπώνυμο, πατρώνυμο, κ.τ.λ.), όσο και τα ιατρικά τους δεδομένα (διάγνωση, συμπτώματα, εξετάσεις, κ.τ.λ.).

Η εφαρμογή πραγματεύεται την ολοκλήρωση των δεδομένων που διατηρούνται από ιατρούς ξεχωριστών ειδικοτήτων, καθένας από τους οποίους χρησιμοποιεί και διαφορετικό τρόπο επεξεργασίας και αποθήκευσης. Ενδεικτικά επιλέχθηκαν μια σχεσιακή βάση δεδομένων (relational database), αρχεία τύπου DICOM προερχόμενα από το ομώνυμο πρότυπο [28] και αρχεία μηνυμάτων HL7 [42].

Η επιλογή των δύο προτύπων προέρχεται από την υψηλή δημοτικότητά τους μεταξύ ιατρών και κατασκευαστών και από το γεγονός ότι επιτελούν συμπληρωματικές λειτουργίες καθώς το πρότυπο DICOM κωδικοποιεί και διαχειρίζεται τα δεδομένα όλου του βασικού απεικονιστικού εξοπλισμού ενός νοσοκομείου, ενώ το HL7 ασχολείται με όλα τα υπόλοιπα θέματα ιατρικά ή μη (παραγγελίες, εισαγωγές-εξαγωγές ασθενών, εξετάσεις κ.τ.λ.).

Ο κώδικας της εφαρμογής είναι γραμμένος σε γλώσσα Java. Περιλαμβάνονται οι βασικές λειτουργίες επεξεργασίας βάσεων δεδομένων (insert, update, delete) μέσω του JDBC (Java DataBase Connectivity) καθώς και κλάσεις τόσο για τη δημιουργία αρχείων HL7 και DICOM, όσο και για την εξαγωγή πληροφοριών από αυτά.

Το κοινό λεξιλόγιο μετατροπής των δεδομένων προέρχεται από το μοντέλο δεδομένων RDF [109] και τις γλώσσες Σημασιολογικού Ιστού RDFS [110] και OWL [107]. Τα πρωτογενή δεδομένα των ιατρών μετατρέπονται σε αντίστοιχους RDF γράφους (RDF graphs), που χρησιμοποιούν και αναφέρονται σε εξωτερικά λεξιλόγια, γνωστά και ως οντολογίες (ontologies), οι οποίες με τη σειρά τους αποτελούν αξιωματικές περιγραφές ενός δεδομένου τομέα γνώσης. Τη διαδικασία αυτή αναλαμβάνουν κλάσεις κώδικα Java που χρησιμοποιούν στοιχεία από τη βιβλιοθήκη του προγραμματιστικού πλαισίου (framework) Σημασιολογικού Ιστού Jena [6], καθώς και το ανεξάρτητο εργαλείο D2RQ [34]. Ο χρήστης μπορεί να υποβάλει έξυπνα ερωτήματα πάνω στα μετασχηματισμένα δεδομένα μέσω της ειδικής γλώσσας ερωτημάτων SPARQL [111]. Το αντίστοιχο κομμάτι της εφαρμογής για κάθε ιατρό διαθέτει μια διεπαφή που ονομάζεται τελικό σημείο SPARQL (SPARQL endpoint) στην οποία και δημοσιεύονται τα μετασχηματισμένα δεδομένα. Οι διεπαφές αυτές δημιουργούνται με τη βοήθεια του Joseki RDF server [53].

Προκειμένου να εξασφαλιστεί η διαφάνεια των πρωτογενών πηγών δεδομένων στον τελικό χρήστη, τα ερωτήματα υποβάλλονται σε ένα εργαλείο διαχείρισης ομόσπονδων ερωτημάτων (federated queries), το οποίο αποφασίζει ποιο υποερώτημα να στείλει σε κάθε τελικό σημείο και κατόπιν συγκεντρώνει και παρουσιάζει τα αποτελέσματα στο χρήστη, δίνοντάς του την αίσθηση ότι θέτει ερωτήματα σε μια μοναδική πηγή δεδομένων.

Η παρούσα εφαρμογή προσπαθεί να δώσει λύσεις σε δύο προβλήματα. Το πρώτο αφορά στη μηχανοργάνωση των υπηρεσιών και λειτουργιών των εθνικών συστημάτων υγείας. Γενικά στο εξωτερικό υπάρχουν αυτόνομα συστήματα που καταγράφουν πληροφορίες ιατρικών φακέλων ασθενών τα οποία κατασκευάζονται συνήθως από ιδιώτες [30, 46, 52]. Στην Ελλάδα όμως, πλην του νεοσύστατου συστήματος της ηλεκτρονικής συνταγογράφησης [122] δεν υφίσταται κάποια γενικής χρήσεως ή/και αναγνωρισμένη από τους κρατικούς φορείς ψηφιακή καταγραφή οποιουδήποτε ιατρικού ιστορικού, αποτελεσμάτων κ.τ.λ., πλην των περιπτώσεων ορισμένων μεμονωμένων ιατρείων και κλινικών που κρατούν τα δικά τους ηλεκτρονικά αρχεία. Επομένως, εφαρμογές σαν την IntegraHEALTH 1.0 μπορούν να βοηθήσουν σε περιπτώσεις όπου χρειάζεται απεμπλοκή από τη χειρόγραφη καταγραφή πληροφοριών, εντατικοποιημένος έλεγχος και στενή παρακολούθηση κινήσεων, καταχωρήσεων, εντολών και αρχείων. Φυσικά οι ίδιες αρχές ισχύουν και για οποιοδήποτε τύπο και είδος δεδομένων (π.χ. βιομηχανικά, οικονομικά δεδομένα, δεδομένα κινήσεως οχημάτων κ.τ.λ.)

Το δεύτερο πρόβλημα το οποίο μπορεί να επιλυθεί βάσει των τεχνολογιών που χρησιμοποιούνται στην IntegraHEALTH 1.0 είναι η διασύνδεση πολλών ετερογενών συστημάτων αποθήκευσης δεδομένων. Βάσει του [108] μπορεί να υποτεθεί ότι είναι πρακτικά δυνατή η μετατροπή κάθε είδους πληροφορίας σε μια κοινή μορφή (εδώ σε προτάσεις RDF), κάτι το οποίο επιτρέπει ολοένα και πιο διευρυμένες αναζητήσεις σε μια πληθώρα τελικών σημείων. Για παράδειγμα, μπορεί κάποιος να αναζητήσει για ένα άτομο δεδομένα ιατρικού χαρακτήρα, ποινικό μητρώο, δεδομένα φορολογικού χαρακτήρα, ή ακόμα και στοιχεία από το φορέα στον οποίο εργάζεται το άτομο αυτό, αν υπάρχει η κατάλληλη εφαρμογή που να το υλοποιεί. Μπορεί επομένως να τεθεί ως

απώτερος στόχος η δημιουργία υπερσυστημάτων γενικού σκοπού ικανών να ανακτήσουν εύκολα οποιαδήποτε πληροφορία για οποιοδήποτε στοιχείο τους ζητηθεί, αφού τα πάντα θα εκφράζονται σε μια κοινή μορφή.

Στο κεφάλαιο 2 παρουσιάζεται το πρόβλημα της ενοποίησης δεδομένων τόσο σε γενικό πλαίσιο όσο και ειδικότερα, για την περίπτωση των ιατρικών δεδομένων. Στο κεφάλαιο 3 μελετώνται οι τεχνολογίες και τα πρότυπα που χρησιμοποιεί η εφαρμογή IntegraHEALTH 1.0. Στο κεφάλαιο 4 δίνεται η αρχιτεκτονική της εφαρμογής, ενώ στο κεφάλαιο 5 περιγράφεται η υλοποίησή της. Στο κεφάλαιο 6 περιγράφονται σενάρια χρήσης της εφαρμογής μέσω της υποβολής διάφορων σύνθετων και «έξυπνων» ερωτημάτων. Τέλος, στο κεφάλαιο 7 δίνονται τα συμπεράσματα της όλης μελέτης και ένας επίλογος.

ΚΕΦΑΛΑΙΟ 2:
ΤΟ ΠΡΟΒΛΗΜΑ ΤΗΣ ΟΛΟΚΛΗΡΩΣΗΣ ΔΕΔΟΜΕΝΩΝ

2.1 Γενικές έννοιες

2.1.1 Η ετερογένεια των πηγών δεδομένων και τα είδη της

Τα προβλήματα στο συνδυασμό πολλαπλών πηγών δεδομένων κάτω από μία μοναδική διεπαφή (interface) ερωτημάτων, υπάρχουν εδώ και αρκετό καιρό. Η γρήγορη υιοθέτηση της τεχνολογίας των βάσεων δεδομένων μετά τη δεκαετία του 1960, οδήγησε, όπως ήταν αναμενόμενο, στην ανάγκη να διαμοιραστούν ή να συγχωνευθούν οι υπάρχουσες πηγές. Τα μοντέρνα συστήματα διαχείρισης βάσεων δεδομένων (Database Management System, DBMS) επιλύουν το πρόβλημα της πρόσβασης και της διαχείρισης μεγάλου όγκου δεδομένων σε μια μεμονωμένη πλατφόρμα, όμως πολλά θέματα προκύπτουν όταν δύο ή περισσότερες βάσεις καλούνται να συνεργαστούν. Το βασικό πρόβλημα είναι η ετερογένεια των διαφόρων πηγών δεδομένων, η οποία μπορεί να εμφανιστεί σε πολλές μορφές, ξεκινώντας από τις πλατφόρμες υλικού και λογισμικού στις οποίες βασίζονται και φτάνοντας μέχρι το μοντέλο δεδομένων και το σχήμα (schema) που χρησιμοποιούν για να παρέχουν μια λογική δομή στα αποθηκευμένα δεδομένα. Μερίδιο ευθύνης φέρει και η διαθέσιμη ποικιλία των ειδών δεδομένων που μπορούν να αποθηκευθούν (απλό κείμενο, έγγραφα, εικόνες, βίντεο, κ.τ.λ.) [48].

Ανάλογα με το επίπεδο της δομής των πηγών στο οποίο εμφανίζεται η ετερογένεια, μπορεί να τοποθετηθεί σε μία από τέσσερις κατηγορίες, οι οποίες ακολουθούν την ιεραρχία που δίνεται παρακάτω [37, 90]:

- Συντακτική ετερογένεια: Πρόκειται για το αποτέλεσμα των διαφορών μεταξύ των μοντέλων αναπαράστασης των δεδομένων (σχεσιακό μοντέλο, μοντέλο οντοτήτας – συσχέτισης, αντικειμενοστραφές μοντέλο).
- Σχηματική ή δομική ετερογένεια: Σχετίζεται με το γεγονός ότι τα διάφορα πληροφοριακά συστήματα αποθηκεύουν τα δεδομένα τους χρησιμοποιώντας διαφορετικές δομές δεδομένων.
- Σημασιολογική ετερογένεια: Προκύπτει από τα περιεχόμενα των πληροφοριών και το νόημα που τους αποδίδεται.
- Συστημική ετερογένεια: Οφείλεται στη χρήση διαφορετικών λειτουργικών συστημάτων ή διαφορετικών πλατφόρμων υλικού.

Ιδιαίτερη προσοχή αξίζει να δοθεί στο φαινόμενο της σημασιολογικής ετερογένειας που πηγάζει από το γεγονός ότι σε αρκετές περιπτώσεις τα ίδια ή επικαλυπτόμενα δεδομένα επαναλαμβάνονται σε δύο ή περισσότερες βάσεις ή/και αναπαριστώνται με διαφορετικό τρόπο από τα σχήματα που τις υποστηρίζουν. Αυτό έχει σαν συνέπεια εσφαλμένες απαντήσεις σε πιθανά ερωτήματα, είτε παρουσιάζοντας εις διπλούν τα ζητούμενα στοιχεία, είτε παραλείποντας πληροφορίες

που σχετίζονται με κάποιο στοιχείο επειδή αυτές εκφράζονται διαφορετικά. Υπάρχουν τέσσερις τύποι σημασιολογικής ετερογένειας [10]:

- Ονομαστική διένεξη (naming conflict): Προκύπτει όταν δύο ή περισσότερες βάσεις χρησιμοποιούν διαφορετικά ονόματα για στήλες που περιέχουν τα ίδια ακριβώς δεδομένα. Π.χ. ο Αριθμός Μητρώου Κοινωνικής Ασφάλισης (Α.Μ.Κ.Α.) ενός ατόμου μπορεί να αποθηκεύεται σε ένα πεδίο μιας βάσης με όνομα «Α.Μ.Κ.Α.» ενώ σε άλλη βάση μπορεί να αποθηκεύεται στο πεδίο «Προσωπικός κωδικός ασφάλισης».
- Διένεξη πεδίου ορισμού (domain conflict): Παρουσιάζεται όταν δύο η περισσότερες βάσεις δεδομένων αναπαριστούν το ίδιο στοιχείο του πραγματικού κόσμου με χρήση διαφορετικών τιμών. Για παράδειγμα ο ίδιος υπάλληλος μπορεί στις βάσεις δύο διαφορετικών τμημάτων μιας εταιρείας (μητρώο υπαλλήλων και μητρώο μισθοδοσίας) να αναφέρεται σε κάθε μία με διαφορετικό προσωπικό κωδικό.
- Διένεξη μεταδεδομένων (metadata conflict): Τέτοιου είδους διενέξεις συμβαίνουν όταν τα ίδια δεδομένα αναπαριστώνται σε μια βάση στο επίπεδο σχήματος, ενώ σε άλλη στο επίπεδο δεδομένων. Π.χ. το ιστορικό μισθοδοσίας ενός υπαλλήλου μπορεί σε μια βάση να αποθηκευθεί σαν μια στήλη στο μητρώο υπαλλήλων (επίπεδο δεδομένων), ενώ σε άλλη να τοποθετηθεί σε δικό του, ξεχωριστό πίνακα (επίπεδο σχήματος).
- Δομική διένεξη (structural conflict): Εμφανίζεται λόγω της διαφορετικής οργάνωσης των δεδομένων σε κάθε βάση. Π.χ. το όνομα ενός υπαλλήλου μπορεί είτε να αποθηκευθεί είτε ολόκληρο σε ένα πεδίο, είτε να χρησιμοποιηθούν πολλά επιμέρους πεδία (όνομα, επώνυμο, πατρώνυμο, κ.τ.λ.). Σε αυτό το είδος ετερογένειας υπάγονται και οι διενέξεις τύπων δεδομένων (datatype conflicts), που συναντώνται όταν το ίδιο στοιχείο αναπαρίσταται με δεδομένα διαφορετικών τύπων (π.χ. χρήση ακέραιου αριθμού και δεκαδικού κινητής υποδιαστολής για την απεικόνιση μιας θερμοκρασίας) [118].

2.1.2 Ορισμός της ολοκλήρωσης δεδομένων

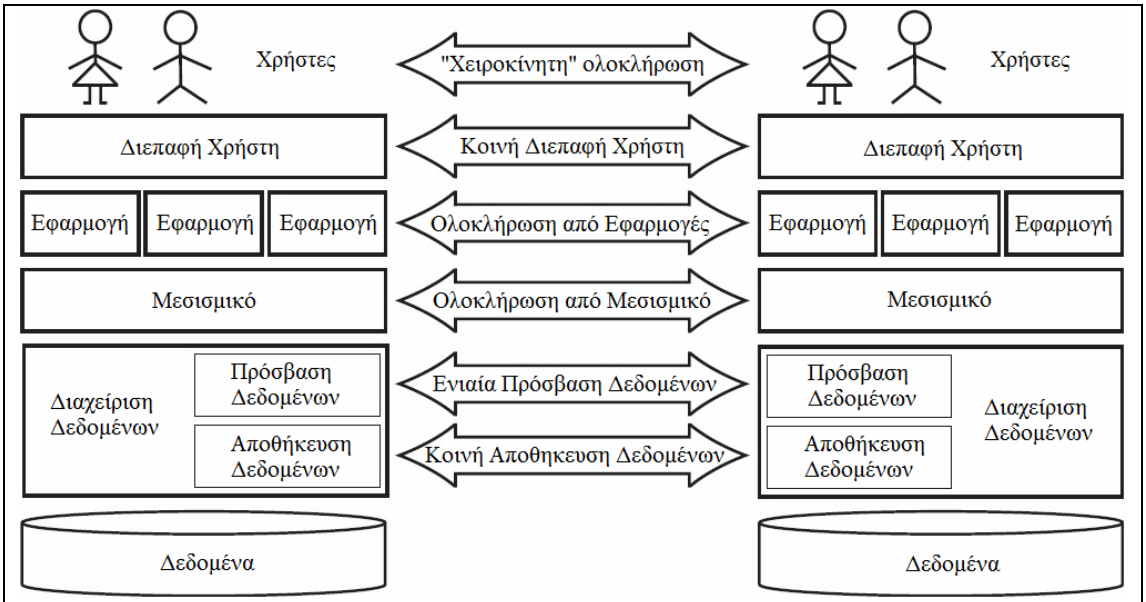
Τα προβλήματα που παρουσιάστηκαν στην προηγούμενη ενότητα αποτελούν κομμάτια ενός μεγαλύτερου προβλήματος που καλείται «Ολοκλήρωση Δεδομένων». Πρόκειται για ένα από τα πιο παλιά πεδία έρευνας και εμφανίστηκε αμέσως μετά την εισαγωγή της τεχνολογίας των βάσεων δεδομένων στον επιχειρηματικό κόσμο. Κατά τους Ziegler και Dittrich [118] «η ολοκλήρωση δεδομένων πολλαπλών συστημάτων πληροφοριών στοχεύει γενικότερα στο συνδυασμό τους, ώστε να σχηματίσουν ένα ενιαίο σύνολο και οι χρήστες να αποκτήσουν την εντύπωση ότι αλληλεπιδρούν με ένα μοναδικό σύστημα πληροφοριών». Στους χρήστες παρέχεται μια ομογενής λογική όψη των δεδομένων τα οποία στην πραγματικότητα είναι διανεμημένα σε ετερογενείς πηγές. Σε αυτήν την όψη μπορούν να υποβάλουν οι χρήστες τα ερωτήματά τους.

Για να γίνει αυτό, όλα τα δεδομένα πρέπει να αναπαρασταθούν χρησιμοποιώντας τις ίδιες αφαιρετικές αρχές (καθολικό μοντέλο δεδομένων και ενοποιημένη

σημασιολογία). Αυτή η διαδικασία συμπεριλαμβάνει τον εντοπισμό και επίλυση των διενέξεων σχήματος και δεδομένων σχετικά με τη δομή και τη σημασιολογία.

2.1.3 Οι διαφορετικές προσεγγίσεις της ολοκλήρωσης δεδομένων

Τα συστήματα διαχείρισης πληροφοριών μπορούν να περιγραφούν χρησιμοποιώντας μια αρχιτεκτονική στρωμάτων, όπως αυτή της εικόνας 2.1. Στο κορυφαίο επίπεδο, οι χρήστες προσπελαύνουν τα δεδομένα και τις υπηρεσίες μέσω διεπαφών που εκτελούνται πάνω σε διαφορετικές εφαρμογές. Οι εφαρμογές μπορεί να χρησιμοποιούν μεσισμικό (middleware) για να προσπελάσουν τα δεδομένα μέσω ενός στρώματος πρόσβασης δεδομένων. Τη διαχείριση των δεδομένων αναλαμβάνει ένα σύστημα αποθήκευσης. Συνήθως συστήματα διαχείρισης βάσεων δεδομένων χρησιμοποιούνται για να συνδυάσουν την πρόσβαση στα δεδομένα και το στρώμα αποθήκευσης.



Εικόνα 2.1: Αρχιτεκτονική διαστρωμάτωσης συστήματος διαχείρισης πληροφοριών και προσεγγίσεις ολοκλήρωσης δεδομένων.

Συνήθως το πρόβλημα της ολοκλήρωσης μπορεί να αντιμετωπιστεί σε καθένα από τα επίπεδα ενός συστήματος, όπως αυτά δίνονται στην εικόνα 2.1. Έτσι είναι διαθέσιμες οι παρακάτω προσεγγίσεις [97, 118]:

- «Χειροκίνητη» ολοκλήρωση ("manual" integration): Σε αυτήν την προσέγγιση όλες οι απαραίτητες λειτουργίες για την επίτευξη της ολοκλήρωσης γίνονται αποκλειστικά από το χρήστη. Ο τελευταίος πρέπει να γνωρίζει πού και σε ποια μορφή βρίσκονται τα προς ολοκλήρωση δεδομενα, να έχει τις απαιτούμενες γνώσεις ώστε να τα εξάγει από τα συστήματα πληροφοριών στα οποία βρίσκονται και τέλος να φέρει σε πέρας τη διαδικασία φυλλομέτρησης των ανακτηθέντων δεδομένων, τον εντοπισμό των

ζητούμενων στοιχείων και την ομογενοποίησή τους χωρίς κάποια αυτόματη διαδικασία από τρίτο λογισμικό.

- Κοινή διεπαφή χρήστη (common user interface): Η παρούσα προσέγγιση διαφέρει από την προηγούμενη μόνο στο γεγονός ότι παρέχεται στους χρήστες μια εφαρμογή – διεπαφή η οποία επικοινωνεί με όλα τα διαθέσιμα συστήματα πληροφοριών, χωρίς να απαιτείται από το χρήση εξειδικευμένη γνώση των τελευταίων. Οι πληροφορίες που ανακτώνται, είτε σύνδεσμοι προς αυτές, εμφανίζονται στη διεπαφή, από όπου ο χρήστης μπορεί να επιλέξει τα στοιχεία που επιθυμεί. Όλες οι άλλες λειτουργίες που απαιτούνται για την ολοκλήρωση των δεδομένων (φυλλομέτρηση, εντοπισμός, ομογενοποίηση) πρέπει να γίνουν «χειροκίνητα» από το χρήστη.
- Ολοκλήρωση από εφαρμογές (integration by applications): Αυτή η προσέγγιση χρησιμοποιεί ειδικές εφαρμογές που προσπελαύνουν τα δεδομένα και επιστρέφουν ολοκληρωμένα αποτελέσματα στο χρήστη. Αυτή η λύση είναι πρακτική για ένα μικρό αριθμό συστημάτων. Ωστόσο οι εφαρμογές μπορούν να γίνουν αρκετά απαιτητικές, καθώς αυξάνεται συνέχεια ο αριθμός των διεπαφών των συστημάτων και των μορφών δεδομένων προς ομογενοποίηση και ολοκλήρωση.
- Ολοκλήρωση από μεσισμικό (integration by middleware): Πρόκειται για μια παραλλαγή της προηγούμενης προσέγγισης, η οποία χρησιμοποιεί την ίδια φιλοσοφία με αυτή του αντικειμενοστραφούς μοντέλου προγραμματισμού. Το μεσισμικό, από τη θέση που έχει στο σύστημα σύμφωνα και με την εικόνα 2.1, μπορεί να σχεδιαστεί για να εκτελέσει κάποιες από τις απαιτούμενες λειτουργίες ολοκλήρωσης δεδομένων αφαιρώντας το «φόρτο εργασίας» από τις εφαρμογές του αμέσως υψηλότερου στρώματος. Ταυτόχρονα, το γεγονός ότι μία μονάδα μεσισμικού μπορεί να εξυπηρετεί περισσότερες από μία «ανώτερες» εφαρμογές επιτρέπει στις τελευταίες να χρησιμοποιούν τις δυνατότητες του μεσισμικού κατά το δοκούν, ακριβώς όπως χρησιμοποιείται μια κλάση σε ένα αντικειμενοστραφές πρόγραμμα. Ωστόσο, η λύση αυτή δεν είναι πανάκεια, καθώς είναι δυνατόν κάποιες λειτουργίες να πρέπει να εκτελεστούν από τις εφαρμογές του ανώτερου στρώματος.
- Ενιαία πρόσβαση δεδομένων (uniform data access): Σε αυτήν την περίπτωση μια λογική ολοκλήρωση επιτυγχάνεται στο επίπεδο πρόσβασης δεδομένων. Σε καθολικές εφαρμογές παρέχεται μια ενοποιημένη όψη των κατανεμημένων δεδομένων, αν και μόνο εικονικά δεδομένα είναι διαθέσιμα σε αυτό το επίπεδο. Ωστόσο, η καθολική παροχή φυσικά ολοκληρωμένων δεδομένων μπορεί να είναι χρονοβόρα, καθώς η πρόσβαση στα δεδομένα, η ομογενοποίηση και η ολοκλήρωση πρέπει να γίνουν στο χρόνο εκτέλεσης.
- Κοινή αποθήκευση δεδομένων (common data storage): Σε αυτήν την προσέγγιση ολοκλήρωσης, τα ίδια τα δεδομένα ή κάποια αντίγραφά τους, συγκεντρώνονται από τις πηγές τους σε ένα κεντρικό χώρο αποθήκευσης, στον οποίο μπορεί ο χρήστης να υποβάλλει ερωτήματα. Ο μεγάλος βαθμός διαθεσιμότητας των δεδομένων εγγυάται τη γρήγορη απάντηση των

ερωτημάτων. Ωστόσο, ο κεντρικός χώρος αποθήκευσης θα πρέπει να ενημερώνεται συχνά για τις αλλαγές που συμβαίνουν στις πηγές ώστε να τις συμπεριλάβει και αυτός.

2.1.4 Διαδικασίες ολοκλήρωσης δεδομένων

Τα συστήματα ολοκλήρωσης δεδομένων πετυχαίνουν το στόχο τους χρησιμοποιώντας το καθένα διαφορετικές διαδικασίες, ανάλογα με το επίπεδο ολοκλήρωσης που απαιτείται. Έτσι μπορεί να υπάρχουν περιπτώσεις όπου τα δεδομένα απλά συλλέγονται χωρίς περαιτέρω αλλαγές, ενώ σε άλλες να υφίστανται και κάποιου είδους σημασιολογική ολοκλήρωση (περαιτέρω επεξεργασία ειδικά για την αντιμετώπιση της σημασιολογικής ετερογένειας που αναφέρεται στην ενότητα 2.1.1). Οι διαδικασίες ολοκλήρωσης περιλαμβάνουν [37]:

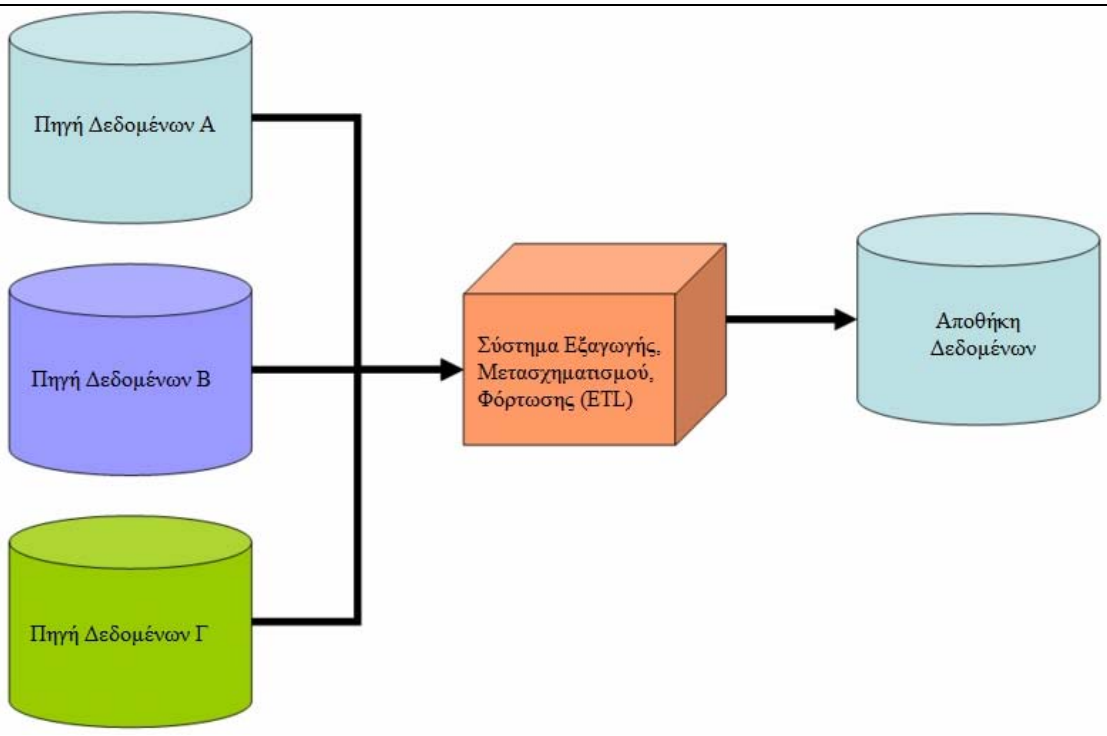
- Αντιστοίχηση δεδομένων/σχημάτων (data/schema matching): Πρόκειται για τη βασική διαδικασία που ακολουθούν τα περισσότερα συστήματα ολοκλήρωσης δεδομένων, όπου τα δεδομένα από διάφορες πηγές αντιστοιχίζονται σε ένα δεδομένο σχήμα. Η θεωρία της προσέγγισης αυτής δίνεται στην ενότητα 2.1.6
- Συλλογή (collection): Τα δεδομένα συλλέγονται από τις πηγές τους αναλλοίωτα. Δεν υπάρχει καμία αντιστοίχησή τους με δεδομένα άλλων πηγών, ή άλλα σχήματα.
- Σύντηξη (fusion): Η ολοκλήρωση των δεδομένων γίνεται με μια απλή εξαγωγή, χωρίς περαιτέρω αφαιρετικούς υπολογισμούς. Αντίθετα όμως με την περίπτωση της συλλογής, η σύντηξη γίνεται συνδυάζοντας σημασιολογικά ισοδύναμα δεδομένα που προέρχονται από τις διάφορες πηγές. Πέρα απ' αυτό, προσπαθεί να προσδιορίσει συνεπείς αναπαραστάσεις, π.χ. αν οι πηγές δεδομένων αναφέρουν αντικρουόμενες τιμές για το ίδιο στοιχείο δεδομένων, η σύντηξη χρησιμοποιεί κανόνες αντιστοίχησης και ευριστικές μεθόδους (heuristics) για να απαλείψει τις όποιες διενέξεις. Αξίζει να σημειωθεί ότι η ολοκλήρωση στο επίπεδο σύντηξης δεδομένων είναι αρκετά δύσκολη, καθώς συχνά είναι αδύνατο να ταυτοποιηθούν τα δεδομένα ή να αποφασιστεί ποια δεδομένη τιμή είναι η σωστή.
- Αφαίρεση (abstraction): Εδώ εφαρμόζονται μετασχηματισμοί στα δεδομένα για να επιτευχθεί το επιθυμητό επίπεδο αφαίρεσης. Σε αυτούς περιλαμβάνονται συναρτήσεις για συνάθροιση (aggregation) δεδομένων, ανακατάταξη οντοτήτων ή ακόμα πιο πολύπλοκες διαδικασίες συλλογιστικής (reasoning)
- Συμπλήρωση (supplementation): Με αυτή τη μέθοδο, τα δεδομένα του συστήματος αποτελούνται όχι μόνο από τα αρχικά δεδομένα των πηγών άλλα και από άλλα, προστιθέμενα δεδομένα που περιγράφουν το περιεχόμενο ή τα συμφραζόμενα (context) των αρχικών (π.χ. σημασιολογικά μεταδεδομένα). Τέτοιου είδους ολοκλήρωση χρησιμοποιείται για να χειριστεί την έμμεση σημασιολογία των δεδομένων. Η διαδικασία αυτή είναι απαραίτητη όπου οι

πηγές δεν παρέχουν κάποιο σχήμα και η ολοκλήρωση βασίζεται σε ένα σχήμα μεταδεδομένων.

- Άλλες διαδικασίες όπως η συνάθροιση (aggregation) δεδομένων ή/και η ομαδοποίησή (grouping) τους βάσει στοιχείων των περιεχομένων τους.

2.1.5 Αποθήκες δεδομένων

Μια από τις πρώτες προτάσεις επίλυσης του προβλήματος ολοκλήρωσης δεδομένων που επιχειρήθηκε από τη δεκαετία του 1970 (θεωρητικά) και υλοποιήθηκε στα τέλη της δεκαετίας του 1980 αφορά τις αποθήκες δεδομένων (data warehouses, εικόνα 2.2). Ένα τέτοιο σύστημα εξάγει, μετασχηματίζει και φορτώνει (extracts, transforms and loads, ETL) [104] τα δεδομένα από διαφορετικές πηγές (βάσεις δεδομένων) σε ένα μοναδικό κοινό σχήμα (schema) στο οποίο μπορούν να υποβληθούν ερωτήματα, κάνοντας τα δεδομένα συμβατά μεταξύ τους. Αυτή η προσέγγιση χρησιμοποιεί μια αρχιτεκτονική στενής σύζευξης (tightly coupled architecture) [91] επειδή τα δεδομένα είναι ήδη συγκεντρωμένα σε ένα συγκεκριμένο μοναδικό χώρο αποθήκευσης κατά τη στιγμή του ερωτήματος. Έτσι απαιτείται μικρός χρόνος για την επεξεργασία και την επίλυση των ερωτημάτων.



Εικόνα 2.2: Δομή Συστήματος Αποθήκης Δεδομένων.

Ωστόσο, η τεχνολογία αυτή παρουσιάζει κάποια προβλήματα [9]:

- Απαιτείται πρόσθετος αποθηκευτικός χώρος για την αποθήκευση των αντιγραμμένων πληροφοριών των επιμέρους συστημάτων σε μια αποθήκη. Η συνεχώς φθίνουσα τιμή των αποθηκευτικών μέσων, αλλά και το φιλτράρισμα

των δεδομένων και η δημιουργία περίληψης τους πριν εισαχθούν στην αποθήκη μετριάζουν το συγκεκριμένο πρόβλημα.

- Μπορεί να υπάρξει πρόβλημα με την ενημερότητα των δεδομένων, το οποίο σημαίνει ότι οι πληροφορίες στην αποθήκη μπορεί να μην είναι πάντα οι πιο πρόσφατες. Όταν μία αρχική πηγή δεδομένων ενημερώνεται, η αποθήκη διατηρεί ακόμα τα παλιά δεδομένα και η διαδικασία εξαγωγής – μετασχηματισμού – φόρτωσης χρειάζεται να επανεκτελεστεί για να επιτευχθεί ο απαραίτητος συγχρονισμός.
- Δυσκολίες παρουσιάζονται στην κατασκευή αποθηκών δεδομένων, όπου υπάρχει μόνο μια διεπαφή ερωτημάτων σε μία σύνοψη των πηγών και καμία πρόσβαση στα πλήρη δεδομένα. Αυτό το πρόβλημα συχνά εμφανίζεται όταν επιχειρείται συγχώνευση διάφορων υπηρεσιών ερωτημάτων όπως στις ταξιδιωτικές διαδικτυακές εφαρμογές ή στις εφαρμογές κατηγοριοποιημένων διαφημίσεων.
- Ένα πιο σοβαρό πρόβλημα είναι το ότι απαιτείται για τα δεδομένα που θα φορτωθούν στην αποθήκη να έχει προσδιορισθεί από πριν η πηγή, η μορφή, το φιλτράρισμα και μια σύνοψή τους, κάτι το οποίο μπορεί να μην συμβαδίζει με κάποιες άλλες απαιτήσεις, π.χ. όταν χρειάζονται απολύτως ενημερωμένα δεδομένα ή υπάρχουν απρόβλεπτες ανάγκες χρηστών.

2.1.6 Η θεωρία και το λογικό πλαίσιο της ολοκλήρωσης δεδομένων

Η θεωρία στην οποία στηρίζεται η ολοκλήρωση δεδομένων είναι ένα υποσύνολο της θεωρίας των βάσεων δεδομένων και τυποποιεί τις έννοιες του προβλήματος σε λογική πρώτης τάξεως (first-order logic). Η εφαρμογή της θεωρίας αυτής δίνει ενδείξεις σχετικά με τις δυνατότητες και τη δυσκολία της ολοκλήρωσης. Αν και οι ορισμοί της μπορεί να φαίνονται αφηρημένοι, έχουν επαρκή γενικότητα για να εξηγήσουν κάθε συμπεριφορά των συστημάτων ολοκλήρωσης δεδομένων.

Με βάση τον ορισμό της ολοκλήρωσης δεδομένων, προκύπτει το συμπέρασμα ότι η κύρια εργασία στην κατασκευή ενός σχετικού συστήματος είναι η καθιέρωση ενός συνόλου αντιστοιχιών (mappings) μεταξύ των πηγών δεδομένων και του καθολικού σχήματος. Το σύνολο αυτό πρέπει να λαμβάνεται κατάλληλα υπόψη στην τεκμηρίωση ενός τέτοιου συστήματος. Για την υλοποίησή του χρησιμοποιείται ένα σύνολο εντολών, έστω W, το οποίο αναλαμβάνει την μετατροπή των δεδομένων από τις μορφές των πηγών σε αυτή που ορίζει το καθολικό σχήμα, εφαρμόζοντας τους κανόνες του συνόλου M (εικόνα 2.3). Στη γενική περίπτωση, το W μπορεί να θεωρηθεί ως ένας αλγόριθμος, ενώ στην πραγματικότητα, όπου χρησιμοποιούνται γλώσσες προγραμματισμού και μοντέλα δεδομένων, μπορεί να χρησιμοποιηθεί ο όρος «κώδικας μετασχηματισμού» (wrapper code), αν θεωρηθεί ότι τα αρχικά δεδομένα «περιέχονται» στα μετασχηματισμένα δεδομένα του καθολικού σχήματος, ή ότι ισοδύναμα τα τελευταία «περιτυλίγουν» τα πρώτα. Στην εφαρμογή IntegraHEALTH 1.0 παρουσιάζονται σχετικά παραδείγματα κώδικα μετασχηματισμού δεδομένων.

Τα 3 αυτά στοιχεία (πηγές, σχήμα, αντιστοιχίες) αποτελούν τα βασικά συστατικά ενός συστήματος ολοκλήρωσης δεδομένων. Σύμφωνα με τον Lenzerini [59], μπορεί να οριστεί ένα τέτοιο σύστημα I σαν μια τριάδα <G, S, M>, όπου:

- G είναι το καθολικό σχήμα, εκφρασμένο σε μια γλώσσα L_G πάνω σε ένα αλφάβητο A_G. Το αλφάβητο περιλαμβάνει ένα σύμβολο για κάθε στοιχείο του G (π.χ. σχέση αν το G είναι σχεσιακό, κλάση αν το G είναι αντικειμενοστραφές κ.τ.λ.).
- S είναι το πηγαίο σχήμα εκφρασμένο σε μια γλώσσα L_S πάνω σε ένα αλφάβητο A_S. Το αλφάβητο περιλαμβάνει ένα σύμβολο για κάθε στοιχείο των πηγών.
- M είναι το σύνολο αντιστοιχιών μεταξύ των G και S, αποτελούμενο από ένα σετ ισχυρισμών (assertions) της μορφής:

$$q_S \rightarrow q_G,$$
$$q_G \rightarrow q_S,$$

όπου τα q_S, q_G, είναι 2 ερωτήματα της ίδιας τάξεως πάνω στο πηγαίο σχήμα S και το καθολικό σχήμα G αντίστοιχα. Τα ερωτήματα q_S εκφράζονται σε μια γλώσσα ερωτημάτων $L_{M,S}$ πάνω στο αλφάβητο A_G. Ο ισχυρισμός $q_S \rightarrow q_G$ καθορίζει ότι η έννοια που αντιπροσωπεύεται από το ερώτημα q_S στις πηγές, αντιστοιχεί στην έννοια του καθολικού σχήματος που αντιπροσωπεύεται από το ερώτημα q_G. (αντίστοιχα και για τον ισχυρισμό $q_G \rightarrow q_S$).

Εικόνα 2.3: Γενική Μορφή Συστήματος Ενοποίησης Δεδομένων.

Το πηγαίο σχήμα περιγράφει τη δομή των πηγών, όπου βρίσκονται τα πραγματικά δεδομένα, ενώ το καθολικό σχήμα παρέχει μία σύμφωνη, ενοποιημένη και εικονική όψη των πηγών. Οι ισχυρισμοί των αντιστοιχιών καθιερώνουν τη σύνδεση μεταξύ των στοιχείων του καθολικού σχήματος και των στοιχείων του πηγαίου σχήματος.

Τα ερωτήματα στο σύστημα I τίθενται με όρους του καθολικού σχήματος G και εκφράζονται στη γλώσσα ερωτημάτων L_G με το αλφάβητο A_G. Ο σκοπός του ερωτήματος είναι να παρέχει τις προδιαγραφές των δεδομένων που ζητείται να εξαχθούν από την εικονική βάση δεδομένων που αντιπροσωπεύεται από το σύστημα ολοκλήρωσης δεδομένων.

Ο παραπάνω ορισμός ενός συστήματος ολοκλήρωσης δεδομένων είναι γενικά επαρκής για να καλύψει σχεδόν κάθε περίπτωση που συναντάται στη σχετική βιβλιογραφία. Προφανώς η φύση της κάθε προσέγγισης εξαρτάται από τα χαρακτηριστικά του συνόλου αντιστοιχιών και από την εκφραστική δύναμη των διάφορων σχημάτων και γλωσσών ερωτημάτων.

Στη συνέχεια καθορίζεται η σημασιολογία ενός συστήματος ολοκλήρωσης δεδομένων. Στα παρακάτω, μια βάση δεδομένων για ένα σχήμα T είναι απλά ένα σύνολο από συλλογές συνόλων, ένα για κάθε σύμβολο στο αλφάβητο του T (π.χ. μία σχέση για κάθε σχεσιακό σχήμα του T, αν το T είναι σχεσιακό, ή ένα σύνολο αντικειμένων για κάθε κλάση του T, αν το T είναι αντικειμενοστραφές κ.τ.λ.). Γίνεται επίσης μια υπόθεση απλοποίησης των πεδίων ορισμού για τα διάφορα σύνολα. Συγκεκριμένα, υποτίθεται ότι οι δομές που απαρτίζουν τις βάσεις δεδομένων που εμπλέκονται με το πλαίσιο αυτό (και οι καθολικές βάσεις και οι βάσεις – πηγές), έχουν πεδίο ορισμού ένα συγκεκριμένο πεδίο Γ.

Για να μπορέσει να οριστεί η σημασιολογία ενός συστήματος ολοκλήρωσης δεδομένων I = <G, S, M> αρχικά θεωρείται μια βάση – πηγή για το I, π.χ. μια βάση δεδομένων D που συμβαδίζει με το πηγαίο σχήμα S και ικανοποιεί όλους τους περιορισμούς του. Με βάση την D μπορεί να προσδιοριστεί ποιο είναι το περιεχόμενο των πληροφοριών του καθολικού σχήματος G. Καλείται καθολική βάση για το I κάθε βάση για το G. Μια καθολική βάση B για το I ονομάζεται σύμφωνη (legal) ως προς την D, εάν:

- Η B είναι σύμφωνη ως προς το G, π.χ. η B ικανοποιεί όλους τους περιορισμούς του G.
- Η B ικανοποιεί το σύνολο αντιστοιχιών M ως προς την D.

Η ιδέα ότι η B ικανοποιεί το σύνολο αντιστοιχιών M ως προς την D εξαρτάται από το πώς θα ερμηνευθούν οι ισχυρισμοί του συνόλου αντιστοιχιών. Μπορεί να σημειωθεί ότι ανεξάρτητα από την ερμηνεία του συνόλου αντιστοιχιών, γενικά, υπάρχουν αρκετές καθολικές βάσεις δεδομένων που είναι σύμφωνες με το I ως προς το D.

Τέλος καθορίζεται η σημασιολογία των ερωτημάτων που τίθενται σε ένα σύστημα ολοκλήρωσης δεδομένων. Όπως σημειώθηκε και παραπάνω, τέτοια ερωτήματα εκφράζονται με όρους συμβόλων του καθολικού σχήματος του I. Γενικά, αν q είναι ένα ερώτημα τάξεως n και DB είναι μια βάση δεδομένων, με q^{DB} υποδηλώνεται το σύνολο των πλειάδων (τάξεως n) στην βάση DB που ικανοποιούν το q. Δοθείσης μιας βάσης-πηγής D για το I, η απάντηση $q^{I,D}$ σε ένα ερώτημα q στο I σχετικά με την D,

είναι το σύνολο των πλειάδων t από αντικείμενα του Γ τέτοια ώστε $t \in q^B$ για κάθε καθολική βάση B η οποία είναι σύμφωνη με το I ως προς την D. Το σύνολο $q^{I,D}$ καλείται σύνολο σίγουρων απαντήσεων του q στο σύστημα I ως προς την D.

2.1.7 Τρόποι μοντελοποίησης συστημάτων

Στην προηγούμενη ενότητα ορίστηκε ένα σύστημα ολοκλήρωσης δεδομένων σαν μια τριάδα <G, S, M>. Η εγκυρότητα του συνόλου αντιστοιχιών M εξαρτάται από τη φύση της επικοινωνίας μεταξύ των σχημάτων G και S. Υπάρχουν δύο δημοφιλείς τρόποι μοντελοποίησης: «Καθολικής Όψεως» (Global As View – GAV) και «Τοπικής Όψεως» (Local As View – LAV) [59].

Εικόνα 2.4: Αναπαράσταση των χώρων των πλειάδων των προσεγγίσεων GAV και LAV. Στην περίπτωση GAV, το σύστημα περιορίζεται στο σύνολο πλειάδων που προκύπτει από τις αντιστοιχίες του κώδικα μετασχηματισμού, ενώ το σύνολο των πλειάδων που εκφράζεται από τις πηγές μπορεί να είναι μεγαλύτερο. Στην περίπτωση LAV, το σύστημα περιορίζεται στο σύνολο των πλειάδων των πηγών, ενώ το σύνολο των πλειάδων που εκφράζονται από το καθολικό σχήμα μπορεί να είναι αρκετά μεγαλύτερο. Έτσι συχνά τα συστήματα LAV έχουν να αντιμετωπίσουν λανθασμένες απαντήσεις.

Τα συστήματα GAV μοντελοποιούν την καθολική βάση ως ένα σύνολο ερωτημάτων πάνω στο S. Στην περίπτωση αυτή το M συνδέει σε κάθε στοιχείο g του G ένα ερώτημα q_S πάνω στο S. Με άλλα λόγια, η γλώσσα ερωτημάτων $L_{M,G}$ επιτρέπει μόνο τις εκφράσεις που απαρτίζονται από σύμβολα του αλφάβητου A_G. Έτσι ένα GAV σύνολο αντιστοιχιών είναι ένα σύνολο ισχυρισμών, ένας για κάθε στοιχείο g του G, που έχουν τη μορφή:

$$g \rightarrow q_S.$$

Η προσέγγιση GAV βασίζεται στην ιδέα ότι το περιεχόμενο του κάθε στοιχείου g του καθολικού σχήματος πρέπει να αποδίδεται με όρους μιας όψης q_S των πηγών. Κατά μία έννοια, το σύνολο M λέει ρητά στο σύστημα πώς να ανακτήσει τα δεδομένα, όταν κάποιος θέλει να αξιολογήσει τα διάφορα στοιχεία του καθολικού

σχήματος. Αυτή η ιδέα είναι αποτελεσματική όταν το σύστημα ολοκλήρωσης δεδομένων βασίζεται σε ένα σταθερό σύνολο πηγών. Η προσέγγιση GAV ευνοεί το σύστημα στην επεξεργασία ερωτημάτων, ωστόσο υπάρχει πρόβλημα στην επέκταση ενός τέτοιου συστήματος με νέες πηγές, καθώς αυτό μπορεί να έχει αντίκτυπο στους ορισμούς των στοιχείων του G, του οποίου οι σχετικές όψεις πρέπει να ξαναοριστούν.

Σε ένα σύστημα LAV, η πηγαία βάση δεδομένων μοντελοποιείται ως ένα σύνολο όψεων πάνω στο G. Σε αυτήν την περίπτωση το M συνδέει σε κάθε στοιχείο s του S ένα ερώτημα q_G πάνω στο G. Με άλλα λόγια, η γλώσσα ερωτημάτων $L_{M,S}$ επιτρέπει μόνο τις εκφράσεις που απαρτίζονται από σύμβολα του αλφάβητου A_S. Έτσι ένα LAV σύνολο αντιστοιχιών είναι ένα σύνολο ισχυρισμών, ένας για κάθε στοιχείο s του S, που έχουν τη μορφή:

$$s \rightarrow q_G.$$

Η προσέγγιση LAV βασίζεται στην ιδέα ότι το περιεχόμενο του κάθε στοιχείου s του καθολικού σχήματος πρέπει να αποδίδεται με όρους μιας όψης q_G του καθολικού σχήματος. Αυτή η ιδέα είναι αποτελεσματική όταν το σύστημα ολοκλήρωσης δεδομένων βασίζεται σε ένα σταθερό και καλά εδραιωμένο καθολικό σχήμα. Η προσέγγιση LAV ευνοεί την επεκτασιμότητα του συστήματος, καθώς η προσθήκη μιας νέας πηγής απλά εμπλουτίζει το M με ένα νέο ισχυρισμό, χωρίς περαιτέρω αλλαγές.

Εκτός από τους 2 προαναφερθέντες τρόπους μοντελοποίησης στο [59] αναφέρεται και η προσέγγιση GLAV, στην οποία οι σχέσεις μεταξύ του καθολικού σχήματος και των πηγών δημιουργούνται χρησιμοποιώντας τόσο GAV, όσο και LAV ισχυρισμούς. Πιο συγκεκριμένα, σε ένα σύνολο αντιστοιχιών GLAV κάθε ισχυρισμός έχει τη μορφή:

$$q_S \rightarrow q_G$$

όπου q_S, q_G είναι ομόσπονδα ερωτήματα πάνω στις πηγές και το καθολικό σχήμα αντίστοιχα. Μια βάση δεδομένων B ικανοποιεί τον παραπάνω ισχυρισμό ως προς μια βάση – πηγή D, αν $q_S^D \subseteq q_G^B$, δηλαδή αν το ερώτημα προς την D περιέχεται στο ερώτημα προς την B (η έννοια του εγκλεισμού ερωτημάτων παρουσιάζεται στην επόμενη ενότητα).

2.1.8 Επεξεργασία ερωτημάτων

Η θεωρία της επεξεργασίας ερωτημάτων στα συστήματα ολοκλήρωσης δεδομένων συνήθως εκφράζεται χρησιμοποιώντας ομόσπονδα ερωτήματα [101]. Ένα τέτοιο ερώτημα μπορεί να θεωρηθεί σαν μια λογική συνάρτηση που εφαρμόζεται στις σχέσεις μιας βάσης δεδομένων όπως η «f(A,B) όπου A<B». Εάν μία πλειάδα αντικατασταθεί στον κανόνα και τον ικανοποιεί, τότε η πλειάδα αυτή μπορεί να θεωρηθεί μέρος του συνόλου των απαντήσεων του ερωτήματος.

Σχετικά με την ολοκλήρωση δεδομένων, ο εγκλεισμός ερωτημάτων είναι μια από τις πιο σημαντικές ιδιότητες των συνενωτικών ερωτημάτων. Ένα ερώτημα A περιέχει

ένα άλλο ερώτημα B (συμβολίζεται ως A ⊃ B) αν τα αποτελέσματα από την εκτέλεση του B είναι ένα υποσύνολο των αποτελεσμάτων της εκτέλεσης του A σε οποιαδήποτε βάση δεδομένων. Τα δύο ερωτήματα ονομάζονται ισοδύναμα, αν τα σύνολα των αποτελεσμάτων τους είναι ίσα για κάθε βάση. Αυτό είναι σημαντικό γιατί και στα GAV και στα LAV συστήματα, ένα χρήστης θέτει συνενωτικά ερωτήματα πάνω σε ένα εικονικό σχήμα που αναπαρίσταται από ένα σύνολο όψεων, ή «υλοποιημένων» συνενωτικών ερωτημάτων. Κατά την ολοκλήρωση ο στόχος είναι να ξαναγραφτούν τα ερωτήματα που αναπαρίστανται από τις όψεις, ώστε τα αποτελέσματά τους να είναι ισοδύναμα με το ερώτημα του χρήστη ή να περιέχονται σε αυτό κατά το μέγιστο βαθμό. Αυτό συνιστά το πρόβλημα της απάντησης ερωτημάτων με χρήση όψεων (view – based query answering).

Στα συστήματα GAV, ο σχεδιαστής γράφει τον κώδικα μετασχηματισμού W για να ορίσει την επανεγγραφή των ερωτημάτων. Κάθε στοιχείο του ερωτήματος του χρήστη αντιστοιχεί σε ένα κανόνα αντικατάστασης, όπως ακριβώς κάθε στοιχείο στο καθολικό σχήμα αντιστοιχεί σε ένα ερώτημα πάνω στην πηγή. Η επεξεργασία ερωτημάτων απλά επεκτείνει ή τροποποιεί το ερώτημα του χρήστη σύμφωνα με τον κανόνα που έχει οριστεί στον ενδιάμεσο κώδικα, ώστε το τελικό ερώτημα να είναι ισοδύναμο.

Στα συστήματα LAV, τα ερωτήματα υφίστανται μια πιο ριζική διαδικασία επανεγγραφής γιατί δεν υπάρχουν εντολές στον κώδικα W για να προσαρμόσουν το ερώτημα του χρήστη σε μια απλή στρατηγική επέκτασης. Το σύστημα ολοκλήρωσης δεδομένων πρέπει να κάνει μια αναζήτηση στο χώρο των πιθανών ερωτημάτων για να βρει την καλύτερη επανεγγραφή. Το αποτέλεσμα μπορεί να μην είναι ισοδύναμο ερώτημα, αλλά μπορεί να περιέχεται στο μέγιστο δυνατό βαθμό από το αρχικό ερώτημα, και οι πλειάδες του αποτελέσματος μπορεί να μην είναι πλήρεις.

2.2 Ολοκλήρωση δεδομένων με χρήση οντολογιών

Μια κατηγορία λύσεων στο πρόβλημα της ολοκλήρωσης δεδομένων, είναι αυτή που χρησιμοποιεί τις οντολογίες, που είναι ειδικές δομές πληροφοριών, για να συνδυάσουν αποτελεσματικά δεδομένα και πληροφορίες από διάφορες ετερογενείς πηγές. Η αποτελεσματικότητα της προσέγγισης αυτής συνδέεται άμεσα με την συνέπεια και την εκφραστικότητα της οντολογίας ή των οντολογιών που χρησιμοποιούνται κατά τη διαδικασία της ολοκλήρωσης.

2.2.1 Οντολογίες

Ο όρος οντολογία (ontology) προέρχεται από τη φιλοσοφία. Σύμφωνα με την ετυμολογία του σημαίνει «ο λόγος περί του όντος», δηλαδή η μελέτη αυτού που υπάρχει. Πρόκειται για κλάδο της μεταφυσικής που ασχολείται με την έννοια της ύπαρξης, τη φύση και την ουσία των «όντων» (αυτών που υπάρχουν), τον τρόπο περιγραφής τους, καθώς και με τις κατηγορίες της ύπαρξης και τις σχέσεις μεταξύ αυτών.

Στη σημερινή εποχή η επιστήμη των υπολογιστών έχει δανειστεί τον συγκεκριμένο όρο. Ο ορισμός της οντολογίας κατά τον T.R. Gruber, ο οποίος βελτιώθηκε αργότερα από τον R. Studer είναι ο εξής: «Μια οντολογία είναι μια ρητή και τυπική προδιαγραφή μιας επίνοιας (conceptualization)» [5].

Στην πράξη, οι οντολογίες είναι λεξιλόγια που περιγράφουν πεδία γνώσης. Περιλαμβάνουν μια πεπερασμένη λίστα όρων που περιλαμβάνουν μεταξύ άλλων κλάσεις (ομάδες αντικειμένων) και ιδιότητες (χαρακτηριστικά αντικειμένων), καθώς και σχέσεις μεταξύ των όρων αυτών. Η βασική σχέση μεταξύ δύο όμοιων όρων είναι αυτή που προκύπτει από την ιεραρχία τους. Όπως και στον αντικειμενοστραφή προγραμματισμό, σε μια οντολογία υπάρχουν κλάσεις, υποκλάσεις και υπερκλάσεις. Το ίδιο πράγμα συμβαίνει και με τις ιδιότητες (υποϊδιότητες, υπεριδιότητες). Άλλες σχέσεις αφορούν πληροφορίες όπως:

- περιορισμούς τιμών (ο κωδικός αριθμός Υ μπορεί να είναι μόνο θετικός ακέραιος),
- προτάσεις μη επικάλυψης (οι κλάσεις Α και Β είναι ξένες μεταξύ τους),
- προδιαγραφές λογικών σχέσεων μεταξύ των αντικειμένων (το στοιχείο Χ έχει ακριβώς ένα κωδικό αριθμό Υ, ο οποίος είναι μοναδικός για το στοιχείο αυτό).

Οι οντολογίες παίζουν σημαντικό ρόλο στη δημιουργία του Σημασιολογικού Ιστού (μια έννοια που αναλύεται στο κεφάλαιο 3). Επιτρέπουν σε εφαρμογές αλλά και σε διάφορους φορείς που έχουν αποθηκευμένες μεγάλες ποσότητες δεδομένων να αντιστοιχίσουν τα δεδομένα τους είτε σε μια κοινή οντολογία είτε σε διαφορετικές οντολογίες με χρήση ενός συνόλου αντιστοιχίσεων μεταξύ των τελευταίων. Το αποτέλεσμα είναι τα δεδομένα να γίνονται «συμβατά» μεταξύ τους καθώς διατυπώνονται ομοιότητες μεταξύ τους, οι οποίες λαμβάνονται υπόψη κατά την ανάγνωση και επεξεργασία των δεδομένων από το σύστημα, ξεπερνώντας τις όποιες διαφορές που προέρχονται από την ετερογένεια των συστημάτων που τα περιέχουν. Διαπιστώνεται έτσι ότι οι οντολογίες υποστηρίζουν τη σημασιολογική διαλειτουργικότητα (semantic interoperability).

Η Τεχνητή Νοημοσύνη έχει μεγάλη παράδοση στην ανάπτυξη και τη χρήση των γλωσσών οντολογιών. Προς το παρόν, οι σημαντικότερες γλώσσες οντολογιών για τον Σημασιολογικό Ιστό είναι οι εξής [5]:

- Η RDF (Resource Description Framework, Πλαίσιο Περιγραφής Πόρων) είναι ένα μοντέλο δεδομένων για αντικείμενα (πόρους) και για τις σχέσεις μεταξύ τους. Παρέχει μια απλή σημασιολογία για το συγκεκριμένο μοντέλο δεδομένων, και μπορεί να εκφραστεί με σύνταξη XML.
- Η RDF Schema είναι μια γλώσσα περιγραφής λεξιλογίου, με την οποία μπορούν να περιγραφούν οι ιδιότητες και οι κλάσεις των πόρων RDF, μαζί με μια σημασιολογία για ιεραρχίες γενίκευσης ιδιοτήτων και κλάσεων.
- Η OWL (Web Ontology Language, Γλώσσα Οντολογιών Ιστού), είναι μια πλουσιότερη γλώσσα περιγραφής λεξιλογίου για την περιγραφή κλάσεων και ιδιοτήτων, όπως είναι οι σχέσεις μεταξύ κλάσεων (π.χ. μη επικάλυψη), η πληθικότητα (π.χ. «ακριβώς ένα»), η ισότητα, η πλουσιότερη τυποποίηση των

ιδιοτήτων, τα χαρακτηριστικά των ιδιοτήτων (π.χ. συμμετρία, μεταβατικότητα), και οι απαριθμητές κλάσεις (enumeration classes).

2.2.2 Ο ρόλος των οντολογιών στην ολοκλήρωση δεδομένων

Οι οντολογίες, ως τυπικά μοντέλα αναπαράστασης, με ρητά ορισμένες έννοιες και σχέσεις που τις συνδέουν, χρησιμοποιούνται για να αντιμετωπίσουν το πρόβλημα της σημασιολογικής ετερογένειας των διαφόρων πηγών δεδομένων. Σε πεδία όπως αυτό της βιοϊατρικής, η ταχεία ανάπτυξη, υιοθέτηση και δημόσια διαθεσιμότητα των οντολογιών έχει δώσει τη δυνατότητα σε όσους ασχολούνται με την ολοκλήρωση δεδομένων να στηριχθούν επάνω τους για τη σημασιολογική ενοποίηση δεδομένων και πληροφοριών.

Χάρη στις οντολογίες, είναι δυνατή πλέον, η ξεκάθαρη ταυτοποίηση οντοτήτων σε ετερογενή συστήματα πληροφοριών και η δήλωση των κατάλληλων σχέσεων που συνδέουν αυτές τις έννοιες μεταξύ τους. Συγκεκριμένα, οι οντολογίες παίζουν τους εξής ρόλους:

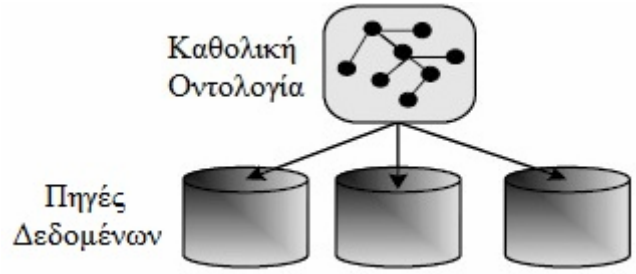
- Προσδιορισμός περιεχομένων: Υπάρχει πλέον η δυνατότητα ακριβής ερμηνείας των δεδομένων από πολλαπλές πηγές μέσω του ρητού ορισμού όρων και σχέσεων στις οντολογίες.
- Μοντέλα ερωτημάτων: Σε μερικά συστήματα, ένα ερώτημα σχηματίζεται χρησιμοποιώντας τις οντολογίες σαν ένα καθολικό σχήμα ερωτημάτων.
- Επαλήθευση: Η οντολογία επαληθεύει το σύνολο αντιστοιχιών που χρησιμοποιείται για την ολοκλήρωση δεδομένων από διαφορετικές πηγές. Αυτά τα σύνολα μπορούν είτε να οριστούν από το χρήστη, είτε να δημιουργηθούν από το σύστημα.

2.2.3 Αρχιτεκτονικές ολοκλήρωσης δεδομένων με οντολογίες

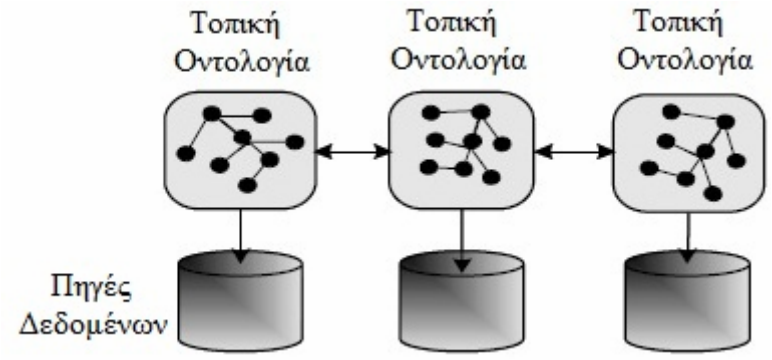
Υπάρχουν τρεις κύριες αρχιτεκτονικές που υλοποιούνται στην ολοκλήρωση δεδομένων με χρήση οντολογιών. Αυτές είναι οι ακόλουθες [37, 114]:

- Αρχιτεκτονική μοναδικής οντολογίας: Χρησιμοποιείται μόνο μία καθολική οντολογία, το λεξιλόγιο της οποίας μπορεί να περιγράψει τα δεδομένα σε όλες τις πηγές και τα ερωτήματα γίνονται μέσα από αυτήν την καθολική οντολογία (εικόνα 2.5α). Αυτή η προσέγγιση μπορεί να φαίνεται απλή, ωστόσο χρειάζεται να γνωρίζει κάποιος τη σημασιολογία όλων των πηγών δεδομένων για να ορίσει την καθολική οντολογία. Μπορεί επίσης, να χρησιμοποιηθεί ένας συνδυασμός από αρκετές εξειδικευμένες οντολογίες ανάλογα με την ποικιλία στην περιγραφή των δεδομένων από τις πηγές.
- Αρχιτεκτονική πολλαπλών οντολογιών: Κάθε πηγή δεδομένων αντιπροσωπεύεται από τη δική της τοπική οντολογία. Έτσι χρειάζονται αντιστοιχίσεις μεταξύ αυτών των οντολογιών (εικόνα 2.5β). Τα ερωτήματα στα ολοκληρωμένα δεδομένα γίνονται διαμέσου της εκάστοτε τοπικής οντολογίας και οι αντιστοιχίσεις χρησιμοποιούνται για να εκτελέσουν τοπικά ερωτήματα στις άλλες οντολογίες.

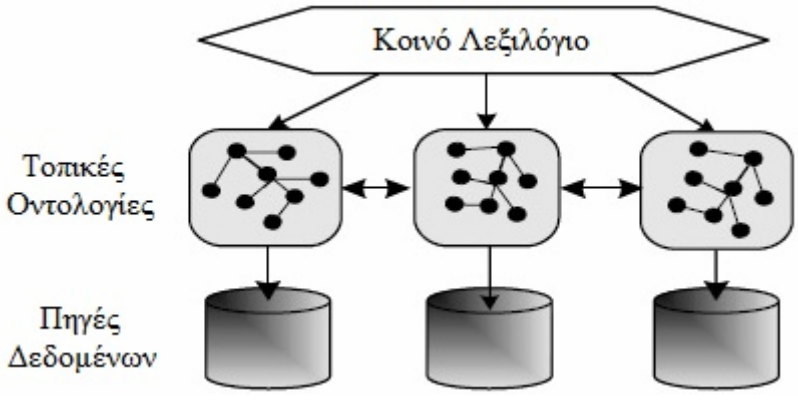
- Υβριδική αρχιτεκτονική: Χρησιμοποιείται για να ξεπεραστούν τα προβλήματα των δύο προηγούμενων προσεγγίσεων. Τα δεδομένα σε κάθε πηγή αντιπροσωπεύονται από μια τοπική οντολογία και ένα κοινό λεξιλόγιο δημιουργείται για να υπάρχει επικοινωνία μεταξύ των οντολογιών αυτών (εικόνα 2.5γ). Αυτή η προσέγγιση έχει σκοπό να εκμεταλλευτεί τα θετικά στοιχεία των δύο πρώτων αρχιτεκτονικών: την ευκολία που δίνει ο τοπικός ορισμός οντολογιών και την υποβολή ερωτημάτων σε αυτές μέσω ενός κοινού λεξιλογίου.



Εικόνα 2.5α: Αρχιτεκτονική Μοναδικής Οντολογίας.



Εικόνα 2.5β: Αρχιτεκτονική Πολλαπλών Οντολογιών.



Εικόνα 2.5γ: Υβριδική Αρχιτεκτονική.

2.3 Η ολοκλήρωση δεδομένων στην ιατροφαρμακευτική περίθαλψη

2.3.1 Πολυπληθείς πηγές δεδομένων και ανοργάνωτα συστήματα πληροφοριών

Η ιατροφαρμακευτική περίθαλψη δεν εκμεταλλεύεται στο έπακρο την υπάρχουσα τεχνολογία πληροφοριακών συστημάτων. Απόδειξη αποτελεί το γεγονός ότι υπάρχουν ακόμα και σήμερα συστήματα υγείας (ένα από αυτά και το ελληνικό), τα οποία στηρίζονται είτε σε χειρόγραφες σημειώσεις (συνταγές φαρμάκων, παραπεμπτικά εξετάσεων, γνωματεύσεις, κ.τ.λ.), είτε σε μεμονωμένα συστήματα αποθήκευσης ιατρικών δεδομένων [96], που είναι όμως αποτελέσματα πρωτοβουλιών των εκάστοτε φορέων υγείας.

Οι χειρόγραφες σημειώσεις παρουσιάζουν έναν σημαντικό αριθμό πλεονεκτημάτων. Είναι ένα απλό σύστημα που δεν απαιτεί ιδιαίτερες γνώσεις, κάνοντάς το κατανοητό και εύχρηστο σε όλους τους εμπλεκόμενους και παρέχει μια σχετική ευκολία στη φυλλομέτρηση (browsing). Ωστόσο, η έλλειψη μηχανοργάνωσης θέτει περιορισμούς στη συγκέντρωση και την επεξεργασία των δεδομένων, ειδικά όταν εμπλέκονται πληροφορίες για έναν ασθενή που προέρχονται από διαφορετικούς ιατρούς. Τα τηρούμενα αρχεία από τους διάφορους παρόχους ιατροφαρμακευτικής περίθαλψης δεν είναι εύκολο να συγκεντρωθούν σε ένα χώρο και να μελετηθούν εύκολα ή/και γρήγορα καθότι απαιτείται αρκετή χειρωνακτική εργασία (μεταφορά φακέλων, έλεγχος των περιεχομένων τους ένα προς ένα κ.τ.λ.)

Μια μερική λύση στα προβλήματα της προηγούμενης λύσης δίνουν τα διάφορα πληροφοριακά συστήματα αποθήκευσης και επεξεργασίας ιατρικών δεδομένων. Όμως, παρά τη μείωση της απαιτούμενης χειρωνακτικής εργασίας για την εισαγωγή, την ταξινόμηση και την αναζήτηση πληροφοριών, εγείρονται νέα προβλήματα. Τα συστήματα που χρησιμοποιούνται είναι κατά βάση αυτόνομα, ετερογενή, και συνήθως έχουν ελάχιστη ή καθόλου συνδεσιμότητα μεταξύ τους καθώς υλοποιούνται σε διαφορετικές πλατφόρμες και ακολουθούν διαφορετικά πρότυπα. Έτσι, όλες οι απαιτούμενες πληροφορίες για τη σύνθεση του πλήρους ιατρικού ιστορικού ενός ατόμου είναι γεωγραφικώς διασκορπισμένες (σε τόσα σημεία, όσα είναι και τα συστήματα) και ασύνδετες αφού είναι μοντελοποιημένες σε ασύμβατες μεταξύ τους μορφές. Τα αποτελέσματα είναι η ύπαρξη του κινδύνου ιατρικών λαθών διάγνωσης και θεραπείας, η διεξαγωγή διπλών εξετάσεων, η έλλειψη συντονισμού και το αυξημένο κόστος θεραπείας.

Για την εξάλειψη των προαναφερθέντων προβλημάτων απαιτείται η ενοποίηση της ιατροφαρμακευτικής περίθαλψης με χρήση των κατάλληλων τεχνολογιών ολοκλήρωσης δεδομένων. Το ζητούμενο είναι η διαλειτουργικότητα μεταξύ των βιοϊατρικών πληροφοριών και των καταχωρημένων δεδομένων ασθενών και ασθενειών. Οι ανασταλτικοί παράγοντες που πρέπει να ξεπεραστούν περιλαμβάνουν το μεγάλο πλήθος τόσο των προτύπων αποθήκευσης και επεξεργασίας ιατρικών δεδομένων, όσο και αυτό των δεδομένων, καθώς λόγω πολυπλοκότητας των κλινικών δεδομένων, εμπλέκονται βάσεις – πηγές με μεγάλο βαθμό ετερογένειας που αναπτύσσονται γρήγορα σε μέγεθος. Έτσι, έννοιες όπως ευελιξία, συντηρησιμότητα

και επεκτασιμότητα γίνονται απαραίτητες προϋποθέσεις ενός σωστού συστήματος ολοκλήρωσης δεδομένων.

2.3.2 Η χρήση οντολογιών στα συστήματα ολοκλήρωσης δεδομένων υγείας

Όπως έχει ήδη αναφερθεί στα προηγούμενα, ένα στοιχείο – κλειδί για την ολοκλήρωση ερευνητικών και κλινικών φορέων είναι η ολοκλήρωση δεδομένων και πηγών που χρησιμοποιούν αυτοί οι φορείς. Πρακτικά, πρέπει να υπάρξουν συνδέσεις μεταξύ των πεδίων (π.χ. μεταξύ γονοτυπικών και φαινοτυπικών [120] πηγών πληροφοριών) αλλά και μεταξύ των αντικειμένων μέσα σε ένα πεδίο (π.χ. μεταξύ γονιδιακών και μακρομοριακών [47] πηγών).

Οι βιοϊατρικές οντολογίες υποστηρίζουν την ολοκλήρωση δεδομένων με δυο διαφορετικούς τρόπους [17]:

- Παρέχοντας σε προσεγγίσεις με αποθήκες δεδομένων ένα λεξιλόγιο το οποίο συμφωνεί με τις προδιαγραφές της τελευταίας και θα χρησιμοποιηθεί για να μετασχηματίσει όλα τα δεδομένα των επιμέρους πηγών σε μια ενιαία μορφή.
- Ως μέρη σε κώδικες μετασχηματισμού (wrapper code) ανάλογων προσεγγίσεων. Οι τελευταίοι ορίζουν ένα καθολικό σχήμα (πάνω στο οποίο γίνονται τα ερωτήματα) και ένα σύνολο αντιστοιχιών μεταξύ αυτού του σχήματος και των υπόλοιπων τοπικών σχημάτων (τα σχήματα των προς ενοποίηση πηγών).

Ανάλογα με το είδος των πεδίων που καλύπτουν, οι βιοϊατρικές οντολογίες μπορούν να χωριστούν σε τρεις κατηγορίες [61]:

- Γενικές ιατρικές οντολογίες: Ασχολούνται με όλο το πεδίο της ιατρικής. Χαρακτηριστικά παραδείγματα: SNOMED CT (SNOMED 2007) [94], GALEN (GALEN, 2007) [36].
- Ειδικές ιατρικές οντολογίες: Περιγράφουν ένα συγκεκριμένο τομέα της ιατρικής. Χαρακτηριστικό παραδείγματα είναι η Foundational Model of Anatomy (FMA, 2007) [33]. Επίσης, ένα σύστημα οντολογιών που αναφέρεται συχνά στην έρευνα για τον καρκίνο είναι το caCORE (NCI CBIIT, 2007) [19], ένα αρκετά εξελιγμένο περιβάλλον που χρησιμοποιεί τις οντολογίες UMLS [102], NCI Thesaurus [74] και NCI Metathesaurus [73] (NCI, 2007).
- Ειδικές βιοϊατρικές οντολογίες: Περιγράφουν ένα συγκεκριμένο τομέα της βιοϊατρικής. Χαρακτηριστικά Παραδείγματα: Gene Ontology (GO, 2007) [38], OBO Foundry (OBO, 2007) [76].

2.3.3 Παραδείγματα συστημάτων ολοκλήρωσης βιοϊατρικών δεδομένων

Στην παρούσα ενότητα θα αναφερθούν ορισμένες λύσεις που έχουν δημιουργηθεί για το επιλύσουν το πρόβλημα της ολοκλήρωσης βιοϊατρικών δεδομένων. Θα παρουσιαστούν οι εμπλεκόμενοι φορείς, τα πεδία εφαρμογής των συστημάτων αυτών καθώς και στοιχεία της αρχιτεκτονικής τους.

2.3.3.1 Health-e-Child Project

Το 2006 ξεκίνησε το Health-e-Child Project με αποστολή να αναπτυχθεί μια ολοκληρωμένη πλατφόρμα περίθαλψης για τους ευρωπαίους παιδίατρους, παρέχοντας ενιαία ολοκλήρωση παραδοσιακών και ανερχόμενων πηγών βιοϊατρικής πληροφορίας. Ο μακροπρόθεσμος στόχος είναι να παρέχεται αδιάκοπη παροχή σε παγκόσμιες αποθήκες βιοϊατρικής γνώσης για εξατομικευμένη και προληπτική περίθαλψη, μεγάλης κλίμακας και βασισμένη σε πληροφορίες βιοϊατρική έρευνα και εκπαίδευση και ενημερωμένη δημιουργία πολιτικής (policy making) [40]. Το όλο εγχείρημα ξεκίνησε από ιατρούς του νοσοκομείου Great Ormond Street του Λονδίνου καθώς και των νοσοκομείων των ινστιτούτων Gaslini στη Τζένοα (Ιταλία), και Necker (Παρίσι) [3]. Συνολικά συμμετείχαν 15 φορείς όπως η Siemens και το CERN. Το έργο ολοκληρώθηκε το 2009, όπου και έδωσε τη σκυτάλη του στο Sim-e-Child [93] που προσπαθεί έως σήμερα να επεκτείνει τις δυνατότητες του προκατόχου του και στην αμερικανική ήπειρο.

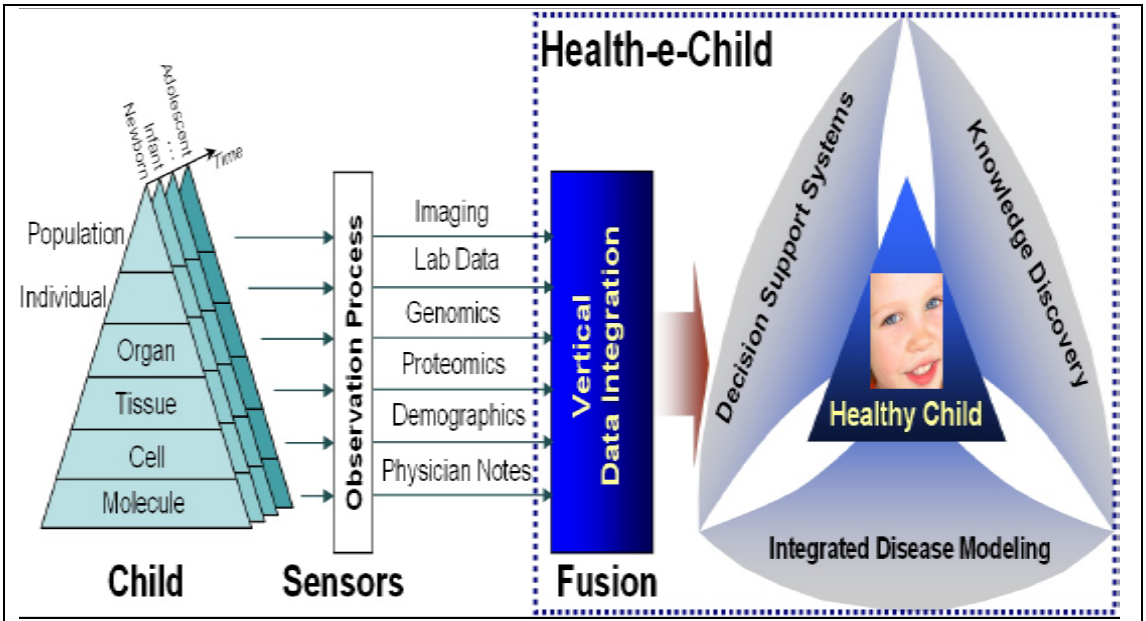
Το σύστημα εφαρμόζει μια ασθενοκεντρική ολοκλήρωση των δεδομένων ακολουθώντας μια διαδικασία που στον τομέα των επιχειρήσεων ονομάζεται «κάθετη ολοκλήρωση» (vertical integration) [13]. Αυτό συμβαίνει γιατί για κάθε ασθενή τα δεδομένα που χρειάζεται να ολοκληρωθούν ανήκουν σε διαφορετικούς τομείς (γενετικές ή/και κλινικές πληροφορίες, επιδημιολογικά στοιχεία, κ.τ.λ.) καθένας από τους οποίους ελέγχεται από διαφορετικά συστήματα. Εργαλεία όπως μοντέλα ασθενειών, αλγόριθμοι εξόρυξης δεδομένων και σύνολα αντιστοιχιών ολοκλήρωσης δεδομένων με οντολογίες βοηθούν την όλη διαδικασία.

Χαρακτηριστικό παράδειγμα του συστήματος είναι η διαδικασία ολοκλήρωσης των πληροφοριών σχετικά με την ασθένεια της παιδικής ιδιοπαθούς αρθρίτιδας (Juvenile Idiopathic Arthritis, JIA) [115], όπου ερωτήματα που σχετίζονται με παράγοντες πρόβλεψης της ασθένειας, μέτρηση και αξιολόγηση της βλάβης που προκαλεί, αποτελεσματικότητας και παρενεργειών των προτεινόμενων θεραπειών, κληρονομικότητα, μελέτη στοιχείων και σύγκριση με την πορεία άλλων ασθενών και τυποποίηση όλων των παραπάνω στοιχείων, βρίσκουν απαντήσεις μέσω της βοήθειας των παρακάτω εργαλείων [3]:

- Αλγόριθμοι ανακάλυψης γνώσης (Knowledge Discovery, KD) που χρησιμοποιούν τεχνικές στατιστικής μάθησης και εξόρυξης δεδομένων για να βρουν μια καλύτερη κατηγοριοποίηση της JIA (ομογενείς ομάδες ασθενών) βασισμένη σε συνδυασμένες αναλύσεις όλων των διαθέσιμων ετερογενών βιοϊατρικών δεδομένων (π.χ. δεδομένα κλινικά, απεικονιστικά, γονοτυπικά, πρωτεϊνωματικά).
- Απεικονιστικές τεχνικές που περιλαμβάνουν ημιαυτόματη κατάτμηση της χονδροπάθειας για την επιτάχυνση της λήψης αποφάσεων σχετικά με τη θεραπεία (ένας χόνδρος με φλεγμονή ίσως να είναι στοιχείο πρόβλεψης της σοβαρότητας της ασθένειας), την ημιαυτόματη αξιολόγηση της βλάβης στην άρθρωση και την καταχώρηση των εικόνων από τις εξετάσεις σε βάθος χρόνου ώστε να μπορεί να γίνει σύγκριση των διαφορετικών μελετών της

ίδιας περίπτωσης ή μεταξύ διαφορετικών περιπτώσεων ασθενών ή ομάδων ασθενών.

- Συστήματα αποφάσεων για εξατομικευμένη αξιολόγηση και θεραπεία παρατηρώντας την εξέλιξη της ασθένειας και του αποτελέσματος της φαρμακευτικής αγωγής, βασισμένη σε διαθέσιμες βιοϊατρικές καταχωρήσεις στοιχείων και προηγούμενη κλινική γνώση.



Εικόνα 2.6: Απεικόνιση της προσέγγισης της «κάθετης» ολοκλήρωσης δεδομένων στο Health-e-Child Project.

Για να υπάρξει ενοποίηση όλων των παραπάνω φορέων ακολουθήθηκε μία «συνένωση» αρκετών ισχυρών βιοϊατρικών οντολογιών, που η κάθε μία πραγματεύονταν ένα συγκεκριμένο αντικείμενο. Χρησιμοποιήθηκαν οντολογίες όπως οι GALEN, Gene Ontology (GO), FMA (Foundation Model of Anatomy), UMLS, ολόκληρες ή μέρη αυτών. Επειδή όμως δεν καλυπτόταν σωστά το πεδίο της παιδιατρικής (π.χ. γιατί κάποιες οντολογίες ορίζουν όρους πάνω σε δεδομένα που προκύπτουν από τη φυσιολογία ενός ενήλικου ατόμου), προτάθηκε και εφαρμόστηκε η εξαγωγή των απαραίτητων όρων και σχέσεων από κάθε οντολογία και η περαιτέρω ενοποίησή τους σε μία οντότητα.

2.3.3.2 Pangea – LE

Το Pangea – LE είναι ένα σύστημα ολοκλήρωσης κλινικών δεδομένων, το οποίο μπορεί να κατηγοριοποιηθεί σαν ένα γενικής χρήσεως μεσισμικό ανάμεσα σε υπάρχοντα ετερογενή συστήματα αποθήκευσης δεδομένων και σε ιατρούς – χρήστες. Τα βασικά χαρακτηριστικά του είναι τα εξής [15]:

- Είναι μη επεμβατικό (non-invasive), που σημαίνει ότι δεν αλλάζει τα αρχικά δεδομένα των πηγών.

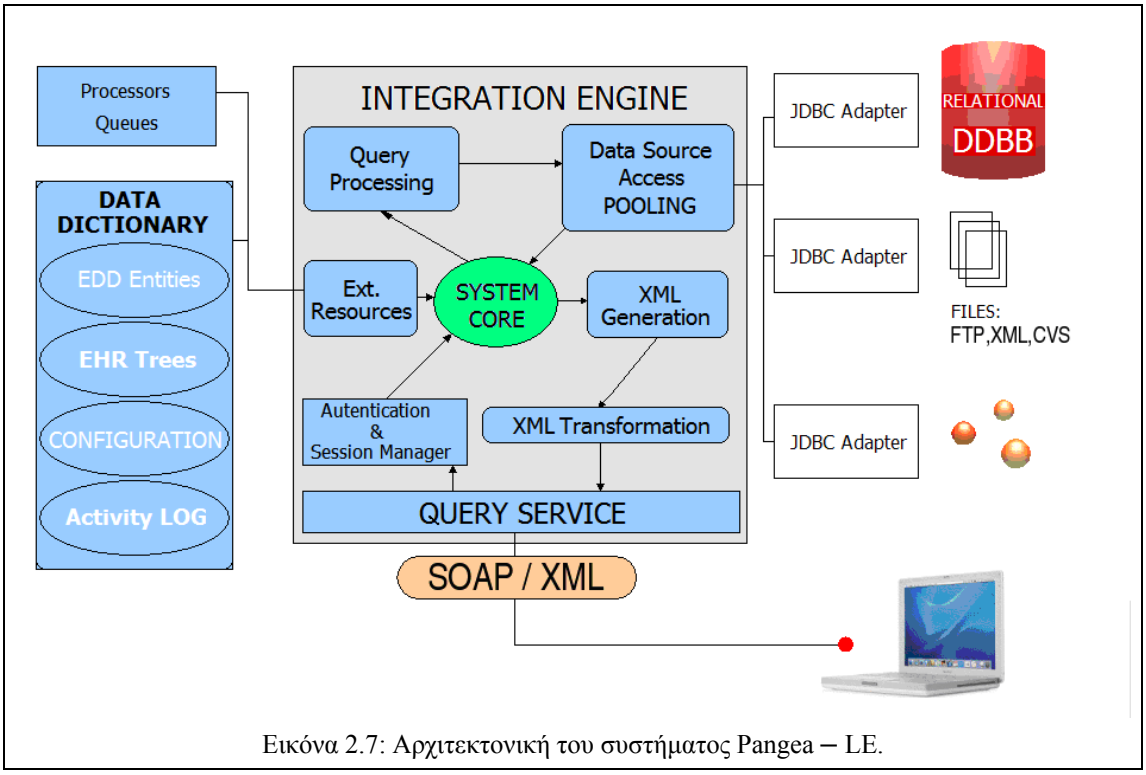
- Είναι «ελαφρύ» (lightweight), καθώς οι τεχνικές απαιτήσεις, η εγκατάσταση και συντήρηση του συστήματος είναι ελάχιστες.
- Είναι προσαρμόσιμο (adaptable), διότι η εισαγωγή νέων πηγών στο σύστημα και τη μορφή εξόδου της πληροφορίας είναι πλήρως ρυθμίσιμα.

Το σύστημα επιτρέπει τον ορισμό και τη διαχείριση μιας καθολικής, ενιαίας και ολοκληρωμένης όψης όλων των κλινικών δεδομένων σχετικά με έναν ασθενή. Αυτή η όψη παρουσιάζεται στον ιατρό – χρήστη μέσω μια εφαρμογής Ιστού σχετική με ηλεκτρονικά αρχεία υγείας (Electronic Health Record (EHR) [43] Web Application). Οι όψεις ετοιμάζονται «επί τόπου» (on the fly) και είναι εξαρτώμενες από ρόλους (role dependant), κάτι που σημαίνει ότι μπορούν να οριστούν διαφορετικά προφίλ (ρόλοι) πρόσβασης στο σύστημα και στα δεδομένα του, ανάλογα με την επαγγελματική ιδιότητα κάθε χρήστη (ιατρικές ειδικότητες, νοσοκόμες, διοικητικά στελέχη, κ.τ.λ.).

Η αρχιτεκτονική του Pangea – LE έχει τέσσερα βασικά στοιχεία [2]:

- Προσαρμογείς: Οι ετερογενείς πηγές δεδομένων προσεγγίζονται μέσα από ένα σύνολο οδηγών JDBC (Java DataBase Connectivity drivers) ή/και μέσα από ένα σύνολο αρχείων ρυθμίσεων για να παραμετροποιήσουν κάθε μία από τις ολοκληρωμένες πηγές.
- Ορισμοί εξαγωγής EHR (EHR Extract Definitions, EED): Πρόκειται για το βασικό συστατικό του Pangea – LE, αφού αυτό περιγράφει τις κλινικές έννοιες που διαμοιράζονται μεταξύ των επιμέρους υποσυστημάτων του συστήματος ολοκλήρωσης. Κάθε πληροφορία που ανταλλάσσεται, οργανώνεται σε μια κλινική οντότητα EED. Διάφορες EED οντότητες μπορούν να οριστούν σε ένα υποσύστημα, μία για κάθε περιγράψιμη έννοια. Κάθε EED οντότητα διαμοιράζεται σαν ένα αδιάσπαστο αντικείμενο. Ουσιαστικά, αποτελεί την ελάχιστη μονάδα πληροφορίας που μπορεί να ανταλλαχθεί μεταξύ δύο υποσυστημάτων. Οι οντότητες EED αρχικοποιούνται από μια υπηρεσία Ιστού του συστήματος (εικόνα 2.7), η οποία δέχεται αιτήσεις σε μορφή XML. Κάθε αίτηση ζητάει ένα συγκεκριμένο είδος EED μαζί με κάποια παράμετρο (π.χ. κωδικό αριθμό ασθενή). Όταν η αίτηση αυτή φτάσει στο σύστημα και εγκριθεί, τότε με κατάλληλες προτάσεις SQL η EED οντότητα συμπληρώνεται με τα ζητούμενα δεδομένα. Κατόπιν εφαρμόζεται μετασχηματισμός XSLT, αν έχει ζητηθεί και τέλος στέλνεται η απόκριση στην αιτούσα εφαρμογή.
- Μετασχηματισμοί XSLT: Κάθε EED οντότητα μπορεί να προσδιορίζει έναν αριθμό αρχείων μετασχηματισμού XSL, τα οποία θα εφαρμοστούν επί του XML μηνύματος απόκρισης. Αυτό επιτρέπει στο Pangea – LE να προσαρμόσει την έξοδό του σε διαφορετικές μορφές μηνυμάτων εφαρμογών, ή άλλες συσκευές (PDAs, Tablet PCs, κ.τ.λ.). Επιπλέον, η ίδια κλινική οντότητα μπορεί να μορφοποιηθεί με διαφορετικό τρόπο γα κάθε ρόλο πρόσβασης χρήστη (user access roles). Οι μετασχηματισμοί μπορούν να εφαρμοστούν είτε στην πλευρά του διακομιστή, είτε στην εφαρμογή του πελάτη, αν αυτή το επιτρέπει.

- Δένδρα φυλλομέτρησης EHR (EHR browsing trees): Το Pangea – LE έχει την ικανότητα να οργανώνει την ανακτηθείσα κλινική πληροφορία σε μια ευέλικτη δομή δεδομένων δένδρου. Όταν η πληροφορία έχει εξαχθεί, διαφορετικές όψεις μπορούν να ρυθμιστούν για να ταξινομήσουν τα EHRs σύμφωνα με όποιο στοιχείο ζητηθεί (ημερομηνία, πηγή πληροφορίας, κ.τ.λ.). Ένα δένδρο EHR είναι επίσης ένα XML έγγραφο, το οποίο ορίζει πώς να δομηθεί, οργανωθεί, συναθροισθεί, συνοψισθεί και καθοδηγηθεί η διαδικασία εξαγωγής της κλινικής πληροφορίας ενός ασθενή. Κατά το χρόνο εκτέλεσης, κάθε EHR δένδρο συνιστά μια όψη ιστορικού υγείας για ένα συγκεκριμένο ρόλο και συντίθεται κυρίως από συνδεδεμένες οντότητες EED. Όταν ένας χρήστης προσπελαύνει την κλινική πληροφορία ενός ασθενή, ανακτάται αρχικά μόνο η ελάχιστη απαιτούμενη πληροφορία για να σχηματιστεί μια συνοπτική όψη του ιστορικού ενός ασθενή. Ακολούθως, ο χρήστης μπορεί να απευθυνθεί σε κάθε κόμβο του δένδρου για να λάβει μια πιο λεπτομερή όψη του κλινικού αντικειμένου που περιγράφεται από τη συνδεδεμένη οντότητα EED. Κάθε δομικό στοιχείο του μοντέλου δεδομένων δένδρου EHR αντιστοιχεί σε κόμβους στον εικονικό έλεγχο EHR δένδρου που χρησιμοποιείται από την εφαρμογή Ιστού εικονοσκόπησης EHR (EHR viewer Web application). Έτσι οι κόμβοι των EHR δένδρων αλληλεπιδρούν με το χρήστη και οι EHR μπορούν να δημιουργούνται κατά παραγγελία. Τα δένδρα φυλλομέτρησης EHR σχεδιάστηκαν μόνο για λόγους οπτικοποίησης και είναι δυνατό να οριστεί ένα διαφορετικό δένδρο για κάθε διαφορετικό ρόλο πρόσβασης.



Εικόνα 2.7: Αρχιτεκτονική του συστήματος Pangea – LE.

2.3.3.3 DebugIT Project

Το DebugIT project (Detecting and Eliminating Bacteria Using Information Technology – Εντοπισμός και Εξάλειψη Βακτηρίων χρησιμοποιώντας Τεχνολογία Πληροφοριών), είναι ένα έργο, το οποίο ξεκίνησε το 2008 και αναπτύσσεται μέχρι και σήμερα [27]. Σ' αυτό συμμετέχουν 14 ευρωπαϊκοί εκπαιδευτικοί και εμπορικοί φορείς με συντονιστή την βελγική AGFA Healthcare [1]. Χρησιμοποιεί δεδομένα από τα υπάρχοντα συστήματα κλινικής πληροφορίας (Clinical Information Systems, CIS) [14] των ευρωπαϊκών νοσοκομείων για να βοηθήσει στην έρευνα κατά των παθογόνων βακτηρίων. Η πρόσβαση σε αυτά τα διαμοιρασμένα και ετερογενή δεδομένα επιτυγχάνεται μέσω ενός εικονικού, πλήρως ολοκληρωμένου χώρου αποθήκευσης κλινικών δεδομένων (Clinical Data Repository, CDR) [89].

Το σύστημα DebugIT παρέχει διαφανή (transparent) πρόσβαση σε μια μοναδική ομογενή όψη των πηγών δεδομένων. Η συγκέντρωση ανεπεξέργαστων δεδομένων (raw data) σε ένα μόνιμο χώρο αποθήκευσης δεν επιτρέπεται λόγω θεμάτων ηθικής και προστασίας προσωπικών δεδομένων. Οι πηγές προσπελαύνονται μέσω SPARQL και τα αποτελέσματα παρουσιάζονται σε μορφή RDF. Για να υλοποιηθούν τα παραπάνω, η αρχιτεκτονική του συστήματος (εικόνα 2.8) αποτελείται από τρία βασικά συστατικά [99]:

- Κώδικες μετασχηματισμού (wrappers), οι οποίοι εξάγουν τα δεδομένα από τα τοπικά CISs, τα μετασχηματίζουν από τα μοντέλα των CISs στο μοντέλο του DebugIT και στη συνέχεια φορτώνουν τα δεδομένα στο τοπικό CDR. Με τα δεδομένα ήδη αποθηκευμένα, εκτελούνται ακόμα δύο διεργασίες:
  - Κανονικοποίηση (normalization) δεδομένων με χρήση οντολογιών.
  - Μετατροπή δεδομένων από το «μοντέλο εισαγωγής» (input model) στο «μοντέλο διεπαφής» (interface model) του τοπικού CDR.

  Οι κώδικες μετασχηματισμού αναπτύσσονται με το πακέτο Talend [98].
- Τοπικά CDRs, τα οποία είναι βάσεις δεδομένων MySQL που αποθηκεύουν δεδομένα DebugIT που εξήχθησαν από το CIS και αποτελούν διεπαφές ανάμεσα στους παρόχους και το σύστημα. Τα δεδομένα που αποθηκεύονται σ' αυτά μπορούν να ελεγχθούν ως προς την εγκυρότητά τους (validated) ενώ μπορεί να προστεθούν σ' αυτά σημειώσεις (annotations). Βασίζονται σε συστήματα διαχείρισης βάσεων δεδομένων για λόγους ευρωστίας και επεκτασιμότητας.

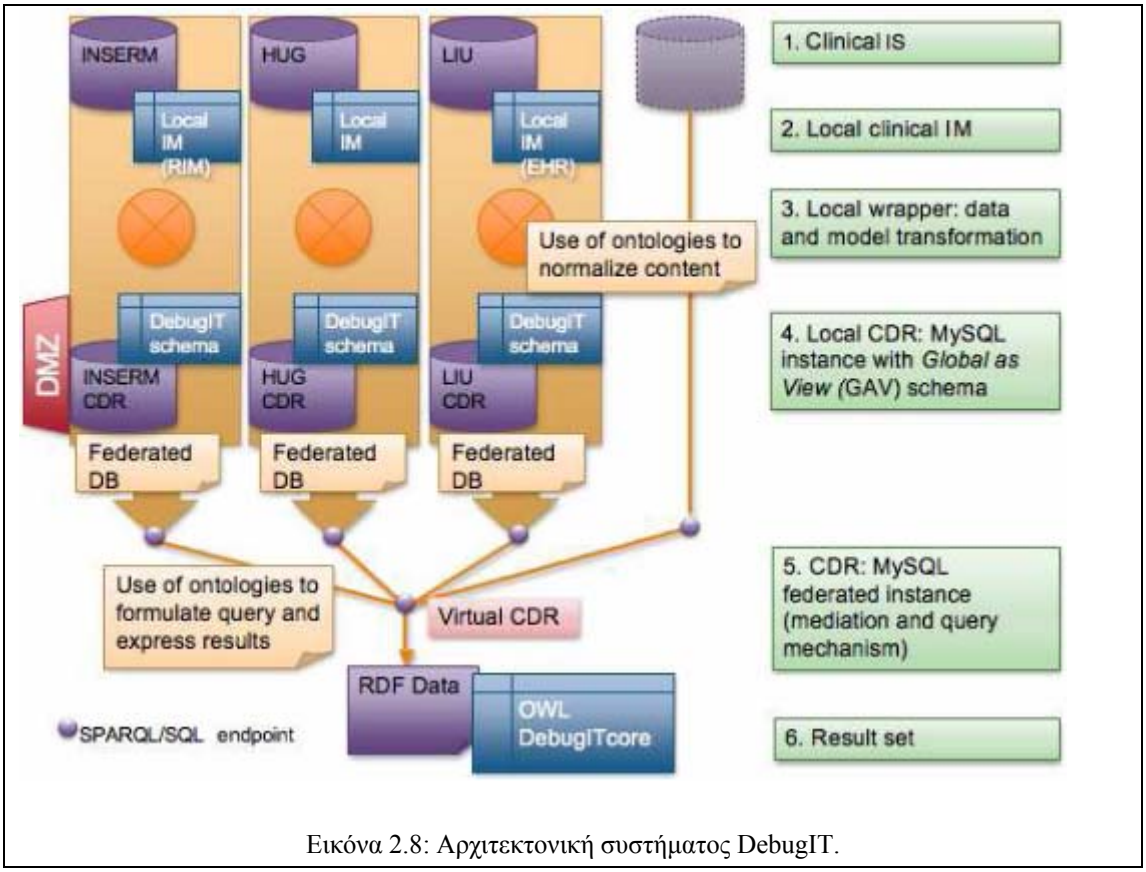
  Κάθε τοπικό CDR περιέχει δύο ξεχωριστά σχήματα: ένα σχήμα EAV/CR (Entity – Attribute – Value with Classes and Relationships, Οντότητα – Χαρακτηριστικό – Τιμή με Κλάσεις και Σχέσεις) [117] που χρησιμοποιείται σαν σχήμα εισαγωγής για τα δεδομένα που εξάγονται από το CIS και ένα άλλο, προσαρμοσμένο για μετασχηματισμένα δεδομένα DebugIT. Το EAV/CR βρίσκεται μεταξύ του CIS και του CDR, ώστε να μην χρειάζονται αλλαγές εκατέρωθεν όταν μια πλευρά τροποποιείται. Το προσαρμοσμένο σχήμα συμπληρώνεται με τα κανονικοποιημένα δεδομένα που προηγουμένως ήταν αποθηκευμένα στο EAV/CR σχήμα. Αποτελεί την διεπαφή ερωτημάτων

του συστήματος και επιλύει τα προβλήματα απόδοσης ερωτημάτων (query performance) και εκφραστικότητας του EAV/CR.

Τα τοπικά CDRs χρησιμοποιούνται για τις παρακάτω εργασίες:

- Κανονικοποίηση των περιεχομένων του CDR.
- Κατασκευή καταλόγων δεδομένων και μοντέλου πληροφορίας.
- Πρόχειρη σύνταξη (draft) της οντολογίας πυρήνα του DebugIT.
- Εφαρμογή μεθόδων εξόρυξης δεδομένων (data mining).
- Πρόχειρη σύνταξη περιπτώσεων χρήσης τελικών σημείων (endpoint use cases).

- Μια MySQL ομόσπονδη βάση δεδομένων (MySQL federated database instance), η οποία αποτελεί τον πυρήνα του συστήματος. Συνδέει όλες τις πηγές δεδομένων για να δημιουργήσει μια καθολική όψη πάνω σε αυτές. Έτσι, καθιστά ικανή την πρόσβαση στα δεδομένα DebugIT μέσα από ένα μοναδικό σημείο ερωτημάτων (query point) χωρίς να αποθηκεύει κεντρικά τα δεδομένα. Το CDR αυτό καθώς και τα τοπικά CDRs «περιτυλίγονται» (wrapped) από υπηρεσίες Ιστού (Web services) με χρήση της πλατφόρμας D2R ώστε να παρέχουν τελικά σημεία SPARQL.



Εικόνα 2.8: Αρχιτεκτονική συστήματος DebugIT.

Το σύστημα χρησιμοποιεί ένα καθολικό σχήμα, ακολουθώντας έτσι μια αρχιτεκτονική στενής σύζευξης. Το σχήμα αυτό μπορεί να φιλοξενήσει όλα τα δεδομένα σχετικά με το έργο και τις σχέσεις τους.

Η αρχιτεκτονική του συστήματος έχει χαρακτηριστικά αποθηκών δεδομένων και ολοκλήρωσης με όψεις. Ο μετασχηματισμός δεδομένων και το κοινό σχήμα είναι χαρακτηριστικά των αποθηκών δεδομένων, ενώ η ανυπαρξία κεντρικής αποθήκευσης είναι το βασικό χαρακτηριστικό ολοκλήρωσης με όψεις. Επίσης παρουσιάζονται στο σύστημα πλεονεκτήματα και μειονεκτήματα και των δύο μεθόδων. Τα πλεονεκτήματα είναι η υψηλή απόδοση (αποθήκες) και η μη έκθεση δεδομένων (όψεις). Τα μειονεκτήματα είναι το κόστος της ενημέρωσης των μοντέλων (αποθήκες και όψεις) και ο συγχρονισμός των δεδομένων (αποθήκες).

ΚΕΦΑΛΑΙΟ 3:
ΤΕΧΝΟΛΟΓΙΕΣ ΚΑΙ ΠΡΟΤΥΠΑ

3.1 Πρότυπα διαχείρισης ιατρικών δεδομένων

Στις παρακάτω υποενότητες θα παρουσιαστούν τα δύο πρότυπα κωδικοποίησης, αποθήκευσης και ανταλλαγής ιατρικών δεδομένων, τα DICOM [28] και HL7 [42] που χρησιμοποιούνται στην εφαρμογή IntegraHEALTH 1.0 και θα γίνει μια απλή αναφορά σε μερικά άλλα τα οποία έχουν δημοσιευθεί.

Όπως αναφέρθηκε και στην εισαγωγή, υπήρχαν δύο λόγοι για την επιλογή των προτύπων. Ο πρώτος λόγος έχει να κάνει με το γεγονός ότι και τα δύο πρότυπα τυγχάνουν ευρείας αποδοχής. Στον Health Level 7 Organization συμμετέχουν εκτός από κατασκευαστές ιατρικών συστημάτων όπως η Interfaceware [50], και επιχειρήσεις – κολοσσοί στο χώρο της ανάπτυξης λογισμικού ή/και μηχανημάτων όπως η Microsoft, η General Electric και η Philips με τα αντίστοιχα τμήματά τους που ασχολούνται με συστήματα ιατροφαρμακευτικής περίθαλψης [42]. Επίσης, σύμφωνα με τα στοιχεία της National Electrical Manufacturers Association (NEMA) [75], της ένωσης που ανέπτυξε το πρότυπο DICOM, το αντίστοιχο τμήμα αντιπροσωπεύει εταιρείες των οποίων οι πωλήσεις αποτελούν πάνω από το 90% της παγκόσμιας αγοράς ιατρικών απεικονιστικών συστημάτων.

Ο δεύτερος λόγος για την επιλογή αυτών των δύο προτύπων είναι η αλληλοκάλυψη που παρέχουν όσον αφορά τη συνολική διαχείριση ενός παρόχου ιατροφαρμακευτικής περίθαλψης, όπως για παράδειγμα ενός νοσοκομείου. Σύμφωνα με το [11], κανένα από τα δύο πρότυπα δεν μπορεί να διαχειριστεί από μόνο του ένα νοσοκομείο. Το πρότυπο DICOM μπορεί να επιτύχει τη διαλειτουργικότητα μεταξύ των απεικονιστικών μηχανημάτων και των διατάξεων πυρηνικής ιατρικής, αλλά τα δεδομένα του δεν αναγνωρίζονται από τα συστήματα διαχείρισης, τα οποία (συνήθως) συμμορφώνονται με το πρότυπο HL7. Αντίστοιχα, τα δεδομένα που ανταλλάσσονται με το πρότυπο HL7 δεν είναι αναγνώσιμα από τα ιατρικά απεικονιστικά μηχανήματα. Συνεπώς για την απρόσκοπτη λειτουργία ενός οργανισμού ιατροφαρμακευτικής περίθαλψης απαιτείται η συνεργασία των δύο προτύπων. Αυτό έχει γίνει ευρέως αντιληπτό, σε σημείο που το μεν πρότυπο DICOM να έχει εκδώσει το μέρος 3.20 του προτύπου με τίτλο DICOM “Transformation of DICOM to and from HL7 Standards” (Μετασχηματισμός του DICOM από και προς τα πρότυπα του HL7) [75], η δε κοινότητα επαγγελματιών υγείας και κατασκευαστών να έχουν ξεκινήσει το έργο IHE (Integrating the HealthCare Enterprise) [49], το οποίο έχει στόχο να προωθήσει τη συντονισμένη χρήση των δύο προτύπων, επιτυγχάνοντας ένα υψηλότερο επίπεδο ολοκλήρωσης μεταξύ τους.

3.1.1 Το πρότυπο DICOM

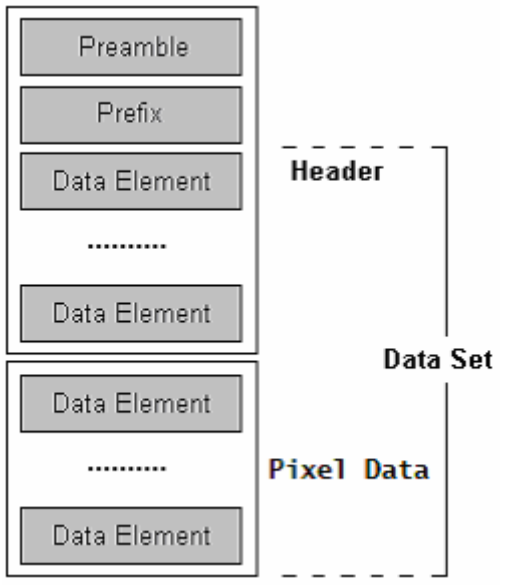
Το πρότυπο DICOM (Digital Imaging and Communications in Medicine) είναι ένα πρότυπο για χειρισμό, αποθήκευση, εκτύπωση και μετάδοση πληροφορίας για

εφαρμογές ιατρικών απεικονίσεων. Περιλαμβάνει έναν ορισμό τύπου αρχείου και ένα πρωτόκολλο επικοινωνίας δικτύου που χρησιμοποιεί το TCP/IP για να επικοινωνούν τα διάφορα συστήματα μεταξύ τους. Αναπτύχθηκε από την επιτροπή προτύπων DICOM, η οποία είναι μέλος της NEMA.

Το DICOM παρέχει τη δυνατότητα ενοποίησης σαρωτών, διακομιστών, σταθμών εργασίας, εκτυπωτών και υλικού δικτύου από πολλαπλούς κατασκευαστές. Έχει ήδη υιοθετηθεί ευρέως από νοσοκομεία και βρίσκει το δρόμο του προς μικρότερες εφαρμογές όπως γραφεία ιατρών (κυρίως ακτινολόγων) και οδοντιάτρων [68].

3.1.2 Δομή αρχείου DICOM

Ο τύπος αρχείου DICOM ομαδοποιεί τις πληροφορίες σε σύνολα δεδομένων (data sets) [69]. Ένα αρχείο ακτινογραφίας λ.χ., μπορεί να περιέχει το ονοματεπώνυμο ή τον προσωπικό κωδικό του ασθενή μέσα στις πληροφορίες του.



Εικόνα 3.1: Βασική Δομή ενός αρχείου DICOM.

Ένα αρχείο DICOM αποτελείται από την επικεφαλίδα, που περιέχει το πρόθεμα και έναν αριθμό αντικειμένων, όπως όνομα, κωδικός, κ.τ.λ., και από ένα ειδικό αντικείμενο που περιέχει τα δεδομένα εικονοστοιχείων (pixel data) της εικόνας [21]. Ένα μοναδικό αρχείο DICOM μπορεί να περιέχει μόνο ένα αντικείμενο με δεδομένα εικονοστοιχείων. Το αντικείμενο αυτό περιέχει είτε μία εικόνα είτε πολλαπλά πλαίσια (frames, «καρέ») επιτρέποντας την αποθήκευση κινούμενης εικόνας μερικών δευτερολέπτων (animation) ή τρισδιάστατων δεδομένων [69]. Τα δεδομένα εικονοστοιχείων μπορούν να συμπιεστούν χρησιμοποιώντας μια ποικιλία προτύπων με ή χωρίς απώλειες (JPEG [56], JPEG Lossless [57], Run-Length Encoding (RLE) [85], κ.τ.λ.).

Τα αρχεία DICOM έχουν κατάληξη ".dcm" και η ονοματολογία των αρχείων ορίζεται ρητά στο μέρος 3.10 του προτύπου [72].

3.1.2.1 Η επικεφαλίδα DICOM (DICOM header)

Σε ένα αρχείο DICOM τα αντικείμενα που δεν περιέχουν δεδομένα εικονοστοιχείων συγκροτούν την επικεφαλίδα του αρχείου. Βρίσκεται στην αρχή της ακολουθίας bit και αποτελείται από τα εξής μέρη [21, 70]:

- Ένα εισαγωγικό πεδίο μήκους 128 byte. Συνήθως στα byte αυτά δίνεται η τιμή δεκαεξαδικού μηδέν (00H), αν το πεδίο δεν χρησιμοποιείται από ένα προφίλ εφαρμογής, ή από μια συγκεκριμένη υλοποίηση του προτύπου.

- Ένα πρόθεμα τεσσάρων byte, τα οποία περιέχει τη συμβολοσειρά 'DICM'.
- Μια σειρά από στοιχεία δεδομένων, καθένα από τα οποία περιέχει μια πληροφορία για την εικόνα που έπεται. Οι πληροφορίες που μπορούν να φέρουν καθορίζονται ρητά από το μέρος 3.6 του προτύπου [71]. Υπάρχει η δυνατότητα κωδικοποίησης πάνω από 3000 χαρακτηριστικών για την εικόνα που μπορεί να φέρει ένα αρχείο.

Στο Παράρτημα Α, παρουσιάζονται παραδείγματα δεδομένων επικεφαλίδας αρχείων DICOM.

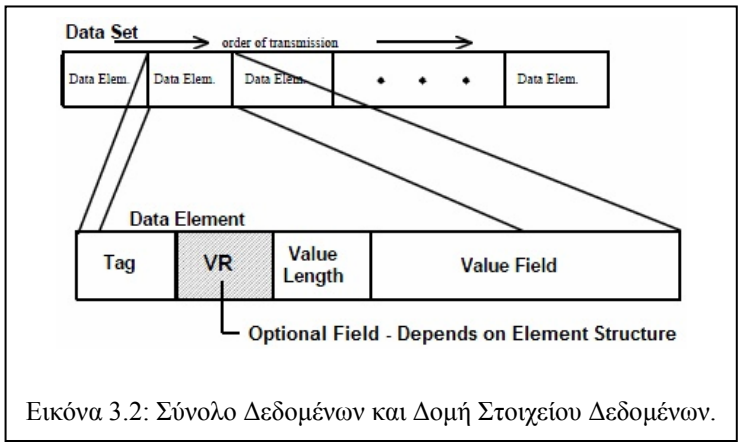
3.1.2.2 Το σύνολο δεδομένων DICOM (DICOM data set)

Ένα σύνολο δεδομένων αναπαριστά μία υπόσταση (instance) ενός αντικειμένου πληροφορίας του πραγματικού κόσμου. Απαρτίζεται από στοιχεία δεδομένων, τα οποία περιέχουν τις κωδικοποιημένες τιμές ή χαρακτηριστικά αυτού του αντικειμένου [70]. Στο πρότυπο DICOM σαν σύνολο δεδομένων μπορεί να οριστεί ολόκληρο το αρχείο πλην των δύο πρώτων πεδίων της επικεφαλίδας (εισαγωγικό πεδίο και πρόθεμα) (εικόνα 3.1).

3.1.2.3 Το στοιχείο δεδομένων DICOM (DICOM data element)

Ένα στοιχείο δεδομένων είναι η πρώτη υποδιαίρεση ενός συνόλου δεδομένων και αποτελείται από πεδία (εικόνα 3.2). Οι ορισμοί των επιμέρους πεδίων ενός στοιχείου δεδομένων είναι οι εξής [70]:

- Ετικέτα στοιχείου δεδομένων (Element Tag): Ένα ζεύγος ακεραίων 16 bit χωρίς πρόσημο, μοναδικό για κάθε είδος πληροφορίας που μπορεί να αποθηκευθεί. Το ζεύγος που αντιστοιχεί σε κάθε είδος ορίζεται ρητά στο μέρος 3.6 του προτύπου [71].
- Αναπαράσταση τιμής (Value Representation, VR): Μία συμβολοσειρά δύο χαρακτήρων που περιέχουν την αναπαράσταση τιμής του στοιχείου. Για μια συγκεκριμένη ετικέτα δεδομένων, η αναπαράσταση ορίζεται από το λεξικό δεδομένων [71].
- Μήκος τιμής (Value Length): Ένας ακέραιος 16 ή 32 bit χωρίς πρόσημο που περιέχει το μήκος του πεδίου τιμής σε bytes (πάντα ζυγός αριθμός).
- Πεδίο τιμής (Value Field): Πρόκειται για ένα ζυγό αριθμό από bytes που περιέχει τα δεδομένα του στοιχείου. Ο τύπος των δεδο-



Εικόνα 3.2: Σύνολο Δεδομένων και Δομή Στοιχείου Δεδομένων.

μένων ορίζεται από την αναπαράσταση τιμής του στοιχείου.

- Στο [71] ορίζεται και η πολλαπλότητα τιμής (Value Multiplicity, VM) που αναφέρει πόσα δεδομένα με αναπαράσταση τιμής VR μπορούν να τοποθετηθούν στο πεδίο τιμής. Σε περίπτωση ύπαρξης πολλαπλών τιμών, οι χαρακτήρες οριοθέτησης ορίζονται από την ενότητα 6.4 του μέρους 3.5 του προτύπου [70].

3.1.2.4 Δεδομένα εικονοστοιχείων (Pixel data)

Στο τελευταίο κομμάτι του αρχείου αποθηκεύονται τα δεδομένα εικονοστοιχείων, τα οποία αντιπροσωπεύουν την εικόνα που περιέχεται σ' αυτό. Κάθε κελί με δεδομένα εικονοστοιχείου περιέχει μια μοναδική τιμή δείγματος εικονοστοιχείου (pixel sample value), η οποία πρέπει να είναι ένας δυαδικός ακέραιος, χωρίς πρόσημο ή με μορφή συμπληρώματος ως προς 2, και προσδιορίζεται από τα εξής στοιχεία δεδομένων [70]:

- Δεσμευμένα bit (bits allocated): Καθορίζει το μέγεθος του κελιού.
- Αποθηκευμένα bit (bits stored): Ορίζει τον συνολικό αριθμό από τα δεσμευμένα bit που θα χρησιμοποιηθούν για να αναπαραστήσουν μία τιμή δείγματος εικονοστοιχείου.
- Bit υψηλότερης τάξης (high bit): Καθορίζει σε ποια θέση θα τοποθετηθεί το μεγιστοτάξιο bit της ακολουθίας των αποθηκευμένων bit.

Το μέρος 3.3 του προτύπου [69] ορίζει τους περιορισμούς, οι οποίοι αφορούν τα παραπάνω στοιχεία. Το πεδίο τιμής ενός στοιχείου δεδομένων που θα περιέχει δεδομένα εικονοστοιχείου, όπως και όλα τα άλλα πεδία τιμών στο DICOM, πρέπει να έχει μήκος ζυγό αριθμό από bytes. Αυτό σημαίνει ότι ίσως χρειαστεί να συμπληρωθεί με δεδομένα που δεν είναι μέρος της εικόνας και δεν θα θεωρηθούν σαν σημαντικά ψηφία (padding).

3.1.3 Το πρότυπο Health Level 7 (HL7)

Το πρότυπο HL7 είναι ένα σαφώς καθορισμένο για τον τομέα εφαρμογής του (domain-specific) κοινό πρότυπο ανταλλαγής δεδομένων υγειονομικής φροντίδας το οποίο στηρίζεται σε ένα σύστημα δοσοληψιών οι οποίες γίνονται με μηνύματα (HL7 messages) τα οποία κωδικοποιούνται κατά ASCII. Έτσι είναι αναγνώσιμα τόσο από υπολογιστές (machine-readable) όσο και από ανθρώπους (human-readable) [23]. Λόγω του μικρού μεγέθους τους αποθηκεύονται και μεταφέρονται εύκολα χρησιμοποιώντας μια ποικιλία διαφορετικών τρόπων: ανταλλαγή αρχείων απλού κειμένου, e-mail, τεχνικές store and forward, χρήση «μεσιτών» μηνυμάτων (message brokers), ενθυλάκωση μηνύματος σε πακέτα πρωτοκόλλων μετάδοσης δεδομένων όπως το TCP/IP.

Δημιουργός του προτύπου είναι ο ομώνυμος οργανισμός Health Level Seven International [42], ο οποίος δημιουργήθηκε το 1987 και είναι μέλος του American National Standards Institute (ANSI) [4]. Η ονομασία HL7 αναφέρεται στο έβδομο στρώμα της διαστρωμάτωσης κατά OSI (Open Systems Interconnection) [119] για

δίκτυα. Δείχνει ότι το πρότυπο επικεντρώνεται στα πρωτόκολλα του στρώματος εφαρμογής ενώ τα χαμηλότερα στρώματα θεωρούνται ως εργαλεία που βοηθούν στο έργο του.

Μέχρι στιγμής έχουν δημοσιευθεί τρεις εκδόσεις του προτύπου. Η έκδοση 1, που δεν έτυχε επαρκούς υποστήριξης, μένοντας έτσι «στα χαρτιά», η σειρά εκδόσεων 2.X, η οποία σταδιακά άρχισε να γίνεται αποδεκτή και χρησιμοποιείται έως και σήμερα, και η έκδοση 3, η οποία χρησιμοποιεί σύνταξη βασισμένη στην XML [23].

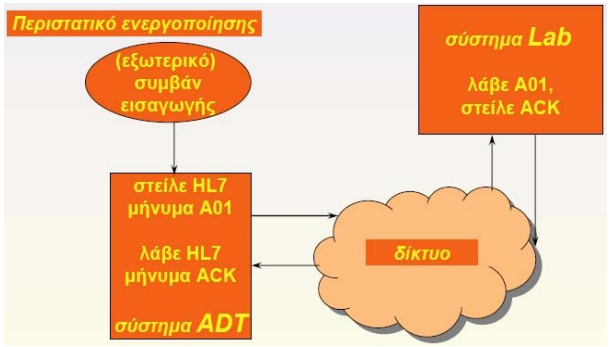
3.1.3.1 HL7 v2.X

Η σειρά εκδόσεων 2.X του προτύπου ξεκίνησε με την έκδοση 2.1 το 1990. Αρχικά, δεν έτυχε ευρείας αποδοχής, όπως και η έκδοση 1, αν και ήταν πληρέστερη και χρησιμοποιήσιμη. Μετά το 1998 δημιουργήθηκαν συστήματα που να υποστηρίζουν τη σειρά, αρχής γενομένης με την έκδοση 2.3.1 [23]. Η πιο πρόσφατη έκδοση της σειράς είναι η 2.7 (2011) [42]

Όλα οι εκδόσεις της σειράς v2.X έχουν συμβατότητα προς τα πίσω (backward compatibility). Αυτό σημαίνει ότι ένα μήνυμα που συντάχθηκε με την έκδοση 2.3 λ.χ. θα είναι αναγνώσιμο από μια εφαρμογή που υποστηρίζει την έκδοση 2.6 [23].

Σύνθεση μηνυμάτων HL7 v2.X

Πριν αναλυθούν οι κανόνες σύνθεσης ενός μηνύματος HL7 αξίζει να αναφερθεί το βασικό «μοντέλο δοσοληψίας», το οποίο καθορίζει τη σύνταξη και την αποστολή τους. Ένα περιστατικό (π.χ. εισαγωγή ασθενούς) ενεργοποιεί τον μηχανισμό, ο οποίος το κωδικοποιεί με ένα κατάλληλο μήνυμα το οποίο και αποστέλλει στο δίκτυο. Οι ενδιαφερόμενοι φορείς απαντούν με μηνύματα επιβεβαίωσης (Acknowledgement, ACK) και επιπλέον μηνύματα για πιθανά παραγόμενα συμβάντα από το αρχικό περιστατικό (π.χ. εντολές εξετάσεων) [123].



Εικόνα 3.3: Βασικό «μοντέλο δοσοληψίας» HL7.

Κάθε μήνυμα HL7 2.X κατηγοριοποιείται με βάση το αναγνωριστικό είδους του. Αυτό καθορίζεται από ένα αλφαριθμητικό έξι χαρακτήρων μοναδικό για κάθε είδος. Στο Παράρτημα Β δίνεται ένας πίνακας με τους πιο διαδεδομένους τύπους μηνυμάτων.

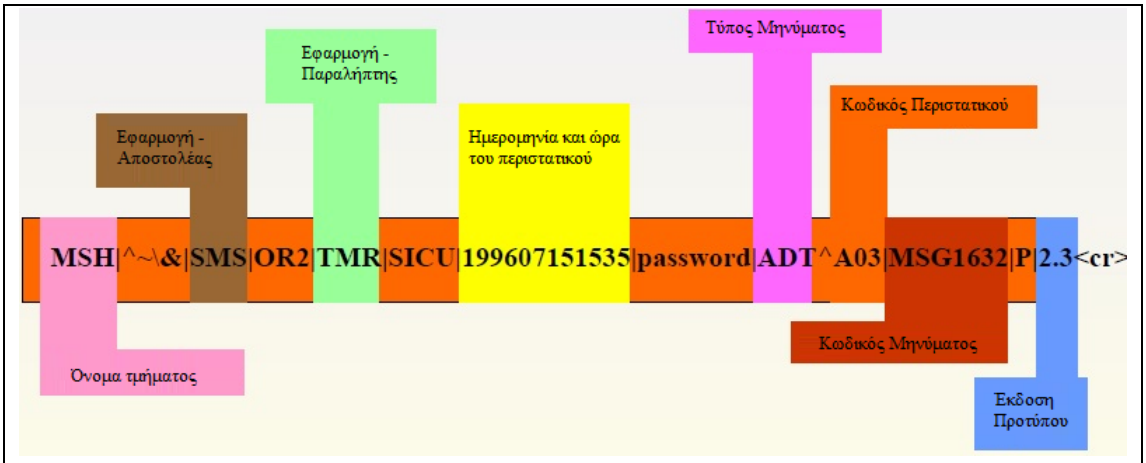
Τμήματα (Segments)

Ένα τμήμα είναι η πρώτη υποδιαίρεση ενός μηνύματος HL7. Ανάλογα με το είδος του μηνύματος που θα αποσταλεί, το πρότυπο ορίζει το είδος και το πλήθος των τμημάτων που απαιτούνται ή μπορούν προαιρετικά να συμπεριληφθούν σε αυτό [62].

Σε ένα μήνυμα κάθε τμήμα γράφεται στη δικιά του γραμμή και ξεκινά με το όνομά του. Αποτελείται από μία συμβολοσειρά τριών χαρακτήρων η οποία ορίζει το είδος πληροφορίας που φέρει το τμήμα (εικόνα 3.4). Στο Παράρτημα Β δίνεται πίνακας με τα συνηθέστερα ονόματα τμημάτων που χρησιμοποιούνται.

Πεδία (Fields)

Τα πεδία είναι τα κύρια συστατικά των τμημάτων. Βάσει του ονόματος ενός τμήματος, που είναι το πρώτο πεδίο του, καθορίζεται από το πρότυπο πόσα πεδία θα περιλαμβάνει καθώς και τι τύπους δεδομένων θα δέχονται αυτά τα πεδία. Ο διαχωρισμών μεταξύ των πεδίων σε ένα τμήμα γίνεται με τον οριοθέτη «|» [62, 123].



Εικόνα 3.4: Τα επιμέρους πεδία ενός τμήματος MSH.

Συστατικά – υποσυστατικά (Components – subcomponents)

Όταν ένας – σύνθετος συνήθως – τύπος δεδομένων ενός πεδίου το ορίζει, μπορεί τα δεδομένα του πεδίου να περιλαμβάνουν παραπάνω από μία συμβολοσειρές, είτε είναι χαρακτήρες (λέξεις), είτε αριθμοί (ακέραιοι, κινητής υποδιαστολής κ.τ.λ.). Τότε κάθε συμβολοσειρά αποτελεί ένα συστατικό του πεδίου. Τα συστατικά χωρίζονται μεταξύ τους από τον οριοθέτη «^» [62, 123]. Στην εικόνα 3.4 φαίνεται ότι το όνομα του μηνύματος (ADTA03) στο οποίο ανήκει το τμήμα χωρίζεται σε δύο συστατικά.

Η ίδια λογική εφαρμόζεται όταν ο ένας από τους υποτύπους ενός σύνθετου τύπου δεδομένων είναι και αυτός σύνθετος. Τότε, ένα συστατικό (του οποίου τα δεδομένα εκφράζονται από τον υποτύπο) χωρίζεται σε υποσυστατικά. Η μόνη διαφορά εδώ είναι ο χαρακτήρας οριοθέτησης, ο οποίος είναι ο «&».

Τύποι δεδομένων (Data types)

Η σειρά προτύπων 2.X παρέχει μια ποικιλία τύπων για την αναπαράσταση κάθε είδους δεδομένων. Σε αυτήν περιλαμβάνονται απλοί τύποι όπως αυτοί των αλφαριθμητικών, ημερομηνίας/ώρας και αριθμητικών τιμών, αλλά και σύνθετοι τύποι, οι οποίοι αποτελούνται από σύνολα υποτύπων, καθένας από τους οποίους μπορεί να έχει περαιτέρω υποτύπους κ.ο.κ. μέχρι να προκύψουν σύνολα από απλούς τύπους [62, 123]. Η δομή αυτή των σύνθετων τύπων δημιουργεί την ανάγκη του ορισμού συστατικών και υποσυστατικών μέσα σε ένα πεδίο. Στο Παράρτημα Β δίνεται πίνακας με τους συνηθέστερους τύπους που χρησιμοποιούνται.

3.1.4 Άλλα πρότυπα

Αν και για τους σκοπούς της παρούσας διπλωματικής εργασίας χρησιμοποιούνται μόνο τα πρότυπα DICOM και HL7, καθώς πρόκειται για τα 2 δημοφιλέστερα παγκοσμίως (στους εκάστοτε τομείς τους), υπάρχουν αρκετά άλλα, τα οποία είναι εξίσου αποδεκτά από κατασκευαστές ανά τον κόσμο. Στην παρούσα ενότητα παρουσιάζονται συνοπτικά τα πιο γνωστά από αυτά.

3.1.4.1 OpenEHR

Το openEHR [77] είναι ένα σύνολο προδιαγραφών ενός ανοιχτού προτύπου το οποίο περιγράφει τη διαχείριση, αποθήκευση, ανάκτηση και ανταλλαγή ιατρικών δεδομένων προερχόμενα από ηλεκτρονικά αρχεία υγείας (electronic health records, EHRs). Τα ηλεκτρονικά αρχεία υγείας αποτελούν μια έννοια, η οποία εκφράζει τη μεταφορά των δεδομένων ενός ασθενή από το χαρτί σε ηλεκτρονική μορφή, κατάλληλη και για μεταφορά. Ο στόχος του openEHR είναι η δημιουργία προσωποκεντρικών και «διαχρονικών» ηλεκτρονικών αρχείων υγείας, όπου όλα τα δεδομένα από όλη τη ζωή του ασθενή θα αποθηκεύονται σε ένα τέτοιο αρχείο. Αυτό επιτυγχάνεται αφήνοντας τις προδιαγραφές των κλινικών πληροφοριών έξω από το μοντέλο πληροφοριών. Οι προδιαγραφές αυτές συγκροτούνται σε «αρχέτυπα», στα οποία καλούνται εκ των υστέρων να συμμορφωθούν τα δεδομένα των openEHR συστημάτων. Το πρότυπο δεν ασχολείται με την ανταλλαγή των ηλεκτρονικών αρχείων υγείας αυτών καθαυτών, καθώς αποτελούν πεδίο δράσης προτύπων όπως το HL7 και το EN13606.

3.1.4.2 EN13606

Το ευρωπαϊκό πρότυπο Health informatics – Electronic Health Record Communication που φέρει την κωδική ονομασία EN13606 [100] έχει σαν στόχο να ορίσει μια αυστηρή και σταθερή αρχιτεκτονική πληροφορίας για την ανταλλαγή μέρους ή όλων των ηλεκτρονικών αρχείων υγείας ενός ατόμου, βοηθώντας στη διαλειτουργικότητα των σχετικών συστημάτων. Αυτό γίνεται με το να διατηρείται το

αρχικό κλινικό νόημα των δεδομένων που τους έδωσε ο συντάκτης καθώς και ο βαθμός εμπιστευτικότητας τους.

Η τεχνική προσέγγιση του προτύπου λαμβάνει υπόψιν τη χρήση αντικειμενο-στραφούς τεχνολογίας και υπηρεσιών Ιστού για να υλοποιηθεί το ηλεκτρονικό αρχείο υγείας σαν ένα κομμάτι μεσισμικού (διακομιστής EHR), «πελάτες» του οποίου θα είναι όχι μόνο συστήματα EHR αλλά και άλλες υπηρεσίες μεσισμικού όπως στοιχεία ασφαλείας, συστήματα ροής εργασίας και συστήματα αποφάσεων. Επίσης θεωρείται σημαντική η αντιστοίχιση με το HL7 v3, ώστε η συμμόρφωση στο EN13606 να προσδιορίζεται μέσα από ένα σχετικό περιβάλλον HL7.

3.1.4.3 CDISC - SDTM

Ο Clinical Data Interchange Standards Consortium (CDISC) [20] είναι ένας μη κερδοσκοπικός οργανισμός ο οποίος έχει δημιουργήσει αρκετά πρότυπα για τη διαλειτουργικότητα των συστημάτων υγείας, τα οποία ορίζει ως μια σειρά μοντέλων που εκφράζονται συνήθως σε XML. Το πιο διαδεδομένο από αυτά είναι το Μοντέλο Πινακοποίησης Δεδομένων Μελετών (Study Data Tabulation Model, SDTM), το οποίο είναι χτισμένο πάνω στην ιδέα της περιγραφής ενός συνόλου παρατηρήσεων για ένα άτομο που συμμετείχε σε μια έρευνα λ.χ., σαν ένα σύνολο μεταβλητών, όπου η κάθε μία αντιστοιχίζεται σε μια στήλη ενός πίνακα ή ενός συνόλου δεδομένων. Κάθε μεταβλητή εμπίπτει σε μία από πέντε κατηγορίες [116]:

- Μεταβλητές ταυτότητας: Ταυτοποιούν την μελέτη, τα θέματα της παρατήρησης, το πεδίο και τον αριθμό ακολουθίας της καταγραφής.
- Μεταβλητές θέματος: Ορίζουν το σημείο εστίασης της παρατήρησης.
- Μεταβλητές χρονισμού: Περιγράφουν το χρονισμό της παρατήρησης (ώρα έναρξης/λήξης, διάρκεια κ.τ.λ.).
- Μεταβλητές προσδιορισμού: Οτιδήποτε πρόσθετο (κείμενο, αριθμητικές τιμές κ.τ.λ.) που προσδιορίζει περαιτέρω τα αποτελέσματα ή τα πρόσθετα χαρακτηριστικά μιας παρατήρησης.
- Μεταβλητές κανόνα: Εκφράζουν έναν αλγόριθμο ή μια εκτελέσιμη μέθοδο που ορίζει την αρχή, το τέλος ή τις συνθήκες επανάληψης του μοντέλου σχεδίασης της δοκιμής.

Μετά το σχηματισμό τους, οι πίνακες μπορούν να υποβληθούν σε άλλες εφαρμογές, εξασφαλίζοντας έτσι την ανταλλαγή των δεδομένων.

3.2 Τεχνολογίες Σημασιολογικού Ιστού

3.2.1 Ο Σημασιολογικός Ιστός

Αν και στη σημερινή του μορφή, ο Παγκόσμιος Ιστός (World Wide Web – WWW) αποτελεί το κατεξοχήν εργαλείο για επικοινωνία μεταξύ ανθρώπων, διεξαγωγή επιχειρηματικών δραστηριοτήτων και ανταλλαγή δεδομένων, παρουσιάζει κάποια προβλήματα, κυρίως στο πεδίο της αναζήτησης πληροφοριών.

Τα βασικά εργαλεία σχεδόν για κάθε δραστηριότητα στον Ιστό είναι οι υπερσύνδεσμοι μεταξύ εγγράφων (hyperlinks) και οι μηχανές αναζήτησης (search engines). Στις τελευταίες έχουν εντοπιστεί πρακτικά προβλήματα [5] όπως:

- Υψηλή ανάκληση με χαμηλή ακρίβεια: Η αναζήτηση ενός όρου δίνει εκτός από τις σχετικές με αυτόν σελίδες και ένα σύνολο από μη σχετικές, οι οποίες απλά τυγχάνει να τον περιέχουν στα κείμενά τους, συχνά με διαφορετική σημασία από αυτή που ζητείται από το χρήστη.
- Χαμηλή ή καθόλου ανάκληση: Μπορεί να μην υπάρξει απάντηση σε κάποιο ερώτημα σχετικά με έναν όρο, ή ο αριθμός των απαντήσεων να είναι πολύ μικρός για να εξαχθούν οι επιθυμητές πληροφορίες.
- Τα αποτελέσματα είναι ιδιαίτερα ευαίσθητα στο λεξιλόγιο: Η χαμηλή ή μηδαμινή ανάκληση για μια λέξη – κλειδί μπορεί να οδηγήσει τη μηχανή αναζήτησης σε χρήση ενός συνωνύμου της δοθείσας λέξης, η οποία μπορεί να επιστρέψει στον χρήστη τα αποτελέσματα που επιθυμεί. Κάτι τέτοιο όμως δεν είναι ικανοποιητικό, γιατί τα σημασιολογικά παρόμοια ερωτήματα θα πρέπει να επιστρέφουν παρόμοια αποτελέσματα.
- Τα αποτελέσματα είναι μεμονωμένες ιστοσελίδες: Ένα από τα πιο συνηθισμένα προβλήματα είναι η περίπτωση κατά την οποία η ζητούμενη πληροφορία βρίσκεται κατακερματισμένη σε διάφορά έγγραφα. Τότε είναι απαραίτητη η ανάκτηση όλων των εγγράφων και η «χειροκίνητη» επεξεργασία τους ώστε να εξαχθεί από το καθένα το ζητούμενο κομμάτι πληροφορίας. Επίσης με μη αυτόματο τρόπο πρέπει να γίνει και η σύνθεση των επιμέρους κομματιών σε ένα νέο, ενιαίο έγγραφο, αν αυτό απαιτείται.

Η κύρια αιτία των προβλημάτων αυτών είναι ότι το νόημα του περιεχομένου του Ιστού δεν είναι προς το παρόν αναγνώσιμο από υπολογιστές (machine-readable) καθώς οι δυνατότητες του υπάρχοντος λογισμικού για ερμηνεία προτάσεων και εξαγωγή χρήσιμων πληροφοριών για τους χρήστες είναι ακόμα πολύ περιορισμένες.

Η λύση που προτείνεται από το World Wide Web Consortium (W3C) είναι η αναπαράσταση του δικτυακού περιεχομένου σε μορφή που να είναι επεξεργάσιμη από υπολογιστές (machine-processable) και η χρήση έξυπνων τεχνικών για την εκμετάλλευση αυτών των αναπαραστάσεων. Η διαδικασία αυτή θα εξελίξει τον Παγκόσμιο Ιστό σε Σημασιολογικό Ιστό (Semantic Web). Όμως για να αποκτήσουν «κατανόηση» οι μηχανές μέσα από τις διαθέσιμες αποθήκες πληροφορίας, απαιτείται μια συστηματική δόμηση των περιεχομένων τους. Αυτή είναι και η μεγαλύτερη πρόκληση σήμερα, καθώς ο τεράστιος Παγκόσμιος Ιστός και τα δεδομένα που περιέχει δεν μπορούν να αναδομηθούν εκ θεμελίων μέσα σε μικρό χρονικό διάστημα.

3.2.1.1 Αρχιτεκτονική και ιεραρχία του Σημασιολογικού Ιστού.

Στην ενότητα 2.2.1 αναφέρθηκαν τρεις γλώσσες του Σημασιολογικού Ιστού: η RDF, η RDF Schema (RDFS) και η OWL. Η ανάγκη για τη χρήση πολλαπλών γλωσσών για την περιγραφή δεδομένων προκύπτει από την αρχιτεκτονική του Ιστού. Στην αρχιτεκτονική αυτή μπορεί να παρατηρηθεί ότι ο Σημασιολογικός Ιστός αποτελείται από στρώματα (layers) [121]. Κάθε στρώμα διαθέτει τις δικές του

λειτουργίες, επεκτείνοντας την τεχνολογία των κατώτερων στρωμάτων. Σε αυτή την «στοίβα» επιπέδων, οι χαμηλότερες θέσεις αφορούν λειτουργίες που σχετίζονται περισσότερο με αυτές του σημερινού Παγκόσμιου Ιστού, ενώ στις ανώτερες συναντώνται περισσότερο πολύπλοκες αναπαραστάσεις γνώσης καθώς και λειτουργίες εξαγωγής συμπερασμάτων και παραγωγής νέας γνώσης (συλλογιστική), κάτι το οποίο πλησιάζει στην ανθρώπινη σκέψη και γνώση. Μπορούν να διακριθούν τα παρακάτω επίπεδα:

- Επίπεδο μεταδεδομένων: Στο επίπεδο αυτό εισάγεται μια βασική γλώσσα αναπαράστασης γνώσης, η οποία επιτρέπει μόνο τη δημιουργία προτάσεων σχετικά με στοιχεία του διαδικτύου. Κάθε στοιχείο αναφέρεται και ως πόρος (resource) του διαδικτύου. Για κάθε πόρο ορίζονται ιδιότητες (properties) που μοντελοποιούν τα χαρακτηριστικά του, τα οποία δίνονται ως τιμές σε αυτές τις ιδιότητες. Η γλώσσα που υλοποιεί το επίπεδο αυτό είναι η RDF.
- Επίπεδο σχήματος: Έχοντας υπόψιν τα βασικά στοιχεία μοντελοποίησης του προηγούμενου επιπέδου, εδώ υλοποιούνται έννοιες όπως αυτή της κλάσης (class) και καθορίζεται η ιεραρχία μεταξύ κλάσεων και ιδιοτήτων όπως γίνεται και στις γλώσσες προγραμματισμού. Η γλώσσα η οποία υλοποιεί το επίπεδο αυτό είναι η γλώσσα RDFS (RDF Schema).
- Λογικό επίπεδο: Στο επίπεδο αυτό υλοποιούνται περισσότερο εκφραστικές γλώσσες αναπαράστασης γνώσης. Οι γλώσσες αυτές χρησιμοποιούν και επεκτείνουν τη λειτουργικότητα του επιπέδου σχήματος παρέχοντας περισσότερες εκφραστικές δυνατότητες. Η γλώσσα η οποία υλοποιεί τη λειτουργικότητα του επιπέδου αυτού είναι η OWL.
- Επίπεδο κανόνων: Στο επίπεδο αυτό η λειτουργικότητα των γλωσσών του λογικού επιπέδου επεκτείνεται ακόμη περισσότερο παρέχοντας τη δυνατότητα καταγραφής κανόνων, οδηγώντας σε αυτό που ονομάζεται συλλογιστική (reasoning). Μέσω της συλλογιστικής με εφαρμογή κανόνων προκύπτει νέα γνώση ως προς τα ήδη περιγραφόμενα δεδομένα, οπότε γίνεται δυνατός ο χειρισμός ακόμα πιο πολύπλοκων ερωτημάτων.

3.2.2 Resource Description Framework (RDF)

3.2.2.1 Βασικές έννοιες

Η RDF είναι μια γλώσσα η οποία χρησιμοποιείται για την απλή περιγραφή πόρων (resources) του διαδικτύου αποδίδοντας μεταπληροφορία σχετικά με τα χαρακτηριστικά τους. Ως πόρος αναφέρεται οποιαδήποτε οντότητα του Παγκόσμιου Ιστού, ενώ ο όρος μπορεί να προσδιορίσει και αντικείμενα τα οποία δεν είναι άμεσα διαθέσιμα στο διαδίκτυο, όπως είναι για παράδειγμα ένα βιβλίο.

Η βασική ιδέα πίσω από την RDF είναι ότι τα προς περιγραφή αντικείμενα (πόροι) αποτελούνται από ένα σύνολο χαρακτηριστικών (ιδιότητες – properties) που η κάθε μία έχει συγκεκριμένη τιμή. Έτσι μια μεταπληροφορία για ένα πόρο αποτελείται από μια ιδιότητα και την τιμή που έχει ο πόρος για την ιδιότητα αυτή.

Μια έκφραση για έναν πόρο ονομάζεται RDF πρόταση (statement). Μια πρόταση αποτελείται από μια τριάδα (triple) ενός υποκειμένου (subject), μιας ιδιότητας/κατηγορήματος (property/predicate) και ενός αντικειμένου (object). Τη θέση του υποκειμένου καταλαμβάνει ο πόρος, τη θέση της ιδιότητας η ιδιότητα που του αποδίδεται, ενώ τη θέση του αντικειμένου η τιμή της ιδιότητας. Η τιμή αυτή μπορεί να είναι κάποιος άλλος πόρος ή κάποια τιμή δεδομένων (λεκτικό – literal). Συντακτικά οι προτάσεις αυτές δηλώνονται από μια διατεταγμένη τριάδα της μορφής, <S, P, O>. όπου τα S, P και O αντιπροσωπεύουν το υποκείμενο, την ιδιότητα (κατηγόρημα) και το αντικείμενο, αντίστοιχα [121].

Από τα παραπάνω μπορεί να γίνει αντιληπτό ότι η RDF, αν και αποκαλείται γλώσσα, στην ουσία είναι ένα μοντέλο δεδομένων, όπου το βασικό δομικό στοιχείο είναι οι προτάσεις. Ένα τέτοιο αφηρημένο μοντέλο χρειάζεται μια καθορισμένη σύνταξη ώστε να αναπαρίσταται και να μεταδίδεται [5], και η RDF διαθέτει μια τέτοια σύνταξη σε XML, ονόματι RDF/XML.

Για λόγους γενικότητας η RDF χρησιμοποιεί αναφορές URI (URI references, URIrefs) για να προσδιορίσει τις οντότητες οι οποίες βρίσκονται στη θέση του υποκειμένου, της ιδιότητας και του αντικειμένου. Μια αναφορά URI αποτελείται από ένα URI (Uniform Resource Identifier, Ενιαίο Αναγνωριστικό Πόρων) και από ένα προαιρετικό αναγνωριστικό τμήματος (fragment identifier). Για παράδειγμα η αναφορά http://www.example.org/index.html#section2 αποτελείται από το URI http://www.example.org/index.html και από το αναγνωριστικό τμήματος section2 το οποίο διαχωρίζεται από το URI με ένα σύμβολο «#».

3.2.2.2 Τρόποι αναπαράστασης μιας πρότασης

Έστω η πρόταση: «Η ιστοσελίδα http://www.example.org/index.html έχει δημιουργό τον John Smith». Σύμφωνα με την μοντελοποίηση RDF αντιστοιχίζεται σε μια τριάδα υποκειμένου – κατηγορήματος – αντικειμένου ως εξής [5]:

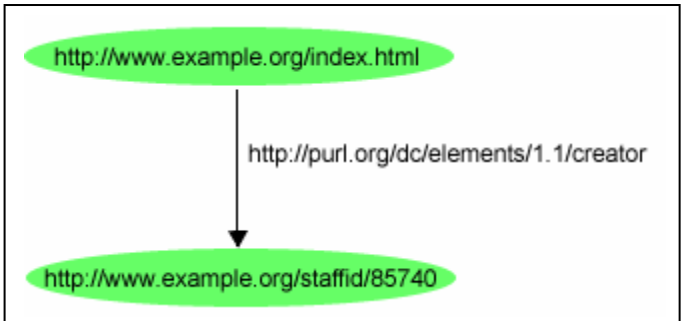
- Ένα υποκείμενο: http://www.example.org/index.html.
- Μια ιδιότητα (κατηγόρημα): http://purl.org/dc/elements/1.1/creator.
- Ένα αντικείμενο: http://www.example.org/staffid/85740 (όπου θεωρούμε ότι το id 85740 είναι ο μοναδικός κωδικός του John Smith).

Οι δημοφιλέστερες μορφές αναπαράστασης της παραπάνω πρότασης, αλλά και κάθε άλλης πρότασης είναι τρεις [109, 121]:

- Αναπαράσταση με μορφή τριάδας: Τα δεδομένα γράφονται στη μορφή <S, P, O> τοποθετώντας την κάθε αναφορά URI σε αγκύλες, τη μία δίπλα στην άλλη. Η πρόταση κλείνει με μία τελεία. Το αρχικό παράδειγμα γράφεται ως εξής: <http://.../index.html> <http://.../creator> <http://.../85740>. Για λόγους συντόμευσης, χρησιμοποιείται συνήθως μια μορφή γραφής, που βασίζεται σε κατάλληλα ονόματα XML (XML qualified names, QNames [113]). Τα ονόματα αυτά αποτελούνται από ένα πρόθεμα, το οποίο αντιστοιχίζεται σε ένα URI, μια διπλή τελεία (:) και το όνομα του πόρου. Επομένως, ορίζοντας τα προθέματα ex, dc, exstaffid για τα URIs http://www.example.org/,

http://purl.org/dc/elements/1.1/ και http://www.example.org/staffid/ αντίστοιχα, η αρχική τριάδα ξαναγράφεται ως: ex:index.html dc:creator exstaff:85740.

- Αναπαράσταση με μορφή γράφου: Σε αυτήν την περίπτωση σχηματίζεται ένας διατεταγμένος γράφος, του οποίου οι κόμβοι αποτελούν το υποκείμενο και το αντικείμενο της πρότασης, ενώ η ιδιότητα απεικονίζεται με ένα τόξο με κατεύθυνση από το υποκείμενο προς το αντικείμενο.



Εικόνα 3.5: Γραφική αναπαράσταση πρότασης RDF.

- Αναπαράσταση σε μορφή XML: Εδώ το αρχικό παράδειγμα κωδικοποιείται σαν ένα έγγραφο XML. Στην ενότητα 3.2.2.5 γίνεται περαιτέρω ανάλυση της μορφής αυτής και δίνεται σχετικό παράδειγμα.

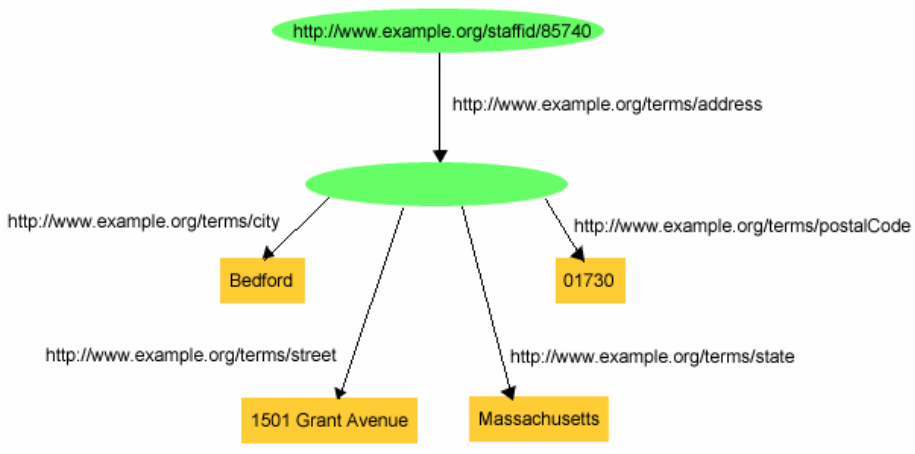
3.2.2.3 Σημασιολογικά δίκτυα – Δομημένες τιμές ιδιοτήτων – Κενοί κόμβοι

Τα περισσότερα δεδομένα στον πραγματικό κόσμο αποτελούνται από αρκετά πολύπλοκες δομές, ώστε να μην φτάνει μία μόνο πρόταση για την αναπαράστασή τους. Για παράδειγμα, η διεύθυνση ενός ατόμου μπορεί να γραφτεί ολόκληρη σαν ένα απλό λεκτικό (plain literal) και να γίνει αντικείμενο μιας πρότασης. Ωστόσο, αν μια εφαρμογή χρειάζεται τα επιμέρους στοιχεία της διεύθυνσης, τότε αυτή πρέπει να αναπαρασταθεί σαν μια δομή που περιλαμβάνει οδό, αριθμό, πόλη κ.τ.λ.

Τέτοιου είδους δομημένη πληροφορία εκφράζεται με σημασιολογικά δίκτυα. Σε αυτά τα δίκτυα, αναπαρίσταται ένα σύνολο προτάσεων σχετικά με πόρους, σχηματίζοντας έτσι σύνθετους γράφους. Η πιο απλή περίπτωση σημασιολογικού δικτύου είναι αυτή στην οποία ένας πόρος έχει πολλαπλές ιδιότητες, όπου η τιμή της κάθε μίας είναι ένα λεκτικό [5].

Αυτού του είδους η αναπαράσταση δομημένης πληροφορίας μπορεί να εμπλέξει την δημιουργία ενδιάμεσων κόμβων που αντιστοιχίζονται σε συγκεντρωτικές έννοιες, όπως η διεύθυνση ενός ατόμου. Τέτοιοι κόμβοι σπανίως αναζητούνται από ερωτήματα χρηστών, οπότε δεν είναι απαραίτητο να έχουν τη δική τους αναφορά URI. Αποτελούν δηλαδή κενούς κόμβους (blank nodes). Ο μοναδικός σκοπός ενός κενού κόμβου είναι να εξασφαλίσει τη συνοχή του γράφου στον οποίον περιέχεται. Οι κενοί κόμβοι ονομάζονται επίσης και ανώνυμοι πόροι (anonymous resources) [109].

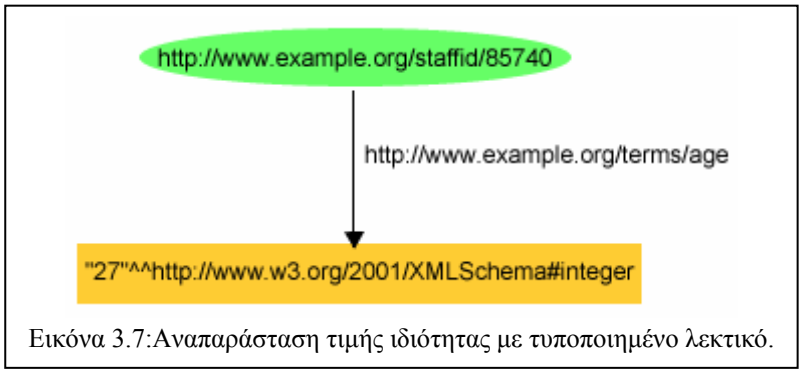
Για την περίπτωση αναπαράστασης σε τριάδες, χρησιμοποιούνται τα αναγνωριστικά κενού κόμβου (blank node identifiers), τα οποία έχουν τη μορφή _:name, για να καταδείξουν την ύπαρξη ενός κενού κόμβου. Από την αρχική πρόταση μπορεί να προκύψει η τριάδα _:x exterms:city "Bedford", όπου _:x η αυθαίρετη ονομασία του κενού κόμβου [109, 121].



Εικόνα 3.6: Σημασιολογικό δίκτυο αναπαράστασης δομημένης πληροφορίας με χρήση κενού κόμβου.

3.2.2.4 Τύποι δεδομένων και λεκτικά (literals)

Έως τώρα, όλες οι σταθερές τιμές που χρησιμοποιούνται σαν αντικείμενα σε προτάσεις RDF έχουν αναπαρασταθεί από απλά (μη τυποποιημένα) λεκτικά, ακόμα και όταν η τιμή της ιδιότητας επιβάλλεται να είναι αριθμός (π.χ. η τιμή μιας ιδιότητας «ηλικία») ή κάποιο άλλο είδος πιο εξειδικευμένης τιμής. Αν και δεν επηρεάζει την αναγνωσιμότητα της πληροφορίας από τον άνθρωπο, εφαρμογές που είναι σχεδιασμένες να αναγνώσουν δεδομένα συγκεκριμένων τύπων ίσως εμφανίσουν προβλήματα ερμηνείας του γράφου [109].



Εικόνα 3.7:Αναπαράσταση τιμής ιδιότητας με τυποποιημένο λεκτικό.

Στην RDF τα τυποποιημένα λεκτικά (typed literals) χρησιμοποιούνται για να παρέχουν τέτοιου είδους πληροφορίες. Ένα τυποποιημένο λεκτικό σχηματίζεται από μία συμβολοσειρά που συνοδεύεται από μια αναφορά URI που αντιστοιχεί στον επιθυμητό τύπο δεδομένων. Τα σύμβολα «^^» χρησιμοποιούνται ως διαχωριστής του ζεύγους [5, 121]. Για παράδειγμα, η περιγραφή της ηλικίας ενός ατόμου μπορεί να γραφτεί: <http://www.example.org/staffid/85740> <http://www.example.org/terms/age> "27"^^<http://www.w3.org/2001/XMLSchema#integer>, ή χρησιμοποιώντας QNames: exstaff:85740 exterms:age "27"^^xsd:integer, ή με έναν γράφο, όπως στην εικόνα 3.7.

3.2.2.5 RDF/XML:Η σύνταξη κατά XML της RDF

Στην ενότητα 3.2.2.2 αναφέρθηκε ότι η σύνταξη κατά XML είναι ένας από τους κύριους τρόπους αναπαράστασης μιας πρότασης και κατ' επέκταση ενός ολόκληρου εγγράφου RDF. Εδώ θα αναλυθεί το παρακάτω παράδειγμα ως προς τα στοιχεία της σύνταξής του, σύμφωνα με το [109]:

```
1. <?xml version="1.0"?>
2. <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.          xmlns:dc="http://purl.org/dc/elements/1.1/">
4.    <rdf:Description rdf:about="http://www.example.org/index.html">
5.       <dc:creator rdf:resource="http://www.example.org/staffid/85740"/>
6.    </rdf:Description>
7. </rdf:RDF>
```

Παράδειγμα 3.1: Έγγραφο RDF/XML.

Η γραμμή 1 αποτελεί τη δήλωση XML, που δείχνει ότι το ακόλουθο περιεχόμενο είναι σε XML και την έκδοση της γλώσσας που χρησιμοποιείται.

Η γραμμή 2 ξεκινά με το στοιχείο (element) rdf:RDF. Αυτό δείχνει ότι το ακόλουθο περιεχόμενο XML (που ξεκινά από αυτή τη γραμμή και λήγει στην γραμμή 7 με την ετικέτα κλεισίματος </rdf:RDF>) θα αναπαραστήσει πληροφορία μοντελοποιημένη σε RDF. Το στοιχείο που ακολουθεί στην ίδια γραμμή, είναι μια δήλωση χώρου ονομάτων XML που έχει τη μορφή xmlns χαρακτηριστικού της ετικέτας rdf:RDF. Αυτή η δήλωση ορίζει ότι όλες οι ετικέτες του εγγράφου που αρχίζουν με το πρόθεμα rdf: είναι μέρος του χώρου ονομάτων που αναφέρεται από το URI http://www.w3.org/1999/02/22-rdf-syntax-ns#. Δηλαδή, όλο το λεξιλόγιο της RDF ορίζεται από αναφορές URI που αρχίζουν με αυτή τη διεύθυνση .

Η γραμμή 3 περιέχει ακόμα μια δήλωση χώρου ονομάτων XML με το πρόθεμα dc:. Κάθε χώρος ονομάτων που ορίζεται στο έγγραφο εκτός από το χώρο της RDF, ονομάζεται εξωτερικός χώρος ονομάτων. Κάθε εξωτερικός χώρος ονομάτων υποτίθεται ότι είναι κι αυτός ένα έγγραφο RDF που ορίζει πόρους, οι οποίοι με τη σειρά τους χρησιμοποιούνται στο έγγραφο RDF που τον εισάγει.

Οι γραμμές 4 – 6 παρουσιάζουν μια πρόταση RDF. Στην RDF/XML μια πρόταση εκφράζεται μέσω της περιγραφής του υποκειμένου της. Στην ετικέτα αρχής της γραμμής 4 βρίσκεται το στοιχείο rdf:Description, το οποίο δηλώνει ότι αυτό που ακολουθεί είναι μια περιγραφή πόρου. Ο πόρος προσδιορίζεται από το χαρακτηριστικό rdf:about, το οποίο έχει για τιμή την αναφορά URI του πόρου. Η γραμμή 5 περιλαμβάνει ένα στοιχείο ιδιότητας, εμφωλευμένο στο rdf:Description, με το QName dc:creator σαν ετικέτα για να αναπαραστήσει το κατηγόρημα (ιδιότητα) της πρότασης. Το αντικείμενο της πρότασης τυγχάνει να είναι πόρος, οπότε γράφεται σαν ένα χαρακτηριστικό του στοιχείου ιδιότητας, το οποίο περιγράφεται με μια ετικέτα κενού στοιχείου (empty-element tag). Όταν το αντικείμενο είναι λεκτικό, τότε γράφεται ανάμεσα σε ξεχωριστές ετικέτες αρχής και τέλους, π.χ. <dc:creator>John Smith</dc:creator>

Χαρακτηριστικά περιγραφής πόρων

Στην προηγούμενη ενότητα εξηγήθηκε πως περιγράφεται ένας πόρος σε ένα RDF έγγραφο μέσω του rdf:Description. Οι προς περιγραφή πόροι εμφανίζονται σαν τιμές κάποιου από τα παρακάτω χαρακτηριστικά [109]:

- rdf:about: Δέχεται σαν τιμή την πλήρη αναφορά URI ενός πόρου. Γενικά σε περιγραφές πόρων η RDF/XML απαγορεύει τη χρήση QNames.
- rdf:ID: Υποδηλώνει τη χρήση τμηματικού αναγνωριστικού αντί για την πλήρη αναφορά URI ενός πόρου, η οποία προκύπτει από τη συνένωση του αναγνωριστικού με κάποιο βασικό URI.
- rdf:nodeID: Χρησιμοποιείται για να δηλώσει ότι ο πόρος είναι ένας κενός κόμβος. Δέχεται ως τιμή ένα αυθαίρετο αναγνωριστικό. Επίσης μπορεί να χρησιμοποιηθεί αντί του rdf:resource σε στοιχείο ιδιότητας, αν η τιμή της είναι κενός κόμβος.
- rdf:resource: Πρόκειται για ειδική περίπτωση χαρακτηριστικού, το οποίο χρησιμοποιείται σε στοιχεία ιδιοτήτων για να εκφράσει πόρους σαν αντικείμενα προτάσεων. Οι τιμές που μπορεί να πάρει, είναι αυτές που ορίζονται στα στοιχεία rdf:about και rdf:ID. Αν εκφράζεται πόρος που περιγράφεται από αναγνωριστικό χαρακτηριστικού rdf:ID, τότε πριν από το αναγνωριστικό πρέπει να μπαίνει ο χαρακτήρας «#»[5].

Χαρακτηριστικά περιγραφής τύπων κλάσεων και δεδομένων

Η έννοια της κλάσης ως ομάδα αντικειμένων με κοινές ιδιότητες ή χαρακτηριστικά, είναι γνωστή από την πληροφορική. Ένα αντικείμενο μπορεί να είναι στιγμιότυπο (instance) μιας κλάσης αν πληροί τις απαιτούμενες ιδιότητές της. Τότε λέγεται ότι ανήκει στον τύπο (type) που ορίζει η κλάση αυτή.

Η RDF/XML υποστηρίζει αυτόν τον μηχανισμό παρέχοντας το χαρακτηριστικό rdf:type. Πρόκειται για μια ιδιότητα με τιμή της έναν πόρο, ο οποίος αναπαριστά μια κλάση αντικειμένων. Σε μια πρόταση, το υποκείμενο της ιδιότητας αυτής αποτελεί στιγμιότυπο της κλάσης – τιμής της ιδιότητας [121].

Παρόμοια μπορεί να προσδιοριστεί και ένας τύπος δεδομένων στον οποίο ανήκει ένα τυποποιημένο λεκτικό. Αυτό γίνεται με το χαρακτηριστικό rdf:datatype που συνοδεύει την ετικέτα αρχής ενός στοιχείου ιδιότητας [109]. Π.χ. <foaf:firstName rdf:datatype="http://www.w3.org/2001/XMLSchema#String">Nick</foaf:firstName>

Στοιχεία – υποδοχείς και λίστες

Τα στοιχεία – υποδοχείς (container elements) χρησιμεύουν στην συλλογή ενός αριθμού πόρων ή χαρακτηριστικών για το σύνολο των οποίων είναι επιθυμητό να διατυπωθεί μια πρόταση. Υπάρχουν τρεις τύποι υποδοχέων στην RDF [5]:

- rdf:Bag: Μη διατεταγμένος υποδοχέας. Τα αντικείμενα που περιέχει δεν ακολουθούν κάποια συγκεκριμένη σειρά ή διάταξη.

- rdf:Seq: Ένας διατεταγμένος υποδοχέας. Τα αντικείμενα που περιέχει είναι ταξινομημένα με κάποιο τρόπο (αλφαβητικά, κ.τ.λ.).
- rdf:Alt: Ένα σύνολο από εναλλακτικές τιμές που αναφέρονται σε μια συγκεκριμένη οντότητα (συνήθως η τιμή μιας ιδιότητας).

Τα περιεχόμενα των στοιχείων – υποδοχέων είναι στοιχεία με ονόματα rdf:_1, rdf:_2, κ.ο.κ. Αντί γι' αυτόν τον συμβολισμό μπορεί να χρησιμοποιηθεί και το όνομα rdf:li για κάθε στοιχείο.

Ένας περιορισμός των υποδοχέων είναι ότι δεν υπάρχει τρόπος να κλείσουν, δηλαδή να δηλωθεί ότι «αυτά είναι όλα τα μέλη του υποδοχέα». Εναλλακτικά μπορεί να χρησιμοποιηθεί το παρακάτω λεξιλόγιο [5], το οποίο υλοποιεί τη δομή μιας λίστας:

- rdf:List: Ορίζει ένα αντικείμενο λίστας.
- rdf:first και rdf:rest: Ιδιότητες που αναφέρονται αντίστοιχα στο πρώτο στοιχείο μιας λίστας και το υπόλοιπό της.
- rdf:nil: Προκαθορισμένος πόρος που υποδηλώνει την κενή λίστα.

Υποστασιοποίηση (Reification)

Η RDF δίνει τη δυνατότητα να ορίζονται προτάσεις που αναφέρονται σε άλλες προτάσεις, ή τα συστατικά τους χρησιμοποιώντας τα παρακάτω στοιχεία [5].

- rdf:Statement: Χρησιμοποιείται κατά τον ίδιο τρόπο με το rdf:Description, με τη μόνη διαφορά ότι ο πόρος στον οποίο αναφέρεται είναι μια πρόταση.
- rdf:subject: Ιδιότητα που αναφέρεται στο υποκείμενο μιας πρότασης.
- rdf:predicate: Ιδιότητα που αναφέρεται στο κατηγόρημα μιας πρότασης.
- rdf:object: Ιδιότητα που αναφέρεται στο αντικείμενο μιας πρότασης.

3.2.3 RDF Schema (RDFS)

Η RDF παρέχει τη δυνατότητα δημιουργίας απλών προτάσεων για τους πόρους των οποίων είναι επιθυμητή η περιγραφή, χρησιμοποιώντας ιδιότητες, τιμές και αναφορές URI για τον προσδιορισμό των συστατικών που συμμετέχουν σε μια πρόταση. Δεν παρέχει όμως δυνατότητα ορισμού και περιγραφής ενός επιπλέον απαιτούμενου λεξιλογίου. Το συγκεκριμένο λεξιλόγιο πρέπει να περιλαμβάνει τον ορισμό των κλάσεων που μπορεί να εμφανίζονται σε μια εφαρμογή, των ιδιοτήτων τους και των πιθανών σχέσεων ιεραρχίας/κληρονομικότητας, είτε μεταξύ των ίδιων των κλάσεων, είτε μεταξύ των ιδιοτήτων τους. Η γλώσσα η οποία παρέχει τη λειτουργικότητα αυτή είναι η RDFS. Σε αυτήν υπάρχει ένα επιπλέον λεξιλόγιο πάνω σε αυτό της RDF το οποίο περιλαμβάνει στοιχεία τα οποία προορίζονται να προσδώσουν την επιπρόσθετη αυτή λειτουργικότητα.

3.2.3.1 Λεξιλόγιο της RDFS

Στις παρακάτω λίστες αναφέρονται όλα τα χαρακτηριστικά που μπορεί να χρησιμοποιηθούν σε ένα έγγραφο RDF/XML για να ορίσουν όρους και λειτουργίες της RDFS, όπως αυτά περιγράφονται στα [5, 110]. Για να χρησιμοποιηθούν όλα αυτά τα χαρακτηριστικά πρέπει στην ετικέτα αρχής του στοιχείου rdf:RDF να προστεθεί το επιπλέον χαρακτηριστικό: xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" που εισάγει στο έγγραφο το χώρο ονομάτων της RDFS.

Βασικές κλάσεις

- rdfs:Resource: Όλα τα πράγματα που περιγράφονται σε RDF ονομάζονται πόροι και είναι στιγμιότυπα της κλάσης αυτής. Αυτή είναι η κλάση των πάντων. Όλες οι άλλες κλάσεις είναι υποκλάσεις της. Η rdfs:Resource είναι στιγμιότυπο της rdfs:Class.
- rdfs:Class: Η κλάση όλων των κλάσεων. Είναι στιγμιότυπο του εαυτού της.
- rdfs:Literal: Πρόκειται για την κλάση όλων των λεκτικών όπως οι συμβολοσειρές και οι ακέραιοι αριθμοί. Σε αυτήν συμπεριλαμβάνονται όλες οι τιμές των ιδιοτήτων που είναι λεκτικά, είτε απλά, είτε τυποποιημένα. Είναι στιγμιότυπο της κλάσης rdfs:class και υποκλάση της rdfs:Resource.
- rdfs:Datatype: Η κλάση όλων των τύπων δεδομένων. Είναι στιγμιότυπο και υποκλάση της rdfs:Class και κάθε στιγμιότυπο της είναι υποκλάση της rdfs:Literal.
- rdfs:XMLLiteral: Η κλάση των κωδικοποιημένων κατά XML λεκτικών. Είναι στιγμιότυπο της rdfs:Datatype και υποκλάση της rdfs:Literal.
- rdf:Property: Η κλάση όλων των ιδιοτήτων. Είναι στιγμιότυπο της rdfs:Class.
- rdf:Statement: Η κλάση όλων των υποστασιοποιημένων προτύπων.

Βασικές ιδιότητες για τον περιορισμό ιδιοτήτων

- rdfs:domain: Καθορίζει το πεδίο ορισμού μιας ιδιότητας P και δηλώνει ότι όλοι οι πόροι που έχουν μια δεδομένη ιδιότητα είναι στιγμιότυπα των κλάσεων του πεδίου ορισμού.
- rdfs:range: Καθορίζει τις κλάσεις των οποίων τα στιγμιότυπα είναι τιμές μιας ιδιότητας P, δηλαδή το σύνολο τιμών της ιδιότητας.

Βασικές ιδιότητες για τον ορισμό σχέσεων

- rdf:type: Αναλύθηκε στην ενότητα 3.2.2.5. Δηλώνει ότι ένας πόρος είναι στιγμιότυπο μιας κλάσης.
- rdfs:subClassOf: Συσχετίζει μια κλάση με μια από τις υπερκλάσεις της. Όλα τα στιγμιότυπα μιας κλάσης είναι στιγμιότυπα και της υπερκλάσης.
- rdfs:subPropertyOf: Συσχετίζει μια ιδιότητα με μια από τις υπεριδιότητές της.

Κλάσεις – υποδοχείς και ιδιότητες – υποδοχείς

- rdfs:Container: Η κλάση όλων των κλάσεων υποδοχέων. Τα στοιχεία – υποδοχείς που αναλύθηκαν στην σχετική υποενότητα της 3.2.2.5 είναι υποκλάσεις της rdfs:Container.
- rdfs:ContainerMembershipProperty: Αυτή η κλάση έχει ως στιγμιότυπα τις ιδιότητες rdf:_1, rdf:_2 κ.ο.κ. που χρησιμεύουν για να δείξουν ότι ένας πόρος είναι μέλος ενός υποδοχέα. Η κλάση είναι υποκλάση της rdf:Property και τα στιγμιότυπά της υποϊδιότητες της ιδιότητας rdfs:member.
- rdfs:member: Είναι η υπεριδιότητα των στιγμιότυπων της προηγούμενης κλάσης (ιδιότητες μελών υποδοχέα).

Βοηθητικές ιδιότητες

- rdfs:seeAlso: Συνδέει έναν πόρο με έναν άλλο πόρο που τον περιγράφει.
- rdfs:isDefinedBy: Είναι υποϊδιότητα της rdfs:seeAlso και συνδέει έναν πόρο με το σημείο όπου μπορεί να βρεθεί ο ορισμός του, συνήθως ένα σχήμα RDF.
- rdfs:comment: Ιδιότητα που εκφράζει σχόλια, αναγνώσιμα από τον άνθρωπο, που μπορεί να συσχετίζονται με κάποιον πόρο.
- rdfs:label: Μια φιλική προς το χρήστη ετικέτα (όνομα) που συσχετίζεται με έναν πόρο. Μεταξύ άλλων, μπορεί να χρησιμεύσει και ως όνομα κόμβου σε μια γραφική αναπαράσταση του εγγράφου RDF.

3.2.4 Web Ontology Language (OWL)

Οι γλώσσες RDF και RDFS επιτρέπουν την αναπαράσταση ενός μέρους της οντολογικής γνώσης. Τα κύρια θεμελιώδη στοιχεία μοντελοποίησης των δύο αυτών γλωσσών αφορούν την οργάνωση των λεξιλογίων σε τυποποιημένες ιεραρχίες: σχέσεις υποκλάσης και υποϊδιότητας, περιορισμούς πεδίου ορισμού και συνόλου τιμών, καθώς και στιγμιότυπα κλάσεων. Ωστόσο λείπουν αρκετές άλλες δυνατότητες. Μερικές από αυτές είναι οι εξής [5]:

- Τοπική εμβέλεια ιδιοτήτων: Το rdfs:range ορίζει το σύνολο τιμών μιας ιδιότητας για όλες τις κλάσεις. Επομένως, δεν είναι δυνατό στην RDFS να δηλωθούν περιορισμοί στο σύνολο τιμών, οι οποίοι θα ισχύουν μόνο για μερικές κλάσεις.
- Μη επικάλυψη κλάσεων: Δεν είναι δυνατόν να οριστούν δύο κλάσεις ως ξένες μεταξύ τους.
- Λογικοί συνδυασμοί κλάσεων, δηλαδή ο ορισμός μιας κλάσης ως ένωση, τομή ή συμπλήρωμα άλλων κλάσεων.
- Περιορισμοί πληθικότητας των στοιχείων μιας κλάσης.
- Ειδικά χαρακτηριστικά ιδιοτήτων: Στην RDFS δεν μπορεί να εκφραστεί το ότι μια ιδιότητα είναι μεταβατική, μοναδική ή αντίστροφη κάποιας άλλης ιδιότητας.

Επομένως απαιτείται μια γλώσσα οντολογιών πλουσιότερη από την RDFS και η οποία θα παρέχει τις παραπάνω, αλλά και επιπλέον δυνατότητες. Η γλώσσα αυτή είναι η OWL. Όλες οι απαιτούμενες λειτουργίες παρέχονται μέσω χαρακτηριστικών, ιδιοτήτων και κλάσεων, που περιλαμβάνονται στο λεξιλόγιο της.

3.2.4.1 Η OWL 2 και τα τρία προφίλ της

Η OWL 2 είναι η τελευταία, ανανεωμένη έκδοση της αρχικής δημοσίευσης της OWL που έγινε το 2004 και που αναφέρεται πλέον σαν «OWL 1». Κληρονομεί όλα τα βασικά στοιχεία λεξιλογίου της αρχικής έκδοσης καθώς και επιπλέον χαρακτηριστικά, όπως νέες δομές που αυξάνουν την εκφραστικότητα, επεκταμένη υποστήριξη για τύπους δεδομένων και απλές δυνατότητες μεταμοντελοποίησης [107]. Στο εξής, με κάθε αναφορά σε OWL εννοείται η έκδοση OWL 2.

Κατά το σχεδιασμό μιας γλώσσας πρέπει να τηρηθεί μια ισορροπία ανάμεσα στην εκφραστική της ισχύ και στην απόδοση υποστήριξης συλλογισμών. Γενικά, όσο πλουσιότερη είναι μια γλώσσα, τόσο λιγότερο αποδοτική γίνεται η υποστήριξη συλλογισμών, ξεπερνώντας συχνά το όριο της μη υπολογισιμότητας [5]. Το W3C αντιμετωπίζει την κατάσταση αυτή στην OWL 2 ορίζοντας για αυτήν τρία προφίλ, καθένα από τα οποία ανταλλάσσει μέρος της εκφραστικής δύναμης της γλώσσας για αυξημένη αποδοτικότητα.

Η OWL 2 EL χρησιμοποιείται σε εφαρμογές που περιέχουν οντολογίες με μεγάλο αριθμό ιδιοτήτων ή/και κλάσεων. Μπορεί να χειριστεί την εκφραστική τους δύναμη και να επιλύσει τα βασικά προβλήματα συμπερασμού (inference) σε πολυωνυμικό χρόνο αναφορικά με το μέγεθος της οντολογίας. Το ακρωνύμιο EL δηλώνει ότι το προφίλ βασίζεται στο EL κομμάτι των περιγραφικών λογικών, που περιγράφει μόνο την υπαρξιακή ποσοτικοποίηση (existential quantification).

Η OWL 2 QL στοχεύει σε εφαρμογές που χρησιμοποιούν μεγάλους όγκους δεδομένων – στιγμιότυπων, όπου η πιο σημαντική εργασία συλλογιστικής είναι η απάντηση ερωτημάτων. Είναι δυνατή η υλοποίηση μηχανισμού απαντήσεων σε συνενωτικά ερωτήματα (conjunctive queries) με χρήση συμβατικών σχεσιακών βάσεων δεδομένων. Με κατάλληλες τεχνικές συμπερασμού οι απαντήσεις μπορούν να δοθούν σε λογαριθμικό χρόνο. Η εκφραστική δύναμη του προφίλ αυτού έχει περιοριστεί στο να συμπεριλαμβάνει εννοιολογικά μοντέλα, όπως το ER (Entity-Relationship, Μοντέλο Οντότητας – Συσχέτισης). Το ακρωνύμιο QL προκύπτει από το γεγονός ότι η απάντηση ερωτημάτων μπορεί να υλοποιηθεί ξαναγράφοντας τα ερωτήματα σε μια σχεσιακή γλώσσα ερωτημάτων (Query Language, QL).

Η OWL 2 RL στοχεύει σε εφαρμογές που απαιτούν επεκτάσιμη συλλογιστική χωρίς να θυσιάζουν μεγάλο μέρος της εκφραστικότητας της γλώσσας. Τα σχετικά συστήματα συμπερασμού μπορούν να υλοποιηθούν με χρήση κανόνων (rule-based engines). Προβλήματα όπως ο έλεγχος στιγμιότυπων και η απάντηση συνενωτικών ερωτημάτων μπορούν να λυθούν σε πολυωνυμικό χρόνο αναφορικά με το μέγεθος της οντολογίας. Το ακρωνύμιο RL αναφέρεται στο γεγονός ότι ο συμπερασμός μπορεί να υλοποιηθεί με μια βασική γλώσσα κανόνων (Rule Language, RL).

3.2.4.2 Περιγραφή της γλώσσας OWL

Καθώς η OWL βασίζεται στις γλώσσες RDF και RDFS, χρησιμοποιεί κατά κύριο λόγο την σύνταξη της RDF που βασίζεται στην XML (RDF/XML). Όλα τα παρακάτω χαρακτηριστικά και τα παραδείγματά τους θα δίνονται σε αυτή τη μορφή.

Κεφαλίδα

Τα έγγραφα OWL αποκαλούνται συνήθως οντολογίες OWL και είναι έγγραφα RDF. Τα δύο πρώτα στοιχεία μιας οντολογίας είναι τα ίδια με αυτά ενός απλού εγγράφου RDF όπως αυτά εξηγούνται στην ενότητα 3.2.2.5. Για μια οντολογία πρέπει να προστεθεί ο χώρος ονομάτων της OWL, ήτοι http://www.w3.org/2002/07/owl#.

Μετά από αυτά τα υποχρεωτικά στοιχεία μπορεί να υπάρχει μια συλλογή ισχυρισμών (assertions) για λόγους «νοικοκυρέματος» του εγγράφου. Οι ισχυρισμοί αυτοί οργανώνονται σε ένα στοιχείο owl:Ontology, το οποίο περιέχει σχόλια, έλεγχο εκδόσεων και εισαγωγή άλλων οντολογιών. Από τους ισχυρισμούς αυτούς μόνο ένας έχει συνέπειες για το λογικό νόημα της οντολογίας: το owl:imports, που απαριθμεί άλλες οντολογίες, το περιεχόμενο των οποίων υποτίθεται ότι είναι μέρος της τρέχουσας οντολογίας [5].

Στοιχεία κλάσεων

Οι κλάσεις ορίζονται με τη χρήση του στοιχείου owl:Class που είναι υποκλάση της rdfs:Class. Η πιο συνηθισμένη περίπτωση ορισμού κλάσης είναι με χρήση ετικέτας κενού στοιχείου (π.χ. <owl:Class rdf:ID="Human"/>). Σε περίπτωση που αυτό δεν αρκεί για να οριστεί πλήρως μια κλάση και χρειάζεται να δηλωθεί ως προς τη σχέση της με άλλες κλάσεις, παρέχονται κάποιες ιδιότητες, όπως [5]:

- owl:disjointWith: Χρησιμοποιείται για να δηλώσει ότι μια κλάση είναι ξένη ως προς μια άλλη κλάση (δεν έχουν κοινά στοιχεία).
- owl:equivalentClass: Χρησιμοποιείται για να δηλώσει την ισοδυναμία μεταξύ δύο κλάσεων.
- rdfs:subClassOf: Το στοιχείο μεταφέρεται αυτούσιο από το λεξιλόγιο της RDFS για να περιγράψει σχέσεις υποκλάσης μεταξύ κλάσεων.
- owl:Thing: Είναι η πιο γενική κλάση, η κλάση των πάντων, η υπερκλάση κάθε κλάσης.
- owl:Nothing: Η κενή κλάση, η οποία είναι και υποκλάση όλων των κλάσεων.

Βασικά στοιχεία ιδιοτήτων

Στην OWL υπάρχουν δύο είδη ιδιοτήτων [5]:

- Ιδιότητες αντικειμένου (Object Properties), οι οποίες συσχετίζουν αντικείμενα με άλλα αντικείμενα. Αποτελούν στιγμιότυπα της κλάσης owl:ObjectProperty.

- Ιδιότητες τύπου δεδομένων (Datatype Properties), οι οποίες συσχετίζουν αντικείμενα με τιμές ενός τύπου δεδομένων. Αποτελούν στιγμιότυπα της κλάσης owl:DatatypeProperty. Όπως και η RDFS, έτσι και η OWL δεν διαθέτει δικούς της προκαθορισμένους τύπους δεδομένων, επιτρέπει όμως τη χρήση των τύπων δεδομένων της XML Schema.

Σε κάθε ορισμό ιδιότητας είναι δυνατόν να οριστούν ένα ή περισσότερα πεδία ορισμού ή/και σύνολα τιμών με χρήση των στοιχείων rdfs:domain και rdfs:range. Σε περίπτωση που υπάρχουν παραπάνω από ένα πεδία ή σύνολα, λαμβάνεται η τομή τους. Επίσης επιτρέπεται η συσχέτιση αντίστροφων ιδιοτήτων, με χρήση της ιδιότητας owl:inverseOf. Παράδειγμα αποτελεί το ζεύγος ιδιοτήτων drives (οδηγεί) και isDrivenBy (οδηγείται από). Τέλος με τη χρήση της ιδιότητας owl:equivalentProperty, μια ιδιότητα μπορεί να οριστεί ως ισοδύναμη μιας άλλης (μπορεί να χρησιμοποιηθεί στη θέση της στο RDF σύνολο χωρίς να αλλάξει το γενικό νόημα).

Ειδικοί τύποι ιδιοτήτων

Τα παρακάτω στοιχεία χρησιμεύουν στον απευθείας ορισμό μερικών ιδιοτήτων των στοιχείων ιδιοτήτων. Εκτός από την owl:FunctionalProperty, που είναι υποκλάση της rdf:Property, οι υπόλοιπες κλάσεις ορισμού ιδιοτήτων είναι υποκλάσεις της owl:ObjectProperty [106].

- owl:TransitiveProperty: Ορίζει μια μεταβατική ιδιότητα. Μια ιδιότητα P είναι μεταβατική, όταν η ισχύς των τριάδων <x, P, y> και <y, P, z>, συνεπάγεται την ισχύ της <x, P, z>.
- owl:SymmetricProperty: Ορίζει μια συμμετρική ιδιότητα. Μια ιδιότητα P είναι συμμετρική, αν η ισχύς της τριάδας <x, P, y> συνεπάγεται και την ισχύ της τριάδας <y, P, x>.
- owl:FunctionalProperty: Ορίζει μια συναρτησιακή ιδιότητα. Μια ιδιότητα P είναι συναρτησιακή, εάν δεν υπάρχουν δύο διακριτές τιμές y_1, y_2 τέτοιες ώστε να ισχύουν για δεδομένο x οι τριάδες <x, P, y_1> και <x, P, y_2> (δηλαδή έχει το πολύ μία τιμή για κάθε αντικείμενο).
- owl:InverseFunctionalProperty: Ορίζει μια αντιστρόφως συναρτησιακή ιδιότητα. Μια ιδιότητα P είναι αντιστρόφως συναρτησιακή, εάν δεν υπάρχουν δύο διακριτές τιμές x_1, x_2 τέτοιες ώστε να ισχύουν για δεδομένο y οι τριάδες <x_1, P, y> και <x_2, P, y> (δηλαδή δύο διαφορετικά αντικείμενα δεν μπορούν να έχουν την ίδια τιμή).

Εναλλακτικοί ορισμοί κλάσεων

Αν και για να οριστεί μια κλάση συνήθως χρησιμοποιείται το όνομά της μέσω ενός URIref, υπάρχουν και ο τρόπος ορισμού μέσω των συνθηκών που ικανοποιούν τα στιγμιότυπά της. Αυτό μπορεί να επιτευχθεί ορίζοντας την σαν την υποκλάση μιας

κλάσης, η οποία μπορεί και να είναι ανώνυμη, που προσδιορίζεται από τις επιθυμητές συνθήκες, οι οποίες σχετίζονται με τρίτες κλάσεις ως εξής:

- Μια πλήρη απαρίθμηση αντικειμένων, που όλα μαζί δημιουργούν τα στιγμιότυπα μιας κλάσης:
- Με περιορισμούς ιδιοτήτων.
- Με την ένωση ή την τομή δύο ή περισσοτέρων περιγραφών κλάσεων.
- Με το συμπλήρωμα μιας κλάσης.

Η OWL διαθέτει τα κατάλληλα εργαλεία για κάθε περίπτωση, τα οποία περιγράφονται αναλυτικά στα [5, 106] και παρουσιάζονται συνοπτικά στις επόμενες υποενότητες.

Περιορισμοί ιδιοτήτων

Οι περιορισμοί ιδιοτήτων έχουν την ακόλουθη γενική μορφή:

```
<owl:Restriction>
   <owl:onProperty rdf:resource=”(όνομα ή URIref μιας ιδιότητας)”/>
   (στοιχείο δήλωσης ακριβώς μίας τιμής, περιορισμός τιμών ή περιορισμός
   πληθικότητας)
</owl:Restriction>
```

Παράδειγμα 3.2: Γενική μορφή περιορισμού ιδιοτήτων.

όπου η κλάση owl:Restriction είναι μια υποκλάση της owl:Class. Μπορούν να εφαρμοστούν και σε ιδιότητες αντικειμένου και σε ιδιότητες τύπου δεδομένων.

Υπάρχουν δύο βασικοί τύποι περιορισμών ιδιοτήτων:

- Περιορισμός είδους τιμών: Περιορίζει το σύνολο τιμών της ιδιότητας όταν χρησιμοποιείται στη συγκεκριμένη περιγραφή κλάσης.
- Περιορισμός πληθικότητας: Περιορίζει το πλήθος των τιμών που μπορεί να έχει μια ιδιότητα μέσα στη δομή της συγκεκριμένης περιγραφής κλάσης.

Οι περιορισμοί είδους τιμών είναι τριών τύπων:

- owl:allValuesFrom: Ιδιότητα που έχει για τιμή της έναν πόρο που αναφέρεται σε κλάση ή ένα σύνολο τιμών. Δηλώνει ότι όλα τα αντικείμενα που περιέχονται στον πόρο – τιμή περιλαμβάνονται στον περιορισμό ιδιότητας.
- owl:someValuesFrom: Αντίθετα με την προηγούμενη ιδιότητα, εδώ δηλώνεται ότι τουλάχιστον ένα από τα αντικείμενα του πόρου – τιμή περιλαμβάνονται στον περιορισμό ιδιότητας.
- owl:hasValue: Δηλώνει μια συγκεκριμένη τιμή (πόρος ή λεκτικό) την οποία πρέπει να έχει η ιδιότητα που καθορίζεται από το owl:onProperty.

Οι περιορισμοί πληθικότητας είναι τριών τύπων. Όλοι συνδέουν την περιγραφή κλάσης στην οποία βρίσκονται, με μια τιμή που ανήκει στον τύπο δεδομένων nonNegativeInteger της XML Schema. Σε μια οντολογία OWL συντάσσονται με το χαρακτηριστικό rdf:datatype="&xsd;nonNegativeInteger".

- owl:maxCardinality: Η κλάση περιέχει το πολύ N σημασιολογικά διακριτές τιμές (πόροι ή τιμές δεδομένων), όπου N η τιμή του περιορισμού.
- owl:minCardinality: Η κλάση περιέχει τουλάχιστον N σημασιολογικά διακριτές τιμές.
- owl:cardinality: Η κλάση περιέχει ακριβώς N σημασιολογικά διακριτές τιμές.

Απαριθμήσεις

Η περιγραφή κλάσης με απαρίθμηση αντικειμένων ορίζεται με την ιδιότητα owl:oneOf. Η τιμή αυτής της ιδιότητας πρέπει να είναι μια λίστα αντικειμένων τα οποία είναι τα στιγμιότυπα της περιγραφόμενης κλάσης. Η λίστα στοιχείων παριστάνεται με τη βοήθεια του κατασκευαστή rdf:parseType=”Collection”.

Λογικοί συνδυασμοί

Οι τρεις επόμενες ιδιότητες αποτελούν τύπους περιγραφής κλάσεων, οι οποίοι είναι πιο προχωρημένοι. Ουσιαστικά αποτελούν για τις κλάσεις τα αντίστοιχα των τελεστών AND, OR, NOT που χρησιμοποιούνται σε λογικές προτάσεις:

- owl:intersectionOf: Ιδιότητα που ορίζει μια κλάση η οποία περιγράφεται από τα στοιχεία της τομής δύο ή περισσότερων κλάσεων.
- owl:unionOf: Ιδιότητα που ορίζει μια κλάση η οποία περιγράφεται από τα στοιχεία της ένωσης δύο ή περισσότερων κλάσεων. Η ιδιότητα αυτή, όπως και η προηγούμενη, συντάσσεται με ένα χαρακτηριστικό rdf:parseType=”Collection”, καθώς απαιτείται η ύπαρξη δύο ή περισσότερων κλάσεων για να οριστεί η ένωση ή η τομή τους.
- owl:complementOf: Ιδιότητα που ορίζει μια κλάση η οποία περιγράφεται από τα στοιχεία του συμπληρώματος μιας άλλης κλάσης. Αυτό σημαίνει ότι η περιγραφόμενη κλάση αποτελείται από όλα τα στοιχεία που δεν ανήκουν στην κλάση – τιμή της ιδιότητας αυτής.

Ταυτότητα αντικειμένων

Αρκετές γλώσσες προγραμματισμού λειτουργούν κάτω από την υπόθεση του «μοναδικού ονόματος»: διαφορετικά ονόματα αναφέρονται σε διαφορετικά πράγματα. Ωστόσο μια τέτοια υπόθεση δεν είναι δυνατή στον Παγκόσμιο Ιστό. Για παράδειγμα, το ίδιο άτομο μπορεί να αναφέρεται σε διαφορετικές τοποθεσίες με διαφορετικούς τρόπους (συνήθως με διαφορετικές αναφορές URI). Κατά συνέπεια η υπόθεση αυτή δεν ισχύει ούτε στην OWL. Εκτός και αν υπάρξει μια ρητή πρόταση που ορίζει ότι δύο ή περισσότερες αναφορές URI αναφέρονται στο ίδιο ή σε διαφορετικά πράγματα, τα εργαλεία της OWL υποθέτουν ότι και οι δύο περιπτώσεις είναι εξίσου πιθανές.

Η OWL παρέχει τρία στοιχεία για τη δήλωση γεγονότων σχετικά με την ταυτότητα αντικειμένων [106].

- owl:sameAs: Ιδιότητα που δηλώνει ότι δύο αναφορές URI αναφέρονται στο ίδιο αντικείμενο.
- owl:differentFrom: Ιδιότητα που δηλώνει ότι δύο αναφορές URI αναφέρονται σε διαφορετικά αντικείμενα.
- owl:AllDifferent: Κλάση, τα στιγμιότυπα της οποίας συνδέονται με λίστες αντικειμένων μέσω της ιδιότητας owl:distinctMenbers. Δηλώνει ότι όλα τα αντικείμενα μιας λίστας είναι διαφορετικά το ένα από το άλλο. Χρησιμοποιείται για συντομία, αντί ενός συνόλου πολλαπλών owl:differentFrom μεταξύ των στοιχείων της λίστας.

3.2.5 Συλλογιστική με κανόνες (Rule – based reasoning)

Για να μπορέσουν τα θεμελιώδη στοιχεία μοντελοποίησης των RDF και RDFS να εκφράσουν το νόημα των δεδομένων που αναπαριστούν, πρέπει πρώτα να τυποποιηθεί το νόημα που τα ίδια εκφράζουν. Με άλλα λόγια απαιτείται να οριστεί η σημασιολογία των στοιχείων των RDF και RDFS. Αυτό γίνεται μέσω της κατηγορηματικής λογικής, γνωστής και ως λογική πρώτης τάξεως. Βάσει αυτής ορίζονται κάποιες προτάσεις στις οποίες υπακούν τα στοιχεία και οι οποίες ονομάζονται αξιώματα. Τα δύο βασικά αξιώματα είναι [5]:

- Triple(R, P, V), που εκφράζει μια πρόταση RDF. R είναι ο πόρος – υποκείμενο, P είναι η ιδιότητα – κατηγόρημα και V η τιμή της ιδιότητας (αντικείμενο)
- Type(X, Y), που συνδέει έναν πόρο X με την κλάση Y στην οποία ανήκει: Ουσιαστικά ισχύει: Type(X, Y) ⇔ Triple(X, rdf:type, Y).

Μέσω αυτών των δύο βασικών κανόνων προκύπτει όλη η αξιωματική θεμελίωση των RDF/RDFS. Ωστόσο για να χρησιμοποιηθεί η αξιωματική σημασιολογία των RDF/RDFS σε αυτή τη μορφή, απαιτείται ένα σύστημα αποδείξεων λογικής πρώτης τάξης, κάτι το οποίο αποδεικνύεται ιδιαίτερα «βαρύ» για τις γλώσσες αυτές από πλευράς απόδοσης της υποστήριξης συλλογισμών, ιδιαίτερα σε περιπτώσεις μεγάλου αριθμού προτάσεων (π.χ. εκατομμύρια προτάσεις Type(X, Y) σε ένα μεγάλο έγγραφο).

Για το λόγο αυτό έχει δοθεί στις RDF/RDFS μια σημασιολογία και ένα σύστημα συμπερασμού εκφρασμένο σε τριάδες RDF. Έτσι διατυπώνονται προτάσεις όπως:

- rdfs:subClassOf rdf:type rdf:Property.
- rdfs:Class rdf:type rdfs:Class.
- rdf:type rdf:type rdf:Property.

Στο Παράρτημα Γ δίνεται πίνακας με τα βασικότερα αξιώματα και προτάσεις της αξιωματικής θεμελίωσης των RDF/RDFS/OWL σε μορφή τριάδων.

Το σύστημα συμπερασμού αποτελείται από κανόνες της μορφής:

ΑΝ το Ε περιέχει συγκεκριμένες τριάδες
ΤΟΤΕ πρόσθεσε στο Ε συγκεκριμένες επιπλέον τριάδες

όπου Ε είναι ένα αυθαίρετο σύνολο τριάδων RDF.

Η γλώσσα OWL, σαν ένα είδος επέκτασης των RDF/RDFS χρησιμοποιεί τις ίδιες τεχνικές κανόνων για συμπερασμό αναλύοντας και χρησιμοποιώντας και τα δικά της στοιχεία. Στο Παράρτημα Γ δίνεται πίνακας ενδεικτικών παραδειγμάτων κανόνων συμπερασμού από όλο το φάσμα των Γλωσσών Σημασιολογικού Ιστού.

Η θεμελιώδης μορφή ενός κανόνα είναι η εξής:

$$A_1, A_2, A_3, \ldots, A_n \rightarrow B$$

Όπου τα A_1, A_2, A_3, …, A_n, Β είναι ατομικοί τύποι. Το Β είναι η κεφαλή και τα A_1, A_2, A_3, …, A_n είναι οι προϋποθέσεις του κανόνα. Το σύνολο $\{A_1, \ldots, A_n\}$ αναφέρεται ως το σώμα του κανόνα. Οποιοσδήποτε από τους ατομικούς τύπος μπορεί να περιέχει μεταβλητές.

Οι κανόνες χωρίζονται σε δύο είδη:

- Επαγωγικοί Κανόνες: Στη συγκεκριμένη περίπτωση ελέγχεται η ισχύς όλων των ατομικών τύπων – προϋποθέσεων του κανόνα. Αν είναι όλοι αληθείς, τότε είναι αληθής και η κεφαλή του.
- Κανόνες Αντίδρασης: Στην περίπτωση αυτή, αν όλες οι προϋποθέσεις του κανόνα είναι αληθείς, τότε εκτελείται η ενέργεια – κεφαλή του κανόνα.

Κατά τη χρήση τους σε συστήματα συλλογιστικής (reasoning systems), οι κανόνες έχουν δύο διαφορετικούς τρόπους ανάγνωσης [25]:

- Ανάγνωση με κανονική αλληλουχία (προς τα εμπρός, forward chaining): Ξεκινά η εκτέλεση του κανόνα με τα διαθέσιμα δεδομένα και χρησιμοποιούνται οι κανόνες συμπερασμού του συστήματος για να αντλήσει περισσότερα δεδομένα ώστε να επιτευχθεί ο στόχος.
- Ανάγνωση με ανάστροφη αλληλουχία (προς τα πίσω, backward chaining): Ο κανόνας εκτελείται ξεκινώντας με μια λίστα στόχων (ή μια υπόθεση) και δουλεύει προσπαθώντας να βρει αν υπάρχουν διαθέσιμα δεδομένα ώστε να υποστηρίζουν τους στόχους (ή την υπόθεση).

3.2.6 Η γλώσσα SPARQL

Η SPARQL [111] είναι μια γλώσσα ερωτημάτων RDF που περιλαμβάνει και ένα σχετικό πρωτόκολλο. Το όνομά της είναι ένα ακρωνύμιο που σημαίνει Simple Protocol And RDF Query Language (Απλό Πρωτόκολλο και Γλώσσα Ερωτημάτων RDF). Προτυποποιήθηκε από το SPARQL Working Group του World Wide Web Consortium και θεωρείται μια τεχνολογία – κλειδί του Σημασιολογικού Ιστού.

Η SPARQL επιτρέπει σε ένα ερώτημα να αποτελείται από διατάξεις τριάδων, λογικές ενώσεις και τομές και άλλες κατ' επιλογή διατάξεις. Χρησιμοποιείται για να ανακτήσει τιμές από δομημένα και ημιδομημένα δεδομένα, να εξερευνήσει δεδομένα θέτοντας ερωτήματα σε άγνωστες σχέσεις, να πραγματοποιήσει σύνθετες ενώσεις ανόμοιων βάσεων δεδομένων και για να μετατρέψει δεδομένα RDF από ένα λεξιλόγιο σε ένα άλλο.

Η μορφή σύνταξης που χρησιμοποιείται για την υποβολή των ερωτημάτων είναι η Turtle [112].

3.2.6.1 Βασικά ερωτήματα SPARQL

Η SPARQL βασίζεται στην ταύτιση υποδειγμάτων γράφων (graph patterns). Το απλούστερο υπόδειγμα γράφου είναι το υπόδειγμα της τριάδας, το οποίο μοιάζει με μια τριάδα RDF, αλλά υπάρχει η δυνατότητα χρήσης μεταβλητής αντί όρου RDF σε οποιαδήποτε από τις τρεις θέσεις (υποκείμενο – κατηγόρημα – αντικείμενο). Ο συνδυασμός υποδειγμάτων τριάδων παράγει ένα βασικό υπόδειγμα γράφου και απαιτείται ακριβής ταύτιση με κάποιο γράφο προκειμένου ένα υπόδειγμα να θεωρηθεί πλήρες [5].

Ένα απλό ερώτημα αποτελείται από τα παρακάτω στοιχεία με αυτή τη σειρά:

- Δηλώσεις προθεμάτων, για συντόμευση των URIs.
- Μια δήλωση αποτελέσματος, που προσδιορίζει το τι πληροφορίες ζητούνται να επιστραφούν από το ερώτημα.
- Το υπόδειγμα του ερωτήματος, που καθορίζει που θα γίνουν οι ερωτήσεις στο σύνολο δεδομένων

Ένα παράδειγμα ερωτήματος δίνεται παρακάτω:

```
1. PREFIX foaf: <http://xmlns.com/foaf/0.1/>
2. SELECT ?name ?mbox
3. WHERE
4.   { ?x foaf:name ?name .
5.     ?x foaf:mbox ?mbox }
```

Παράδειγμα 3.3: Απλό ερώτημα SPARQL.

Η γραμμή 1, αποτελεί την μοναδική δήλωση προθέματος στο ερώτημα. Για κάθε οντολογία, στο χώρο ονομάτων της οποίας αναφέρονται κάποια στοιχεία των τριάδων ενός ερωτήματος, προτείνεται ο ορισμός ενός προθέματος για την αποφυγή δυσανάγνωστων ερωτημάτων από τις πολλαπλές επαναλήψεις ενός URI.

Οι γραμμές 2 έως 5 αποτελούν το ερώτημα. Τα ερωτήματα SPARQL έχουν τη δομή τύπου SELECT – FROM – WHERE, παρόμοια με την SQL [5].

- Το SELECT (γραμμή 2) παίζει το ρόλο της δήλωσης αποτελέσματος, η οποία περιλαμβάνει δύο μεταβλητές με όνομα name και mbox, οι οποίες και ζητούνται από το ερώτημα.
- Η λέξη WHERE καθώς και οι τριάδες που ακολουθούν (γραμμές 3 έως 5) ορίζουν τη μορφή γράφου που πρέπει να ικανοποιούν τα ζητούμενα δεδομένα. Κάθε τριάδα τελειώνει με μια τελεία (.), όπως ορίζει η RDF για τις προτάσεις της, εκτός από την τελευταία τριάδα κάθε μπλοκ τριάδων που ορίζεται μεταξύ αγκυλών ({}). Επίσης παρατηρείται η χρήση μεταβλητών στις θέσεις των συστατικών των τριάδων.

Το αποτέλεσμα ενός ερωτήματος είναι μια ακολουθία λύσεων, που απεικονίζει τους τρόπους με τους οποίους η μορφή του γράφου του ερωτήματος ταιριάζει με τα δεδομένα [111].

Τα αποτελέσματα κάθε ερωτήματος SELECT δίνονται σε μορφή πίνακα, η οποία προκύπτει από το XML πρότυπο σειριοποίησης SPARQL Query Results XML Format. Κάθε λύση δίνει ένα τρόπο με τον οποίο οι επιλεγμένες μεταβλητές μπορούν να δεσμευθούν με όρους RDF ώστε η μορφή του ερωτήματος να ταιριάζει στα δεδομένα. Το σύνολο αποτελεσμάτων αποτελείται όλες τις πιθανές λύσεις.

3.2.6.2 Άλλα είδη ερωτημάτων SPARQL

Πέρα από τα βασικά ερωτήματα SELECT στο [111] αναφέρονται και μερικοί άλλοι τύποι ερωτημάτων, καθένας με τη δική του λειτουργικότητα:

- CONSTRUCT: Συντάσσεται με όρισμα ένα υπόδειγμα γράφου μέσα σε αγκύλες. Το ερώτημα αυτό επιστρέφει έναν RDF γράφο με μορφή αυτή του ορίσματος. Ακολουθείται από μια πρόταση WHERE που περιλαμβάνει το δικό της υπόδειγμα γράφου. Ο επιστρεφόμενος γράφος αποτελείται από ένα σύνολο τριάδων ίδιο με αυτό του ορίσματος για κάθε στοιχείο της ακολουθίας λύσεων του υποδείγματος του WHERE.
- ASK: Συντάσσεται με όρισμα ένα υπόδειγμα γράφου μέσα σε αγκύλες. Επιστρέφει μια απάντηση yes ή no αναλόγως με το αν υπάρχει λύση για το υπόδειγμα – όρισμα.
- DESCRIBE: Συντάσσεται με μεταβλητές ή/και URI. Ακολυθείται (συνήθως) από μια πρόταση WHERE που περιαμβάνει το δικό της μπλοκ προτάσεων. Επιστρέφει ένα γράφο που περιέχει προτάσεις που σχετίζονται με τους πόρους που αποτελούν λύσεις του υποδείγματος του μπλοκ της WHERE (ή αλλιώς μια «περιγραφή» των πόρων αυτών). Το πόσες και ποιες πληροφορίες καθορίζεται από το φορέα που περιέχει τα RDF δεδομένα και την υπηρεσία ερωτημάτων του.

3.2.6.3 Λεκτικά και κενοί κόμβοι

Υπάρχουν περιπτώσεις όπου τα αποθηκευμένα δεδομένα σε έναν RDF γράφο είναι λεκτικά, τα οποία μπορεί και να είναι τυποποιημένα. Σε αυτήν την περίπτωση η πλήρης μορφή του λεκτικού πρέπει να δίνεται στο ερώτημα, αλλιώς δεν επιστρέφεται το επιθυμητό αποτέλεσμα. Εξαίρεση αποτελούν οι ακέραιοι που τοποθετούνται σε ένα SPARQL ερώτημα, οι οποίοι αυτόματα θεωρούνται ότι είναι τυποποιημένα λεκτικά με μορφοποίηση xsd:integer. Το ίδιο γίνεται και για τους τύπους xsd:float και xsd:double. Οποιαδήποτε άλλη μορφή τυποποίησης αριθμών πρέπει να δηλωθεί ρητά [111].

Τα αποτελέσματα ενός ερωτήματος μπορεί να περιέχουν κενούς κόμβους. Σε τέτοιες περιπτώσεις, οι κόμβοι συμβολίζονται με "_:" και ακολουθούνται από μια ετικέτα. Ίδιες ετικέτες αντιστοιχούν σε ίδιους κόμβους. Κατά την επιστροφή των

αποτελεσμάτων είναι δυνατόν να χρησιμοποιηθούν διαφορετικές ετικέτες, καθότι η ονομασία των κενών κόμβων είναι αυθαίρετη. Έτσι, η επιστροφή ετικετών _:a, _:b, αντί των _:x, _:y που περιέχει ο αρχικός γράφος είναι φυσιολογική [111].

3.2.6.4 Βασικό λεξιλόγιο της SPARQL

Η SPARQL διαθέτει μια σειρά από προτάσεις, οι οποίες περιγράφονται στο [111], για να καλύψει περιπτώσεις όπως ερωτήματα σε πολλαπλούς γράφους και «φιλτράρισμα» ή ταξινόμηση των αποτελεσμάτων βάσει συγκεκριμένων τιμών ή μεταβλητών.

- FROM/FROM NAMED: Στα RDF σύνολα υπάρχει συνήθως ένας προκαθορισμένος γράφος στον οποίο προστίθενται τα νέα δεδομένα. Πέραν αυτού, μπορούν να υπάρχουν στο ίδιο σύνολο και άλλοι γράφοι οι οποίοι κατονομάζονται με το δικό τους URI. Οι εκφράσεις FROM/FROM NAMED χρησιμοποιούνται σε ένα ερώτημα πριν το μπλοκ τριάδων της έκφρασης WHERE. Συντάσσονται με URIs τα οποία αναφέρονται σε τέτοιους κατονομαζόμενους γράφους (named graphs).
- GRAPH: Έκφραση ισοδύναμη με τις προηγούμενες δύο, που χρησιμοποιείται μέσα σε μπλοκ τριάδων μιας έκφρασης WHERE. Συντάσσεται είτε με URI κατονομασμένου γράφου, ή με μεταβλητή, που παίρνει μόνο URI σαν τιμή.
- FILTER: Χρησιμοποιείται σαν επιλογέας υποσυνόλου των αποτελεσμάτων που επιστρέφει το ερώτημα που τον περιέχει Η επιλογή γίνεται βάσει περιορισμών που θέτει ο χρήστης. Η σύνταξή της είναι: FILTER (έκφραση). Στη θέση της έκφρασης μπορεί να χρησιμοποιηθεί οποιαδήποτε κανονική έκφραση, η οποία μπορεί να περιέχει, μεταβλητές, μαθηματικούς τύπους, λεκτικά και αριθμητικές τιμές. Για την ειδική περίπτωση όπου η επιλογή βασίζεται σε μια συμβολοσειρά, χρησιμοποιείται η συνάρτηση regex ως εξής: FILTER regex(μεταβλητή, συμβολοσειρά).
- OPTIONAL: Συναντάται μέσα σε μπλοκ τριάδων WHERE και μπορεί να περιλαμβάνει το δικό της μπλοκ. Επιτρέπει την σύνταξη ερωτημάτων στα οποία μπορούν να προστίθενται πληροφορίες στη λύση τους, όπου αυτές είναι διαθέσιμες, χωρίς όμως να απορρίπτουν τη λύση, αν ένα μέρος της δεν ταιριάζει στο αρχικό σχήμα του ερωτήματος (προαιρετική αντιστοίχιση). Κάθε τριάδα ή σύνολο τριάδων που ορίζει και ένα «προαιρετικό» στοιχείο πρέπει να περικλείεται στη δική του πρόταση OPTIONAL.
- UNION: Τελεστής που χρησιμοποιείται για να ενώσει τα αποτελέσματα δύο υποδειγμάτων γράφων (το καθένα στο δικό του μπλοκ) και να τα εμφανίσει σαν ένα ενιαίο αποτέλεσμα.
- GROUP BY: Ακολουθεί το μπλοκ τριάδων της πρότασης WHERE. Συντάσσεται με μια ή περισσότερες μεταβλητές, σύμφωνα με τις οποίες ομαδοποιούνται τα αποτελέσματα του ερωτήματος.
- HAVING: Συντάσσεται συνήθως μετά την GROUP BY και συνοδεύεται από μια συνθήκη σύμφωνα με την οποία «φιλτράρονται» οι ομάδες που

προκύπτουν από την GROUP BY κατά τον ίδιο τρόπο που λειτουργεί η πρόταση FILTER σε μη ομαδοποιημένα δεδομένα.

- Αθροιστικές συναρτήσεις: Συναρτήσεις που εφαρμόζουν παραστάσεις σε ομάδες λύσεων. Χρησιμοποιούνται σε περιπτώσεις όπου ο χρήστης που θέτει το ερώτημα θέλει να δει ένα αποτέλεσμα το οποίο υπολογίζεται πάνω σε ένα σύνολο λύσεων και όχι σε μια μεμονωμένη λύση. Οι πιο συνηθισμένες είναι:
  - COUNT: Μετράει τις εμφανίσεις μιας τιμής στη μεταβλητή ή παράσταση ορίσματος.
  - SUM: Υπολογίζει το άθροισμα των διαφορετικών τιμών της μεταβλητής ή παράστασης που δίνεται σαν όρισμα.
  - MIN/MAX: Βρίσκει την ελάχιστη/μέγιστη τιμή της μεταβλητής ή της παράστασης ορίσματος.
  - AVG: Βρίσκει τον μέσο όρο των διαφόρων τιμών της μεταβλητής ή της παράστασης ορίσματος.
- ORDER BY (DESC): Χρησιμοποιείται για να ταξινομήσει τα επιστρεφόμενα αποτελέσματα σύμφωνα με τις τιμές μιας μεταβλητής κατά αύξουσα σειρά. (φθίνουσα αν χρησιμοποιείται ο όρος DESC).
- DISTINCT: Χρησιμοποιείται αμέσως μετά την πρόταση SELECT για να αποτρέψει την εμφάνιση διπλών λύσεων. Μια πιο ελαστική παραλλαγή της είναι η REDUCED, η οποία επιτρέπει ένα άνω όριο πολλαπλών εγγραφών.
- OFFSET: Χρησιμοποιώντας την πρόταση αυτή με όρισμα έναν ακέραιο θετικό αριθμό N ή μια παράσταση που δίνει σχετικό αποτέλεσμα, προκαλεί την εμφάνιση των όποιων λύσεων υπάρχουν μετά τις πρώτες N λύσεις.
- LIMIT: Χρησιμοποιώντας την πρόταση αυτή με όρισμα έναν ακέραιο θετικό αριθμό N ή μια παράσταση που δίνει σχετικό αποτέλεσμα, προκαλεί την εμφάνιση των πρώτων N λύσεων του ερωτήματος.

3.2.6.5 Τελικά σημεία SPARQL (SPARQL endpoints) και ομόσπονδα ερωτήματα (federated queries)

Ένα τελικό σημείο είναι μια υπηρεσία της SPARQL που δίνει τη δυνατότητα σε χρήστες να υποβάλουν ερωτήματα σε απομακρυσμένες πηγές πληροφοριών μέσω της γλώσσας SPARQL. Αποτελούν μια ειδική περίπτωση των τελικών σημείων υπηρεσιών Ιστού (Web Service Endpoints). Δηλαδή πρόκειται για οντότητες μέσα στον Ιστό στις οποίες μπορούν να απευθύνονται μηνύματα υπηρεσιών Ιστού (στη συγκεκριμένη περίπτωση ερωτήματα SPARQL). Πρακτικά αυτό σημαίνει ότι μπορούν να δέχονται ερωτήματα και να στέλνουν τα αποτελέσματα μέσω του πρωτοκόλλου HTTP. Έτσι μπορεί να ανακτηθεί πληροφορία από διάφορα σημεία του Ιστού αν είναι γνωστά τα URIs των τελικών σημείων τους.

Ερωτήματα, ή τμήματα ερωτημάτων μπορούν να υποβάλλονται σε τελικά σημεία χρησιμοποιώντας πριν το κατάλληλο μπλοκ τριάδων την πρόταση SERVICE με όρισμα το πλήρες URI του σημείου.

Η ύπαρξη των τελικών σημείων επιτρέπει την υποβολή ερωτημάτων που συχνά αναζητούν στοιχεία από δύο ή περισσότερες απομακρυσμένες βάσεις πληροφοριών. Τέτοιου είδους ερωτήματα ονομάζονται ομόσπονδα ερωτήματα (federated queries). Ένα δομικό στοιχείο των ερωτημάτων αυτών είναι η ικανότητά τους να θέτουν τα υποερωτήματά τους σε ένα άλλο τελικό σημείο SPARQL όσο αυτά εκτελούνται.

Για την ανάπτυξη των ομόσπονδων ερωτημάτων χρησιμοποιούνται οι προτάσεις SERVICE και BINDINGS. Η πρώτη χρησιμοποιείται, όπως εξηγείται και παραπάνω, για να εισάγει σε ένα ερώτημα URIs τελικών σημείων. Η δεύτερη παρέχει μια μη ταξινομημένη ακολουθία λύσεων, η οποία συνδυάζεται με τα αποτελέσματα της αποτίμησης του ερωτήματος με μια διαδικασία ένωσης. Μπορεί να χρησιμοποιηθεί από μια εφαρμογή για να παρέχει συγκεκριμένες απαιτήσεις για τα αποτελέσματα ενός απλού ερωτήματος. Στην περίπτωση των ομόσπονδων ερωτημάτων χρησιμοποιείται από τη μηχανή SPARQL σε συνδυασμό με την πρόταση SERVICE για να στείλει ένα πιο περιορισμένο ερώτημα στο απομακρυσμένο τελικό σημείο.

Η παραπάνω περίπτωση είναι η επίσημα προτεινόμενη εκδοχή του W3C για τα ομόσπονδα ερωτήματα. Όμως απαιτείται η γνώση των επιμέρους URI των τελικών σημείων (για να δοθούν ως ορίσματα στην πρόταση SERVICE) ή/και η γνώση του τι δεδομένα υπάρχουν σε κάθε τελικό σημείο. Επομένως είναι δυνατή η απόκρυψη της αρχικής μορφής των δεδομένων, αλλά όχι της κατανομής τους. Για να αντιμετωπιστεί αυτή η κατάσταση έχουν δημοσιευθεί ειδικές εφαρμογές, οι οποίες παίρνουν ένα γενικό ερώτημα προς ένα σύνολο τελικών σημείων και με κατάλληλους αλγόριθμους προβαίνουν σε μια κατάτμησή του με την έννοια ότι σχηματίζουν ένα σύνολο από υποερωτήματα, ένα για κάθε τελικό σημείο, ανάλογα με τα δεδομένα που αυτό περιέχει. Χαρακτηριστικά παραδείγματα είναι το DARQ [26], το οποίο βασίζεται στην υπάρχουσα μηχανή επεξεργασίας ερωτημάτων της Jena, αλλά και το FedX [31], το οποίο χρησιμοποιείται και στην εφαρμογή IntegraHEALTH 1.0.

3.3 Σχεσιακές βάσεις δεδομένων

3.3.1 Εισαγωγή – Βασική ορολογία

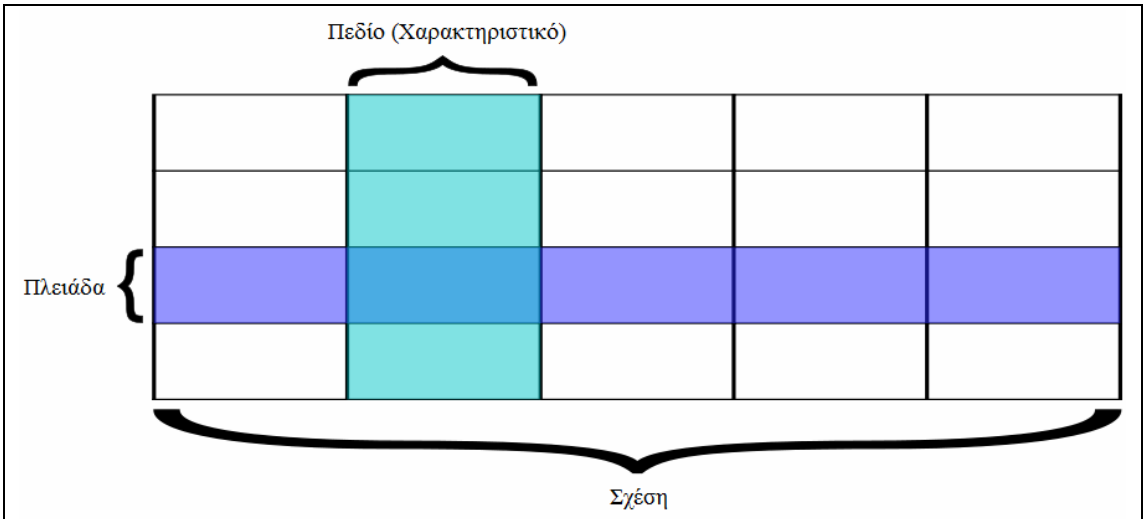
Μια σχεσιακή βάση δεδομένων είναι μια συλλογή δεδομένων οργανωμένη ως ένα σύνολο συσχετισμένων πινάκων από τους οποίους τα δεδομένα μπορούν να προσπελαθούν με πολλούς διαφορετικούς τρόπους (ανάγνωση, εγγραφή, τροποποίηση) χωρίς να είναι απαραίτητη η αναδιοργάνωση των πινάκων [86]. Ο σκοπός της είναι η οργανωμένη αποθήκευση πληροφορίας και η δυνατότητα εξαγωγής της πληροφορίας αυτής, ιδίως σε πιο οργανωμένη μορφή, σύμφωνα με ερωτήματα που τίθενται στη σχεσιακή βάση δεδομένων. Τη σχεσιακή βάση δεδομένων επινόησε ο Edgar Codd το 1970.

Οι ερωτήσεις προς τη βάση δεδομένων γίνονται συνήθως μέσω της γλώσσας SQL (Structured Query Language).

3.3.2 Σχέσεις (Πίνακες)

Μια σχέση ορίζεται σαν ένα σύνολο πλειάδων που έχουν τα ίδια χαρακτηριστικά. Μια πλειάδα αναπαριστά ένα μοναδικό αντικείμενο και τις σχετικές με αυτό πληροφορίες. Τα αντικείμενα αναφέρονται σε οντότητες ή έννοιες του πραγματικού κόσμου. Η σχέση απεικονίζεται σαν ένας πίνακας, ο οποίος οργανώνεται σε γραμμές και στήλες [92] (εικόνα 3.8).

Σε κάθε στήλη ενός πίνακα αντιστοιχίζεται ένα πεδίο, το οποίο αναφέρεται σε μια ιδιότητα ενός αντικειμένου. Οι γραμμές, που ονομάζονται εγγραφές του πίνακα, αντιστοιχίζονται σε αντικείμενα. Η έννοια της σχέσης, αν και αναφέρεται σε όλο τον πίνακα, αποτυπώνεται σε κάθε κελί του, το οποίο στην ουσία απεικονίζει την πρόταση: «Το αντικείμενο Χ έχει μια ιδιότητα Υ με τιμή αυτήν που δίνεται στο κελί».



Εικόνα 3.8: Απεικόνιση βασικής έννοιας σχεσιακής βάσης δεδομένων.

3.3.3 Πεδίο ορισμού και περιορισμοί

Για κάθε χαρακτηριστικό υπάρχει ένα σύνολο από επιτρεπόμενες τιμές που ονομάζεται πεδίο ορισμού του χαρακτηριστικού [92]. Κάθε τιμή του χαρακτηριστικού πρέπει να είναι στοιχείο του συγκεκριμένου συνόλου.

Στο πεδίο ορισμού ενός χαρακτηριστικού μπορούν να τεθούν και επιπλέον περιορισμοί. Για παράδειγμα, ένας περιορισμός σε ένα χαρακτηριστικό ακέραιου αριθμού, μπορεί να το αναγκάσει να λαμβάνει τιμές μόνο από το 1 έως το 10. Περιορισμοί μπορούν να τεθούν σε μια πλειάδα ή ακόμα και σε μια ολόκληρη σχέση.

3.3.4 Κλειδιά

Από την περιγραφή της ενότητας 3.3.2 προκύπτει η ανάγκη για τη διασφάλιση της μοναδικότητας των πλειάδων σε μια σχέση. Αυτό γίνεται με τα κλειδιά που είναι σύνολα χαρακτηριστικών με διαφορετική τιμή για κάθε πλειάδα.

Σύμφωνα με το [92], ένα σύνολο από ένα ή περισσότερα χαρακτηριστικά που όταν χρησιμοποιηθούν μαζί προσδιορίζουν μοναδικά μια σχέση μέσα στη βάση

δεδομένων ονομάζεται υπερ-κλειδί. Τα υπερ-κλειδιά που δεν έχουν υποσύνολα που να είναι κι αυτά υπερ-κλειδιά ονομάζονται υποψήφια κλειδιά μιας σχέσης. Το υποψήφιο κλειδί που επιλέγεται από τον σχεδιαστή της βάσης για να προσδιορίσει τη σχέση ονομάζεται πρωτεύον κλειδί. Τα υπόλοιπα κλειδιά ονομάζονται εναλλακτικά κλειδιά. Τα κλειδιά ονομάζονται απλά ή σύνθετα ανάλογα με το πόσα χαρακτηριστικά (στήλες) χρησιμοποιούνται για την κατασκευή τους. Σαν απλό κλειδί ορίζεται το κλειδί που αποτελείται από μία στήλη, ενώ σαν σύνθετο αυτό που απαρτίζεται από περισσότερες [86].

Μια ειδική περίπτωση κλειδιού αποτελεί το ξένο κλειδί [92]. Πρόκειται για μια στήλη ενός πίνακα, οι τιμές της οποίας αποτελούν το πρωτεύον κλειδί ενός άλλου πίνακα. Έτσι επιτρέπονται συνδέσεις μεταξύ των πινάκων μέσω της πράξης της συνένωσης (join) δημιουργώντας πιο πολύπλοκες σχέσεις σε μια βάση.

3.3.5 Αποθηκευμένες διαδικασίες

Μια αποθηκευμένη διαδικασία είναι ένα σύνολο από προτάσεις SQL που μπορεί να κληθεί ονομαστικά. Με άλλα λόγια είναι εκτελέσιμος κώδικας, ένα πρόγραμμα που εκτελεί μια συγκεκριμένη εργασία και μπορεί να κληθεί κατά τον ίδιο τρόπο που μπορεί να κληθεί σε ένα πρόγραμμα μια συνάρτηση ή μια μέθοδος [80]. Οι αποθηκευμένες διαδικασίες επιτελούν συνηθισμένες λειτουργίες, όπως την εισαγωγή μιας πλειάδας σε μία σχέση, τη συλλογή στατιστικής πληροφορίας, ή την ενθυλάκωση σύνθετης λογικής και υπολογισμών.

3.3.6 Σχεσιακές πράξεις

Τα ερωτήματα που γίνονται σε σχεσιακές βάσεις δεδομένων και οι παραγόμενες σχεσιακές μεταβλητές που βρίσκονται μέσα στις βάσεις, εκφράζονται μέσω της σχεσιακής άλγεβρας. Στην αρχική σχεσιακή άλγεβρα, ο Edgar Codd παρουσίασε οκτώ σχεσιακούς τελεστές σε δύο ομάδες των τεσσάρων. Οι πρώτοι τέσσερις βασίστηκαν στις παραδοσιακές μαθηματικές πράξεις επί συνόλων: ένωση, τομή, διαφορά και καρτεσιανό γινόμενο. Οι αντίστοιχοι τελεστές της SQL είναι οι UNION, INTERSECT, EXCEPT ή MINUS και CROSS JOIN. Οι υπόλοιποι τέσσερις τελεστές περιλαμβάνουν ειδικές πράξεις μόνο για τις σχεσιακές βάσεις [22, 92].

- Επιλογή: Ανακτά πλειάδες από μια σχέση, περιορίζοντας τα αποτελέσματα μόνο σε εκείνα που πληρούν ορισμένα κριτήρια. Η δομή SELECT – WHERE είναι η ισοδύναμη της επιλογής στην SQL.
- Προβολή: Εξάγει μόνο καθορισμένα χαρακτηριστικά από μια πλειάδα ή ένα σύνολο πλειάδων.
- Φυσική συνένωση (natural join): Δύο σχέσεις συνδέονται με βάση τα κοινά τους στοιχεία. Στην SQL χρησιμοποιείται ο τελεστής INNER JOIN.
- Σχεσιακή διαίρεση: Σύνθετη πράξη, η οποία χρησιμοποιεί τις πλειάδες μιας σχέσης για να κατατμήσει μια δεύτερη. Πρόκειται για την αντίθετη πράξη του καρτεσιανού γινομένου.

ΚΕΦΑΛΑΙΟ 4: ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ IntegraHEALTH 1.0

4.1 Γενικό πλάνο

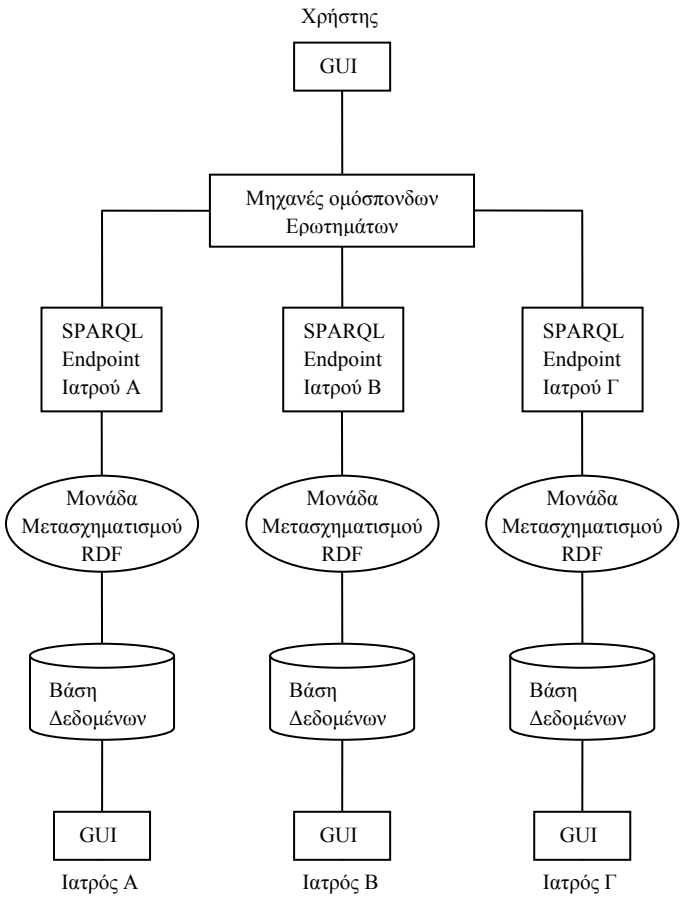
Η IntegraHEALTH 1.0 είναι μια εφαρμογή ολοκλήρωσης δεδομένων ιατροφαρμακευτικής περίθαλψης, η οποία συλλέγει δεδομένα από ανεξάρτητους μεταξύ τους ιατρούς σχετικά με τους ασθενείς τους και αφού τα μετατρέψει σε μορφή RDF, δίνει τη δυνατότητα σε χρήστες να υποβάλουν τα ερωτήματά τους.

Ως μια ενδεικτική υλοποίηση αυτής της ιδέας επιλέχθηκε η περίπτωση όπου τρεις ιατροί (ένας παθολόγος, ένας ακτινολόγος και ένας μικροβιολόγος) τροφοδοτούν το σύστημα με δεδομένα και ένας χρήστης του υποβάλει ομόσπονδα ερωτήματα. Ο κάθε ιατρός έχει στον υπολογιστή του μια βάση δεδομένων για όλους τους ασθενείς του με τα προσωπικά στοιχεία του καθενός και πληροφορίες για την ασθένεια του καθώς και στοιχεία που προκύπτουν από εξετάσεις και μετρήσεις π.χ. ακτινογραφίες ή αιματολογικές εξετάσεις. Η εξαγωγή των πληροφοριών αυτών από τις βάσεις και ο μετασχηματισμός τους στο μοντέλο RDF παρέχει στο σύστημα τα δεδομένα στα οποία θα υποβληθούν τα ερωτήματα.

Σχηματικά, η αρχιτεκτονική του συστήματος απεικονίζεται στην εικόνα 4.1. Όπως μπορεί να φανεί στην εικόνα αυτή, ακολουθείται το πρότυπο ομοσπονδίας πολλαπλών τελικών σημείων SPARQL [40]. Αναλύοντας το σύστημα από πάνω προς τα κάτω, αποτελείται από τα εξής μέρη:

- Γραφικές διεπαφές χρήστη (Graphical User Interfaces, GUI) ιατρών: Καθένας από τους τρεις ιατρούς διαθέτει μια εξειδικευμένη γραφική διεπαφή χρήστη προσαρμοσμένη στις ανάγκες της ειδικότητάς του. Η κάθε διεπαφή αποτελείται από μια φόρμα εισαγωγής δεδομένων, που περιλαμβάνει πεδία για όλα τα απαιτούμενα δημογραφικά στοιχεία ενός ασθενή (όνομα, επώνυμο, πατρώνυμο, κ.τ.λ.), καθώς και για όλα τα στοιχεία που αφορούν στα θέματα υγείας του. Υλοποιεί έτσι το μηχανισμό δημιουργίας, ανάγνωσης, ενημέρωσης και διαγραφής (Create, Read, Update, Delete, CRUD) [58] που απαιτείται για την πρόσβαση κάθε ιατρού στη βάση δεδομένων που του αντιστοιχεί.
- Βάσεις δεδομένων: Πρόκειται για τους χώρους αποθήκευσης των δεδομένων των ασθενών κάθε ιατρού. Από αυτές αντλούνται τα δεδομένα για να σταλούν στις μονάδες μετασχηματισμού RDF της εφαρμογής για περαιτέρω επεξεργασία. Στο κεφάλαιο 1 αναφέρθηκε η χρήση τριών πρότυπων επεξεργασίας και αποθήκευσης των δεδομένων των ιατρών (απλή βάση δεδομένων, DICOM, HL7). Η επιλογή των προτύπων έγινε ανάλογα με τις ειδικότητες των ιατρών. Συγκεκριμένα, ο πρώτος ιατρός (παθολόγος) χρησιμοποιεί μια απλή βάση δεδομένων, αφού οι διαγνώσεις του δεν βασίζονται σε πολύπλοκες εξετάσεις και αριθμητικά δεδομένα. Ο δεύτερος

ιατρός (ακτινολόγος) αποθηκεύει στη βάση του αρχεία DICOM, επειδή χρησιμοποιεί απεικονιστικές τεχνικές για τις διαγνώσεις του, από τις οποίες είναι δυνατόν να εξαχθούν και κάποιες μετρήσεις. Τέλος, ο τρίτος ιατρός (μικροβιολόγος) χρησιμοποιεί το πρότυπο HL7 για να αποθηκεύσει το σύνολο των αριθμητικών αποτελεσμάτων των αιματολογικών και βιοχημικών μετρήσεών του σε αρχεία μηνυμάτων, τα οποία αποθηκεύει στη βάση. Οι δύο τελευταίοι ιατροί, χρησιμοποιούν τη δυνατότητα του συστήματος διαχείρισης σχεσιακών βάσεων δεδομένων MySQL να αποθηκεύει αρχεία σε σειριοποιημένη μορφή. Οι αντίστοιχες γραφικές διεπαφές παρέχουν όλη την απαιτούμενη λειτουργία για τη μετατροπή των αρχείων σε μεγάλα δυαδικά αντικείμενα (Binary Large Objects, BLOBs) [65] κατά την αποθήκευσή τους στη βάση, και την εφαρμογή της αντίστροφης διαδικασίας για την εξαγωγή τους από τη βάση.



Εικόνα 4.1: Αρχιτεκτονική της εφαρμογής IntegraHEALTH 1.0.

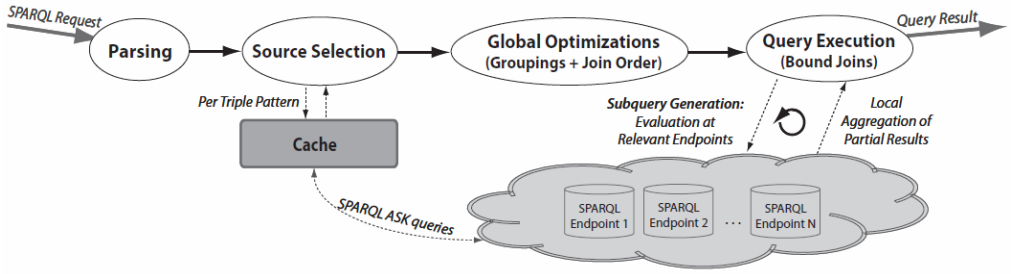
- Μονάδες μετασχηματισμού RDF: Αποτελούν την «καρδιά» του συστήματος. Η εφαρμογή περιλαμβάνει μία για κάθε συνδεδεμένο ιατρό. Είναι υπεύθυνες για τη διαδικασία εξαγωγής των δεδομένων των ιατρών από τις βάσεις, τον μετασχηματισμό τους σε προτάσεις RDF με βάση κάποιες δοθείσες οντολογίες που παρέχουν το επιθυμητό λεξιλόγιο και την αποθήκευση αυτών

των κειμένων σε μεταβλητές μοντέλων της Jena, οι οποίες αποθηκεύονται σε ειδικές, παραμετροποιημένες για τη διαχείριση RDF γράφων βάσεις δεδομένων, γνωστών και ως triple stores. Από τα δεδομένα αυτών των βάσεων, οι οποίες θεωρούνται κομμάτι των μονάδων, λαμβάνεται όλη η απαιτούμενη πληροφορία από τον Joseki RDF server, προκειμένου να δημιουργήσει τα τελικά σημεία SPARQL, ένα για κάθε ιατρό.

- Τελικά σημεία SPARQL: Τα triple stores που δημιουργούνται από τις μονάδες μετασχηματισμού RDF των ιατρών δίνονται ως ορίσματα σε αρχεία ρυθμίσεων του Joseki. Με βάση αυτά τα αρχεία δημιουργούνται τρία στιγμιότυπα (instances) καθένα από τα οποία συνδέεται με ένα triple store. Κάθε στιγμιότυπο δημιουργεί ένα τελικό σημείο SPARQL στο οποίο δημοσιεύει τα RDF δεδομένα των βάσεων στο URL του. Εκεί καταλήγουν τα ερωτήματα του χρήστη και από τα δημοσιευμένα δεδομένα προκύπτουν οι επιθυμητές απαντήσεις. Τα τελικά σημεία θεωρούνται ότι είναι ενεργά συνεχώς, καθώς ενημερώνονται αυτόματα για τις όποιες αλλαγές στα triple stores.
- Γραφική διεπαφή χρήστη/Μηχανές ομόσπονδων ερωτημάτων SPARQL: Πρόκειται για τη φόρμα υποβολής ερωτημάτων του χρήστη προς το σύστημα, η οποία περιλαμβάνει και ξεχωριστό πεδίο για την επιστροφή των απαντήσεων. Συνδέεται με τη μηχανή ομόσπονδων ερωτημάτων που θα επιλέξει ο χρήστης και η οποία θα μεταφέρει το ερώτημα στα τελικά σημεία. Υπάρχει η επιλογή της προκαθορισμένης μηχανής ερωτημάτων της γλώσσας SPARQL και η επιλογή του εργαλείου FedX. Στην πρώτη περίπτωση ο χρήστης πρέπει να συμπεριλάβει προτάσεις SERVICE στο ερώτημά του, καθεμιά από τις οποίες έχει όρισμα ένα URI ενός από τα τελικά σημεία SPARQL που αναφέρονται στο προηγούμενο κομμάτι. Κάτω από κάθε πρόταση SERVICE σχηματίζεται ένα μπλοκ με αγκύλες ({}) στο οποίο τοποθετείται ένα υπόδειγμα γράφου, σχηματίζοντας έτσι ένα υποερώτημα το οποίο θα υποβληθεί μόνο στο τελικό σημείο που αναφέρεται στην πρόταση SERVICE. Ανάλογα με τα ζητούμενα στοιχεία, το ερώτημα μπορεί να περιέχει υποερωτήματα προς ένα ή περισσότερα τελικά σημεία. Ο χρήστης έτσι επιφορτίζεται με το έργο της διατύπωσης ενός ομόσπονδου ερωτήματος, το οποίο απαιτεί γνώση της αρχικής κατανομής των δεδομένων.

  Από την άλλη πλευρά, η χρήση της μηχανής ομόσπονδων ερωτημάτων FedX δεν απαιτεί καμία τέτοια κίνηση. Ο χρήστης υποβάλει το ερώτημά του σαν ένα απλό SPARQL ερώτημα και η ίδια η μηχανή αναλαμβάνει να διασπάσει το ερώτημα σε κατάλληλα υποερωτήματα προς όλα τα διαθέσιμα τελικά σημεία ακολουθώντας μια διαδικασία που αναφέρεται στο [88], δίνεται σχηματικά στην εικόνα 4.2 και αποτελείται από τα εξής βήματα:
  - Ανάλυση (parsing) του ερωτήματος.
  - Επιλογή πηγών (source selection), δηλαδή των τελικών σημείων από τα οποία θα αντληθούν τα δεδομένα μέσω μιας σειράς ερωτημάτων ASK προς το σύνολο των τελικών σημείων που έχουν δοθεί σαν ορίσματα στη

μηχανή. Τα ερωτήματα γίνονται ξεχωριστά για κάθε υπόδειγμα τριάδων. Τα αποτελέσματα της επιλογής αποθηκεύονται προσωρινά (cached).



Εικόνα 4.2: Διαδικασία απόκρισης της μηχανής FedX σε ομόσπονδα ερωτήματα.

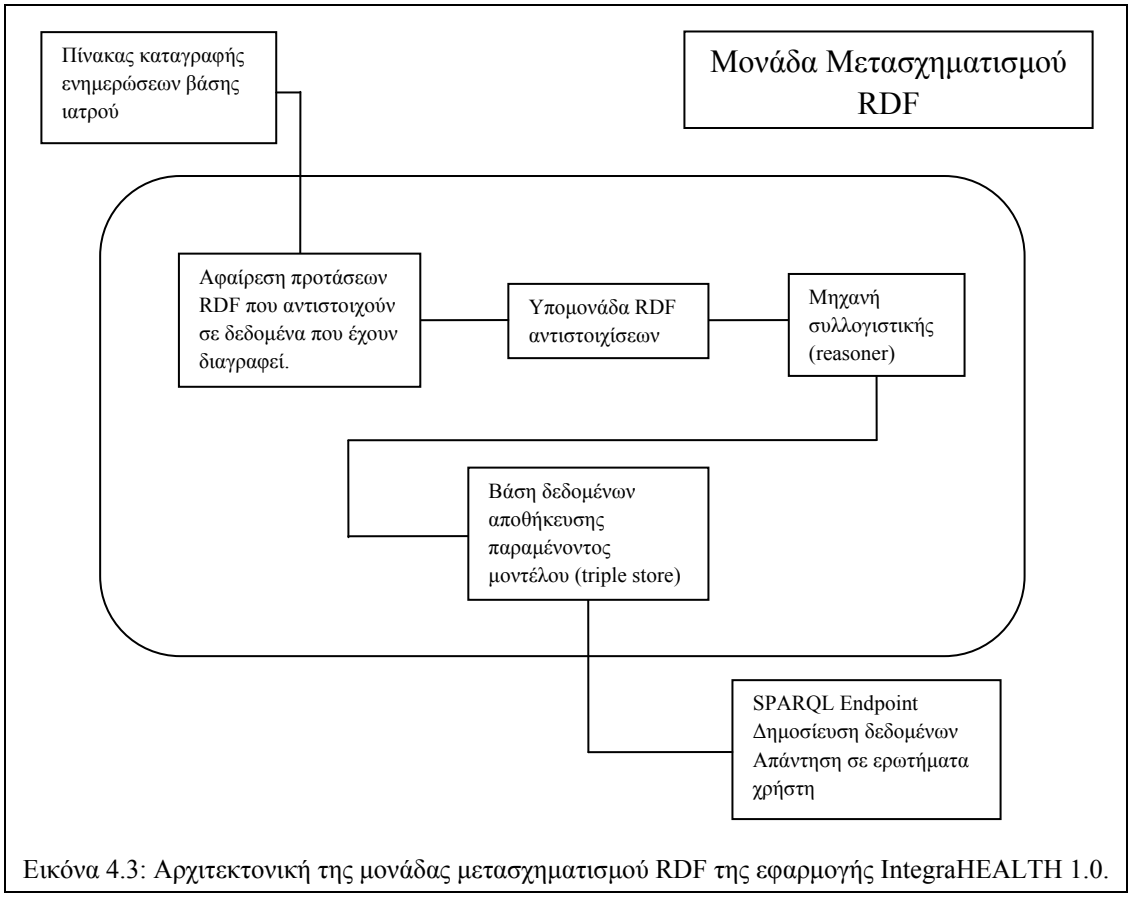
- Εφαρμογή καθολικών βελτιστοποιήσεων με διάταξη συνενώσεων (join order) και την εφαρμογή αποκλειστικών ομάδων (exclusive groups). Για τη διάταξη συνενώσεων γίνεται χρήση ενός βελτιστοποιητή που είναι βασισμένος σε κανόνες, ο οποίος ταξινομεί μια λίστα ορισμάτων συνένωσης (join arguments) σύμφωνα με μια εκτίμηση κόστους βασισμένη σε ευριστικούς κανόνες. Με βάση αυτήν την ταξινόμηση επιλέγονται οι πιο χαμηλού κόστους πράξεις συνένωσης ώστε να σχηματιστεί το τελικό σύνολο αποτελεσμάτων. Οι αποκλειστικές ομάδες είναι σύνολα υποδειγμάτων γράφου που αναφέρονται στην ίδια πηγή δεδομένων (τελικό σημείο SPARQL). Μια τέτοια ομάδα μπορεί να σταλθεί στην πηγή στην οποία αναφέρεται ως ομόσπονδο ερώτημα, μειώνοντας περαιτέρω τον ολικό χρόνο επεξεργασίας του αρχικού ερωτήματος.
- Εκτέλεση ερωτήματος (query execution): Αφού εκτελεστούν όλα τα προηγούμενα βήματα, βάσει των αποτελεσμάτων τους, δημιουργούνται τα κατάλληλα υποερωτήματα, στέλνονται προς τα επιμέρους τελικά σημεία, γίνεται συνάθροιση των επιμέρους αποτελεσμάτων και στη συνέχεια η διαδικασία επαναλαμβάνεται ώσπου να προκύψει το τελικό αποτέλεσμα. Στο στάδιο αυτό χρησιμοποιείται η τεχνική των δεσμευμένων συνενώσεων (bound joins), η οποία βασίζεται στην ομαδοποίηση ενός συνόλου αντιστοιχίσεων σε ένα μοναδικό υποερώτημα με τη χρήση προτάσεων UNION της SPARQL.

4.2 Η μονάδα μετασχηματισμού RDF της IntegraHEALTH 1.0

Ο αριθμός και η πολυπλοκότητα των διαδικασιών που συμβαίνουν στις μονάδες μετασχηματισμού RDF του συστήματος, καθώς και η εμπλοκή διαφόρων εξωτερικών εφαρμογών και εργαλείων (D2RQ, Joseki, κ.τ.λ.) προκειμένου να επιτευχθεί ο μηχανισμός εξαγωγής, μετασχηματισμού και φόρτωσης (ETL) που χρησιμοποιεί η εφαρμογή, επιβάλλουν την λεπτομερέστερη περιγραφή της αρχιτεκτονικής των μονάδων. Η εφαρμογή διαθέτει μια μονάδα για κάθε ιατρό που συνδέεται στο σύστημα που είναι «εξειδικευμένη» για τα δεδομένα κάθε ιατρού (πλήθος, είδος,

λεξιλόγιο αντιστοιχίσεων (οντολογίες)). Ωστόσο, παρά τις όποιες διαφορές, όλες οι μονάδες ακολουθούν την ίδια αρχιτεκτονική, η οποία παρουσιάζεται σχηματικά στην εικόνα 4.3. Στην ίδια εικόνα παρουσιάζεται και η σειρά κλήσης των κλάσεων και μεθόδων των επιμέρους συστατικών της μονάδας κατά την εκτέλεση του μετασχηματισμού δεδομένων.

Πριν αναλυθούν οι επιμέρους διαδικασίες που λαμβάνουν χώρα στη μονάδα, αξίζει να αναφερθεί ότι στις βάσεις των ιατρών χρησιμοποιούνται πίνακες καταγραφής (logs) των συναλλαγών που γίνονται στη βάση. Τα δεδομένα αυτών των πινάκων τροφοδοτούνται στη μονάδα και έτσι ξεκινά η διαδικασία της ανανέωσης των παραμένοντων μοντέλων σε διαφορετικό χρόνο από αυτό της ενημέρωσης των στοιχείων των βάσεων (ενδεικτικά έχει επιλεχθεί η είσοδος ενός χρήστη στο σύστημα, κάτι το οποίο εξασφαλίζει τη διαθεσιμότητα σε μορφή RDF όλων των τελευταίων αλλαγών των επιμέρους βάσεων). Οι πίνακες αυτοί χρησιμοποιούνται διότι λόγω της ETL προσέγγισης που χρησιμοποιεί η εφαρμογή είναι απαραίτητη η ύπαρξη ενός βασικού στοιχείου που θα στηρίζει έναν μηχανισμό αποδοτικής ενημέρωσης των RDF μοντέλων χωρίς να απαιτείται η επαναδημιουργία τους, κάτι που όσο πιο μεγάλες γίνονται οι βάσεις, τόσο πιο πολλούς υπολογιστικούς πόρους θα χρειάζεται. Εξασφαλίζεται έτσι μια σχετική εξοικονόμηση χρόνου και πόρων με την εισαγωγή – εξαγωγή μεμονωμένων τμημάτων στους γράφους, που αναφέρονται σε εγγραφές βάσεων ή αρχεία.



Εικόνα 4.3: Αρχιτεκτονική της μονάδας μετασχηματισμού RDF της εφαρμογής IntegraHEALTH 1.0.

4.2.1 Αφαίρεση προτάσεων RDF που αντιστοιχούν σε δεδομένα που έχουν διαγραφεί

Η επιλογή μιας ETL προσέγγισης για τον σχηματισμό RDF γράφων από τα πρωτογενή δεδομένα των ιατρών επιβάλλει τη δημιουργία ενός μηχανισμού ο οποίος αναφέρθηκε στη προηγούμενη ενότητα και ο οποίος θα φροντίζει για την ενημέρωση του RDF γράφου κάθε ιατρού με τις πιο πρόσφατες αλλαγές στη βάση του τελευταίου, οι οποίες είναι ήδη αποθηκευμένες στον αντίστοιχο πίνακα καταγραφής. Ο μηχανισμός αυτός υλοποιείται σε 2 στάδια, χρησιμοποιώντας διαδοχικά δύο από τις διαθέσιμες λειτουργίες της μονάδας μετασχηματισμού. Η πρώτη λειτουργία για την επίτευξη αυτού του συγχρονισμού είναι η διαγραφή των υπογράφων του ήδη υπάρχοντος RDF γράφου που έχουν σαν ρίζες τους κόμβους που αντιστοιχούν στα πρωτεύοντα κλειδιά διαγραμμένων στοιχείων από τη βάση του ιατρού. Οι προς διαγραφή κόμβοι μπορεί είτε να προέρχονται από εντολές διαγραφής στοιχείων, είτε από εντολές ενημέρωσης, οπότε και θα επαναδημιουργηθούν στο επόμενο βήμα. Στην περίπτωση της ενημέρωσης δεδομένων, η διαγραφή RDF κόμβων και αντικατάστασή τους από νέους είναι αναπόφευκτη, καθώς λόγω των μηχανισμών αποθήκευσης και μετασχηματισμού δεδομένων που έχουν επιλεχθεί είναι αδύνατη η απευθείας ενημέρωση των μοντέλων μέσω άμεσης διαγραφής ή αντικατάστασης συγκεκριμένων τριάδων. Το σκεπτικό πίσω από τη λειτουργία αυτή μπορεί να αποτυπωθεί στον παρακάτω ψευδοκώδικα:

```
{
  actions1 ←εύρεση_πράξεων_ενημέρωσης_ΒΔ;
  actions2 ←εύρεση_πράξεων_διαγραφής_ΒΔ;
  for action in actions1{διάγραψε_σχετικές_RDF_προτάσεις;}
  for action in actions2{διάγραψε_σχετικές_RDF_προτάσεις;}
}
```

Παράδειγμα 4.1: Ψευδοκώδικας αφαίρεσης προτάσεων από έναν RDF γράφο.

4.2.2 Υπομονάδα RDF αντιστοιχίσεων

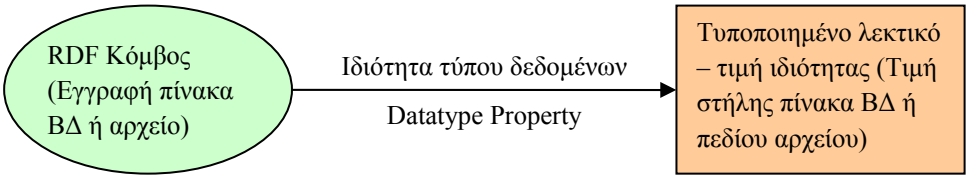
Πρόκειται για το σημαντικότερο τμήμα της μονάδας μετασχηματισμού. Είναι υπεύθυνο για τη μετατροπή των στοιχείων του πινάκα καταγραφής της βάσης του ιατρού σε προτάσεις RDF. Αποτελεί το δεύτερο μέρος του μηχανισμού ενημέρωσης των RDF γράφων των ιατρών. Κάθε μονάδα μετασχηματισμού RDF χρησιμοποιεί και διαφορετικό εργαλείο δημιουργίας RDF προτάσεων στο τμήμα αυτό, ανάλογα με το είδος, την ποσότητα και το πρότυπο αποθήκευσης των δεδομένων κάθε ιατρού. Η βασική λειτουργία όμως είναι η ίδια παντού: οι εγγραφές από τον πίνακα καταγραφής που αναφέρονται σε εισαγωγή ή ενημέρωση δεδομένων δίνονται ως ορίσματα στην υπομονάδα και για κάθε εγγραφή του πίνακα δημιουργείται ένα νέο σύνολο προτάσεων RDF το οποίο ενώνεται με τον ήδη υπάρχοντα γράφο. Ο αντίστοιχος ψευδοκώδικας είναι ο εξής:

```
{
   actions1 ←εύρεση_πράξεων_ενημέρωσης_ΒΔ;
   actions2 ←εύρεση_πράξεων_δημιουργίας_νέας_εγγραφής_ΒΔ;
   for action in actions1{
      σχημάτισε_RDF_προτάσεις;
      ενσωμάτωσε_RDF_προτάσεις_στον_υπάρχοντα_γράφο;
   }
   for action in actions2{
      σχημάτισε_RDF_προτάσεις;
      ενσωμάτωσε_RDF_προτάσεις_στον_υπάρχοντα_γράφο;
   }
}
```

Παράδειγμα 4.2: Γενικευμένος ψευδοκώδικας λειτουργίας υπομονάδας RDF αντιστοιχίσεων.

Ο σχηματισμός προτάσεων RDF γίνεται σε κάθε περίπτωση ως εξής: Κάθε πίνακας βάσης και κάθε είδος (τύπος) αποθηκευμένου αρχείου αντιστοιχίζονται σε μια κλάση μιας οντολογίας και κάθε στήλη ενός πίνακα ή επιμέρους πεδίο ενός αποθηκευμένου αρχείου αντιστοιχίζονται σε μια ιδιότητα μιας οντολογίας. Για κάθε εγγραφή ή αρχείο σχηματίζεται ένας νέος RDF κόμβος με τόσες ιδιότητες όσες και οι στήλες ή τα πεδία αντίστοιχα. Οι τιμές αυτών των ιδιοτήτων βασίζονται στις τιμές των στηλών ή των πεδίων. Το όνομα του κόμβου προέρχεται από το πρωτεύον κλειδί του πίνακα της βάσης ή το όνομα του αρχείου, το οποίο φροντίζεται να παρέχεται στα αρχεία κατά μοναδικό τρόπο.

Υπάρχουν δύο επιλογές αντιστοίχισης τιμών ανάλογα με τον τύπο της διαθέσιμης ιδιότητας. Για τις ιδιότητες τύπου δεδομένων (datatype properties) η τιμή της στήλης ή του πεδίου συνδυάζεται με τον κατάλληλο τύπο δεδομένων σχηματίζοντας ένα τυποποιημένο λεκτικό. Το λεκτικό αυτό αποτελεί και την τιμή της ιδιότητας. Στο σημασιολογικό δίκτυο που δημιουργείται οι ιδιότητες αυτές απεικονίζονται γραφικά ως εξής:

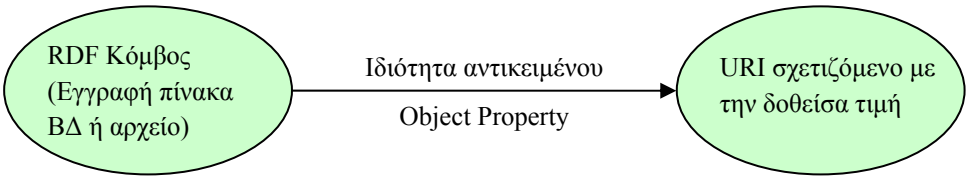


Εικόνα 4.4: Αναπαράσταση RDF πρότασης που σχηματίζεται για μια ιδιότητα τύπου δεδομένων.

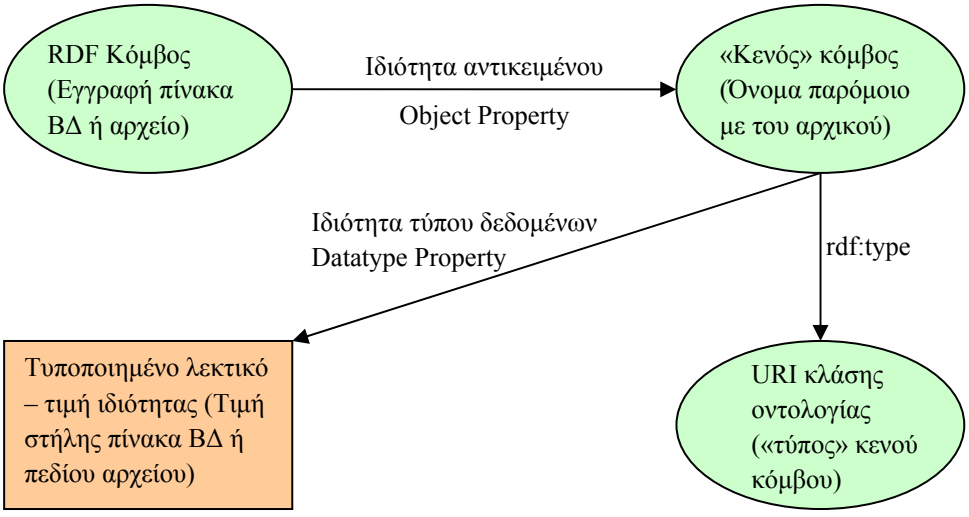
Όπως φαίνεται και από την παραπάνω εικόνα, στο ήδη υπάρχον RDF κείμενο προστίθεται μια μόνο πρόταση για κάθε ιδιότητα τύπου δεδομένων, η οποία είναι η <RDF_Κόμβος, Ιδιότητα_τύπου_δεδομένων, Τιμή_στήλης_πίνακα/πεδίου_αρχείου>. Η περίπτωση του Α.Μ.Κ.Α. ενός ασθενή είναι ένα χαρακτηριστικό παράδειγμα ιδιότητας τύπου δεδομένων.

Οι ιδιότητες αντικείμένου (object properties) αντιμετωπίζονται με δύο τρόπους:

- Δίνεται απευθείας ένα URI ως τιμή στην ιδιότητα. Το URI αναφέρεται σε στοιχείο μιας διαθέσιμης οντολογίας που σχετίζεται με τη δοθείσα τιμή της στήλης του πίνακα ή του πεδίου του αρχείου. Έτσι στο RDF κείμενο προστίθεται μία πρόταση, η <RDF_Κόμβος, Ιδιότητα_αντικειμένου, URI_σχετιζόμενο_με_τιμή> με αναπαράσταση αυτή της εικόνας 4.5α.
- Δημιουργείται ένας νέος κόμβος ο οποίος αν και έχει την ίδια λειτουργικότητα με έναν κενό κόμβο, εντούτοις ονοματίζεται με βάση το όνομα του αρχικού κόμβου. Ο «κενός» αυτός κόμβος έχει δύο ιδιότητες: την rdf:type, μια ιδιότητα αντικειμένου που λαμβάνει σαν τιμή ένα URI μιας κλάσης μιας διαθέσιμης οντολογίας, της οποίας θεωρείται στιγμιότυπο και μια ιδιότητα τύπου δεδομένων με αντικείμενο την τιμή στήλης πίνακα ΒΔ ή πεδίου αρχείου. Το σημασιολογικό δίκτυο που προστίθεται σε αυτήν την περίπτωση δίνεται στην εικόνα 4.5β.



Εικόνα 4.5α: Αναπαράσταση RDF πρότασης που σχηματίζεται για μια ιδιότητα αντικειμένου με απευθείας ανάθεση URI.



Εικόνα 4.5β: Σημασιολογικό δίκτυο που σχηματίζεται για την ανάθεση μιας τιμής-λεκτικού σε μια ιδιότητα αντικειμένου.

Από τους δύο παραπάνω τρόπους, ο πρώτος χρησιμοποιείται στη βάση δεδομένων του παθολόγου, όπου σε ξεχωριστό πίνακα δίνονται URIs οντολογιών που αναφέρονται σε ασθένειες, ενώ ο δεύτερος χρησιμοποιείται για κάθε στήλη βάσης ή πεδίο αρχείου που περιέχει τιμές που έχουν δοθεί από τους ιατρούς, όπως για

παράδειγμα οι μετρήσεις σωματικού λίπους (ακτινολόγος) και πλήθους ερυθρών αιμοσφαιρίων (μικροβιολόγος) ή η διεύθυνση του ασθενή (δημογραφικό στοιχείο κοινό για όλους τους ιατρούς).

Ανάλογα με τον τρόπο που ο κάθε ιατρός αποθηκεύει τα δεδομένα του χρησιμοποιείται ένα από τα παρακάτω εργαλεία:

- Για τις περιπτώσεις όπου χρησιμοποιούνται αρχεία (ακτινολόγος – DICOM, μικροβιολόγος – HL7), η εφαρμογή ανακτά τα αρχεία των οποίων τα ονόματα βρίσκονται στους πίνακες καταγραφής. Κάθε αρχείο αναλύεται και σχηματίζεται μια λίστα με όλες τις τιμές των επιμέρους πεδίων που περιέχει. Κάθε στοιχείο της λίστας αντιστοιχίζονται σε μια RDF πρόταση ή ένα σημασιολογικό δίκτυο όπως αναφέρεται παραπάνω. Όλες οι τιμές που περιέχονται στα αρχεία είναι απλά λεκτικά, τα οποία κατά την εκτέλεση του κώδικα που κάνει τις αντιστοιχίσεις, μετατρέπονται σε τυποποιημένα. Ο αντίστοιχος ψευδοκώδικας είναι ο εξής:

```
{
  files ← Αρχεία_προς_εισαγωγή_ή_ενημέρωση
  for file in files{
    ανέλυσε(parse)_αρχείο;
    φτιάξε_κόμβο_που_αναφέρεται_σε_αρχείο;
    list ← λίστα_τιμών_με_τιμές_πεδίων_αρχείου;
    for element in list{
      if (ιδιότητα_αντιστοιχίας = Ιδιότητα_τύπου_δεδομένων){
        κατασκεύασε_RDF_πρόταση;
        πρόσθεσε_πρόταση_σε_RDF_κόμβο_αρχείου;
      }
      if (ιδιότητα_αντιστοιχίας = Ιδιότητα_αντικειμένου){
        κατασκεύασε_RDF_σημασιολογικό_δίκτυο;
        πρόσθεσε_σημασιολογικό_δίκτυο_σε_RDF_κόμβο_αρχείου;
      }
    }
  }
}
```

Παράδειγμα 4.3: Ψευδοκώδικας δημιουργίας RDF προτάσεων από τα επιμέρους πεδία ενάς αρχείου.

- Για την περίπτωση της βάσης δεδομένων (παθολόγος) χρησιμοποιείται το εργαλείο D2RQ το οποίο μετατρέπει σχεσιακές βάσεις δεδομένων σε έγγραφα RDF. Η βασική ιδέα πίσω από αυτήν την εφαρμογή είναι ότι η έννοια της σχέσης, η απεικόνισή της με έναν πίνακα, ο ορισμός της σημασίας των γραμμών, των στηλών και του μεμονωμένου κελιού μπορεί να ωθήσει κάποιον στο να εκφράσει ότι στην ουσία μια σχέση (πίνακας) μιας βάσης δεδομένων αποτελείται από προτάσεις τριών στοιχείων (υποκείμενο – κατηγόρημα – αντικείμενο) (κελιά). Επομένως κάθε πίνακας σχεσιακής βάσης

δεδομένων μπορεί να μεταφραστεί σε τριάδες κατά RDF, να αποθηκευθεί και να επεξεργαστεί κατάλληλα.

Για παράδειγμα, σε έναν απλό πίνακα «στοιχεία ασθενή» με πρωτεύον κλειδί τον Α.Μ.Κ.Α. του κάθε ασθενή και με τις υπόλοιπες στήλες να είναι οι Όνομα, Επώνυμο, Πατρώνυμο, Ασθένεια, κάθε διαφορετική εγγραφή (γραμμή του πίνακα) αναφέρεται σε διαφορετικό ασθενή. Το πρωτεύον κλειδί μπορεί να χρησιμοποιηθεί στο όνομα – URI ενός RDF κόμβου, ο οποίος θα είναι στιγμιότυπο της κλάσης μιας οντολογίας στην οποία αντιστοιχίζεται ο πίνακας, ως σύνολο αντικειμένων που έχουν όλα τα ίδια χαρακτηριστικά (εγγραφές). Κάθε στήλη του πίνακα αντιστοιχίζεται σε μια ιδιότητα μιας οντολογίας και η τιμή του κάθε κελιού είναι η τιμή του κόμβου για τη συγκεκριμένη ιδιότητα. Έτσι από έναν τέτοιο πίνακα προκύπτει το παρακάτω σύνολο προτάσεων για κάθε εγγραφή.

<URI_κόμβου_που_περιέχει_τιμή_κλειδιού, Ιδιότητα_Ονόματος, Τιμή_Ονόματος>.
<URI_κόμβου_που_περιέχει_τιμή_κλειδιού, Ιδιότητα_Επώνυμου, Τιμή_Επώνυμου>.
<URI_κόμβου_που_περιέχει_τιμή_κλειδιού, Ιδιότητα_Πατρώνυμου, Τιμή_Πατρώνυμου>.
<URI_κόμβου_που_περιέχει_τιμή_κλειδιού, Ιδιότητα_Α.Μ.Κ.Α., Τιμή_Α.Μ.Κ.Α.>.
<URI_κόμβου_που_περιέχει_τιμή_κλειδιού, Ιδιότητα_Ασθένειας, Τιμή_Ασθένειας>.

Το εργαλείο D2RQ έχει το δικό του χώρο ονομάτων ο οποίος χρησιμοποιείται για να ορίσει RDF έγγραφα σε μορφή N3 τα οποία ομαδοποιούν τις αντιστοιχίσεις που επιλέγει ο σχεδιαστής για κάθε στήλη και κάθε πίνακα. Η «ιεραρχία» δημιουργίας ενός τέτοιου εγγράφου είναι η εξής: Αρχικά ορίζεται ένα αντικείμενο το οποίο περιλαμβάνει στοιχεία όπως το είδος της πλατφόρμας της βάσης (MySQL, Oracle, κ.τ.λ.) και το URI της. Στη συνέχεια ορίζονται αντικείμενα αντιστοίχισης κλάσεων που αντιστοιχίζουν τους πίνακες της βάσης σε κλάσεις οντολογιών. Τα στοιχεία αυτά αναφέρεται ότι ανήκουν στο αρχικό στοιχείο περιγραφής της βάσης. Τέλος κάθε στήλη ενός πίνακα αντιστοιχίζεται σε μια ιδιότητα οντολογίας με μια σειρά αντικειμένων αντιστοίχισης ιδιοτήτων. Όλα τα στοιχεία ιδιοτήτων πρέπει να ανήκουν σε κάποιο στοιχείο αντιστοίχισης κλάσης.

Όσον αφορά τη μονάδα μετασχηματισμού, η διαδικασία δημιουργίας RDF γράφου από τη σχεσιακή βάση περιορίζεται μόνο στην κλήση μιας μεθόδου του εργαλείου D2RQ με όρισμα το όνομα του ειδικού αρχείου αντιστοιχίσεων. Κατόπιν το ίδιο το εργαλείο αναλαμβάνει να εκτελέσει τις απαιτούμενες μετατροπές.

4.2.3 Μηχανή συλλογιστικής (Reasoner)

Κάθε μονάδα μετασχηματισμού περιλαμβάνει και την δική της μηχανή συλλογιστικής, ο ρόλος της οποίας είναι να εφαρμόζεται επί του συνόλου των παραχθέντων RDF προτάσεων από την υπομονάδα αντιστοιχίσεων και του λεξιλογίου (οντολογίες) που η τελευταία χρησιμοποιεί. Το αποτέλεσμα είναι η παραγωγή νέας γνώσης, μέσω της προσθήκης προτάσεων RDF που σχηματίζονται με βάση τους κανόνες που περιλαμβάνει η μηχανή συλλογιστικής και το μηχανισμό που περιγράφεται στην ενότητα 3.2.5 (ΑΝ το RDF σύνολο περιέχει συγκεκριμένες τριάδες ΤΟΤΕ πρόσθεσε στο RDF σύνολο συγκεκριμένες επιπλέον τριάδες).

Για τους σκοπούς της εργασίας χρησιμοποιείται ένας συνδυασμός δύο από τις διαθέσιμες μηχανές συλλογιστικής της Jena. Συγκεκριμένα, χρησιμοποιείται η ενσωματωμένη RDFS μηχανή μαζί με κανόνες που έχουν οριστεί χειροκίνητα σε ξεχωριστό αρχείο κανόνων, το οποίο και δίνεται ως όρισμα για την κατασκευή της συνδυασμένης μηχανής.

Η λύση που προσφέρει η Jena δεν είναι μοναδική, καθώς υπάρχουν στο διαδίκτυο αρκετές άλλες ελεύθερες μηχανές συλλογιστικής όπως ο Pellet [83], ο HermiT [44] και ο FaCT [24]. Πολλοί μάλιστα είναι συμβατοί με την Java, κάτι το οποίο σημαίνει ότι μπορούν να κληθούν από τις μεθόδους μιας εφαρμογής, αρκεί να υπάρχουν οι κατάλληλες αναφορές στα αρχεία .jar των μηχανών.

4.2.4 Βάση δεδομένων αποθήκευσης παραμένοντος μοντέλου (Triple store)

Η κάθε μονάδα μετασχηματισμού αποθηκεύει τα RDF δεδομένα της σε ένα triple store το οποίο είναι μια ειδικά διαμορφωμένη βάση υποδοχής RDF προτάσεων, όπως εξηγήθηκε και στα προηγούμενα. Οι βάσεις αυτές χρησιμοποιούνται από τον Joseki RDF Server κατά την κλήση του τελευταίου για το σχηματισμό τελικών σημείων SPARQL. Τα δεδομένα που περιέχουν δημοσιεύονται στο τελικό σημείο και γίνονται διαθέσιμα στους εξωτερικούς χρήστες.

Η εφαρμογή IntegraHEALTH 1.0 χρησιμοποιεί το ενσωματωμένο triple store που διαθέτει η Jena. Όμως, όπως και με την περίπτωση των μηχανών συλλογιστικής, στο διαδίκτυο υπάρχουν διαθέσιμα και άλλα triple stores όπως το Virtuoso [105].

4.3 Παραδοχές υλοποιημένης εφαρμογής

Για τη διευκόλυνση της ανάπτυξης της εφαρμογής IntegraHEALTH 1.0, κρίθηκε απαραίτητο να γίνουν ορισμένες παραδοχές, καθώς το ζητούμενο είναι να δοθεί έμφαση στο μηχανισμό ολοκλήρωσης των μετασχηματισμένων σε RDF δεδομένων:

1. Στο σύστημα είναι δυνατόν να είναι ταυτόχρονα συνδεδεμένοι δύο ή τρεις ιατροί, αλλά και ο χρήστης. Σε μια τέτοια περίπτωση, ο χρήστης μπορεί να υποβάλει ερωτήματα και να λάβει απαντήσεις σε μια έκδοση των RDF γράφων που είναι αποθηκευμένοι στα triple stores, η οποία δεν λαμβάνει υπόψιν τις αλλαγές που γίνονται από τους ιατρούς την ώρα που υποβάλλονται

τα ερωτήματα. Ο μηχανισμός ETL πρέπει να επανεκτελεστεί, προκειμένου να ληφθούν υπόψιν όλες οι αλλαγές.

2. Το σύστημα κατασκευάζεται θεωρώντας ως προϋπάρχοντα τα συστήματα αποθήκευσης δεδομένων των ιατρών και προσαρμόζεται σε αυτά, χωρίς να υπάρχει απαίτηση από τους ιατρούς να αλλάξουν οτιδήποτε πάνω τους.
3. Τα όποια απεικονιστικά μηχανήματα των ιατρών είναι ρυθμισμένα ώστε να δίνουν ένα μοναδικό όνομα αρχείου σε κάθε παραγόμενη εικόνα, το οποίο δεν περιλαμβάνει κάποιο από τα δημογραφικά δεδομένα του ασθενή, όντας έτσι σύμφωνο με το μέρος 3.10 του προτύπου DICOM [72].
4. Με εξαίρεση τα πακέτα που περιέχουν τις κλάσεις των γραφικών διεπαφών χρήστη των ιατρών, όλες οι άλλες κλάσεις είναι ορατές από τα πακέτα που υλοποιούν τις μονάδες μετασχηματισμού RDF. Με άλλα λόγια επιλέγεται να δοθούν «υπερεξουσίες» στις μονάδες.
5. Το σύστημα λειτουργεί έχοντας ως «μονάδα ομαδοποίησης» των πληροφοριών την ιατρική επίσκεψη καθώς ένας ασθενής μπορεί να επισκεφθεί πολλές φορές πολλούς ιατρούς, ενώ η επίσκεψη είναι μία κατ' άτομο, ιατρό και ημέρα (σπανίως έχει νόημα να επισκεφθεί κάποιος τον ίδιο ιατρό δύο φορές σε μία ημέρα για διαφορετικές ασθένειες). Έτσι προσφέρει περισσότερες ευκολίες στη μοντελοποίηση.
6. Το λεξιλόγιο που χρησιμοποιείται για το μετασχηματισμό των πληροφοριών (οντολογίες) κάθε ιατρού είναι προφορτωμένο στο αντίστοιχο triple store που θα φιλοξενεί το παραμένον μοντέλο κάθε ιατρού.

ΚΕΦΑΛΑΙΟ 5:
ΥΛΟΠΟΙΗΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ IntegraHEALTH 1.0

5.1 Προγραμματιστικά και αναπτυξιακά εργαλεία

Πριν αρχίσει η περιγραφή των στοιχείων της εφαρμογής, παρατίθενται στις επόμενες υποενότητες όλα τα μη ευρέως γνωστά προγραμματιστικά εργαλεία τα οποία έχουν αναφερθεί μόνο ονομαστικά στην εισαγωγή αλλά και στο προηγούμενο κεφάλαιο.

5.1.1 Το πλαίσιο Σημασιολογικού Ιστού Jena

Η Jena [6] είναι ένα πλαίσιο ανοιχτού κώδικα της Java που χρησιμοποιείται για την ανάπτυξη εφαρμογών Σημασιολογικού Ιστού. Παρέχει ένα προγραμματιστικό περιβάλλον για τις RDF, RDFS, OWL και SPARQL, μέσω μιας σειράς πακέτων κλάσεων. Μερικές από τις δυνατότητές του περιλαμβάνουν:

- Μια διεπαφή προγραμματισμού εφαρμογών RDF (RDF API).
- Ανάγνωση και εγγραφή αρχείων RDF σε μορφή RDF/XML, N3 και N-Triple.
- Μια διεπαφή προγραμματισμού εφαρμογών OWL (OWL API).
- Παραμένουσα (persistent) αποθήκευση και αποθήκευση στη μνήμη (in-memory).
- Μηχανή ερωτημάτων SPARQL.
- Εφαρμογή συλλογιστικής με έτοιμες μηχανές συλλογιστικής (reasoners) ή/και με ορισμό κανόνων από το χρήστη.

5.1.1.1 Βασικές λειτουργίες χειρισμού RDF

Τα βασικά δομικά στοιχεία της Jena για την κατασκευή RDF συνόλων είναι οι διεπαφές Model (μοντέλο), Resource (πόρος) και Property (ιδιότητα). Ένας RDF γράφος αντιστοιχίζεται σε μια μεταβλητή τύπου Model. Οι διαθέσιμες λειτουργίες ενός μοντέλου είναι οι εξής:

- Εισαγωγή προτάσεων: Γίνεται είτε με απευθείας προσθήκη προτάσεων, ή με τμηματική δημιουργία τους μέσω των εντολών των διεπαφών Resource, Property και Literal (λεκτικό).
- Ανάκτηση στοιχείων: Παρέχεται μια σειρά μεθόδων list, η οποία ανακτά προτάσεις ή συστατικά τους που αντιστοιχούν σε συγκεκριμένες συνθήκες.
- Εμφάνιση αποτελεσμάτων στο χρήστη: Ένα μοντέλο μπορεί να γραφτεί ολόκληρο σε ένα αρχείο εξόδου ή στην οθόνη του υπολογιστή (σε μορφές όπως έγγραφο XML), ή να ληφθεί αναλυτική λίστα όλων των προτάσεων του με την εντολή listStatements και να εμφανιστεί στην οθόνη.

5.1.1.2 Βασικά αντικείμενα χειρισμού OWL

Επειδή η γλώσσα OWL έχει περισσότερες λειτουργίες και πιο πλούσιο λεξιλόγιο από τις RDF/RDFS, η Jena παρέχει γι' αυτήν ένα εξειδικευμένο σύνολο λειτουργιών. Ένα μοντέλο OWL (μοντέλο οντολογίας) περιγράφεται από ένα αντικείμενο τύπου OntModel (αντίστοιχα υπάρχουν και οι διεπαφές OntResource, OntProperty). Η κατασκευή ενός προεπιλεγμένου μοντέλου υποστηρίζει τα εξής:

- Το πλήρες λεξιλόγιο της γλώσσας OWL.
- Αποθήκευση του μοντέλου στη μνήμη.
- Συλλογιστική επιπέδου RDFS, η οποία παράγει συμπεράσματα βασισμένη στις σχέσεις ιεραρχίας μεταξύ κλάσεων και ιδιοτήτων.

Ανάλογα με τις ανάγκες της εφαρμογής για χρήση διαφορετικού λεξιλογίου, ή επιπέδου συλλογιστικής, δίνεται και το κατάλληλο όρισμα στην μέθοδο κατασκευαστή ενός OntModel, ModelFactory.createOntologyModel (OntModelSpec.OWL_MEM);. Στο Παράρτημα Δ δίνεται μια πλήρη λίστα με τις πιθανές τιμές του συγκεκριμένου ορίσματος.

Τα μοντέλα οντολογίας μπορούν να εισάγουν ολόκληρες οντολογίες ή τμήματα αυτών από τρίτες πηγές ώστε να ληφθούν τα όποια απαραίτητα στοιχεία. Αυτό γίνεται με εντολή read με όρισμα μια διαδρομή αρχείου (filepath) ή ένα URL στο οποίο υπάρχει ο χώρος ονομάτων της οντολογίας. Οι πληροφορίες μπορούν να προσπελαστούν με μια σειρά μεθόδων όπως getOntProperty, getOntResource, κ.τ.λ.

Τέλος, διατίθεται ένα σύνολο μεθόδων που καλύπτει όλο το φάσμα των επιπλέον λειτουργιών της OWL. Ενδεικτικά παραδείγματα αποτελούν οι μέθοδοι getSomeValuesFromRestriction, createTransitiveProperty, getObjectPropery, και createEnumeratedClass.

5.1.1.3 Παραμένοντα μοντέλα (Persistent models)

Η προκαθορισμένη συμπεριφορά της Jena απέναντι σε μοντέλα RDF/RDFS/OWL είναι να υποστηρίζει την αποθήκευσή τους στη μνήμη του υπολογιστή. Κάθε φορά που χρειάζεται να ανακτηθεί ή να αποθηκευθεί πληροφορία σε ένα τέτοιο μοντέλο εκτελούνται οι απαραίτητες διαδικασίες, ενώ το μοντέλο εξακολουθεί να παραμένει ανοιχτό και να καταλαμβάνει το δικό του χώρο στη μνήμη του υπολογιστή μέχρι η εφαρμογή να τερματιστεί. Αυτό όμως έχει σοβαρές επιπτώσεις στην γενική ταχύτητα εκτέλεσης της εφαρμογής.

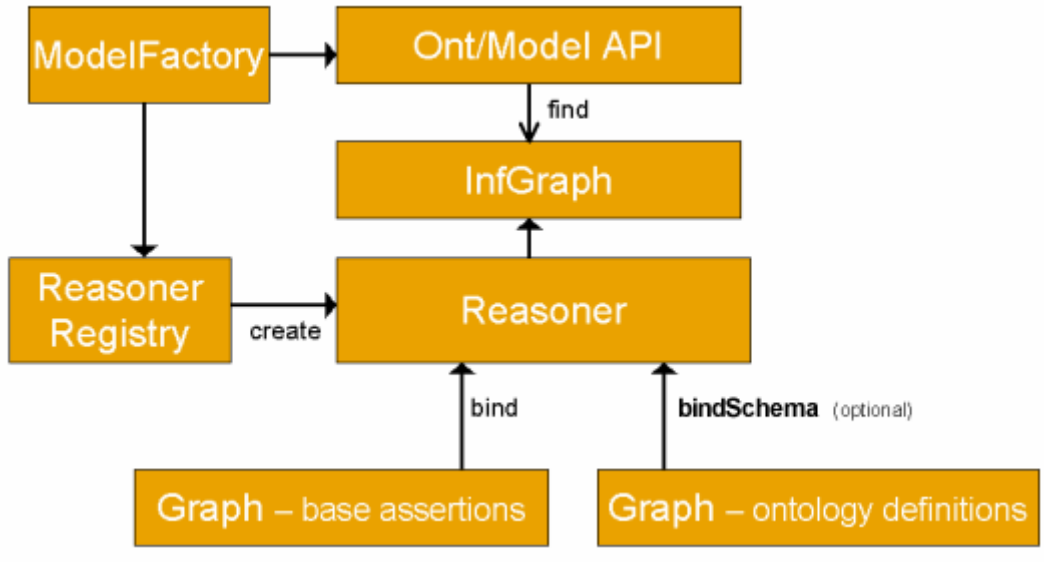
Η λύση δίνεται με το να αποθηκεύονται τα μοντέλα σε σειριοποιημένη μορφή σε βάσεις δεδομένων. Η διαδικασία αυτή ονομάζεται διατήρηση (persistence). Ο σχεδιαστής μπορεί να χρησιμοποιήσει οποιαδήποτε πλατφόρμα ανάπτυξης βάσεων (MySQL, Oracle, κ.τ.λ.) και αφού τη συνδέσει με την Java μέσω του JDBC, μπορεί να δημιουργήσει ένα ή περισσότερα μοντέλα με τη μέθοδο createModel. Κατά την κατασκευή της η βάση δεδομένων μπορεί να είναι απολύτως κενή, καθώς η createModel δημιουργεί όσους πίνακες χρειάζονται για την αποθήκευση του μοντέλου και των πληροφοριών του. Μετά την αποθήκευση, ο σχεδιαστής μπορεί να

«αποσυνδεθεί» από το μοντέλο και τη βάση με εντολές close και να επανέλθει σε αυτό όποτε θέλει στο μέλλον με τη μέθοδο openModel [51]. Έτσι, κατά τη διάρκεια εκτέλεσης ενός προγράμματος δεσμεύεται κάθε φορά τόση μνήμη όση χρειάζεται για να προσπελαστεί το μοντέλο στη βάση.

5.1.1.4 Υποστήριξη συλλογιστικής

Η Jena υποστηρίζει διαδικασίες συλλογιστικής στα μοντέλα της. Ο σχετικός μηχανισμός περιλαμβάνει ένα σύνολο από προεπιλεγμένες μηχανές συλλογιστικής (reasoners), δυνατότητες εισαγωγής εξωτερικών μηχανών, αλλά και εργαλεία για κατασκευή μηχανών μέσω κανόνων που ορίζει ο σχεδιαστής.

Η βασική διαδικασία συμπερασμού περιλαμβάνει τη χρήση της διεπαφής InfModel, η οποία επεκτείνει την Model. Ένα InfModel δημιουργείται από ένα Model ή OntModel, όταν συνδυάζεται με μια μηχανή συλλογιστικής. Οι μηχανές συλλογιστικής στην Jena μπορούν να αποτελέσουν ξεχωριστά αντικείμενα και να τροποποιηθούν κατά βούληση. Η σύνδεση μεταξύ των δύο συστατικών γίνεται με τη μέθοδο bindSchema. Μέσω άλλων μεθόδων bind, μια μηχανή συλλογιστικής μπορεί να συνδεθεί σε οποιοδήποτε σύνολο εξωτερικών RDF δεδομένων. Οι πρόσθετες προτάσεις που θα προκύψουν από τον συμπερασμό, μπορεί να υπολογιστούν από την αρχή όταν καλείται το μοντέλο, να υπολογίζονται δυναμικά, ακολουθώντας τις μεταβολές του, ή να υπολογίζονται μόνο κατά παραγγελία και να αποθηκεύονται κατάλληλα.



Εικόνα 5.1: Δομή του μηχανισμού συμπερασμού της Jena.

Οι διαθέσιμες μηχανές συλλογιστικής στην Jena είναι οι εξής:

- Μεταβατική μηχανή: Παρέχει υποστήριξη για να αποθηκεύονται και να διατρέχονται πλέγματα κλάσεων και ιδιοτήτων. Υλοποιεί μόνο τις

μεταβατικές και ανακλαστικές ιδιότητες των rdfs:subPropertyOf και rdfs:subClassOf.

- Μηχανή RDFS: Υλοποιεί ένα διαμορφώσιμο υποσύνολο των RDFS σχέσεων μεταξύ προτάσεων.
- Μηχανές OWL, OWL Mini, OWL Micro: Ένα σύνολο χρήσιμων άλλα ανολοκλήρωτων υλοποιήσεων υποσυνόλων της OWL.
- Μηχανή συλλογιστικής κανόνων: Μια μηχανή βασισμένοη σε κανόνες ορισμένους από το χρήστη.

Η μηχανή συλλογιστικής OWL παρουσιάζει μια ιδιαιτερότητα από τις άλλες εσωτερικές μηχανές της Jena. Λόγω της απαιτούμενης ισορροπίας μεταξύ εκφραστικής ισχύος και της αποδοτικής υποστήριξης συλλογισμών, η Jena διαθέτει τρεις μηχανές, η κάθεμια με τις δικές της δυνατότητες:

- Η πλήρης μηχανή OWL: Παρέχει πλήρη υποστήριξη σε όλες τις λειτουργίες της γλώσσας. Είναι όμως αρκετά αργή κατά την διαδικασία συμπερασμού και πρέπει να χρησιμοποιείται όταν είναι απολύτως απαραίτητη.
- OWL Mini: Είναι σχεδόν ισοδύναμη με την προηγούμενη, καθώς παραλείπει μόνο τις κανονικές συνεπαγωγές (forward entailments) από τους περιορισμούς minCardinality και someValuesFrom. Αποφεύγει δηλαδή την εμπλοκή της με κενούς κόμβους, οι οποίοι μπορεί να οδηγήσουν σε άπειρες επεκτάσεις.
- OWL Micro: Είναι η «μικρότερη» και πιο γρήγορη από τις τρεις μηχανές. Υποστηρίζει πλήρως το μοντέλο RDFS συν τα βασικά στοιχεία ιδιοτήτων (ενότητα 3.2.4.2) καθώς και τα intersectionOf, unionOf και hasValue. Παραλείπει όλους τους περιορισμούς πληθικότητας και τα αξιώματα ισότητας, που της επιτρέπει να πετυχαίνει αρκετά μεγάλη απόδοση.

5.1.1.5 Η μηχανή γενικών κανόνων της Jena

Λόγω της γενικότητας και της ανοιχτής αρχιτεκτονικής της μηχανής ορισμού κανόνων από το χρήστη, κρίνεται σκόπιμη η ξεχωριστή ανάλυσή της από τα υπόλοιπα στοιχεία συμπερασμού που διαθέτει η Jena.

Όπως έχει ήδη διατυπωθεί στα προηγούμενα, η Jena διαθέτει μια μηχανή συλλογιστικής γενικού σκοπού, η οποία βασίζεται σε κανόνες και υλοποιεί τις μηχανές RDFS και OWL αλλά είναι επίσης διαθέσιμη για γενική χρήση. Υποστηρίζει συμπερασμό βάσει κανόνων σε RDF γράφους και παρέχει στρατηγικές κανονικής αλληλουχίας (forward chaining), ανάστροφης αλληλουχίας (backward chaining) καθώς και υβριδικές υλοποιήσεις.

Οι διάφορες συνθέσεις της μηχανής είναι όλες προσβάσιμες μέσω ενός μοναδικού παραμετροποιήσιμου αντικειμένου που είναι στιγμιότυπο της κλάσης GenericRuleReasoner. Κατ' ελάχιστο, ο κατασκευαστής της κλάσης απαιτεί ένα σύνολο κανόνων σαν όρισμα. Κατά τα άλλα, η μηχανή συλλογιστικής που προκύπτει μπορεί να χρησιμοποιηθεί με τον ίδιο τρόπο που χρησιμοποιούνται και όλες οι άλλες προκατασκευασμένες μηχανές.

Ένας κανόνας ορίζεται σαν ένα αντικείμενο της κλάσης Rule. Τα σύνολα κανόνων που χρειάζεται η μηχανή συλλογιστικής ως ορίσματα είναι απλά μια λίστα κανόνων (αντικείμενο της διεπαφής List<E>, όπου E είναι ο τύπος των στοιχείων της λίστας).

Για ευκολία στην ανάγνωση η Jena επιτρέπει τη χρήση QNames για τις αναφορές URI. Το σύνολο των γνωστών προθεμάτων είναι αυτά που είναι δηλωμένα με το αντικείμενο PrintUtil. Τα προκαθορισμένα είναι τα rdf, rdfs, owl, daml, xsd και ένα δοκιμαστικό πρόθεμα χώρου ονομάτων eg, αλλά περισσότερα προθέματα είναι δυνατόν να οριστούν μέσω των μεθόδων του PrintUtil στον κώδικα ενός προγράμματος Java. Επιπρόσθετα, μπορούν να οριστούν προθέματα στο αρχείο κανόνων, όταν χρησιμοποιείται ένα, για τον ορισμό μιας μηχανής συλλογιστικής.

Οι κανόνες μπορούν να φορτωθούν και να αναλυθούν (parsed) μέσω της εντολής Rule.rulesFromURL (για αρχεία) ή Rule.parseRules (για ρεύματα εισόδου ή συμβολοσειρές (strings). Η μηχανή συλλογιστικής κατασκευάζεται με την εντολή new GenericRuleReasoner.

Η δομή του αρχείου πρέπει να είναι η ακόλουθη:

- @prefix pre: <http://domain/url#>: Το αρχείο ξεκινά υποχρεωτικά με όλες τις δηλώσεις χώρων ονομάτων που θα χρησιμοποιηθούν σε αυτό, καθώς και των προθεμάτων τους. Τα προθέματα έχουν τοπική ισχύ (δηλαδή επηρεάζουν μόνο το αρχείο στο οποίο βρίσκονται).
- @include <urlToRuleFile>: Μετά τις δηλώσεις προθεμάτων ο σχεδιαστής μπορεί να δηλώσει πρόσθετα αρχεία, από τα οποία μπορούν να εισαχθούν κανόνες, ή να καλέσει τις προκαθορισμένες μηχανές συλλογιστικής της Jena (RDFS, OWL, OWLMicro, OWLMini). Τέτοιου είδους δηλώσεις τοποθετούνται οπουδήποτε στο αρχείο, αλλά πάντα πριν τους κανόνες που ορίζει ο σχεδιαστής.

Στο αρχείο κανόνων, πέρα από του κλασικούς κανόνες υπάρχουν και σύνολα ενσωματωμένων προτάσεων – συναρτήσεων. Κάθε τέτοια πρόταση μπορεί να κληθεί από έναν κανόνα και μπορεί να τοποθετηθεί σε οποιοδήποτε σημείο του. Στο Παράρτημα Δ δίνεται μια λίστα με τις διαθέσιμες προτάσεις.

5.1.2 Η πλατφόρμα D2RQ

5.1.2.1 Εισαγωγή

Η πλατφόρμα D2RQ [34] είναι ένα σύνολο πακέτων και υπηρεσιών που επιτρέπει τη «μετάφραση» πινάκων σχεσιακών βάσεων δεδομένων σε έγγραφα με RDF τριάδες, τη σύνδεση όλου αυτού του συστήματος με την Jena καθώς και την «δημοσίευση» των εγγράφων αυτών σε ξεχωριστό τελικό σημείο SPARQL.

Συγκεκριμένα, η πλατφόρμα αποτελείται από τα παρακάτω [35]:

- Γλώσσα αντιστοιχίσεων D2RQ: Μια δηλωτική γλώσσα αντιστοιχίσεων για την περιγραφή της σχέσης μεταξύ μιας οντολογίας και ενός σχεσιακού μοντέλου δεδομένων.

- Μηχανή D2RQ: Ενα plug-in για τη Jena που υλοποιείται σαν ένας γράφος. Ένας γράφος D2RQ μετατρέπει μια τοπική βάση δεδομένων σε ένα εικονικό γράφο RDF διαθέσιμο μόνο για ανάγνωση. Ξαναγράφει κλήσεις API Jena ή Sesame [78], διαδικασίες find και ερωτήματα SPARQL σε συγκεκριμένα SQL ερωτήματα. Τα σύνολα αποτελεσμάτων από αυτά τα ερωτήματα SQL μετασχηματίζονται σε τριάδες RDF ή σύνολα αποτελεσμάτων SPARQL, τα οποία στέλνονται στα υψηλότερα επίπεδα του πλαισίου.
- D2R Server: Περιλαμβάνει ένα τελικό σημείο SPARQL όπου δημοσιεύει σχεσιακές βάσεις μοντελοποιημένες σε RDF. Δίνει τη δυνατότητα σε φυλλομετρητές RDF και HTML να πλοηγηθούν στα περιεχόμενα των βάσεων και επιτρέπει σε εφαρμογές να θέσουν ερωτήματα. Είναι χτισμένος πάνω στην μηχανή D2RQ.

Η πλατφόρμα D2RQ έχει δοκιμαστεί επιτυχώς με τις παρακάτω πλατφόρμες βάσεων δεδομένων: Oracle, MySQL, PostgreSQL, Microsoft SQL Server.

5.1.2.2 Εκτέλεση από τη γραμμή εντολών

Οι βασικές διαδικασίες της πλατφόρμας D2RQ μπορούν να κληθούν από τη γραμμή εντολών [35]. Υποστηρίζονται δύο εντολές, η generate-mapping, η οποία παράγει ένα επεξεργάσιμο αρχείο αντιστοιχίσεων σε μορφή N3 το οποίο αναλύει το σχήμα μιας βάσης και η dump-rdf, η οποία δέχεται ως όρισμα ένα αρχείο μορφής N3 ή τις παραμέτρους μιας βάσης δεδομένων και παράγει απ' ευθείας τον αντίστοιχο RDF γράφο.

Η σύνταξη της generate-mapping είναι η ακόλουθη:

generate-mapping [-u username] [-p password] [-d driverclass] [-o outfile.n3] [-b base uri] jdbcURL

Καθένα από τα ορίσματα αναλύεται περαιτέρω ως εξής:

- jdbcURL: Πρόκειται για το URL της JDBC σύνδεσης της βάσης.
- -u username: Το όνομα σύνδεσης του χρήστη της βάσης
- -p password: Ο μυστικός κωδικός του χρήστη της βάσης.
  -d driverclass: Το πλήρες και ορθό κατά Java όνομα κλάσης του οδηγού.
- -o outfile.n3: Το σύνολο αντιστοιχιών που προκύπτει θα αποθηκευθεί στο αρχείο αυτό σε σύνταξη N3. Αν η παράμετρος αυτή παραλειφθεί, το σύνολο θα γραφτεί στο προκαθορισμένο ρεύμα και μέσο εξόδου.
- -b baseURI: Χρησιμοποιείται για να κατασκευάσει ένα λεξιλογικό χώρο ονομάτων ο οποίος θα χρησιμοποιηθεί αυτόματα σαν δεδομένα σύνδεσης από τον D2R Server, ακολουθώντας τη σύμβαση http://baseURI/vocab/resource/. Αυτό θα πρέπει να είναι και το ίδιο URI που χρησιμοποιείται όταν καλείται ο διακομιστής. Η προεπιλεγμένη τιμή είναι: http://localhost:2020/.

Η εντολή dump-rdf παρέχει ένα τρόπο να «ριφθούν» όλα τα δεδομένα μιας βάσης σε ένα μοναδικό αρχείο RDF. Αυτό μπορεί να γίνει με ή χωρίς τη χρήση αρχείου

αντιστοιχιών (mapping file). Αν δεν δίνεται αρχείο, τότε καλείται αυτόματα η generate-mapping και δημιουργεί ένα προκαθορισμένο σύνολο αντιστοιχιών για τη μετάφραση.

Η σύνταξη της εντολής είναι η: dump-rdf -m mapping.n3 [output parameters]. Εάν δεν παρέχεται αρχείο αντιστοιχιών, τότε η σύνδεση με τη βάση πρέπει να καθοριστεί στην γραμμή εντολής. Οπότε προκύπτει η παρακάτω σύνταξη:

dump-rdf -u username [-p password] -d driverclass -j jdbcURL [output parameters]

Αρκετοί προαιρετικοί παράγοντες επηρεάζουν το έγγραφο εξόδου RDF:

- -f format: Η σύνταξη RDF που θα χρησιμοποιηθεί για την έξοδο.
- -s fetchSize: Πρόκειται για τον αριθμό των γραμμών που θα ανακτώνται με κάθε αίτημα της βάσης. Η προεπιλεγμένη τιμή από την εντολή είναι 500 ή η τιμή Integer.MIN_VALUE για την MySQL ούτως ώστε να επιτρέψει την λειτουργία συνεχής ροής.
- -b baseURI: Ένα βασικό URI για την επίλυση σχετικών μορφών URI
- -o outfile: Το όνομα του αρχείου προορισμού.

5.1.2.3 Λειτουργία στο περιβάλλον της Jena

Το βασικό στοιχείο για την λειτουργία της πλατφόρμας D2RQ στο πλαίσιο του API της Jena είναι το ModelD2RQ, μια κλάση που επεκτείνει τη διεπαφή Model της Jena. Δημιουργείται με την εντολή new ModelD2RQ. Από τη στιγμή που θα οριστεί ένα τέτοιο αντικείμενο, η συμπεριφορά του και οι μέθοδοι που χρησιμοποιούνται απ' αυτό είναι πανομοιότυπα με αυτά του απλού Model της Jena.

5.1.2.4 Η γλώσσα D2RQ

Η γλώσσα αντιστοιχιών D2RQ είναι μια δηλωτική γλώσσα για την περιγραφή των σχέσεων μεταξύ μιας σχεσιακής βάσης και των RDFS λεξιλογίων ή των OWL οντολογιών. Ένα σύνολο αντιστοιχίσεων D2RQ είναι ένα RDF έγγραφο. Ο χώρος ονομάτων είναι το URI http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1# και συμβολίζεται με το πρόθεμα d2rq.

Μια οντολογία αντιστοιχίζεται σε ένα σχήμα βάσης δεδομένων χρησιμοποιώντας τα βασικά στοιχεία d2rq:ClassMap, καθένα από τα οποία έχει ένα ή περισσότερα d2rq:PropertyBridge, τα οποία προσδιορίζουν πώς δημιουργούνται οι ιδιότητες ενός στιγμιότυπου της κλάσης. Το αρχικό στοιχείο απ' όπου ξεκινάει το γράψιμο ενός συνόλου αντιστοιχιών D2RQ είναι το d2rq:ClassMap. Ένα ClassMap αναπαριστά μια κλάση μιας οντολογίας ή ένα σύνολο παρόμοιων κλάσεων [35].

Η δομή ενός κλασσικού συνόλου αντιστοιχιών D2RQ και η ιεραρχία που ακολουθεί δίνεται στην εικόνα 5.2.

Λεξιλόγιο και βασικές ιδιότητες του στοιχείου ClassMap

Στο [35] αναφέρονται όλες οι βασικές ιδιότητες που μπορεί να περιέχει ένα στοιχείο d2rq:ClassMap. Οι σημαντικότερες είναι οι εξής:

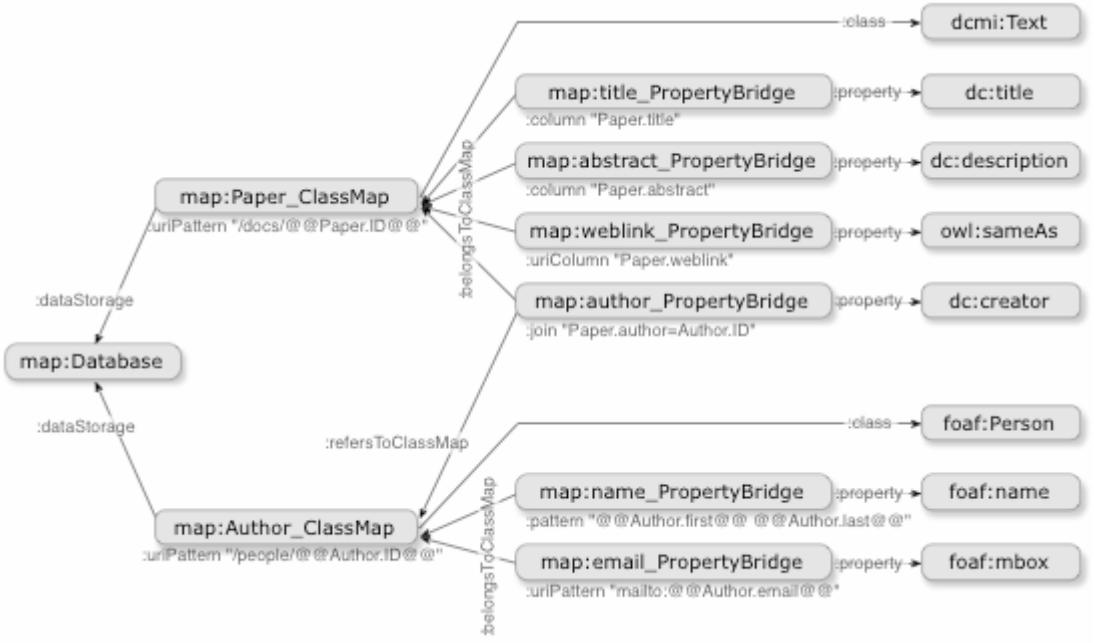
- d2rq:dataStorage: Αναφέρεται σε μια d2rq:Database όπου βρίσκονται αποθηκευμένα τα δεδομένα του στιγμιότυπου.
- d2rq:Class: Μια RDFS ή OWL κλάση. Όλοι οι πόροι που προκύπτουν από αυτό το ClassMap είναι στιγμιότυπα αυτής της κλάσης.
- d2rq:uriPattern: Καθορίζει μια μορφή URI που θα χρησιμοποιείται για να ταυτοποιήσει τα στιγμιότυπα του ClassMap.
- d2rq:uriColumn: Μια στήλη βάσης δεδομένων που περιέχει αναφορές URI για την ταυτοποίηση των στιγμιότυπων του ClassMap. Το όνομα της στήλης πρέπει να έχει τη μορφή: «ΌνομαΠίνακα.ΌνομαΣτήλης».
- d2rq:bNodeIdColumns: Μια λίστα με ονόματα στηλών σε μορφή «ΌνομαΠίνακα.ΌνομαΣτήλης» , χωρισμένα με κόμματα. Τα στιγμιότυπα του ClassMap θα είναι κενοί κόμβοι, ένας για κάθε ξεχωριστή πλειάδα των στηλών αυτών.
- d2rq:constantValue: Αυτό το ClassMap θα έχει μόνο ένα στιγμιότυπο, το οποίο θα ονομάζεται από την τιμή της ιδιότητας. Αυτή μπορεί να είναι ένας κενός κόμβος ή ένα URI.
- d2rq:additionalProperty: Προσθέτει μια επιπλέον ιδιότητα (AdditionalProperty) σε κάθε στιγμιότυπο της κλάσης. Χρησιμοποιείται για να προστεθούν ιδιότητες rdfs:seeAlso ή άλλες προτάσεις που έχουν σταθερή τιμή για κάθε στιγμιότυπο.
- d2rq:condition: Ορίζει μια συνθήκη SQL WHERE. Ένα στιγμιότυπο της κλάσης θα δημιουργηθεί μόνο για τις πλειάδες της βάσης που ικανοποιούν τη συνθήκη.
- d2rq:classDefinitionLabel: Προσδιορίζει μια ετικέτα που θα χρησιμοποιηθεί σαν ένα rdfs:label για όλους τους σχετικούς ορισμούς κλάσης. Υποστηρίζονται πολλαπλές ετικέτες (π.χ. μία για κάθε διαφορετική γλώσσα)
- d2rq:classDefinitionComment: Προσδιορίζει ένα σχόλιο που θα χρησιμοποιηθεί σαν ένα rdfs:comment για όλους τους σχετικούς ορισμούς κλάσης. Υποστηρίζονται πολλαπλά σχόλια.

Λεξιλόγιο και βασικές ιδιότητες του στοιχείου PropertyBridge

Παρομοίως με την προηγούμενη ενότητα, εδώ περιγράφονται οι πιο σημαντικές ιδιότητες του στοιχείου d2rq:PropertyBridge, όπως αυτές δίνονται στο [35]:

- d2rq:belongsToClassMap: Καθορίζει ότι η PropertyBridge ανήκει σε μία d2rq:ClassMap. Το πεδίο αυτό πρέπει να καθορίζεται για κάθε PropertyBridge.
- d2rq:property: Η ιδιότητα RDF που συνδέει την ClassMap με το αντικείμενο ή το λεκτικό που δημιουργείται από την PropertyBridge. Το πεδίο αυτό πρέπει

να καθορίζεται για κάθε PropertyBridge. Αν καθορίζονται παραπάνω από μια d2rq:property, τότε δημιουργείται μια τριάδα για κάθε ιδιότητα ανά πόρο.



Εικόνα 5.2: Δομή ενός συνόλου αντιστοιχιών D2RQ

- d2rq:column: Χρησιμοποιείται για ιδιότητες με τιμές λεκτικού. Έχει ως τιμή το όνομα της στήλης της βάσης που περιέχει τα λεκτικά. Το όνομα της στήλης πρέπει να έχει τη μορφή: «ΌνομαΠίνακα.ΌνομαΣτήλης».
- d2rq:pattern: Χρησιμοποιείται για ιδιότητες με τιμές λεκτικού. Μπορεί να χρησιμοποιηθεί για να επεκτείνει και να συνδυάσει τιμές στηλών πριν χρησιμοποιηθούν σαν μια λεκτική τιμή ιδιότητας. Αν μια μορφή περιέχει περισσότερες από μία στήλες, τότε μία συμβολοσειρά διαχωρισμού, η οποία δεν πρέπει να περιλαμβάνεται στις τιμές των στηλών, πρέπει να χρησιμοποιηθεί μεταξύ των ονομάτων των στηλών, ώστε να επιτρέψει στην D2RQ να μετατρέψει αντίστροφα τα δοθέντα λεκτικά σε τιμές στηλών.
- d2rq:sqlExpression: Χρησιμοποιείται για ιδιότητες με τιμές λεκτικού. Παράγει τιμές λεκτικών αποτιμώντας μια έκφραση SQL. Θέτοντας ερώτημα για μια τέτοια υπολογισμένη τιμή ίσως αποτελέσει βαρύ φόρτο εργασίας για τη βάση.
- d2rq:datatype: Χρησιμοποιείται για ιδιότητες με τιμές λεκτικού. Καθορίζει τον τύπο δεδομένων RDF των λεκτικών.
- d2rq:uriColumn: Χρησιμοποιείται για ιδιότητες με τιμές URI. Δηλώνει μια στήλη βάσης δεδομένων που περιέχει URIs. Το όνομα της στήλης πρέπει να έχει τη μορφή: «ΌνομαΠίνακα.ΌνομαΣτήλης».
- d2rq:uriPattern: Χρησιμοποιείται για ιδιότητες με τιμές URI. Μπορεί να χρησιμοποιηθεί για να επεκτείνει και να συνδυάσει τιμές στηλών πριν χρησιμοποιηθούν σαν μια URI τιμή ιδιότητας. Αν μια μορφή περιέχει περισσότερες από μία στήλες, τότε μία συμβολοσειρά διαχωρισμού, η οποία

δεν πρέπει να περιλαμβάνεται στις τιμές των στηλών, πρέπει να χρησιμοποιηθεί μεταξύ των ονομάτων των στηλών, ώστε να επιτρέψει στην D2RQ να μετατρέψει αντίστροφα τα δοθέντα URI σε τιμές στηλών.

- d2rq:refersToClassMap: Χρησιμοποιείται για ιδιότητες που ανταποκρίνονται σε ένα ξένο κλειδί. Αναφέρει ένα άλλο d2rq:ClassMap που δημιουργεί τα στιγμιότυπα που χρησιμοποιούνται σαν τιμές του PropertyBridge. Μία ή περισσότερες d2rq:join ιδιότητες πρέπει να καθοριστούν για να επιλέξουν τα σωστά στιγμιότυπα.
- d2rq:constantValue: Χρησιμοποιείται για ιδιότητες που έχουν την ίδια σταθερή τιμή σε όλα τα στιγμιότυπα του ClassMap. Η τιμή μπορεί να είναι ένα λεκτικό, ένας κενός πόρος ή ένα URI.
- d2rq:join: Αν οι στήλες που χρησιμοποιούνται για να δημιουργήσουν το αντικείμενο της ιδιότητας (λεκτικά ή URIs) δεν είναι από τον πίνακα που εκφράζεται από το συγκεκριμένο ClassMap, αλλά από έναν άλλο πίνακα (πιθανώς και διαφορετικής βάσης), τότε οι πίνακες πρέπει να συνενωθούν χρησιμοποιώντας μία ή περισσότερες ιδιότητες d2rq:join.
- d2rq:condition: Ορίζει μια συνθήκη SQL WHERE. Η PropertyBridge θα παράγει μια πρόταση αν η συνθήκη ευσταθεί. Μια συνηθισμένη χρήση είναι η μη εμφάνιση προτάσεων με κενές λεκτικές τιμές: d2rq:condition "Table.Column <> "".
- d2rq:propertyDefinitionLabel: Καθορίζει μια ετικέτα που θα χρησιμοποιηθεί σαν rdfs:label για όλους τους συσχετιζόμενους ορισμούς ιδιοτήτων. Υποστηρίζονται πολλαπλές ετικέτες (π.χ. μία για κάθε γλώσσα).
- d2rq:propertyDefinitionComment: Καθορίζει ένα σχόλιο που θα χρησιμοποιηθεί σαν rdfs:comment για όλους τους συσχετιζόμενους ορισμούς ιδιοτήτων. Υποστηρίζονται πολλαπλά σχόλια.

5.1.3 Ο Joseki RDF server

Ο Joseki RDF server [53] είναι στην ουσία ένα servlet το οποίο παρέχει μια διαπροσωπεία Ιστού (Web nterface) για την εκτέλεση ερωτημάτων SPARQL σε έναν RDF γράφο. Αναπτύχθηκε από την Hewlett Packard και δημοσιεύθηκε με μια άδεια ισοδύναμη της MIT license [79]. Μπορεί να εγκατασταθεί σε οποιονδήποτε υποδοχέα servlet ή διακομιστή εφαρμογών Ιστού, όπως ο Tomcat [7]. Διαθέτει μια σειρά από αρχεία .jar τα οποία τον κάνουν άμεσα προσβάσιμο σε προγράμματα Java, αλλά είναι δυνατόν να κληθεί και από τη γραμμή εντολών πηγαίνοντας στο φάκελο που περιέχει τα αρχεία του και δίνοντας την εντολή:

\bin\rdfserver [--port αριθμός πόρτας] [όνομα αρχείου ρυθμίσεων (configuration file)]

Το πρώτο όρισμα χρησιμοποιείται από το χρήστη για να κάνει τον server να λειτουργήσει σε μια πόρτα διαφορετική από την 2020, που είναι η προεπιλογή [55]. Το δεύτερο όρισμα, απαιτείται πάντα σε κάθε κλήση του Joseki και καθορίζει το

όνομα του αρχείου ρυθμίσεων. Πρόκειται για το αρχείο που περιλαμβάνει τον ορισμό υπηρεσιών που απαιτεί ο χρήστης από τον διακομιστή, τα δεδομένα και τους γράφους πάνω στους οποίους θα τεθούν τα ερωτήματα, κ.τ.λ. Το αρχείο είναι υποχρεωτικό να βρίσκεται στον βασικό φάκελο του Joseki στον υπολογιστή και κατασκευάζεται με μη αυτόματο τρόπο από το χρήστη βασιζόμενο στα στοιχεία της επόμενης ενότητας.

5.1.3.1 Δομή αρχείων ρυθμίσεων

Ένα αρχείο ρυθμίσεων στον Joseki περιλαμβάνει υπηρεσίες. Μια υπηρεσία υλοποιείται από έναν επεξεργαστή, ο οποίος μπορεί να εκτελέσει ερωτήματα επί ενός σταθερού και προκαθορισμένου συνόλου δεδομένων, ή μπορεί δυναμικά να σχηματίσει το σύνολο δεδομένων από ένα ερώτημα, ή να κάνει και τα δύο, ανάλογα πάλι με το ερώτημα.

Τα αρχεία ρυθμίσεων είναι και αυτά RDF γράφοι. Το προκαθορισμένο όρισμα είναι το joseki-config.ttl και συχνά γράφεται σε σύνταξη Turtle ή N3 παρά σε RDF/XML, αν και ο διακομιστής μπορεί να διαβάσει δεδομένα σε οποιαδήποτε μορφή σειριοποίησης RDF. Μια προτεινόμενη σειρά σύνταξης των στοιχείων του αρχείου (η οποία δεν είναι δεσμευτική γιατί αφού το αρχείο είναι RDF γράφος τότε εξ' ορισμού δεν έχει σημασία η σειρά των προτάσεων) δίνεται παρακάτω [54]:

- Προθέματα: Αρχικά ορίζονται τα προθέματα που θα χρησιμοποιηθούν και μετά κάποια βασική πληροφορία σε μορφή σχολίων για το αρχείο (προαιρετικό).
- Ορισμός διακομιστή: Πρόκειται για το βασικό στοιχείο ενός αρχείου, αφού χωρίς αυτό δεν είναι δυνατόν να υπάρξει κάποιος φορέας, ο οποίος θα δημοσιεύσει τα RDF δεδομένα για να τους γίνουν ερωτήματα. Η απλούστερη περίπτωση διακομιστή ορίζεται καλώντας την κλάση java: org.joseki.util.ServiceInitSimple.
- Βασική υπηρεσία: Αφού καθοριστεί ένα αντικείμενο διακομιστή, πρέπει να αναφερθεί ρητά ποια υπηρεσία εκτελεί. Αρχικά προσδιορίζεται ο τύπος του RDF αντικειμένου, μια αναφορά στον τύπο της υπηρεσίας (συνήθως παίρνει τιμή "sparql") και μετά αναφέρεται ο επεξεργαστής ο οποίος επιφορτίζεται με το έργο της εκτέλεσης της συγκεκριμένης υπηρεσίας. (Συνήθως) αναφέρεται και το σύνολο δεδομένων με το οποίο θα ασχοληθεί η υπηρεσία. Γενικά όμως δεν είναι υποχρεωτική η αναφορά αυτή.
- Επεξεργαστής: Στη συνέχεια καλείται μια κλάση java η οποία θα υλοποιήσει τον επεξεργαστή ερωτημάτων. Ανάλογα με τις ανάγκες του χρήστη μπορούν να οριστούν μερικές παράμετροι του επεξεργαστή, με τη βασικότερη να είναι η δυνατότητα να δέχεται ή όχι ερωτήματα με προτάσεις FROM/FROM NAMED (επιτρέπει ή όχι ρητά σύνολα δεδομένων ή φόρτωση από τον Ιστό).
- Ορισμός συνόλων δεδομένων: Ένα σύνολο δεδομένων είναι μια συλλογή από έναν ανώνυμο, προκαθορισμένο γράφο και κανένα ή περισσότερους ονομασμένους γράφους. Ερωτήματα προσπελαύνουν τον ονομασμένο γράφο μέσω της λέξης GRAPH στη SPARQL. Κάθε γράφος είναι ένα Jena Model

και αυτοί περιγράφονται με το λεξιλόγιο του Jena Assembler, που είναι ένα αντικείμενο της διεπαφής Assembler και ουσιαστικά αυτό είναι που κατασκευάζει αντικείμενα στη Jena (συνήθως μοντέλα). Η ολοκληρωμένη μορφή της περιγραφής ενός συνόλου δίνεται παρακάτω:

```
# A dataset of one model as the default graph

_:ds1 rdf:type ja:RDFDataset ;
   ja:defaultGraph   _:model0 ;
   rdfs:label          "Dataset _:ds1" ;
   ja:namedGraph
      [ ja:graphName  <http://example.org/name1> ;
        ja:graph      _:model1 ] ;
   ja:namedGraph
      [ ja:graphName  <http://example.org/name2> ;
        ja:graph      _:model2 ] ;
   .
```

Παράδειγμα 5.1: Σύνολο δεδομένων με ένα προκαθορισμένο και δύο ονομαζόμενα σύνολα δεδομένων.

Όπως φαίνεται, το συγκεκριμένο σύνολο δεδομένων περιλαμβάνει τρεις γράφους, τον προκαθορισμένο και δυο ονομασμένους, που αναφέρονται με το URI στο οποίο βρίσκονται και διαθέτουν ο καθένας από ένα αντικείμενο graph που ορίζει το μοντέλο από το οποίο αντλούν τα δεδομένα τους.

- Περιγραφές μοντέλων: Κάθε αντικείμενο ja:graph που υπάρχει στους γράφους του παραδείγματος 5.1 αντιστοιχίζεται σε ένα μοντέλο Jena. Υπάρχουν δύο τρόποι για να οριστεί ένα μοντέλο:
  - Με αναφορά σε εξωτερικά αρχεία: Μπορούν να δηλωθούν όπως στο παράδειγμα 5.2 με χρήση του στοιχείου ja:content και μετά κάθε αρχείο ξεχωριστά με το δικό του ja:externalContent. Σε περίπτωση που δύο ή περισσότερα αρχεία δηλώνονται στο μοντέλο, τότε γίνεται συνένωση αυτών των αρχείων και μετά προχωρά η όποια άλλη διαδικασία.
  - Με αναφορά σε βάση δεδομένων: Ένα μοντέλο το οποίο χρησιμοποιείται στη δημιουργία γράφων μπορεί να είναι αποθηκευμένο σε μια βάση (persistent model). Για να δηλωθεί, απαιτούνται τα στοιχεία της βάσης που το περιέχει.

```
## ---------------------------------------------------------
## Individual graphs (Jena calls them Models)
## (syntax of data files is determined by file extension - defaults to RDF/XML)

_:model0 rdf:type ja:MemoryModel ;
   rdfs:label "Model (plain, merge the 2 RDF files)" ;
   ja:content [
      ja:externalContent <file:D1.ttl> ;
      ja:externalContent <file:D2.ttl> ;
    ] ;
   .
```

```
_:model1 rdf:type ja:MemoryModel ;
   rdfs:label "Model (D1.ttl for content)" ;
   ja:content [
      ja:externalContent <file:D1.ttl> ;
     ] ;
   .

_:model2 rdf:type ja:MemoryModel ;
   rdfs:label "Model (D2.ttl for content)" ;
   ja:content [
      ja:externalContent <file:D2.ttl> ;
    ]
   .

## use of a persistent model
_:modelDB rdf:type ja:RDBModel ;
   ja:connection
   [
      ja:dbType        "MySQL" ;
      ja:dbURL         "jdbc:mysql://localhost/data" ;
      ja:dbUser        "user" ;
      ja:dbPassword  "password" ;
      ja:dbClass       "com.mysql.jdbc.Driver" ;
   ] ;
   ja:modelName "ame"
   .
```

Παράδειγμα 5.2: Jena Models οριζόμενα από εξωτερικά αρχεία που περιλαμβάνουν γράφους RDF και παραμένοντα μοντέλα (περίπτωση βάσης MySQL)

5.1.4 Το εργαλείο ομόσπονδων ερωτημάτων FedX

Το FedX είναι ένα λογισμικό ανοιχτού κώδικα το οποίο αναπτύχθηκε από τη γερμανική εταιρεία Fluid Operations [31] και αποτελεί μια λύση για την επεξεργασία διαμοιρασμένων ερωτημάτων σε ανοικτά συνδεδεμένα δεδομένα (Linked Open Data) [12]. Είναι υλοποιημένο σε Java και επεκτείνει το πλαίσιο Sesame [78] (πάνω στο οποίο είναι χτισμένες γλώσσες ερωτημάτων όπως η SPARQL) με ένα στρώμα ομοσπονδίας (federation layer). Είναι ενσωματωμένο στο Sesame σαν ένα στρώμα αποθήκευσης και συμπερασμού (Storage And Inference Layer, SAIL). Η υποκείμενη υποδομή Sesame επιτρέπει σε ετερογενείς πηγές δεδομένων να χρησιμοποιηθούν σαν τελικά σημεία μέσα στην ομοσπονδία. Πάνω σε αυτήν την υποδομή το FedX υλοποιεί τη λογική που απαιτείται για βελτιστοποίηση και αποτελεσματική εκτέλεση ενός ερωτήματος σε διανεμημένο περιβάλλον.

Η επεξεργασία ομόσπονδων ερωτημάτων στο FedX αποτελείται από τα εξής βήματα:

- Διατύπωση ενός ερωτήματος προς ένα σύνολο πηγών δεδομένων το οποίο έχει δηλωθεί στο FedX ως μια ομοσπονδία.
- Χωρισμός του ερωτήματος σε υποερωτήματα τα οποία και αποστέλλονται προς τις επιμέρους πηγές.

- Συγκέντρωση των αποτελεσμάτων, συγχώνευση αυτών στον ενοποιητή (federator) και παρουσίαση στο χρήστη.

Η όλη διαδικασία είναι διαφανής για τον χρήστη, δηλαδή τα δεδομένα φαίνεται ότι είναι εικονικώς ολοκληρωμένα σε έναν RDF γράφο [88].

Το πιο κρίσιμο στοιχείο στην απόδοση μιας τέτοιας μηχανής επεξεργασίας ερωτημάτων είναι η χρήση τεχνικών βελτιστοποίησης, ώστε να μειωθεί ο αριθμός των αιτημάτων (requests) στα τελικά σημεία. Το FedX χρησιμοποιεί τις ακόλουθες τεχνικές [88]:

- Πηγές προτάσεων (statement sources): Κάθε πρόταση του SPARQL ερωτήματος ελέγχεται ως προς το ποια ή ποιες πηγές περιλαμβάνουν λύσεις γι΄αυτήν, ώστε να συμπεριληφθεί μόνο στα υποερωτήματα των πηγών αυτών.
- Ώθηση φίλτρων (filter pushing): Όποτε είναι αυτό δυνατό, εκφράσεις φιλτραρίσματος SPARQL ωθούνται στο σύστημα για να επιτρέψουν μια πρώιμη αποτίμηση του ερωτήματος.
- Παράλληλη επεξεργασία (parallel processing): Εκμεταλλεύεται ο ταυτοχρονισμός μέσω της πολυνηματικής εκτέλεσης υπολογισμών ένωσης (union) και συνένωσης (join).
- Σειρά συνένωσης (join order): Επηρεάζει σε μεγάλο βαθμό την απόδοση, αφού ο αριθμός των ενδιαμέσων αποτελεσμάτων προσδιορίζει τον ολικό χρόνο εκτέλεσης του ερωτήματος. Στο FedX χρησιμοποιείται η τεχνική μέτρησης μεταβλητών μαζί με διάφορες ευριστικές τεχνικές (heuristics) για να προσδιοριστεί το κόστος κάθε συνένωσης. Ακολουθώντας μια «άπληστη» (greedy) προσέγγιση, οι συνενώσεις εκτελούνται με αύξουσα σειρά κόστους.
- Συνένωση δεσμεύσεων (bound joins): Για να μειωθεί ο αριθμός των αιτημάτων και κατά συνέπεια του ολικού χρόνου εκτέλεσης, οι συνενώσεις υπολογίζονται σε μια συνένωση μπλοκ εμφωλευμένων βρόχων (block nested loop join).
- Ομαδοποιήσεις: Οι προτάσεις που έχουν τις ίδιες σχετικές πηγές δεδομένων συνεκτελούνται σε ένα απλό SPARQL ερώτημα ώστε να ωθηθούν συνενώσεις στο συγκεκριμένο τελικό σημείο.

5.1.4.1 Χρήση του FedX σε περιβάλλον Java

Η τεκμηρίωση της εφαρμογής FedX, που διανέμεται μαζί με την εφαρμογή από την αρχική της σελίδα [31], παρέχει όλες τις πληροφορίες για τη χρήση της σε περιβάλλον Java. Αρχικά πρέπει να δηλωθούν όλα τα τελικά σημεία στα οποία θα υποβληθεί το καθολικό ερώτημα. Αυτό μπορεί να γίνει με πέντε τρόπους:

- Χρησιμοποιώντας μία απλή ομοσπονδία SPARQL ως ένα αντικείμενο SailRepository, στο οποίο δίνονται τα URI των τελικών σημείων ως ορίσματα:

  ```
  Config.initialize();
  SailRepository repo = FedXFactory.initializeSparqlFederation(
  ```

```
Arrays.asList("http://dbpedia.org/sparql",
"http://api.talis.com/stores/nytimes/services/sparql"));
```

- Δημιουργώντας μια απλή ομοσπονδία SPARQL χωρίς τη χρήση άλλου αντικειμένου:
  ```
  Config.initialize();
  FedXFactory.initializeSparqlFederation(Arrays.asList(
  "http://dbpedia.org/sparql",
  "http://api.talis.com/stores/nytimes/services/sparql"));
  ```
- Με χρήση ενός αρχείου ρυθμίσεων δεδομένων, το οποίο θα περιέχει τα URI αλλά και το είδος των τελικών σημείων:
  ```
  Config.initialize();
  Repository repo = FedXFactory.initializeFederation("dataSourceConfig.ttl");
  ```
  Το αρχείο ρυθμίσεων θα περιέχει προτάσεις σαν αυτές:
  ```
  @prefix fluid: <http://fluidops.org/config#>.
  (δήλωση προθεμάτων)
  <http://DBpedia> fluid:store "SPARQLEndpoint";
  fluid:SPARQLEndpoint "http://dbpedia.org/sparql".
  (δήλωση ενός τελικού σημείου SPARQL)
  <http://DBpedia> fluid:store "NativeStore";
  fluid:RepositoryLocation "repositories\\native-storage.dbpedia36".
  (δήλωση ενός τελικού σημείου NativeStore)
  ```
- Με χρήση ενός αρχείου ρυθμίσεων FedX:
  ```
  String fedxConfig = "examples/fedxConfig-withPrefixDecl.prop";
  FedXFactory.initializeFederation(fedxConfig,
  Collections.<Endpoint>emptyList());
  ```
- Κατασκευάζοντας μια λίστα που έχει ως αντικείμενα τελικά σημεία SPARQL:
  ```
  List<Endpoint> endpoints = new ArrayList<Endpoint>();
  endpoints.add(EndpointFactory.loadSPARQLEndpoint("dbpedia",
  "http://dbpedia.org/sparql"));
  endpoints.add(EndpointFactory.loadSPARQLEndpoint("nytimes",
  "http://api.talis.com/stores/nytimes/services/sparql"));
  Config.initialize();
  ```
  Σαν ορίσματα στη μέθοδο loadSPARQLEndpoint για αρχικοποίηση τελικών σημείων δίνονται το URI ή και το όνομα του τελικού σημείου.

Μετά τις αρχικοποιήσεις, δίνεται σαν μια μεταβλητή τύπου String το ερώτημα στο μηχανισμό, και ανάλογα με τη μέθοδο δήλωσης τελικών σημείων ακολουθείται η προετοιμασία του ερωτήματος και η μετατροπή του σε ένα αντικείμενο TupleQuery είτε με σύνδεση στο αντικείμενο SailRepository ή Repository με την εντολή: ΌνομαΑντικειμένου.getConnection().prepareTupleQuery(QueryLanguage.SPARQL, ΚείμενοΕρωτήματος), είτε με τη χρήση του εργαλείου QueryManager σε κάθε άλλη περίπτωση (εντολή QueryManager.prepareTupleQuery(ΚείμενοΕρωτήματος). Τέλος με την εντολή query.evaluate() προκύπτει ένα αντικείμενο TupleQueryResult που περιέχει τα αποτελέσματα της αποτίμησης του ερωτήματος.

5.1.5 Εξωτερικά πακέτα κλάσεων Java που χρησιμοποιήθηκαν στην ανάπτυξη της εφαρμογής

Για να χρησιμοποιηθούν μερικά από τα εργαλεία που αναφέρθηκαν σε προηγούμενες ενότητες, ήταν απαραίτητη η εισαγωγή κάποιων σχετικών αρχείων .jar στην εφαρμογή IntegraHEALTH 1.0 και η κλήση των περιεχομένων τους από ορισμένες κλάσεις της. Πέρα από αυτά τα εργαλεία, η εφαρμογή χρησιμοποιεί και κάποια σύνολα αρχείων .jar από εφαρμογές ή πλαίσια (frameworks) που είναι κατασκευασμένα αποκλειστικά για χρήση σε περιβάλλον Java:

- dcm4che: Πρόκειται για μια συλλογή εφαρμογών για επιχειρήσεις και συστήματα ιατροφαρμακευτικής περίθαλψης. Αναπτύχθηκε από τον ομώνυμο οργανισμό [29] σε περιβάλλον Java και το βασικό κομμάτι της αφορά μια εύρωστη υλοποίηση του προτύπου DICOM (κατασκευή αρχείων, ανάκτηση πληροφοριών από αρχεία, κ.τ.λ.).
- Java Image I/O Technology [81]: Πρόκειται για μια διεπαφή προγραμματισμού εφαρμογών (Application Programming Interface, API), η οποία δίνει τη δυνατότητα εργασίας με εικόνες που αποθηκεύονται σε αρχεία. Περιλαμβάνει πακέτα με εργαλεία για το χειρισμό των δημοφιλέστερων τύπων αρχείων εικόνας (BMP, JPEG κ.τ.λ.) και υποστηρίζει τη δημιουργία επιπρόσθετων εργαλείων για μη συνηθισμένους τύπους.

5.2 Βοηθητικά στοιχεία της εφαρμογής IntegraHEALTH 1.0

Για την πληρέστερη εξήγηση του λογισμικού της εφαρμογής, που δίνεται στην ενότητα 5.3, κρίνεται απαραίτητη η αναφορά και υποτυπώδη ανάλυση των επιμέρους εξωτερικών βοηθητικών στοιχείων της εφαρμογής, τα οποία είτε συνεργάζονται με τον κώδικα της εφαρμογής, είτε απαιτούν τη δημιουργία ανεξάρτητων αρχείων ρυθμίσεων ή ορισμάτων.

5.2.1 Οι τρεις βάσεις των ιατρών

Στο σύστημα υπάρχουν τρεις βάσεις δεδομένων ονόματι doctor1, doctor2 και doctor3 οι οποίες δημιουργήθηκαν με το σύστημα διαχείρισης βάσεων δεδομένων MySQL και αποτελούν τις βάσεις δεδομένων του παθολόγου, του ακτινολόγου και του μικροβιολόγου αντίστοιχα. Η βάση του παθολόγου περιλαμβάνει τους πίνακες:

- elements, που αποθηκεύει όλα τα στοιχεία του ασθενή με την ακόλουθη σειρά: Όνομα (fName), Επώνυμο (lName), Πατρώνυμο (fatherName), Ημερομηνία Γέννησης (birthday), Α.Μ.Κ.Α. (amka), Ασφαλιστικός Οργανισμός (securityOrg), Διεύθυνση (address), Τηλέφωνο (phone), Ημερομηνία Εξέτασης (examinationDate), Ασθένεια (disease), Παρατηρήσεις (remarks), Συνολικό Ποσό (sum), με τη στήλη disease να αποτελεί ξένο κλειδί του πίνακα, καθώς είναι το πρωτεύον κλειδί του πίνακα diseases. Λόγω της παραδοχής 5 της ενότητας 4.3 ορίζεται ένα σύνθετο πρωτεύον κλειδί από τα πεδία amka και examinationDate.

- diseases, που περιέχει URI της οντολογίας NCI Thesaurus που αναφέρονται σε ασθένειες και θα χρησιμοποιηθούν σαν αντικείμενα σε σχετικές RDF προτάσεις που θα δημιουργηθούν. Περιλαμβάνει τα πεδία diseaseID (πρωτεύον κλειδί) και diseaseURI.
- elements_log, ο οποίος είναι ως προς τη δομή του ίδιος με τον πίνακα elements αλλά χρησιμεύει σαν πίνακας καταγραφής με λειτουργία όπως αυτή εξηγείται στην ενότητα 4.2.

Για τη μετατροπή αυτής της βάσης σε RDF χρησιμοποιείται η πλατφόρμα D2RQ. Στο λογισμικό της εφαρμογής IntegraHEALTH 1.0 περιλαμβάνονται στο πακέτο core.handlers (ενότητα 5.3.1.7) κλήσεις προς σχετικές κλάσεις της πλατφόρμας, αλλά και το αρχείο αντιστοιχίσεων D2RQ που αναλαμβάνει τη μετατροπή.

Η βάση του ακτινολόγου περιλαμβάνει τους πίνακες:

- dicom_elements, που αποθηκεύει τα αρχεία DICOM που δημιουργούνται από τα δεδομένα που εισάγει ο ιατρός. Περιλαμβάνει τα πεδία amka, dicomFileName (πρωτεύον κλειδί), dicomFile (εδώ αποθηκεύεται το αρχείο σαν ένα BLOB) και sum, που, όπως και στην βάση του παθολόγου, αντιστοιχεί στο ποσό που οφείλει ο ασθενής ως πληρωμή, αλλά δεν συμπεριλαμβάνεται στα κωδικοποιήσιμα κατά DICOM στοιχεία, ώστε να αποτελέσει μέρος της επικεφαλίδας του αντίστοιχου αρχείου. Η επιλογή του πρωτεύοντος κλειδιού έγινε βάσει της παραδοχής 3 της ενότητας 4.3.
- action_log, που αποτελεί τον πίνακα καταγραφής της βάσης. Περιλαμβάνει τα πεδία modificationCode (αύξων αριθμός καταγραφής), actionTaken (καταγραφή του τύπου αλλαγής στη βάση, ήτοι INSERT, UPDATE ή DELETE), oldFileName (παλιό όνομα αρχείου) και newFileName (νέο όνομα αρχείου). Τα δύο τελευταία πεδία δεν λαμβάνουν τιμή ανάλογα με την αλλαγή που έχει γίνει στη βάση. Δεν υπάρχει νέο όνομα αρχείου μετά από διαγραφή στοιχείων, ούτε παλιό όνομα αρχείου μετά από εισαγωγή στοιχείων. Οι τιμές των πεδίων αυτών αναφέρονται στο πρωτεύον κλειδί του πίνακα dicom_elements και σύμφωνα με αυτές εκτελούνται οι δύο πρώτες διεργασίες της μονάδας μετασχηματισμού RDF του ακτινολόγου.

Η βάση του μικροβιολόγου περιλαμβάνει τους πίνακες hl7_elements και action_log με λειτουργίες πανομοιότυπες με αυτές των πινάκων της βάσης του ακτινολόγου με μόνη εξαίρεση στον πίνακα hl7_elements τη μη καταγραφή του οφειλόμενου ποσού, καθώς αυτό μπορεί να κωδικοποιηθεί μέσα σε ένα μήνυμα HL7 και μάλιστα σε τμήμα τύπου PV1.

Στους πίνακες καταγραφής των βάσεων οι ιατροί δεν έχουν καμία πρόσβαση. Τα δεδομένα σε αυτούς εισάγονται αυτόματα μετά από κάθε αλλαγή στη βάση. Αυτό πραγματοποιείται με μια κατηγορία αποθηκευμένων διαδικασιών που ονομάζονται σκανδάλες (triggers) [66] και στη συγκεκριμένη περίπτωση συντάσσονται με τέτοιο τρόπο ώστε να ενεργοποιούνται αμέσως μετά την εισαγωγή, ενημέρωση ή διαγραφή δεδομένων, κρατώντας ένα αντίγραφο της πλειάδας που αφορούσε η τελευταία διαδικασία και εισάγοντάς το στον πίνακα καταγραφής. Η πλήρης λίστα των διαδικασιών αυτών δίνεται στο παράρτημα Ε.

5.2.2 Οι οντολογίες που χρησιμοποιούνται

Κατά την υλοποίηση της εφαρμογής IntegraHEALTH 1.0, λαμβάνοντας υπόψιν την παραδοχή 2 της ενότητας 4.3, αναζητήθηκε ένα σύνολο οντολογιών το οποίο θα κάλυπτε όσο το δυνατόν περισσότερα στοιχεία των τριών ιατρών. Μετά από αλλεπάλληλες δοκιμές που σχετίζονταν κυρίως με τους περιορισμούς και τις σχέσεις μεταξύ των διαφόρων συστατικών των οντολογιών (πχ. αν ένα αντικείμενο έχει μια συγκεκριμένη ιδιότητα τότε ανήκει σε μια συγκεκριμένη κλάση κ.τ.λ.) επιλέχθηκαν οι παρακάτω οντολογίες:

- MGED Ontology [63], χρησιμοποιείται κυρίως για τα δημογραφικά στοιχεία του ασθενή και για τη διάγνωση και τις παρατηρήσεις του ιατρού. Το θετικό της στοιχείο είναι ότι οι ιδιότητες που έχουν επιλεχθεί δεν έχουν συγκεκριμένα πεδία ορισμού (domains) ή σύνολα τιμών (ranges), καθιστώντας τες αρκετά ευέλικτες στη χρήση. Το URI του χώρου ονομάτων της είναι το http://mged.sourceforge.net/ontologies/MGEDOntology.owl# και το πρόθεμά της είναι το mged.
- Friend Of A Friend (FOAF) Ontology [32], η οποία δρα επικουρικά στην προηγούμενη, καλύπτοντας τα κενά της. Το URI του χώρου ονομάτων της είναι το http://xmlns.com/foaf/0.1/ και το πρόθεμά της είναι το foaf.
- NCI Thesaurus: Πρόκειται για μια οντολογία που έχει κατασκευαστεί από το National Cancer Institute των Η.Π.Α. και περιλαμβάνει έναν αρκετά μεγάλο αριθμό όρων, οι οποίοι εκφράζονται κυρίως με κλάσεις και καλύπτουν σχεδόν όλο το φάσμα της ιατροφαρμακευτικής περίθαλψης από περιπτώσεις ασθενειών και συμπτωμάτων (συνηθισμένες ή μη) έως την καταγραφή επισκέψεων, στοιχεία έρευνας, πληροφορίες γενετικής κ.τ.λ. Το URI του χώρου ονομάτων της είναι το http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl# και το πρόθεμά της είναι το ncit.

Οι οντολογίες MGED και NCI Thesaurus θεωρούνται ιατρικές οντολογίες. Το πλήρες λεξιλόγιό τους και οι σχέσεις μεταξύ των στοιχείων τους είναι δυνατόν να βρεθούν στον ιστότοπο bioportal [16], μαζί με άλλες οντολογίες σχετικές με την ιατροφαρμακευτική περίθαλψη.

5.2.3 Τα τρία τελικά σημεία SPARQL

Στο κεφάλαιο 4 αλλά και στην εισαγωγή έγινε αναφορά στη χρήση του Joseki RDF server για τη δημιουργία τελικών σημείων SPARQL. Επειδή γίνεται χρήση ενός μόνο προσωπικού υπολογιστή για την κατασκευή του όλου συστήματος, δε γίνεται χρήση της δυνατότητας του Joseki να ενσωματώνεται σε διακομιστές Ιστού (Web servers), οπότε δημιουργήθηκαν τρία αρχεία δέσμης (.bat αρχεία) τα οποία περιλαμβάνουν εντολές για τη δημιουργία τριών «τοπικών» τελικών σημείων, υπό την έννοια ότι όλα βρίσκονται στον ίδιο υπολογιστή, κατασκευάζοντας έτσι ένα εικονικό δίκτυο.

Τα αρχεία δέσμης ονομάζονται SQL.bat, DICOM.bat και HL7.bat και εκτελούν την εντολή bin\rdfserver με τις αντίστοιχες παραλλαγές:

- bin\rdfserver --port 2030 SQL.ttl, όπου σύμφωνα με τη βοήθεια της εντολής προκύπτει ότι δημιουργείται ένα τελικό σημείο στη διεύθυνση http://localhost:2030/sparql στο οποίο είναι δημοσιευμένα τα δεδομένα του triple store που περιέχει το παραμένον μοντέλο του παθολόγου, όπως δηλώνεται στο αρχείο ρυθμίσεων SQL.ttl.
- bin\rdfserver --port 2040 DICOM.ttl, που δηλώνει ότι τα RDF δεδομένα του ακτινολόγου δημοσιεύονται στη διεύθυνση http://localhost:2040/sparql.
- bin\rdfserver --port 2050 HL7.ttl, που δηλώνει ότι τα RDF δεδομένα του μικροβιολόγου δημοσιεύονται στη διεύθυνση http://localhost:2050/sparql.

Κατά κανόνα, τα τελικά σημεία είναι πάντα συνδεδεμένα και ενημερώνονται αυτόματα για τις αλλαγές στα RDF μοντέλα που δημοσιεύονται. Στην περίπτωση της IntegraHEALTH 1.0 αυτή η συμπεριφορά μπορεί να προσομοιωθεί, εκτελώντας τα τρία αυτά αρχεία πριν την εκτέλεση της ίδιας της εφαρμογής.

5.3 Το λογισμικό της εφαρμογής IntegraHEALTH 1.0

Στην παρούσα ενότητα θα παρουσιαστεί όλο το λογισμικό που αναπτύχθηκε για την εφαρμογή IntegraHEALTH 1.0. Αρχικά θα περιγραφεί όλος ο κώδικας Java κατά πακέτα και θα αναλυθούν τα κυριότερα και πιο νευραλγικά σημεία του. Κατόπιν θα δοθούν κάποιες περιγραφές των επιμέρους ανεξάρτητων αρχείων που κατασκευάστηκαν για να ρυθμίσουν τις βοηθητικές εφαρμογές του προγράμματος (D2RQ, Joseki, FedX). Όλες οι κλάσεις και τα αρχεία παρατίθενται αναλυτικά στο Παράρτημα Ε.

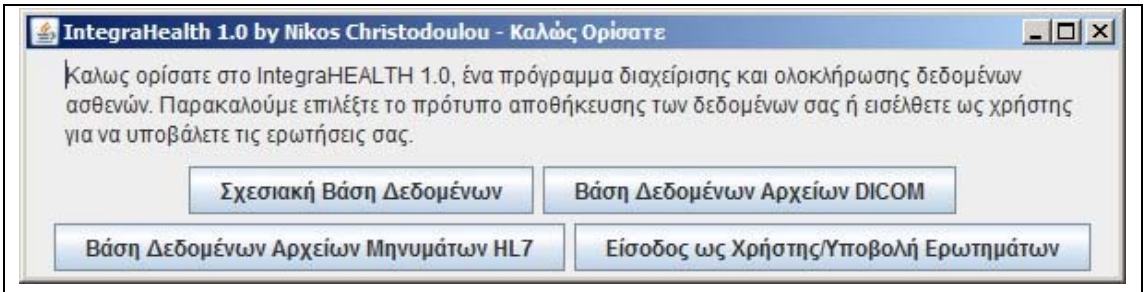
5.3.1 Ο κώδικας Java της εφαρμογής

5.3.1.1 Το πακέτο core.gui.

Στο πακέτο αυτό περιλαμβάνεται η κλάση StartUpFrame.java, η οποία παράγει την αρχική γραφική διεπαφή χρήστη που βλέπει ο χρήστης της εφαρμογής. Οι μέθοδοι που περιέχονται στην κλάση αυτή είναι οι εξής:

- Μια μέθοδος κατασκευαστή, ομώνυμη της κλάσης, που κατασκευάζει τη γραφική διεπαφή.
- Μια μέθοδος actionPerformed, η οποία εκτελεί μια διαφορετική διαδικασία για κάθε κουμπί το οποίο μπορεί να πατήσει ο χρήστης.
- Μια μέθοδος main, που είναι η βασική εκτελέσιμη μέθοδος της εφαρμογής.

Η γραφική διεπαφή χρήστη που κατασκευάζεται από την κλάση δίνεται στην παρακάτω εικόνα:



Εικόνα 5.3: Αρχική γραφική διεπαφή χρήστη της εφαρμογής IntegraHEALTH 1.0

Αξίζει να σημειωθεί ότι όταν ο χρήστης επιλέξει οποιοδήποτε από τα τρία πρώτα κουμπιά, καλείται μία από τις κλάσεις που παρουσιάζονται στην επόμενη ενότητα και εμφανίζουν μόνο τις γραφικές διεπαφές χρήστη των ιατρών, ενώ αν επιλέξει το κουμπί «Είσοδος ως Χρήστης/Υποβολή Ερωτημάτων», τότε το αντίστοιχο μπλοκ εντολών της μεθόδου actionPerformed εκτελεί διαδοχικά τις παρακάτω διεργασίες:

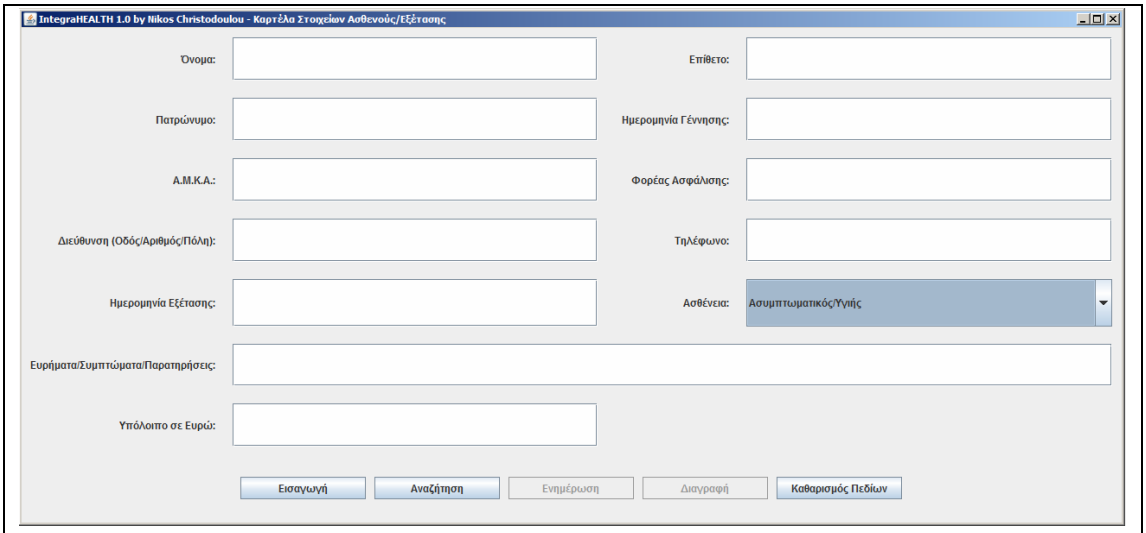
- Ενημέρωση των παραμένοντων μοντέλων από τους πίνακες καταγραφής των βάσεων των ιατρών (εκτελούνται όλες οι λειτουργίες των μονάδων μετασχηματισμού RDF της εφαρμογής, όπως αυτές αναφέρονται στην ενότητα 4.2).
- Εμφάνιση της γραφικής διεπαφής χρήστη που αντιστοιχεί στο προφίλ χρήστη του συστήματος, η οποία αναλύεται στην ενότητα 5.3.1.3.

5.3.1.2 Τα *πακέτα* doctor1.gui, doctor2.gui, doctor3.gui

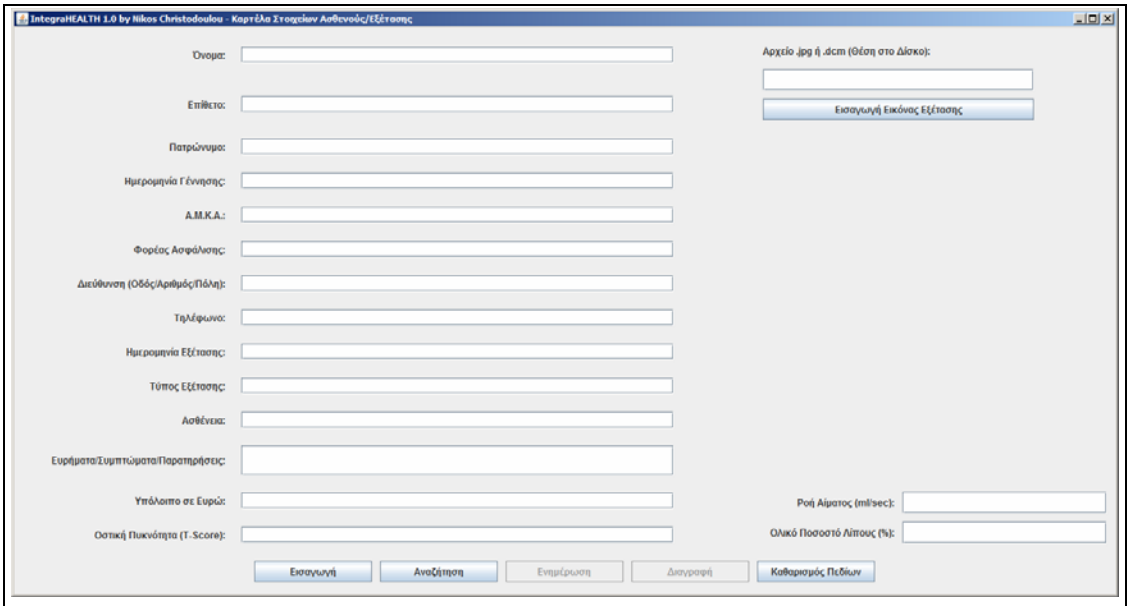
Τα τρία αυτά πακέτα φιλοξενούν από μία κλάση το καθένα, που είναι οι SQLFrame.java, DICOMFrame.java και HL7Frame.java αντίστοιχα, οι οποίες κατασκευάζουν τις γραφικές διεπαφές χρήστη των ιατρών. Και οι τρεις κλάσεις περιλαμβάνουν τις ίδιες μεθόδους:

- Μια μέθοδο κατασκευαστή, ομώνυμη της κλάσης, που κατασκευάζει τη γραφική διεπαφή.
- Μια βοηθητική μέθοδο addComponent, η οποία τοποθετεί τα διάφορα συστατικά (κουμπιά, ετικέτες, κ.τ.λ.) στο κατασκευαζόμενο παράθυρο.
- Μια μέθοδο actionPerformed, η οποία εκτελεί μια διαφορετική διαδικασία για κάθε κουμπί το οποίο μπορεί να πατήσει ο χρήστης.

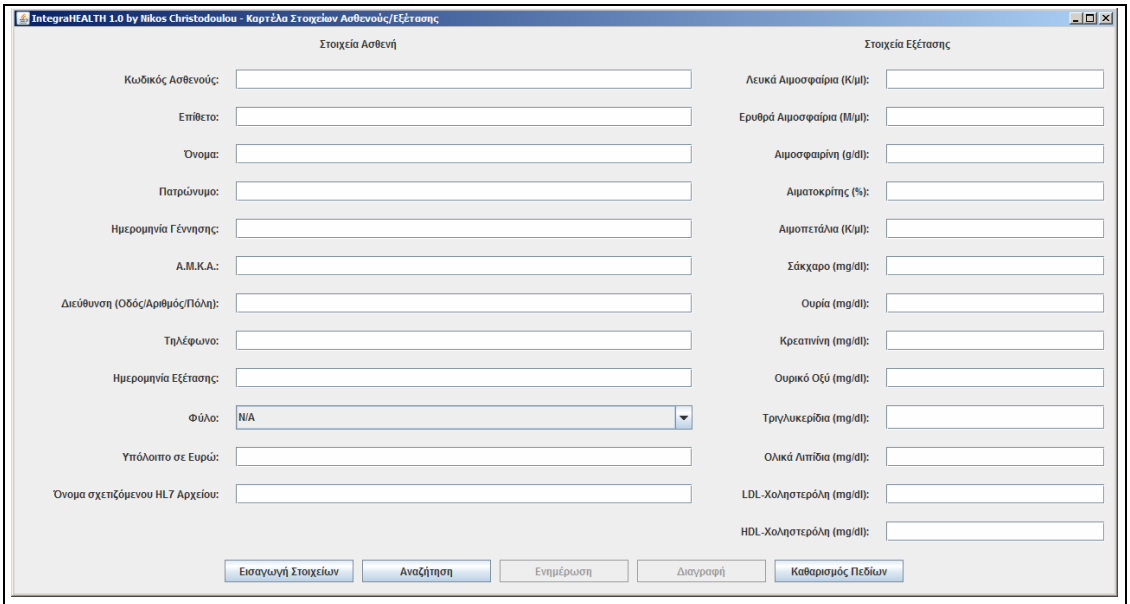
Και οι τρεις κλάσεις περιλαμβάνουν κλήσεις προς τις κλάσεις που υλοποιούν τους μηχανισμούς CRUD που εξασφαλίζουν σε κάθε ιατρό την πρόσβαση στη βάση του. Στις περιπτώσεις του δεύτερου και τρίτου ιατρού, πριν τα δεδομένα αποσταλούν στη βάση, καλούνται μέθοδοι από άλλα πακέτα (τα οποία θα παρουσιαστούν στη συνέχεια), που τα μετατρέπουν σε αρχεία DICOM ή HL7 αντίστοιχα. Ανάλογες κλήσεις υπάρχουν και προς τις μεθόδους ανάλυσης αρχείων προκειμένου να εξαχθούν τα δεδομένα από τα αρχεία των βάσεων ώστε να μπορούν να παρουσιαστούν στη γραφική διεπαφή χρήστη, σε περίπτωση αναζήτησης στοιχείων για κάποιο ασθενή. Οι γραφικές διεπαφές χρήστη που παράγονται από τις κλάσεις αυτές δίνονται στις παρακάτω εικόνες.



Εικόνα 5.4: Γραφική διεπαφή χρήστη του πρώτου ιατρού (παθολόγου)



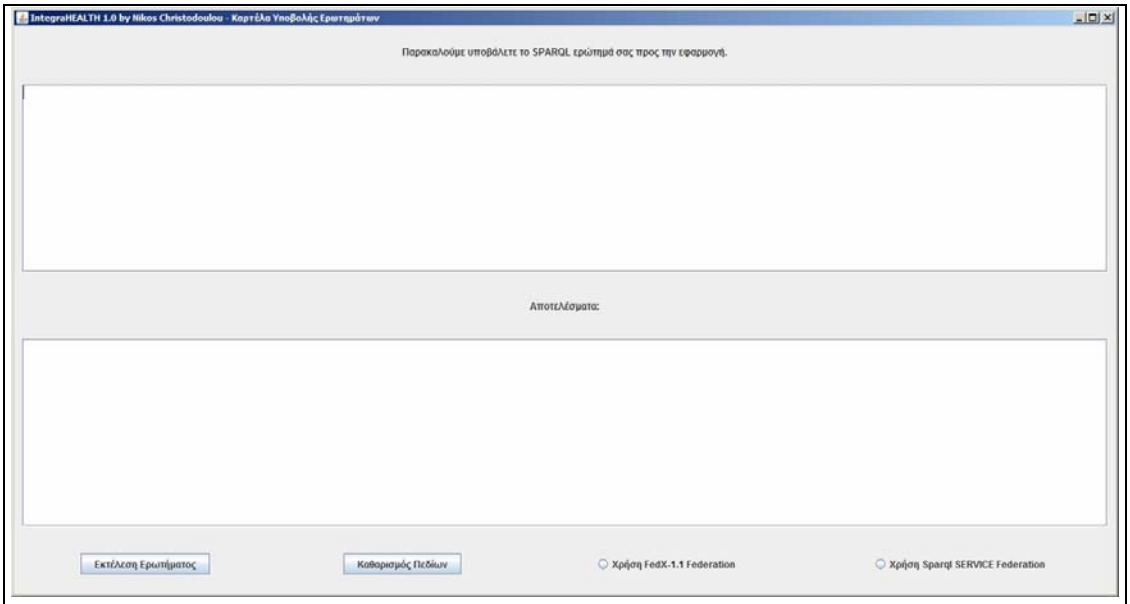
Εικόνα 5.5: Γραφική διεπαφή χρήστη του δεύτερου ιατρού (ακτινολόγου)



Εικόνα 5.6: Γραφική διεπαφή χρήστη του τρίτου ιατρού (μικροβιολόγου)

5.3.1.3 Το πακέτο user.gui

Στο συγκεκριμένο πακέτο περιλαμβάνεται η κλάση UserFrame.java, η οποία επιτελεί διπλό έργο. Αφ' ενός κατασκευάζει τη γραφική διεπαφή του χρήστη του συστήματος, αφ' ετέρου έχει ενσωματωμένες τις κλήσεις προς τις δύο μηχανές επεξεργασίας ερωτημάτων της εφαρμογής (μια απλή μηχανή ερωτημάτων SPARQL και το εργαλείο FedX για ομόσπονδα ερωτήματα). Η γραφική διεπαφή που δημιουργεί η κλάση αυτή δίνεται στην παρακάτω εικόνα:



Εικόνα 5.7: Γραφική διεπαφή του χρήστη του συστήματος

Η κλάση περιέχει τις ακόλουθες μεθόδους:

- Μια pubic μέθοδο κατασκευαστή, ομώνυμη της κλάσης, που κατασκευάζει τη γραφική διεπαφή.
- Μια βοηθητική μέθοδο private void addComponent, η οποία τοποθετεί τα διάφορα συστατικά (κουμπιά, ετικέτες, κ.τ.λ.) στο κατασκευαζόμενο παράθυρο.
- Μια μέθοδο public void actionPerformed, η οποία εκτελεί μια διαφορετική διαδικασία για κάθε κουμπί το οποίο μπορεί να πατήσει ο χρήστης.

Ο χρήστης, όταν ανοίξει τη συγκεκριμένη γραφική διεπαφή, καλείται να υποβάλει το ερώτημά του στο πεδίο κειμένου που βρίσκεται στο πάνω μισό του παραθύρου, να επιλέξει τη μηχανή εκτέλεσης ερωτημάτων που επιθυμεί, επιλέγοντας υποχρεωτικά ένα από τα δύο αμοιβαίως αποκλειόμενα κουμπιά (σε περίπτωση μη επιλογής, εμφανίζεται μήνυμα λάθους) και τέλος να επιλέξει το κουμπί «Εκτέλεση Ερωτήματος», το οποίο, ανάλογα με τη μηχανή απάντησης ερωτημάτων, αποκρίνεται με δύο τρόπους. Αν έχει επιλεγεί η προκαθορισμένη μηχανή απάντησης της SPARQL, τότε δημιουργεί το ερώτημα από το κείμενο που έχει δώσει ο χρήστης και αφού το ελέγξει η εφαρμογή, υποβάλλει σε κάθε τελικό σημείο το υποερώτημα που της αντιστοιχεί, σύμφωνα με τις προτάσεις SERVICE του ερωτήματος.

Σε περίπτωση που επιλεχθεί η μηχανή επεξεργασίας ομόσπονδων ερωτημάτων FedX, τότε το αντίστοιχο μπλοκ προτάσεων φορτώνει τα ονόματα και τα URI των τελικών σημείων από ένα αρχείο ρυθμίσεων εκτός της εφαρμογής, ακολουθώντας ουσιαστικά το τρίτο κατά σειρά παράδειγμα της ενότητας 5.1.4.1. Σε κάθε περίπτωση, τα δεδομένα επιστρέφονται στο χρήστη στο δεύτερο πεδίο κειμένου στο κάτω μέρος του παραθύρου, μορφοποιημένα κάθε φορά σύμφωνα με τις επιταγές της εκάστοτε μηχανής.

Και στις δύο περιπτώσεις εμφανίζεται η ταχύτητα υπολογισμού του ερωτήματος, κρατώντας ένα στιγμιότυπο της ώρας του συστήματος σε millisecond αμέσως πριν και αμέσως μετά την εκτέλεσή του. Αποτελεί την πιο ακριβή προσέγγιση που μπορεί να ληφθεί από τις μεθόδους της Java, καθώς ακόμα και οι εντολές δημιουργίας των στιγμιότυπων ώρας (timestamps) απαιτούν κάποια milliseconds για την εκτέλεσή τους.

5.3.1.4 Το πακέτο doctor1.handlers

Πρόκειται για το πακέτο, το οποίο υλοποιεί το μηχανισμό CRUD του παθολόγου ο οποίος χρησιμοποιεί απλή σχεσιακή βάση δεδομένων. Περιέχει τις κλάσεις: BasicFunctions.java, DataFunctions.java, DBTuple.java. Η πρώτη κλάση περιέχει τις μεθόδους loadDriver, connectToBase και disconnectFromBase οι οποίες είναι υπεύθυνες για τη σύνδεση του προγράμματος με την βάση δεδομένων του ιατρού μέσω JDBC. Η δεύτερη κλάση περιλαμβάνει τις μεθόδους createInBase, fetchFromBase, updateInBase και deleteFromBase, που, όπως υποδηλώνουν και τα ονόματά τους, ασχολούνται με την εισαγωγή, ανάκτηση, ενημέρωση και διαγραφή στοιχείων από τη βάση, αντίστοιχα. Η τρίτη κλάση περιλαμβάνει μια σειρά μεταβλητών, που απεικονίζει μια εγγραφή (πλειάδα) του πίνακα elements της βάσης του πρώτου ιατρού. Με αυτά τα αντικείμενα – πλειάδες γίνεται η ανταλλαγή των

πληροφοριών μεταξύ της βάσης και της γραφικής διεπαφής χρήστη, καθώς επιτρέπει τη μαζική μεταφορά ομαδοποιημένης πληροφορίας χωρίς να απαιτείται η αναφορά όλων των πεδίων ως ορίσματα στις μεθόδους της δεύτερης κλάσης. Περιλαμβάνονται οι εξής μεταβλητές: Όνομα (FName), Επώνυμο (LName), Πατρώνυμο (FatherName), Ημερομηνία Γέννησης (Birthday), Α.Μ.Κ.Α. (Amka), Ασφαλιστικός Οργανισμός (SecurityOrganization), Διεύθυνση (Address), Τηλέφωνο (Phone), Ημερομηνία Εξέτασης (ExaminationDate), Ασθένεια (Disease), Παρατηρήσεις (Remarks), Συνολικό Ποσό (sum).

5.3.1.5 Το πακέτο doctor2.handlers

Το συγκεκριμένο πακέτο είναι αντίστοιχο του πακέτου doctor1.handlers και υλοποιεί το σύστημα CRUD του ακτινολόγου. Περιέχει τις κλάσεις BasicFunctions.java, DICOMDBDataFunctions.java και DICOMTuple.java των οποίων οι λειτουργίες είναι ισοδύναμες με τις αντίστοιχες κλάσεις του προηγούμενου πακέτου. Για λόγους υπόδειξης της διαφορετικότητας των πληροφοριών που χειρίζεται ο κάθε ιατρός παρατίθεται η σειρά – πλειάδα μεταβλητών της κλάσης DICOMTuple: Όνομα (FName), Επώνυμο (LName), Πατρώνυμο (FatherName), Ημερομηνία Γέννησης (Birthday), Α.Μ.Κ.Α. (Amka), Ασφαλιστικός Οργανισμός (SecurityOrganization), Διεύθυνση (Address), Τηλέφωνο (Phone), Ημερομηνία Εξέτασης (ExaminationDate), Τύπος Εξέτασης (ExaminationType), Ασθένεια (Disease), Παρατηρήσεις (Remarks), Πυκνότητα Οστών (BoneDensity), Ροή Αίματος (BloodFlow), Ποσοστό Σωματικού Λίπους (BodyFat), Συνολικό Ποσό (sum), Όνομα Αρχείου (FileName). (Σύμφωνα με τα [18, 64, 87, 103], είναι δυνατόν να μετρηθούν το σωματικό λίπος, η ροή του αίματος και η οστική πυκνότητα με απεικονιστικές τεχνικές).

Εκτός από τις προηγούμενες κλάσεις, περιλαμβάνονται και οι απαραίτητες κλάσεις για τη δημιουργία αρχείων DICOM και την εξαγωγή πληροφοριών από αυτά. Αυτές είναι οι JpegToDicom.java, DicomToJpeg.java και ListDicomHeader.java. Η JpegToDicom.java μετατρέπει ένα αρχείο τύπου JPEG (κατάληξη .jpg) σε αρχείο τύπου dicom (κατάληξη .dcm), προσθέτοντας σε αυτό τα δεδομένα του ασθενή, σχηματίζοντας έτσι την απαραίτητη επικεφαλίδα DICOM (ενότητα 3.1.2.1). Η DicomToJpeg.java παίρνει ένα αρχείο DICOM και εξάγει τα δεδομένα εικονοστοιχείου από αυτό, δίνοντας την αρχική εικόνα JPEG του αρχείου, ενώ η ListDicomHeader.java αναλύει την επικεφαλίδα ενός αρχείου DICOM και εξάγει σε ένα πίνακα όλα τα δεδομένα της.

Πιο αναλυτικά, η κλάση JpegToDicom.java περιέχει τρεις μεθόδους:

- Την public void jpg2dcm που δέχεται ένα αντικείμενο DICOMTuple ως όρισμα και κατασκευάζει από αυτό ένα αρχείο DICOM.
- Την public File getDicomFile, η οποία επιστρέφει το αρχείο που παράγει η jpg2dcm.
- Την public String getDicomFileName, η οποία επιστρέφει το όνομα του αρχείου που παράγει η jpg2dcm.

Η jpg2dcm καλείται από τη μέθοδο actionPerformed της κλάσης DICOMFrame.java που αναφέρθηκε στην ενότητα 5.3.1.2. Αρχικά, δημιουργεί ένα κενό αρχείο DICOM στο οποίο δίνει ως όνομα το όνομα αρχείου της παραγόμενης JPEG εικόνας προσθέτοντας πριν από αυτό την ημερομηνία εξέτασης του ασθενή. Η κίνηση αυτή βοηθάει τον ιατρό στην εύρεση της κατάλληλης εξέτασης ενός ασθενή σε περίπτωση πολλαπλών επισκέψεων του τελευταίου, χωρίς να αντιβαίνει στους κανόνες ονοματολογίας αρχείων DICOM (μέρος 3.10 του προτύπου [72]). Στη συνέχεια το αρχείο εικόνας JPEG φορτώνεται σε ένα αντικείμενο BufferedImage και λαμβάνονται από αυτό τέσσερις ακέραιες τιμές: ο αριθμός των χρωματικών συνιστωσών (ColorComponents), το μέγεθος του εικονοστοιχείου σε bits (BitsPerPixel), το μέγεθος των δεσμευμένων bit (BitsAllocated) και τα δείγματα ανά εικονοστοιχείο (SamplesPerPixel).

Μετά τη λήψη αυτών των στοιχείων, κατασκευάζεται η επικεφαλίδα του αρχείου DICOM, παίρνοντας ένα αντικείμενο DicomObject, το οποίο δημιουργείται από μια κλάση του πακέτου dcm4che. Κατόπιν με την εκτέλεση μιας σειράς μεθόδων putString, putInt και putDate, εισάγονται όλα τα επιθυμητά δεδομένα στην επικεφαλίδα. Κάθε μια από τις παραπάνω μεθόδους προσθέτει ένα στοιχείο δεδομένων DICOM στην επικεφαλίδα. Τα ορίσματά τους είναι το όνομα της ετικέτας του στοιχείου, η αναπαράσταση τιμής του και τα δεδομένα που θα τοποθετηθούν στο πεδίο τιμής του στοιχείου, τα οποία πρέπει να συμφωνούν με το είδος της μεθόδου που χρησιμοποιείται (π.χ. στην putString τα δεδομένα πρέπει να είναι μια συμβολοσειρά και όχι ένας ακέραιος αριθμός). Αρχικά εισάγεται το σύνολο χαρακτήρων που χρησιμοποιείται στο αρχείο και στη συνέχεια τα αριθμητικά στοιχεία που σχετίζονται με την εικόνα, οι πληροφορίες που δίνει ο ιατρός, προσπελαύνοντας κάθε φορά την κατάλληλη μεταβλητή του εισαχθέντος αντικειμένου DICOMTuple, η ημερομηνία και ώρα δημιουργίας του αρχείου και τέλος μερικά μοναδικά αναγνωριστικά.

Στη συνέχεια δημιουργείται ένα ρεύμα εξόδου προς το κενό αρχείο DICOM και γράφονται σε αυτό τα δεδομένα της επικεφαλίδας. Έπειτα το αρχείο JPEG φορτώνεται σε ένα ρεύμα εισόδου, το οποίο συνδέεται με το ρεύμα εξόδου προς το αρχείο DICOM και γίνεται η μεταφορά της εικόνας χρησιμοποιώντας έναν πίνακα byte με 65536 στοιχεία σαν έναν ενδιάμεσο καταχωρητή (buffer) μεγέθους 64KB. Μετά και από αυτή τη διαδικασία ελέγχεται το μέγεθος της εικόνας και αν έχει μονό μήκος προστίθεται ένα επιπλέον μηδενικό ώστε το μήκος του στοιχείου δεδομένων της εικόνας να έχει ζυγό μήκος, όπως αναφέρεται στην ενότητα 3.1.2.4. Στο τέλος κλείνουν όλα τα ρεύματα εισόδου και εξόδου και το νέο αρχείο DICOM είναι έτοιμο. Η πρόσβαση στο σχηματιζόμενο αρχείο ή στο όνομά του γίνεται από τις μεθόδους getDicomFile, getDicomFileName αντίστοιχα.

Η κλάση DicomToJpeg.java αποτελείται από δύο μεθόδους:

- Την public void dcm2jpg που δέχεται ένα αρχείο DICOM ως όρισμα και εξάγει από αυτό τα δεδομένα εικόνας τα οποία αποθηκεύει σε ένα αρχείο JPEG.
- Την public File getJpegFile, η οποία επιστρέφει το αρχείο που παράγει η dcm2jpg.

Η dcm2jpg καλείται από τη μέθοδο fetchFromDicomBase της κλάσης DICOMDBDataFunctions.java, όταν επιχειρείται ανάκτηση δεδομένων από τη βάση. Ξεκινά ορίζοντας ένα κενό αντικείμενο BufferedImage στο οποίο θα τοποθετηθεί η εξαγόμενη εικόνα και φορτώνοντας από τις κλάσεις της διεπαφής Java Image I/O Technology τον κατάλληλο αναγνώστη (reader) για τον τύπο αρχείου DICOM και τις παραμέτρους του. Το αρχείο DICOM φορτώνεται σε ένα ειδικό ρεύμα εισόδου εικόνας, το οποίο δίνεται στον αναγνώστη με την εντολή ΟνομαΑναγνώστη.read(0, ΌνομαΠαραμέτρων) (στον κώδικα είναι γραμμένη ως Reader.read(0, param)) και στη συνέχεια εξάγονται τα δεδομένα του στο αντικείμενο BufferedImage.

Μετά την ανάγνωση του αρχείου DICOM, σχηματίζεται ένα κενό JPEG αρχείο με όνομα ίδιο με αυτό του DICOM αρχείου από το οποίο προήλθε και τοποθετείται σε ένα ρεύμα εξόδου, το οποίο συνδέεται με έναν κωδικοποιητή JPEG (η σχετική κλάση παρέχεται στη διεπαφή προγραμματισμού εφαρμογών Java Image I/O Technology). Με την εντολή ΌνομαΚωδικοποιητή.encode(ΌνομαΑντικειμένουBufferedImage) (στον κώδικα είναι γραμμένη ως encoder.encode(MyJpegImage)) τα δεδομένα εικόνας κωδικοποιούνται κατά JPEG και φορτώνονται στο σχετικό αρχείο. Το ρεύμα εξόδου κλείνει και το αρχείο JPEG είναι έτοιμο να μεταφερθεί σε όποια εφαρμογή καλέσει τη μέθοδο getJpegFile.

Για να γίνει η πλήρης εξαγωγή των πληροφοριών από το αρχείο DICOM και η παρουσίασή τους στη γραφική διεπαφή χρήστη του ακτινολόγου, η μέθοδος fetchFromDicomBase μετά την DicomToJpeg.java καλεί την κλάση ListDicomHeader.java στέλνοντάς της το αρχείο DICOM που εξήγαγε από τη βάση, αφού πρώτα το φορτώσει σε ένα αντικείμενο DicomObject μέσω ενός ειδικού ρεύματος εισόδου DicomInputStream. Το αντικείμενο αυτό δίνεται σαν όρισμα στη μοναδική μέθοδο της κλάσης ListDicomHeader, την public String [] [] listHeader, που επιστρέφει ένα δισδιάστατο πίνακα στοιχείων καθένα από τα οποία περιέχει μια συμβολοσειρά.

Η listHeader, αφού λάβει το DicomObject, δημιουργεί ένα αντικείμενο της κλάσης Iterator στο οποίο εκχωρεί τον αντίστοιχο iterator που σχετίζεται με δεδομένα επικεφαλίδας και είναι ενσωματωμένος στο DicomObject. Όσο το αντικείμενο iterator της μεθόδου βρίσκει στοιχεία δεδομένων στο αντικείμενο που περιέχει το αρχείο, εξάγονται σε ξεχωριστές μεταβλητές, το όνομα της ετικέτας του στοιχείου, η ετικέτα του στοιχείου και η αναπαράσταση τιμής του. Αν η τελευταία έχει τιμή SQ (Sequence of zero or more items, ακολουθία από μηδέν ή περισσότερα αντικείμενα) τότε στην αντίστοιχη σειρά του πίνακα εξόδου καταχωρούνται η ετικέτα, η αναπαράσταση και το όνομα της ετικέτας με αυτή τη σειρά και η μέθοδος καλεί τον εαυτό της για να εξετάσει τα περιεχόμενα του συγκεκριμένου στοιχείου δεδομένων. Διαφορετικά τοποθετείται σε μια ξεχωριστή μεταβλητή το περιεχόμενο του πεδίου τιμής του στοιχείου δεδομένων, οπότε στον πίνακα εξόδου καταχωρούνται η ετικέτα, η αναπαράσταση τιμής και η τιμή του στοιχείου με αυτή τη σειρά.

Για την έξοδο επιλέγεται ένας πίνακας 26x3 κελιών καθώς 26 είναι τα στοιχεία που εισάγονται στην επικεφαλίδα DICOM κατά τη δημιουργία τους από την κλάση JpegToDicom. Τονίζεται ότι τόσο από τις πληροφορίες του κεφαλαίου 3, όσο και από τα παραδείγματα του παραρτήματος Α, προκύπτει ότι είναι δυνατόν να επιλεχθεί από

τον κατασκευαστή ενός αρχείου DICOM πόσα και ποια χαρακτηριστικά θα χρησιμοποιήσει για να συντάξει την επικεφαλίδα του αρχείου, αφού η υποχρεωτική συμπερίληψη όλων των χαρακτηριστικών (των οποίων ο αριθμός υπενθυμίζεται ότι είναι πάνω από 3000) θα οδηγούσε σε υπερμεγέθη και ασύμφορα ως προς την επεξεργασία και την αποθήκευση αρχεία, τα οποία μάλιστα πιθανόν και να περιείχαν αρκετά στοιχεία δεδομένων χωρίς τιμές, αν η απεικονιστική διάταξη του ιατρού δεν παρείχε τις σχετικές πληροφορίες.

5.3.1.6 Το πακέτο doctor3.handlers

Το εν λόγω πακέτο κινείται στην ίδια περίπου γραμμή με το προηγούμενο καθώς υλοποιεί το σύστημα CRUD του μικροβιολόγου, περιλαμβάνοντας τις διαδικασίες μετατροπής πληροφορίας σε αρχεία μηνυμάτων HL7 και εξαγωγής πληροφορίας από αυτά. Περιέχει τις κλάσεις BasicFunctions.java, HL7DBDataFunctions.java και HL7Tuple.java με λειτουργία ισοδύναμη των αντίστοιχων κλάσεων των πακέτων των άλλων ιατρών. Η σειρά μεταβλητών της HL7Tuple.java είναι η εξής: Κωδικός Ασθενή (PatientID) (επιπλέον κωδικοποίηση που χρησιμοποιεί ο ιατρός για τους ασθενείς του), Όνομα (FName), Επώνυμο (LName), Πατρώνυμο (FatherName), Ημερομηνία Γέννησης (Birthday), Φύλο (Gender), Α.Μ.Κ.Α. (Amka), Διεύθυνση (Address), Τηλέφωνο (Phone), Ημερομηνία Εξέτασης (ExaminationDate), Συνολικό Ποσό (sum), Όνομα Αρχείου (FileName), ένας δισδιάστατος πίνακας από συμβολοσειρές με τις ονομασίες και τις μονάδες μέτρησης 13 βασικών στοιχείων αίματος και βιοχημικών στοιχείων και ένας μονοδιάστατος πίνακας δεκαδικών κινητής υποδιαστολής (Float) με τις μετρήσεις των εν λόγω στοιχείων. Τα στοιχεία είναι τα: Λευκά Αιμοσφαίρια, Ερυθρά Αιμοσφαίρια, Αιμοσφαιρίνη, Αιματοκρίτης, Αιμοπετάλια, Σάκχαρο, Ουρία, Κρεατινίνη, Ουρικό Οξύ, Τριγλυκερίδια, Ολικά Λιπίδια, LDL («κακή» χοληστερόλη) και HDL («καλή» χοληστερόλη).

Οι κλάσεις που υλοποιούν τις απαραίτητες μετατροπές δεδομένων από και προς τα HL7 αρχεία είναι οι HL7MessageCreator.java και HL7MessageParser.java. Για τις ανάγκες της εφαρμογής IntegraHEALTH 1.0 επιλέχθηκαν δύο τύποι μηνυμάτων HL7: κάθε επίσκεψη ασθενή δημιουργεί ένα μήνυμα ADTA04 (Register a Patient), ενώ η πιθανή ενημέρωση στοιχείων μιας επίσκεψης, δημιουργεί ένα μήνυμα ADTA08 (Update Patient Information), το οποίο αντικαθιστά το μήνυμα ADTA04 από το οποίο προήλθε (περισσότερα στοιχεία για τα μηνύματα ADT δίνονται στο [95]).

Η κλάση HL7MessageCreator.java περιλαμβάνει τις εξής μεθόδους:

- public void buildADTMessage με ορίσματα ένα αντικείμενο HL7Tuple και μια συμβολοσειρά με τον κωδικό του μηνύματος που θα κατασκευαστεί.
- Μια σειρά μεθόδων private String buildXXXSegment (όπου XXX το αναγνωριστικό του τμήματος) που κατασκευάζουν διαδοχικά τα τμήματα που θα περιλαμβάνονται στα μηνύματα HL7. Σύμφωνα με το προφορτωμένο αρχείο με όλες της πληροφορίες της έκδοσης 2.6 της εφαρμογής Chameleon της εταιρείας Interfaceware, σε ένα μήνυμα ADT πρέπει να υπάρχουν τα τμήματα MSH (Message Header), EVN (Event Type), PID (Patient

Identification), PV1 (Patient Visit). Επομένως περιλαμβάνεται μία μέθοδος για καθένα από τα προαναφερθέντα τμήματα, αλλά και για το τμήμα OBX (Observation/Result), το οποίο είναι προαιρετικό και μπορεί να υφίσταται πολλές φορές σε ένα μήνυμα.

- private String getDateTime που όταν κληθεί λαμβάνει την τρέχουσα ημερομηνία και ώρα, η οποία χρησιμοποιείται στα τμήματα MSH και EVN.
- public File getHL7MessageFile, που επιστρέφει το αρχείο μηνύματος που δημιουργείται από την buildADTMessage.
- public String getHL7MessageFileName, που επιστρέφει το όνομα του αρχείου μηνύματος που δημιουργείται από την buildADTMessage.

Η μέθοδος buildADTMessage καλείται από την μέθοδο actionPerformed της κλάσης HL7Frame.java όταν γίνεται μια νέα καταχώρηση ή μια ενημέρωση στοιχείων. Καλεί διαδοχικά τις μεθόδους για την κατασκευή των τμημάτων MSH, EVN, PID, PV1. Στη συνέχεια ελέγχει το αντικείμενο HL7Tuple για το ποιες μετρήσεις έχουν τιμές, και κατασκευάζει από ένα τμήμα OBX για κάθε μία, καλώντας την αντίστοιχη μέθοδο. Μετά κατασκευάζει ένα κενό αρχείο .hl7 και το κατονομάζει χρησιμοποιώντας τον κωδικό του μηνύματος, τον προσωπικό κωδικό του ασθενή (μεταβλητή PatientID της κλάσης HL7Tuple.java) και την ημερομηνία εξέτασης (αντίθετα με τα αρχεία DICOM, τα αρχεία .hl7 είναι απλά μια προτεινόμενη μορφή μηνυμάτων HL7 η οποία εγγυάται την εύκολη μεταφορά τους, επομένως δεν υφίστανται κάποιοι κανόνες ή περιορισμοί στην ονοματολογία τους). Τέλος φορτώνει το αρχείο σε ένα ρεύμα εξόδου, το οποίο συνδέεται σε ένα αντικείμενο OutputStreamWriter και εκεί γράφονται ένα-ένα τα τμήματα του μηνύματος, ακολουθούμενα από το χαρακτήρα αλλαγής γραμμής. Μετά το κλείσιμο των ρευμάτων, το αρχείο είναι έτοιμο και μπορεί να ληφθεί το ίδιο ή το όνομά του με τις μεθόδους getHL7MessageFile και getHL7MessageFileName, όπως και στην περίπτωση της JpegToDicom.java.

Οι μέθοδοι κατασκευής τμημάτων χρησιμοποιούν τις πληροφορίες της έκδοσης 2.6 του προτύπου από την εφαρμογή Chameleon για να κατασκευάσουν μια συμβολοσειρά που θα περιέχει τα επιθυμητά άλλά και τα ελάχιστα απαραίτητα πεδία, συστατικά ή και υποσυστατικά. Αν και η ακριβής δομή των τμημάτων δεν άπτεται του σκοπού της διπλωματικής εργασίας και δεν θα μελετηθεί περισσότερο, κάθε τμήμα περιέχει τα παρακάτω στοιχεία που εισάγει ο μικροβιολόγος:

- MSH: Περιέχει μόνο τον κωδικό του μηνύματος, την ημερομηνία και την ώρα δημιουργίας και άλλες εισαγωγικές πληροφορίες που σχηματίζονται αυτόματα χωρίς να τις παρέχει ο ιατρός.
- EVN: Περιέχει μόνο το δεύτερο μισό του κωδικού του μηνύματος και την ημερομηνία και ώρα δημιουργίας.
- PID: Περιέχει τα πεδία Κωδικός Ασθενή, Όνομα, Επώνυμο, Πατρώνυμο, Ημερομηνία Γέννησης, Φύλο, Α.Μ.Κ.Α., Διεύθυνση, Τηλέφωνο.
- PV1: Περιέχει τα πεδία Ημερομηνία Εξέτασης και Συνολικό Ποσό καθώς και το δεύτερο μισό του κωδικού του μηνύματος.

- OBX: Περιέχει το όνομα, την τιμή και τη μονάδα μέτρησης της αιματολογικής ή βιοχημικής μέτρησης για την οποία έχει κατασκευαστεί.

Η κλάση HL7MessageParser εκτελεί την αντίστροφη διαδικασία από την HL7MessageCreator. Περιλαμβάνει μόνο μία μέθοδο την pubic HL7Tuple readHL7FileBySegment, η οποία δέχεται σαν όρισμα ένα αρχείο μηνύματος HL7 και επιστρέφει ένα αντικείμενο HL7Tuple με όλα τα δεδομένα προς παρουσίαση στη γραφική διεπαφή χρήστη του μικροβιολόγου. Η μέθοδος καλείται από την μέθοδο fetchFromHL7Base της κλάσης HL7DBDataFunctions.java. Ξεκινά συνδέοντας το αρχείο σε ένα ρεύμα εισόδου και το διαβάζει ανά γραμμή δηλαδή ανά τμήμα. Ανάλογα με το πρώτο πεδίο κάθε τμήματος (το όνομά του) λαμβάνει το κατάλληλο στοιχείο και το προσθέτει στην κατάλληλη μεταβλητή του αντικειμένου HL7Tuple. Για όσες μετρήσεις δεν έχουν γίνει τίθεται στην κατάλληλη μεταβλητή του μονοδιάστατου πίνακα δεκαδικών τιμών η ενδεικτική τιμή «-0.0». Το ρεύμα εισόδου κλείνει και το αντικείμενο HL7Tuple είναι έτοιμο προς αποστολή.

5.3.1.7 Το πακέτο core.handlers

Πρόκειται για το πακέτο το οποίο εκτελεί όλες τις λειτουργίες που αναφέρονται στις μονάδες μετασχηματισμού RDF της εφαρμογής. Περιλαμβάνει τις κλάσεις:

- BasicFunctionsAndVariables.java, που περιέχει διαδικασίες για είσοδο και έξοδο από όλες τις βάσεις δεδομένων της εφαρμογής, κατασκευή και άνοιγμα παραμένοντος μοντέλου και τα URIs των οντολογιών που χρησιμοποιούνται για τη μετατροπή δεδομένων σε RDF.
- UpdateModels.java, η οποία ενημερώνει τα παραμένοντα μοντέλα από τα περιεχόμενα των πινάκων καταγραφής των βάσεων των ιατρών.
- Reasoning.java, η οποία επενεργεί σε μοντέλα εφαρμόζοντας τεχνικές συμπερασμού.
- SQLData2RDF.java, DICOMData2RDF.java και HL7Data2RDF.java, που σχηματίζουν RDF κόμβους από τα δεδομένα των βάσεων των ιατρών, ή διαγράφουν κόμβους από τα αντίστοιχα μοντέλα.

Με εξαίρεση την BasicFunctionsAndVariables.java, οι υπόλοιπες κλάσεις θα μελετηθούν σε ξεχωριστή υποενότητα η κάθε μία.

Η κλάση UpdateModels.java

Η συγκεκριμένη κλάση περιλαμβάνει τρεις μεθόδους:

- public static void updateSQLModel
- public static void updateDICOMModel
- public static void updateHL7Model

οι οποίες, όπως υποδηλώνει και το όνομά τους, είναι επιφορτισμένες με το να ενημερώνουν τα παραμένοντα μοντέλα των triple store των μονάδων μετασχηματισμού RDF, βασιζόμενες στις πρόσφατες ενημερώσεις των βάσεων των τριών ιατρών, όπως αυτές δίνονται στους πίνακες καταγραφής τους.

Η μέθοδος updateSQLModel ασχολείται με το μοντέλο που προκύπτει από τα δεδομένα του παθολόγου. Ξεκινά ανοίγοντας το αντίστοιχο triple store, ονόματι sql_rdf_model και φορτώνοντας το μοντέλο που έχει ήδη αποθηκευμένο. Ακολουθεί ένα ερώτημα προς τον πίνακα elements_log της βάσης doctor_1 για εγγραφές με το χαρακτηριστικό DELETE ή UPDATE και η κλήση της μεθόδου deleteRDFNode της κλάσης SQLData2RDF.java για κάθε επιστρεφόμενη εγγραφή, η οποία διαγράφει από το μοντέλο τους κόμβους που αντιστοιχούν στα αποτελέσματα του ερωτήματος.

Στη συνέχεια δημιουργεί ένα μοντέλο ModelD2RQ με όρισμα το αρχείο Doctor_1_Log_Map.n3. Το εν λόγω αρχείο μετατρέπει τα περιεχόμενα του πίνακα elements_log της βάσης doctor1 (με εξαίρεση τις εγγραφές που περιέχουν το χαρακτηριστικό DELETE) σε προτάσεις RDF, οι οποίες φορτώνονται στο ModelD2RQ. Το μοντέλο αυτό στη συνέχεια προστίθεται με μια μέθοδο add στο ανοιχτό μοντέλο της βάσης. Κατόπιν διαγράφονται όλα τα δεδομένα του πίνακα καταγραφής και κλείνει η σύνδεση προς τη βάση doctor1.

Τέλος, καλεί τη μέθοδο NewInferenceModel της κλάσης Reasoning.java με ορίσματα το ενημερωμένο παραμένον μοντέλο και το αρχείο SQL_RULES.rules που περιλαμβάνει τους κανόνες συμπερασμού που έχουν επιλεχθεί για τα δεδομένα του παθολόγου. Εκεί τα δύο στοιχεία συνδέονται επιστρέφοντας ένα μοντέλο τύπου InfModel, τα περιεχόμενα του οποίου προστίθενται εκ νέου στο παραμένον μοντέλο (κατά την πρόσθεση αποτρέπεται η εισαγωγή διπλών προτάσεων στο μοντέλο). Τότε κλείνει και η σύνδεση προς το triple store sql_rdf_model και το μοντέλο είναι έτοιμο προς δημοσίευση.

Οι μέθοδοι updateDICOMModel και updateHL7Model ακολουθούν ακριβώς την ίδια διαδικασία με τη διαφορά ότι λόγω της αποθήκευσης αρχείων και όχι μεμονωμένων τιμών στη βάση, για τη δημιουργία κόμβων RDF ακολουθείται η εξής διαδικασία:

- Υποβολή δεύτερου ερωτήματος στη βάση του ιατρού για όλες τις εγγραφές με χαρακτηριστικά INSERT ή UPDATE.
- Κλήση της μεθόδου fetchFromXBase της κλάσης XDBDataFunctions.java (όπου X = DICOM, HL7) για κάθε εγγραφή – αποτέλεσμα του παραπάνω ερωτήματος.
- Ανάκτηση των ζητούμενων αρχείων από τις βάσεις, τα οποία επεξεργάζονται με τις ειδικές μεθόδους που παρουσιάστηκαν στις προηγούμενες ενότητες.
- Τροφοδοσία των δεδομένων των αρχείων υπό τη μορφή αντικειμένων XTuple στις μεθόδους xTuple2RdfNode (όπου x = dicom, hl7) των κλάσεων XData2RDF.java για τη δημιουργία νέων RDF κόμβων.

Οι υπόλοιπες διαφορές μεταξύ των δύο αυτών μεθόδων και της updateSQLModel εντοπίζονται στα ονόματα των triple store (x_rdf_model) και στα αρχεία κανόνων που χρησιμοποιούνται από τη μέθοδο NewInferenceModel της κλάσης Reasoning.java (X_RULES.rules).

Η κλάση Reasoning.java

Εδώ φιλοξενείται η μέθοδος public static InfModel NewInferenceModel, η οποία όπως αναφέρθηκε και παραπάνω δέχεται σαν ορίσματά της ένα μοντέλο και μια συμβολοσειρά ή οποία είναι η διαδρομή του αρχείου κανόνων. Η διαδικασία ξεκινά ορίζοντας μια λίστα αντικειμένων Rule (κανόνας), στην οποία φορτώνονται οι κανόνες του εισαγόμενου αρχείου με τη μέθοδο Rule.rulesFronURI(ΔιαδρομήΑρχείουΚανόνων). Στη συνέχεια κατασκευάζεται ένα αντικείμενο τύπου GenericRuleReasoner με όρισμα την προαναφερθείσα λίστα κανόνων. Τέλος δημιουργείται ένα μοντέλο συμπερασμού (InfModel) με χρήση του κατασκευαστή ModelFactory.createInfModel(ΌνομαΜηχανήςΣυλλογιστικής, ΌνομαΜοντέλου), το οποίο και επιστρέφεται σαν τιμή της μεθόδου.

Αν και πρόκειται για μια «μικρή» μέθοδο (6 γραμμές κώδικας) προτιμήθηκε η συγγραφή ξεχωριστής κλάσης καθαρά για λόγους αντικειμενοστρέφειας.

Η κλάση DICOMData2RDF.java

Η κλάση αυτή αναλαμβάνει το έργο της μετατροπής των δεδομένων του ακτινολόγου από αρχεία DICOM σε RDF. Περιλαμβάνει δύο μεθόδους:

- public static void dicomTuple2RdfNode, που λαμβάνει ως όρισμα ένα αντικείμενο DICOMTuple και κατασκευάζει από αυτό έναν RDF κόμβο.
- public static void deleteRDFNode, που λαμβάνει το όνομα ενός αρχείου DICOM και διαγράφει το σημασιολογικό δίκτυο που έχει ως ρίζα τον κόμβο που σχετίζεται με το αρχείο.

Η dicomTuple2RdfNode ξεκινά τη διαδικασία σχηματίζοντας ένα URI που θα σχετίζεται με το αρχείο από το οποίο προήλθε η DICOMTuple που της δόθηκε σαν είσοδος. Αυτό το πετυχαίνει λαμβάνοντας το URI http://example.org/doctor2/ και συνδέοντάς το με το όνομα του DICOM αρχείου, το οποίο περιλαμβάνεται στην DICOMTuple.

Στη συνέχεια συνδέεται με τη βάση dicom_rdf_model που περιέχει το παραμένον μοντέλο και αρχίζει την κατασκευή του RDF κόμβου και του σημασιολογικού του δικτύου ορίζοντας ένα αντικείμενο τύπου Resource στο οποίο δημιουργείται ένας πόρος με τιμή το URI που σχηματίστηκε προηγουμένως. Μετά από αυτό το βήμα συνεχίζει προσθέτοντας τα υπόλοιπα στοιχεία σαν ιδιότητες του πόρου ακολουθώντας την παρακάτω διαδικασία:

- Ορίζεται ο τύπος (κλάση) του πόρου προσθέτοντας την ιδιότητα rdf:type και αντικείμενο την κλάση Health_Care_Visit της οντολογίας NCI Thesaurus. Η συγκεκριμένη κλάση επιλέχθηκε διότι λόγω της παραδοχής 5 της ενότητας 4.3 τα στοιχεία αναφέρονται ουσιαστικά σε επισκέψεις ιατρών (αυτό άλλωστε κωδικοποιείται με τα πρωτεύοντα κλειδιά των βάσεων των ιατρών) και για το γεγονός ότι δεν έχει καμία απαίτηση ή περιορισμό ως προς τις ιδιότητες που μπορεί να δεχθεί.

- Το όνομα του ασθενή αντιστοιχίζεται στην ιδιότητα τύπου δεδομένων (DatatypeProperty) has_first_name της οντολογίας MGED και κωδικοποιείται σαν ένα τυποποιημένο λεκτικό με τύπο δεδομένων XSDstring.
- Το επώνυμο του ασθενή αντιστοιχίζεται στην ιδιότητα τύπου δεδομένων (DatatypeProperty) has_last_name της οντολογίας MGED και κωδικοποιείται σαν ένα τυποποιημένο λεκτικό με τύπο δεδομένων XSDstring.
- Το πατρώνυμο του ασθενή αντιστοιχίζεται στην ιδιότητα τύπου δεδομένων (DatatypeProperty) givenName της οντολογίας FOAF και κωδικοποιείται σαν ένα τυποποιημένο λεκτικό με τύπο δεδομένων XSDstring.
- Για την ημερομηνία γέννησης του ασθενή χρησιμοποιείται η ιδιότητα αντικειμένου (ObjectProperty) has_nodes της οντολογίας MGED η οποία λαμβάνει σαν τιμή της έναν νέο πόρο με URI που προκύπτει από το URI του αρχικού πόρου προσθέτοντας τη συμβολοσειρά "/Birthday". Ο κόμβος αυτός περιλαμβάνει τις ιδιότητες:
  - rdf:type με τιμή την κλάση Date_of_Birth της οντολογίας NCI Thesaurus που υποδηλώνει τον τύπο του κόμβου.
  - has_value, που είναι μια ιδιότητα τύπου δεδομένων της οντολογίας MGED με τιμή την ημερομηνία γέννησης του ασθενή, η οποία κωδικοποιείται σαν ένα τυποποιημένο λεκτικό με τύπο δεδομένων XSDdate.

  Για λόγους συντομίας, από εδώ και στο εξής η διαδικασία αυτή θα αναφέρεται ως Διαδικασία Ανάθεσης Τιμής Ιδιότητας Αντικειμένου (Δ.Α.Τ.Ι.Α.)
- Ο Α.Μ.Κ.Α. του ασθενή αντιστοιχίζεται στην ιδιότητα τύπου δεδομένων (DatatypeProperty) unique_identifier της οντολογίας MGED και κωδικοποιείται σαν ένα τυποποιημένο λεκτικό με τύπο δεδομένων XSDstring.
- Για τον ασφαλιστικό οργανισμό του ασθενή ακολουθείται η Δ.Α.Τ.Ι.Α. με την ιδιότητα αντικειμένου has_nodes της οντολογίας MGED, το επιπλέον αναγνωριστικό "/SecurityOrg" στο αρχικό URI για το σχηματισμό του URI του πόρου – αντικειμένου, την κλάση Primary_Healthcare_Payer της οντολογίας NCI Thesaurus, την ιδιότητα τύπου δεδομένων has_value της οντολογίας MGED και τον τύπο δεδομένων XSDstring για την τυποποίηση του λεκτικού.
- Η διεύθυνση του ασθενή αντιστοιχίζεται στην ιδιότητα τύπου δεδομένων (DatatypeProperty) has_address της οντολογίας MGED και κωδικοποιείται σαν ένα τυποποιημένο λεκτικό με τύπο δεδομένων XSDstring.
- Το τηλέφωνο του ασθενή αντιστοιχίζεται στην ιδιότητα τύπου δεδομένων (DatatypeProperty) has_phone της οντολογίας MGED και κωδικοποιείται σαν ένα τυποποιημένο λεκτικό με τύπο δεδομένων XSDstring.
- Για την ημερομηνία εξέτασης του ασθενή ακολουθείται η Δ.Α.Τ.Ι.Α. με την ιδιότητα αντικειμένου has_nodes της οντολογίας MGED, το επιπλέον αναγνωριστικό "/ExaminationDate" στο αρχικό URI για το σχηματισμό του URI του πόρου – αντικειμένου, την κλάση Physical_Examination_Date της οντολογίας NCI Thesaurus, την ιδιότητα τύπου δεδομένων has_value της

οντολογίας MGED και τον τύπο δεδομένων XSDdate για την τυποποίηση του λεκτικού.

- Για την διάγνωση ακολουθείται η Δ.Α.Τ.Ι.Α. με την ιδιότητα αντικειμένου has_disease_state της οντολογίας MGED, το επιπλέον αναγνωριστικό "/Diagnosis" στο αρχικό URI για το σχηματισμό του URI του πόρου – αντικειμένου, την κλάση Diagnosis της οντολογίας NCI Thesaurus, την ιδιότητα τύπου δεδομένων has_value της οντολογίας MGED και τον τύπο δεδομένων XSDstring για την τυποποίηση του λεκτικού.
- Για τις παρατηρήσεις (συμπτώματα) ακολουθείται η Δ.Α.Τ.Ι.Α. με την ιδιότητα αντικειμένου has_clinical_finding της οντολογίας MGED, το επιπλέον αναγνωριστικό "/Findings" στο αρχικό URI για το σχηματισμό του URI του πόρου – αντικειμένου, την κλάση Finding της οντολογίας NCI Thesaurus, την ιδιότητα τύπου δεδομένων has_value της οντολογίας MGED και τον τύπο δεδομένων XSDstring για την τυποποίηση του λεκτικού.
- Για το συνολικό ποσό πληρωμής ακολουθείται η Δ.Α.Τ.Ι.Α. με την ιδιότητα αντικειμένου has_nodes της οντολογίας MGED, το επιπλέον αναγνωριστικό "/Payment" στο αρχικό URI για το σχηματισμό του URI του πόρου – αντικειμένου, την κλάση Payment της οντολογίας NCI Thesaurus, την ιδιότητα τύπου δεδομένων has_value της οντολογίας MGED και τον τύπο δεδομένων XSDfloat για την τυποποίηση του λεκτικού.
- Για το όνομα του DICOM αρχείου ακολουθείται η Δ.Α.Τ.Ι.Α. με την ιδιότητα αντικειμένου has_nodes της οντολογίας MGED, το επιπλέον αναγνωριστικό "/FileName" στο αρχικό URI για το σχηματισμό του URI του πόρου – αντικειμένου, την κλάση External_Filename της οντολογίας NCI Thesaurus, την ιδιότητα τύπου δεδομένων has_value της οντολογίας MGED και τον τύπο δεδομένων XSDstring για την τυποποίηση του λεκτικού.
- Για τον τύπο της εξέτασης ακολουθείται η Δ.Α.Τ.Ι.Α. με την ιδιότητα αντικειμένου has_nodes της οντολογίας MGED, το επιπλέον αναγνωριστικό "/ExaminationMethod" στο αρχικό URI για το σχηματισμό του URI του πόρου – αντικειμένου, την κλάση Examination_Method της οντολογίας NCI Thesaurus, την ιδιότητα τύπου δεδομένων has_value της οντολογίας MGED και τον τύπο δεδομένων XSDstring για την τυποποίηση του λεκτικού.
- Για τη μέτρηση της ροής του αίματος ακολουθείται η Δ.Α.Τ.Ι.Α. με την ιδιότητα αντικειμένου has_measurement της οντολογίας MGED, το επιπλέον αναγνωριστικό "/BloodFlow" στο αρχικό URI για το σχηματισμό του URI του πόρου – αντικειμένου, την κλάση Blood_Flow_Rate της οντολογίας NCI Thesaurus, την ιδιότητα τύπου δεδομένων has_value της οντολογίας MGED και τον τύπο δεδομένων XSDfloat για την τυποποίηση του λεκτικού.
- Για τη μέτρηση του ποσοστού σωματικού λίπους ακολουθείται η Δ.Α.Τ.Ι.Α. με την ιδιότητα αντικειμένου has_measurement της οντολογίας MGED, το επιπλέον αναγνωριστικό "/BodyFat" στο αρχικό URI για το σχηματισμό του URI του πόρου – αντικειμένου, την κλάση Adipose_Tissue της οντολογίας

NCI Thesaurus, την ιδιότητα τύπου δεδομένων has_value της οντολογίας MGED και τον τύπο δεδομένων XSDfloat για την τυποποίηση του λεκτικού.

- Για τη μέτρηση της οστικής πυκνότητας ακολουθείται η Δ.Α.Τ.Ι.Α. με την ιδιότητα αντικειμένου has_measurement της οντολογίας MGED, το επιπλέον αναγνωριστικό "/BoneDensity" στο αρχικό URI για το σχηματισμό του URI του πόρου – αντικειμένου, την κλάση Bone_Mineral_Density_Test της οντολογίας NCI Thesaurus, την ιδιότητα τύπου δεδομένων has_value της οντολογίας MGED και τον τύπο δεδομένων XSDfloat για την τυποποίηση του λεκτικού.

Μετά την εισαγωγή όλων των δεδομένων, του κόμβου και των υποκόμβων στο μοντέλο, κλείνει η σύνδεση προς τη βάση dicom_model_rdf.

Η μέθοδος deleteRDFNode ξεκινά συνδεόμενη με τη βάση dicom_model_rdf και εισέρχεται στο μοντέλο που υπάρχει εκεί. Τότε χρησιμοποιεί το όνομα του αρχείου που έλαβε σαν όρισμα και ελέγχει αν υπάρχει κάποιος κόμβος με όνομα URI http://example.org/doctor2/ΌνομαΑρχείουDICOM. Αν τον εντοπίσει τότε τον διαγράφει χρησιμοποιώντας την εντολή remove που διαθέτει το μοντέλο ως αντικείμενο Model με όρισμα την μέθοδο listStatements που με τη σειρά της έχει όρισμα ένα αντικείμενο τύπου SimpleSelector, ο οποίος έχει σαν όρισμα μια τριάδα που αποτελείται από το URI του κόμβου και δύο τιμές null, κάτι που σημαίνει ότι επιλέγει όλες τις προτάσεις με υποκείμενο το συγκεκριμένο URI. Η εντολή έχει την εξής μορφή: ΌνομαΜοντέλου.remove(ΌνομαΜοντέλου.listStatements(new SimpleSelector (ΌνομαΜοντέλου.getResource(URIΚόμβου), null, (RDFNode) null))) Μετά εκτελεί την ίδια εντολή για κάθε υποκόμβο με όρισμα κάθε φορά το παραγόμενο URI μέσω της Δ.Α.Τ.Ι.Α. (Δηλαδή όπου URIΚόμβου στην παραπάνω εντολή το όρισμα θα είναι URIΚόμβου+"/Diagnosis", URIΚόμβου+"/Findings", κ.ο.κ.

Μετά τη διαγραφή όλων των σχετικών προτάσεων κλείνει η σύνδεση προς τη βάση dicom_rdf_model.

Η κλάση HL7Data2RDF.java

Η κλάση αυτή αναλαμβάνει το έργο της μετατροπής των δεδομένων του μικροβιολόγου από αρχεία μηνυμάτων HL7 σε RDF. Ως προς τη δομή και τη λειτουργία της είναι πανομοιότυπη με την DICOMData2RDF.java. Οι αντίστοιχες μέθοδοι που περιλαμβάνει είναι οι public static void hl7Tuple2RdfNode και public static void deleteRDFNode, με λειτουργίες ίδιες με αυτές των ομολόγων τους στην προηγούμενη κλάση. Οι μόνες διαφορές εντοπίζονται στο πλήθος των πληροφοριών προς αντιστοίχιση σε προτάσεις RDF και στο διαφορετικό βασικό URI που είναι το http://example.org/doctor3/ΌνομαΑρχείουΜηνύματοςHL7.

Από τις διαθέσιμες πληροφορίες ενός μηνύματος HL7, οι Όνομα, Επώνυμο, Πατρώνυμο, Ημ. Γέννησης, Α.Μ.Κ.Α., Διεύθυνση, Τηλέφωνο, Ημ. Εξέτασης, Συνολικό Ποσό και Όνομα Αρχείου καθώς και η αρχική ιδιότητα που σχετίζονται με τον τύπο του σχηματιζόμενου κόμβου, αντιστοιχίζονται στις ίδιες κλάσεις και

ιδιότητες όπως και στην DICOMData2RDF.java. Για τα υπόλοιπα δεδομένα οι αντιστοιχίσεις έχουν ως εξής:

- Το φύλο του ασθενή αντιστοιχίζεται στην ιδιότητα τύπου δεδομένων (DatatypeProperty) gender της οντολογίας FOAF και κωδικοποιείται σαν ένα τυποποιημένο λεκτικό με τύπο δεδομένων XSDstring.
- Ο προσωπικός κωδικός του ασθενή αντιστοιχίζεται στην ιδιότητα τύπου δεδομένων (DatatypeProperty) has_ID της οντολογίας MGED και κωδικοποιείται σαν ένα τυποποιημένο λεκτικό με τύπο δεδομένων XSDstring.
- Για τις 13 αιματολογικές μετρήσεις ακολουθείται η Δ.Α.Τ.Ι.Α. με την ιδιότητα αντικειμένου has_measurement της οντολογίας MGED, ένα επιπλέον αναγνωριστικό στο αρχικό URI για το σχηματισμό του URI του πόρου – αντικειμένου, μια κλάση της οντολογίας NCI Thesaurus, την ιδιότητα τύπου δεδομένων has_value της οντολογίας MGED και τον τύπο δεδομένων XSDfloat για την τυποποίηση του λεκτικού, αν ο ιατρός έχει δώσει τιμή για τη μέτρηση. Αν αυτό δεν έχει γίνει τότε για την ενδεικτική τιμή «-0» χρησιμοποιείται ο τύπος δεδομένων XSDstring. Στον παρακάτω πίνακα δίνονται τα στοιχεία που χρησιμοποιούνται για κάθε μέτρηση.

Μέτρηση	Αναγνωριστικό	Κλάση οντολογίας NCI Thesaurus
Λευκά Αιμοσφαίρια	/Leukocytes	Leukocyte_Count
Ερυθρά Αιμοσφαίρια	/Erythrocytes	Erythrocyte_Count
Αιμοσφαιρίνη	/Hemoglobin	Hemoglobin_Measurement
Αιματοκρίτης	/Hematocrit	Hematocrit
Αιμοπετάλια	/Platelets	Platelet_Count
Σάκχαρο	/BloodSugar	Fasting_Blood_Sugar_Measurement
Ουρία	/Urea	Urea_Measurement
Κρεατινίνη	/Creatinine	Creatinine_Measurement
Ουρικό Οξύ	/UricAcid	Uric_Acid_Crystal_Measurement
Τριγλυκερίδια	/Triglyceride	Triglyceride_Measurement
Ολικά Λιπίδια	/Lipids	Lipid_Measurement
LDL	/LDL	Serum_LDL_Cholesterol_Measurement
HDL	/HDL	Serum_HDL_Cholesterol_Measurement

Πίνακας 5.1: Στοιχεία για την Δ.Α.Τ.Ι.Α. που αφορά τις μετρήσεις του μικροβιολόγου.

Η κλάση SQLData2RDF.java

Πρόκειται για την κλάση μετατροπής δεδομένων σε RDF που ασχολείται με το μοντέλο του παθολόγου. Αν και το όνομά της για λόγους ομοιομορφίας είναι της ίδιας μορφής με τις δύο παραπάνω ομόλογες κλάσεις της, ουσιαστικά δεν έχει την ίδια λειτουργικότητα με αυτές, καθώς περιέχει μόνο τη μέθοδο public static void deleteRDFNode, αφού για την αντίστοιχη λειτουργία εισαγωγής δεδομένων στο μοντέλο, χρησιμοποιείται το αρχείο Doctor1_Log_Map.n3 σε συνδυασμό με κατάλληλες κλάσεις και μεθόδους του εργαλείου D2RQ, όπως περιγράφθηκε στην αρχή της ενότητας. Η συγκεκριμένη κλάση δρα επικουρικά ως προς το αρχείο αυτό,

καθώς στην πλατφόρμα D2RQ δεν υπάρχει μηχανισμός παρακολούθησης των αλλαγών που γίνονται στα δεδομένα μιας σχεσιακής βάσης (εισαγωγή, ενημέρωση, διαγραφή), παρά μόνο η μετατροπή ενός στιγμιότυπού της σε RDF με χρήση του κατάλληλου αρχείου αντιστοιχίσεων. Η αδυναμία αυτή οδήγησε στη δημιουργία της παρούσας κλάσης, αλλά και στην παραδοχή 1 της ενότητας 4.3.

Η μέθοδος deleteRDFNode εκτελεί την ίδια λειτουργία με τις αντίστοιχες μεθόδους των δύο προηγούμενων κλάσεων. Εκτός από τον κυρίως κόμβο, ο οποίος έχει URI http://example.org/doctor1/ΠρωτεύονΚλειδί διαγράφονται και οι σχετικοί υποκόμβοι για την Ημερομηνία Γέννησης, τον Ασφαλιστικό Οργανισμό, την Ημερομηνία Εξέτασης, τις Παρατηρήσεις (συμπτώματα) και το Συνολικό Ποσό. Οι λεπτομέρειες για την υλοποίηση των κόμβων και τα αναγνωριστικά που χρησιμοποιούνται για να επεκτείνουν το παραπάνω βασικό URI δίνονται σε επόμενη ενότητα όπου θα περιγραφεί το αρχείο Doctor1_Log_Map.n3.

5.3.1.8 Το πακέτο core.firstTime

Στο πακέτο αυτό περιέχεται η κλάση FirstTimeInitialization.java η οποία πρέπει να εκτελεστεί μία φορά πριν την πρώτη εκτέλεση της κυρίως εφαρμογής. Περιέχει τρεις μεθόδους:

- public static void main, μια εκτελέσιμη μέθοδο, η δεύτερη στον κώδικα της IntegraHEALTH 1.0.
- private static void initializeCommonModelPart, η οποία συγκεντρώνει σε ένα μοντέλο τα αρχεία των τριών οντολογιών που αναφέρθηκαν στην ενότητα 5.2.2 και ορίζει τα προθέματά τους.
- public static void createPersistentModel, που συνδέεται σε ένα triple store μιας μονάδας μετασχηματισμού RDF και δημιουργεί ένα παραμένον μοντέλο εισάγοντας τις πληροφορίες του μοντέλου που δημιουργεί η προηγούμενη μέθοδος.

5.3.2 Τα εξωτερικά αρχεία της εφαρμογής

5.3.2.1 Το αρχείο Doctor1_Log_Map.n3

Όπως εξηγήθηκε και στις προηγούμενες ενότητες, το συγκεκριμένο αρχείο χρησιμοποιείται από την πλατφόρμα D2RQ για να μετατρέψει τα δεδομένα της βάσης doctor_1 σε προτάσεις RDF. Όντας ένα αρχείο τύπου n3, πρόκειται ουσιαστικά για ένα έγγραφο RDF, το οποίο έχει την εξής δομή:

- Προθέματα: Περιλαμβάνονται τα προθέματα rdf, rdfs, d2rq, foaf, ncit, mged των οποίων τα URI των χώρων ονομάτων έχουν ήδη δοθεί στο έγγραφο, καθώς και τα ακόλουθα:
  - map: <#>. Αναφέρεται στο ίδιο το έγγραφο και προηγείται κάθε στήλης πίνακα που αντιστοιχίζεται σε μια ιδιότητα RDF.
  - db: <jdbc:mysql://localhost/doctor1#>. Πρόκειται για το χώρο ονομάτων που προκύπτει από το URI της βάσης.

  - xsd: <http://www.w3.org/2001/XMLSchema#>. Ο χώρος ονομάτων για τους τύπους δεδομένων που χρησιμοποιούνται σε τυποποιημένα λεκτικά.
  - jdbc: <http://d2rq.org/terms/jdbc/>. Ο χώρος ονομάτων που ορίζουν οι κατασκευαστές του D2RQ για το Java DataBase Connectivity.
- Ένα στοιχείο d2rq:Database που αντιστοιχίζεται σε όλη τη βάση δεδομένων και έχει τις ιδιότητες:
  - d2rq:jdbcDriver, με τιμή το όνομα της κλάσης java του οδηγού για το σύστημα MySQL.
  - d2rq:jdbcDSN, με τιμή το πλήρες URI της βάσης (jdbc:mysql://localhost/doctor1).
  - d2rq:username, που αναφέρεται στο όνομα χρήστη της βάσης.
  - d2rq:password, που αναφέρεται στον κωδικό πρόσβασης της βάσης.
  - jdbc:autoReconnect που ορίζει τη συμπεριφορά του JDBC όταν η σύνδεση με τη βάση παραμένει ανενεργή πολλή ώρα και δοκιμάζει να κλείσει από μόνη της [67].
  - jdbc:zeroDateTimeBehavior, που ορίζει τη συμπεριφορά του JDBC όταν υπάρχουν μηδενικές τιμές ημερομηνίας ή ώρας στις στήλες [67].
- Ένα στοιχείο d2rq:ClassMap που αντιστοιχίζεται στον πίνακα elements_log και έχει τις ιδιότητες:
  - d2rq:datastorage, που εκφράζει το χώρο αποθήκευσης δεδομένων από τον οποίο θα αντλήσει δεδομένα, με τιμή την προηγούμενη αντιστοίχιση.
  - d2rq:uriPattern, που περιέχει το URI κάθε εγγραφής του πίνακα (http://example.org/doctor1/AMKA_ΗμερομηνίαΕξέτασης).
  - d2rq:class, που έχει ως τιμή την κλάση στην οποία αντιστοιχίζεται η σχέση που εκφράζει ο πίνακας και είναι η Health_Care_Visit της οντολογίας NCI Thesaurus.
  - Επίσης περιλαμβάνει μια προϋπόθεση (d2rq:condition) η οποία έχει την τιμή "elements_log.actionTaken <> 'DELETE'" που σημαίνει ότι δεν θα υπάρξει μετατροπή της εγγραφής σε RDF αν η στήλη actionTaken έχει την τιμή «DELETE».

  Από τα προηγούμενα γίνεται αντιληπτό ότι κάθε εγγραφή του πίνακα elements_log θα θεωρείται ως ένας ξεχωριστός RDF κόμβος, ακριβώς όπως ένα αρχείο DICOM ή HL7 των άλλων βάσεων που και αυτά αντιστοιχούν σε εγγραφές. Το προαναφερθέν URI θα είναι το βασικό αναγνωριστικό κάθε κόμβου και βάσει αυτού θα κατασκευάζονται τα URI όσων πληροφοριών απαιτούν την Δ.Α.Τ.Ι.Α. για να εκφραστούν σε RDF.
- Μια σειρά από στοιχεία d2rq:PropertyBridge, καθένα από τα οποία αντιστοιχίζει σε κάθε στήλη του πίνακα μια ιδιότητα οντολογίας. Όλα τα στοιχεία περιλαμβάνουν την ιδιότητα d2rq:belongsToClassMap που αναφέρεται στο προηγούμενο στοιχείο ClassMap στο οποίο και ανήκουν. Για τις υπόλοιπες ιδιότητες d2rq, ανάλογα με το είδος της ιδιότητας της οντολογίας υπάρχουν δύο διαφορετικά σύνολα:
  - Για τις ιδιότητες τύπου δεδομένων περιλαμβάνονται:

- d2rq:property, που περιέχει το QName της ιδιότητας.
- d2rq:column, που περιέχει το όνομα της στήλης του πίνακα η οποία δίνει τιμές στην ιδιότητα.
- d2rq:datatype, που είναι ο τύπος δεδομένων που χρησιμοποιείται για την μετατροπή της τιμής του πίνακα σε τυποποιημένο λεκτικό.

Με αυτόν τον τρόπο αντιστοιχίζονται τα στοιχεία Όνομα, Επώνυμο, Πατρώνυμο, Α.Μ.Κ.Α., Διεύθυνση, Τηλέφωνο στις ίδιες ιδιότητες οντολογιών όπως και στις περιπτώσεις των αρχείων DICOM και HL7

- Για ιδιότητες τύπου δεδομένων περιλαμβάνονται:
  - d2rq:property, που περιέχει το QName της ιδιότητας.
  - d2rq:refersToClassMap, η οποία έχει ως τιμή ένα άλλο αντικείμενο. d2rq:ClassMap, το οποίο είναι το αντικείμενο της ιδιότητας τύπου δεδομένων.

  Τα στοιχεία Ημερομηνία Γέννησης, Ασφαλιστικός Οργανισμός, Ημερομηνία Εξέτασης, Παρατηρήσεις και Συνολικό Ποσό ακολουθούν αυτό το μοτίβο και αντιστοιχίζονται στην ιδιότητα has_nodes της οντολογίας MGED έχοντας το καθένα το δικό του d2rq:ClassMap σαν αντικείμενο. Εξαίρεση αποτελεί το στοιχείο Διάγνωση το οποίο αντιστοιχίζεται στην ιδιότητα has_disease_state της οντολογίας MGED (όπως και στις περιπτώσεις των DICOM και HL7) αλλά αντί για ξεχωριστό ClassMap περιλαμβάνει:
  - Μια ιδιότητα d2rq:join που συνδέει τη στήλη diseases του πίνακα elements_log με τη στήλη diseaseID του πίνακα diseases υλοποιώντας έτσι τη σχέση ξένου κλειδιού που υπάρχει μεταξύ των δύο αυτών πινάκων και αναφέρεται στην ενότητα 5.2.1.
  - Μια ιδιότητα d2rq:uriColumn που δηλώνει ότι το στοιχείο μπορεί να λάβει σαν αντικείμενο τις τιμές της στήλης diseaseURI του πίνακα diseases (αυτό συμβαίνει λόγω του προηγούμενου d2rq:join).

- Μετά από την πρώτη σειρά από στοιχεία d2rq:PropertyBridge ακολουθούν 5 d2rq:ClassMaps τα οποία σχετίζονται με τις προαναφερθείσες ιδιότητες αντικειμένου. Κάθε d2rq:ClassMap περιλαμβάνει:
  - Την ίδια προϋπόθεση d2rq:condition με το αρχικό ClassMap (δεν γίνεται η αντιστοίχιση αν το πεδίο actionTaken στη βάση έχει τιμή «DELETE»).
  - Την ίδια τιμή της ιδιότητας d2rq:dataStorage που είναι τα αρχικό στοιχείο d2rq:Database.
  - Μια ιδιότητα d2rq:uriPattern με τιμή το URI του κόμβου, το οποίο σχηματίζεται από το βασικό URI της εγγραφής με την προσθήκη ενός επιπλέον αναγνωριστικού. Τα αναγνωριστικά που χρησιμοποιούνται για τις 5 ιδιότητες αντικειμένου είναι τα ίδια που χρησιμοποιούνται και στις περιπτώσεις αρχείων DICOM και HL7.
- Μετά από κάθε d2rq:ClassMap ακολουθούν δύο d2rq:PropertyBridge. Το πρώτο περιέχει:
  - Την ιδιότητα d2rq:property με τιμή την ιδιότητα rdf:type.

- Την ιδιότητα d2rq:uriPattern με τιμή το URI της κλάσης – αντικειμένου της rdf:type. Και τα 5 στοιχεία που απεικονίζονται στις ιδιότητες αυτές έχουν τα ίδια URI κλάσεων της οντολογίας NCI Thesaurus με τις περιπτώσεις των αρχείων DICOM και HL7.

Το δεύτερο d2rq:PropertyBridge περιέχει:

- Την ιδιότητα d2rq:property με τιμή την ιδιότητα has_value της οντολογίας MGED.
- Την ιδιότητα d2rq:column με τιμή τη στήλη του πίνακα από όπου λαμβάνεται η τιμή της ιδιότητας.
- Την ιδιότητα d2rq:datatype με τιμή τον τύπο δεδομένων που χρησιμοποιείται για την μετατροπή της τιμής του πίνακα σε τυποποιημένο λεκτικό.

Και τα δύο d2rq:PropertyBridge περιέχουν την ιδιότητα d2rq:belongsToClassMap με τιμή το d2rq:ClassMap στο οποίο ανήκουν.

5.3.2.2 Το αρχείο SQL_RULES.rules

Πρόκειται για το πρώτο από τα τρία αρχεία κανόνων τα οποία δίνονται ως όρισμα στη μέθοδο NewInferenceModel της κλάσης Reasoning.java όταν γίνεται ο συμπερασμός στο ενημερωμένο παραμένον μοντέλο δεδομένων του παθολόγου. Για την κατασκευή του ακολουθούνται τα βήματα της ενότητας 5.1.1.5. Δίνεται αρχικά το πρόθεμα pre που αντιστοιχεί στο URI http://jena.hpl.hp.com/prefix# και στη συνέχεια τα προθέματα των τριών οντολογιών που χρησιμοποιούνται στο σύστημα. Δηλώνεται ότι στους κανόνες θα συμπεριληφθεί και η μηχανή συλλογιστικής RDFS που διαθέτει η Jena και τέλος δίνεται ο μοναδικός κανόνας για τα δεδομένα του πρώτου ιατρού ο οποίος διατυπώνεται στη μορφή που χρησιμοποιείται στην ενότητα 3.2.5:

Κανόνας ταύτισης πόρων: AN:

1) Ένας πόρος A έχει το ίδιο A.M.K.A. με έναν πόρο B.
2) Ο πόρος A και ο πόρος B έχουν από ένα κόμβο με τύπο ncit: Physical_Examination_Date και οι κόμβοι αυτοί έχουν την ίδια τιμή.

ΤΟΤΕ:

Οι πόροι A και B είναι ο ίδιος πόρος.

Ο παραπάνω κανόνας διατυπώθηκε για την περίπτωση όπου στην βάση doctor1 αλλάξουν τα πρωτεύοντα κλειδιά (π.χ. αν εισαχθεί ένα πεδίο Κωδικός Εξέτασης), ώστε να διατηρείται η ισχύς της παραδοχής 5 (ενότητα 4.3).

5.3.2.3 Το αρχείο DICOM_RULES.rules

Είναι το δεύτερο αρχείο κανόνων και ασχολείται με τα δεδομένα του μοντέλου που προκύπτει από τη βάση του ακτινολόγου. Περιλαμβάνει όλα τα στοιχεία του προηγούμενου αρχείου μαζί με τρεις νέους κανόνες οι οποίοι σχετίζονται με τις αριθμητικές μετρήσεις που μπορεί να συμπεριλάβει ο ιατρός στο αρχείο DICOM ενός ασθενή:

- Κανόνας οστεοπόρωσης: AN:
  1) Ένας πόρος A έχει μια μέτρηση.
  2) Η μέτρηση έχει τύπο ncit:Bone_Mineral_Density_Test.
  3) Η μέτρηση έχει τιμή μικρότερη του -2.5.

  TOTE:

  Στον πόρο A προστίθεται η τριάδα

  <A, mged:has_disease_state, ncit:Osteoporosis> που υποδηλώνει ότι ο ασθενής που αναφέρεται στο συγκεκριμένο πόρο έχει οστεοπόρωση.
- Κανόνας πιθανής οστεοπόρωσης: AN:
  1) Ένας πόρος A έχει μια μέτρηση.
  2) Η μέτρηση έχει τύπο ncit: Bone_Mineral_Density_Test.
  3) Η μέτρηση έχει τιμή μεταξύ -1 και -2.5.

  TOTE:

  Στον πόρο A προστίθεται η τριάδα

  <A, rdfs:label, "possible Osteoporosis"^^xsd:string> που υποδηλώνει ότι ο ασθενής που αναφέρεται στο συγκεκριμένο πόρο είναι πιθανό να εμφανίσει οστεοπόρωση.
- Κανόνας παχυσαρκίας1: AN:
  1) Ένας πόρος A έχει μια μέτρηση.
  2) Η μέτρηση έχει τύπο ncit:Adipose_Tissue.
  3) Η μέτρηση έχει τιμή μεγαλύτερη του 32.

  TOTE:

  Στον πόρο A προστίθεται η τριάδα

  <A, mged:has_disease_state, ncit:Obesity> που υποδηλώνει ότι ο ασθενής που αναφέρεται στο συγκεκριμένο πόρο είναι παχύσαρκος.
- Κανόνας παχυσαρκίας2: AN:
  1) Ένας πόρος A έχει μια μέτρηση.
  2) Η μέτρηση έχει τύπο ncit:Adipose_Tissue.
  3) Η μέτρηση έχει τιμή ανάμεσα στο 25 και το 32.

  TOTE:

  Στον πόρο A προστίθεται η τριάδα

  <A, rdfs:label, "possible Obesity"^^xsd:string> που υποδηλώνει ότι ο ασθενής που αναφέρεται στο συγκεκριμένο πόρο είναι είτε παχύσαρκος άνδρας είτε γυναίκα στα όρια της παχυσαρκίας.

Οι συγκεκριμένοι κανόνες διατυπώθηκαν λαμβάνοντας υπόψιν το [103] που αναφέρει τις ενδεικτικές φυσιολογικές τιμές ποσοστού λίπους και οστικής πυκνότητας.

5.3.2.4 Το αρχείο HL7_RULES.rules

Είναι το τρίτο και τελευταίο αρχείο κανόνων της εφαρμογής και εφαρμόζεται επί του RDF μοντέλου που προκύπτει από τα δεδομένα του μικροβιολόγου. Περιλαμβάνει όλα τα στοιχεία του αρχείου SQL_RULES.rules, έναν επιπλέον κανόνα ταύτισης πόρων και ένα σύνολο κανόνων αναφορικά με τις πιθανές καταστάσεις του

ασθενή ανάλογα με το φύλο του και τις αριθμητικές τιμές δεδομένων που μπορεί να προκύψουν. Οι συγκεκριμένοι κανόνες διατυπώθηκαν λαμβάνοντας υπόψιν τα [124, 125, 126] που αναφέρουν τις ενδεικτικές φυσιολογικές τιμές στοιχείων και ουσιών στο αίμα.

- 2ος κανόνας ταύτισης πόρων: AN:
  1) Ένας πόρος Α έχει τον ίδιο προσωπικό κωδικό με έναν πόρο Β.
  2) Ο πόρος Α και ο πόρος Β έχουν από ένα κόμβο με τύπο ncit: Physical_Examination_Date και οι κόμβοι αυτοί έχουν την ίδια τιμή.

  TOTE:

  Οι πόροι Α και Β είναι ο ίδιος πόρος.
- 1ος κανόνας αναιμίας (άνδρες/γυναίκες): AN:
  1) Ένας πόρος Α αναφέρεται σε άνδρα/γυναίκα.
  2) Ο πόρος Α έχει μια μέτρηση.
  3) Η μέτρηση έχει τύπο ncit:Hematocrit.
  4) Η μέτρηση έχει τιμή κάτω από 40/37.

  TOTE:

  Στον πόρο Α προστίθεται η τριάδα <A, mged:has_disease_state, ncit:Anemia> που υποδηλώνει ότι ο/η ασθενής που αναφέρεται στο συγκεκριμένο πόρο έχει αναιμία.
- 2ος κανόνας αναιμίας (άνδρες/γυναίκες): AN:
  1) Ένας πόρος Α αναφέρεται σε άνδρα/γυναίκα.
  2) Ο πόρος Α έχει μια μέτρηση.
  3) Η μέτρηση έχει τύπο ncit:Hemoglobin_Measurement.
  4) Η μέτρηση έχει τιμή κάτω από 13/11.6.

  TOTE:

  Στον πόρο Α προστίθεται η τριάδα <A, mged:has_disease_state, ncit:Anemia> που υποδηλώνει ότι ο/η ασθενής που αναφέρεται στο συγκεκριμένο πόρο έχει αναιμία.
- Κανόνας υπερουριχαιμίας (υπερβολικό ουρικό οξύ στο αίμα) (ανδρες/γυναίκες): AN:
  1) Ένας πόρος Α αναφέρεται σε άνδρα/γυναίκα.
  2) Ο πόρος Α έχει μια μέτρηση.
  3) Η μέτρηση έχει τύπο ncit:Hemoglobin Uric_Acid_Crystal_Measurement.
  4) Η μέτρηση έχει τιμή πάνω από 7.5/6.2.

  TOTE:

  Στον πόρο Α προστίθεται η τριάδα <A, mged:has_disease_state, ncit:Hyperuricemia> που υποδηλώνει ότι ο/η ασθενής που αναφέρεται στο συγκεκριμένο πόρο έχει υπερουριχαιμία.
- Για τις υπόλοιπες μετρήσεις που δεν απαιτούν προσδιορισμό φύλου του ασθενή χρησιμοποιείται ο ακόλουθος γενικός κανόνας:

  Κανόνας ασθένειας Χ: AN:
  1) Ένας πόρος Α έχει μια μέτρηση.
  2) Η μέτρηση έχει τύπο ncit:Y.

3) Η μέτρηση έχει τιμή πάνω/κάτω από μια τιμή Z.
ΤΟΤΕ:
Στον πόρο Α προστίθεται η τριάδα
<A, mged:has_disease_state, W> που υποδηλώνει ότι ο ασθενής που αναφέρεται στο συγκεκριμένο πόρο έχει την ασθένεια Χ.
Οι διάφορες περιπτώσεις παρουσιάζονται στον παρακάτω πίνακα:

Ασθένεια Χ	Κλάση Υ της οντολογίας NCI Thesaurus (μετρήσιμο μέγεθος)	Τιμή Z	Κλάση W της οντολογίας NCI Thesaurus (ασθένεια)
Θρομβοκυτοπενία	Platelet_Count	<150	Thrombocytopenia
Λοίμωξη	Leukocyte_Count	>11	Infectious_Disorder
Υψηλή «κακή» χοληστερίνη	Serum_LDL_ Cholesterol_Measurement	>130	Type_II_Hyperlipidemia
Υπερτριγλυ-κεριδαιμία	Triglyceride_ Measurement	>150	Hypertriglyceridemia
Νεφροπάθεια	Creatinine_ Measurement	>1,5	Nephropathy
Υπερλιπιδαιμία	Lipid_Measurement	>1000	Hyperlipidemia
Υπεργλυκαιμία	Fasting_Blood_ Sugar_Measurement	>110	Hyperglycemia
Υπογλυκαιμία	Fasting_Blood_ Sugar_Measurement	<60	Hypoglycemia

Πίνακας 5.2: Στοιχεία του γενικού κανόνα εντοπισμού ασθένειας.

5.3.2.5 Το αρχείο Doctor1-2-3.ttl

Το συγκεκριμένο αρχείο είναι ένα αρχείο ρυθμίσεων για το εργαλείο FedX. Χρησιμοποιείται από την κλάση UserFrame.java όταν ο χρήστης επιλέγει να απαντηθεί το ερώτημά του από το συγκεκριμένο εργαλείο. Περιέχει προτάσεις που δηλώνουν ότι τα τρία τελικά σημεία που χρησιμοποιεί η εφαρμογή είναι τελικά σημεία SPARQL καθώς και τα URI τους, τα οποία είναι αυτά που αναφέρονται στην ενότητα 5.2.3

5.3.2.6 Τα τρία αρχεία ρυθμίσεων του Joseki RDF server

Πρόκειται για τα τρία πανομοιότυπα ως προς τη βασική δομή τους αρχεία ρυθμίσεων SQL.ttl, DICOM.ttl, HL7.ttl, τα οποία χρησιμοποιούνται στις κλήσεις του Joseki για τη δημιουργία των τριών τελικών σημείων SPARQL με τις εντολές που δίνονται στην ενότητα 5.2.3. Τα αρχεία είναι διαρθρωμένα σύμφωνα με τις υποδείξεις της ενότητας 5.1.3.1 ως εξής:

- Προθέματα: Δηλώνονται τα βασικά προθέματα για ένα αρχείο ρυθμίσεων, όπως αυτά υποδεικνύονται στο [54] με μόνες αλλαγές τη χρήση του προθέματος xsd του χώρου ονομάτων που περιλαμβάνει τους XML τύπους δεδομένων και την παράλειψη του προθέματος rdf και του ομώνυμου χώρου,

καθώς δεν χρησιμοποιείται πουθενά. Μετά από αυτά περιλαμβάνεται ένα αναγνωριστικό σχόλιο.

- Ορισμός διακομιστή: Ορίζεται ο βασικός διακομιστής joseki καλώντας την κλάση java:org.joseki.util.ServiceInitSimple.
- Βασική υπηρεσία: Καθορίζεται μια βασική υπηρεσία τελικού σημείου SPARQL σαν ένας RDF κόμβος με ιδιότητες αντικειμένου το σύνολο δεδομένων (ονόματι doctor1, doctor2, και doctor3 για παθολόγο, ακτινολόγο και μικροβιολόγο αντίστοιχα) πάνω στο οποίο θα ενεργήσει και ο επεξεργαστής που θα την εκτελέσει.
- Επεξεργαστής: Είναι ο κόμβος – αντικείμενο της αντίστοιχης ιδιότητας της προαναφερθείσας υπηρεσίας. Δηλώνεται ένας επεξεργαστής σταθερού συνόλου δεδομένων (δεν δέχεται ερωτήματα με προτάσεις FROM/FROM NAMED). Περιέχει μια ιδιότητα αντικειμένου με τιμή τον κόμβο που περιέχει την κλάση java υλοποίησής του και δίνεται στο έγγραφο μετά από τον επεξεργαστή και πριν τον ορισμό του συνόλου δεδομένων.
- Ορισμός συνόλου δεδομένων: Εδώ υπάρχει ένας RDF κόμβος με το όνομα του συνόλου δεδομένων που περιγράφεται στην αντίστοιχη ιδιότητα αντικειμένου στον κόμβο της υπηρεσίας (doctorx, x = 1,2,3). Περιέχει μόνο μια ιδιότητα αντικειμένου που έχει σαν τιμή τον κόμβο που περιέχει την περιγραφή του RDF γράφου από όπου θα αντλούνται τα δεδομένα. Το όνομα του κόμβου είναι sqlDBModel, dicomDBModel και hl7DBModel για τα αντίστοιχα αρχεία SQL.ttl, DICOM.ttl, HL7.ttl.
- Περιγραφή παραμένοντος μοντέλου: Είναι ο κόμβος xDBModel (x = sql, dicom, hl7) που αναφέρεται στις ιδιότητες του συνόλου δεδομένων. Η δομή του είναι ίδια με αυτή της τελευταίας δήλωσης του παραδείγματος 5.2. Τα μόνα που αλλάζουν σε κάθε αρχείο είναι η τιμή των ιδιοτήτων ja:dbURL, που είναι το URL του εκάστοτε triple store που φιλοξενεί το παραμένον μοντέλο και ja:modelName που είναι το όνομα του μοντέλου. Η τιμή της πρώτης είναι το URL jdbc:mysql://localhost/x_rdf_model, και της δεύτερης το X_Model, όπου x = sql, dicom, hl7 και X = SQL, DICOM, HL7 για τα αντίστοιχα αρχεία SQL.ttl, DICOM.ttl, HL7.ttl.

ΚΕΦΑΛΑΙΟ 6:
ΣΕΝΑΡΙΑ ΧΡΗΣΗΣ

Στο κεφάλαιο αυτό παρουσιάζεται μια σειρά ομόσπονδων ερωτημάτων που υποβλήθηκαν στην IntegraHEALTH 1.0. Για κάθε ερώτημα δίνονται δύο τρόποι εκτέλεσης. Ο πρώτος περιέχει προτάσεις SERVICE, που καθορίζουν ρητά τα υποδείγματα βασικών γράφων που θα ταυτοποιηθούν σε κάθε τελικό σημείο SPARQL, ενώ ο δεύτερος χρησιμοποιεί ένα απλό SPARQL ερώτημα (χωρίς χρήση της λέξης SERVICE), το οποίο, μέσω του εργαλείου FedX, διασπάται σε επιμέρους SPARQL ερωτήματα που διοχετεύονται στα τελικά σημεία SPARQL. Όλα τα ερωτήματα θεωρείται ότι έχουν στην αρχή τους τα παρακάτω προθέματα:

PREFIX ncit: <http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#>
PREFIX mged: <http://mged.sourceforge.net/ontologies/MGEDOntology.owl#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

Τα ερωτήματα γίνονται διαδοχικά το ένα μετά το άλλο στην ίδια περίοδο λειτουργίας της εφαρμογής.

1. Ζητούνται από το σύστημα όλα τα ονοματεπώνυμα και τα τηλέφωνα των ασθενών (απλό ερώτημα).
   Ερώτημα προς SPARQL:

```
SELECT DISTINCT ?name ?surname ?phone{
   {SERVICE <http://localhost:2030/sparql>{
      ?x mged:has_first_name ?name.
      ?x mged:has_last_name ?surname.
      ?x mged:has_phone ?phone.
   }}UNION
   {SERVICE <http://localhost:2040/sparql>{
      ?x mged:has_first_name ?name.
      ?x mged:has_last_name ?surname.
      ?x mged:has_phone ?phone.
   }}UNION
   {SERVICE <http://localhost:2050/sparql>{
      ?x mged:has_first_name ?name.
      ?x mged:has_last_name ?surname.
      ?x mged:has_phone ?phone.
}}}
ORDER BY ?surname
```

Αποτέλεσμα:

name	surname	phone
"EVLAMPIA"^^ *1	"ATHANASIADI"^^ *1	"2105833648"^^ *1
"EYLAMPIA"^^ *1	"ATHANASIADI"^^ *1	"2105833648"^^ *1
"EYLAMPIA"^^ *1	"ATHANASIADI"^^ *1	"2410819032"^^ *1
"GEORGIOS"^^ *1	"CHATZIARAS"^^ *1	"2410333333"^^ *1
"GEORGIOS"^^ *1	"CHATZIARAS"^^ *1	"6977691299"^^ *1
"NIKOLAOS"^^ *1	"CHRISTODOULOU"^^ *1	"2444031210"^^ *1
"NIKOLAOS"^^ *1	"CHRISTODOULOU"^^ *1	"6976719929"^^ *1
"ZAXARIAS"^^ *1	"CHRISTOPOULOS"^^ *1	"2108765312"^^ *1
"ATHANASIOS"^^ *1	"IOANNOY"^^ *1	"2117819083"^^ *1
"EVAGGELOS"^^ *1	"KOSTOPOULOS"^^ *1	"2410555777"^^ *1
"EVAGGELOS"^^ *1	"KOSTOPOYLOS"^^ *1	"2410555777"^^ *1
"ARISTEIDHS"^^ *1	"MAKRHS"^^ *1	"2410658143"^^ *1
"KORALLIA"^^ *1	"MEGAPANOY"^^ *1	"2105698667"^^ *1
"STAVROS"^^ *1	"MEINTANIS"^^ *1	"6978733652"^^ *1
"STEFANOS"^^ *1	"PALAIOLOGOS"^^ *1	"2411237950"^^ *1
"THEMISTOKLIS"^^ *1	"PAPAPAYLOU"^^ *1	"6949833562"^^ *1
"PETROS"^^ *1	"PETAS"^^ *1	"2111278133"^^ *1
"PANAGIOTIS"^^ *1	"POYLOPOYLOS"^^ *1	"2103496755"^^ *1
"STAMATIS"^^ *1	"STAVROY"^^ *1	"6976820930"^^ *1
"HLIAS"^^ *1	"TSAKALOS"^^ *1	"2410608444"^^ *1
"HLIAS"^^ *1	"TSAKALOS"^^ *1	"6998720388"^^ *1

όπου *1= ^^<http://www.w3.org/2001/XMLSchema#string> (η αντικατάσταση γίνεται για λόγους συντομίας).
Το ερώτημα απαντήθηκε σε 48 ms.

Ερώτημα προς FedX:

```
SELECT DISTINCT ?name ?surname ?phone{
   ?x mged:has_first_name ?name.
   ?x mged:has_last_name ?surname.
   ?x mged:has_phone ?phone.
}
ORDER BY ?surname
```

Αποτέλεσμα:

```
[surname="ATHANASIADI"^^ *1;phone="2410819032"^^ *1;name="EYLAMPIA"^^ *1]
[surname="ATHANASIADI"^^ *1;phone="2105833648"^^ *1;name="EVLAMPIA"^^ *1]
[surname="ATHANASIADI"^^ *1;phone="2105833648"^^ *1;name="EYLAMPIA"^^ *1]
[surname="CHATZIARAS"^^ *1;phone="2410333333"^^ *1;name="GEORGIOS"^^ *1]
[surname="CHATZIARAS"^^ *1;phone="6977691299"^^ *1;name="GEORGIOS"^^ *1]
[surname="CHRISTODOULOU"^^ *1;phone="2444031210"^^ *1;name="NIKOLAOS"^^ *1]
```

```
[surname="CHRISTODOULOU"^^ *1;phone="6976719929"^^ *1;name="NIKOLAOS"^^ *1]
[surname="CHRISTOPOULOS"^^ *1;phone="2108765312"^^ *1;name="ZAXARIAS"^^ *1]
[surname="IOANNOY"^^ *1;phone="2117819083"^^ *1;name="ATHANASIOS"^^ *1]
[surname="KOSTOPOULOS"^^ *1;phone="2410555777"^^ *1;name="EVAGGELOS"^^ *1]
[surname="KOSTOPOYLOS"^^ *1;phone="2410555777"^^ *1;name="EVAGGELOS"^^ *1]
[surname="MAKRHS"^^ *1;phone="2410658143"^^ *1;name="ARISTEIDHS"^^ *1]
[surname="MEGAPANOY"^^ *1;phone="2105698667"^^ *1;name="KORALLIA"^^ *1]
[surname="MEINTANIS"^^ *1;phone="6978733652"^^ *1;name="STAVROS"^^ *1]
[surname="PALAIOLOGOS"^^ *1;phone="2411237950"^^ *1;name="STEFANOS"^^ *1]
[surname="PAPAPAYLOU"^^ *1;phone="6949833562"^^ *1;name="THEMISTOKLIS"^^ *1]
[surname="PETAS"^^ *1;phone="2111278133"^^ *1;name="PETROS"^^ *1]
[surname="POYLOPOYLOS"^^ *1;phone="2103496755"^^ *1;name="PANAGIOTIS"^^ *1]
[surname="STAVROY"^^ *1;phone="6976820930"^^ *1;name="STAMATIS"^^ *1]
[surname="TSAKALOS"^^ *1;phone="2410608444"^^ *1;name="HLIAS"^^ *1]
[surname="TSAKALOS"^^ *1;phone="6998720388"^^ *1;name="HLIAS"^^ *1]
```

Το ερώτημα απαντήθηκε σε 340 ms.

Το παραπάνω ερώτημα επιλέχθηκε σκόπιμα για να δείξει την ευκολία που παρέχει η μηχανή FedX στην απάντηση απλών ερωτημάτων που αφορούν όλα τα διαθέσιμα τελικά σημεία του συστήματος. Στην περίπτωση που στο ερώτημα χρησιμοποιούνται προτάσεις SERVICE απαιτείται μια πιο σύνθετη διατύπωσή του με χρήση του τελεστή UNION διότι τα URIs των RDF κόμβων των εξετάσεων είναι διαφορετικά για κάθε ιατρό, επομένως η παράλειψή τους θα επέστρεφε στοιχεία μόνο από τον «πρώτο» ιατρό σύμφωνα με τη σειρά διατύπωσης του ερωτήματος, που σε αυτήν την περίπτωση είναι ο παθολόγος. Στην περίπτωση της μηχανής FedX, αρκεί η υποβολή ενός απλού ερωτήματος SPARQL, αφού η ίδια η μηχανή αναλαμβάνει να υποβάλλει το ερώτημα στο σύνολο των τελικών σημείων.

2. Ζητούνται από το σύστημα οι τιμές ροής του αίματος όσων ασθενών πάσχουν από θρομβοκυτοπενία (ασθένεια όπου τα αιμοπετάλια στο αίμα είναι χαμηλά).

Ερώτημα προς SPARQL:

```
SELECT ?bloodflow{
   SERVICE <http://localhost:2050/sparql>{
     ?x mged:unique_identifier ?id.
     ?x mged:has_disease_state ncit:Thrombocytopenia.
   }
   SERVICE <http://localhost:2040/sparql>{
     ?y mged:unique_identifier ?id.
     ?y mged:has_measurement ?mr.
     ?mr a ncit:Blood_Flow_Rate.
     ?mr mged:has_value ?bloodflow.
}}
ORDER BY ?bloodflow
```

Αποτέλεσμα:

```
-----------------------------------------------------------------------
| bloodflow                                                           |
=======================================================================
| "90.0"^^<http://www.w3.org/2001/XMLSchema#float>                    |
| "92.0"^^<http://www.w3.org/2001/XMLSchema#float>                    |
-----------------------------------------------------------------------
```

Το ερώτημα απαντήθηκε σε 16ms.

Ερώτημα προς FedX:

```
SELECT ?bloodflow{
   ?x mged:unique_identifier ?id.
   ?x mged:has_disease_state ncit:Thrombocytopenia.

   ?y mged:unique_identifier ?id.
   ?y mged:has_measurement ?mr.
   ?mr a ncit:Blood_Flow_Rate.
   ?mr mged:has_value ?bloodflow.
}
ORDER BY ?bloodflow
```

Αποτέλεσμα:

```
[bloodflow="90.0"^^<http://www.w3.org/2001/XMLSchema#float>]
[bloodflow="92.0"^^<http://www.w3.org/2001/XMLSchema#float>]
```

Το ερώτημα απαντήθηκε σε 48ms.

Στο ερώτημα αυτό γίνεται ανάκτηση πληροφοριών που έχουν καταχωρηθεί από τους ιατρούς (και στη συγκεκριμένη περίπτωση από τον ακτινολόγο) με βάση προτάσεις που παράγονται από την δράση των μηχανών συλλογιστικής πάνω στα δεδομένα των ιατρών. Η πληροφορία του ότι ένας ασθενής πάσχει από θρομβοκυτοπενία, παράγεται από τους κανόνες που τέθηκαν στο αρχείο HL7_RULES.rules.

Μια άλλη παρατήρηση είναι το γεγονός ότι επειδή το ερώτημα έχει κάποιες σταθερές τιμές βάσει των οποίων αναζητούνται πληροφορίες στους RDF γράφους (URI κλάσεων σε μορφή QName σε θέση αντικειμένων προτάσεων), ενώ το προηγούμενο ερώτημα δεν έχει, η μόνη συντακτική διαφορά μεταξύ των δύο παραλλαγών του ερωτήματος είναι η χρήση ή μη των προτάσεων SERVICE.

Τέλος, αναφέρεται ότι για κάθε επιμέρους στοιχείο του ερωτήματος (μετρήσεις ροής αίματος, ασθενείς με θρομβοκυτοπενία) χρησιμοποιείται ξεχωριστό σύνολο προτάσεων που εκτελεί την αναζήτηση, είτε αυτό περιέχεται

σε μπλοκ πρότασης SERVICE είτε όχι. Τα σύνολα διαχωρίζονται με χρήση διαφορετικής μεταβλητής υποκειμένου (?x, ?y, κ.ο.κ.). Συνδετικός κρίκος μεταξύ αυτών των αναζητήσεων είναι οι Α.Μ.Κ.Α. των ασθενών οι οποίοι αποθηκεύονται σε μια μεταβλητή κοινή σε όλες τις ομάδες προτάσεων που εκτελούν τις αναζητήσεις. Η ίδια μέθοδος χρησιμοποιείται και στα επόμενα ερωτήματα.

3. Ζητούνται από το σύστημα τα ονοματεπώνυμα και το συνολικό χρέος (προς όλους τους ιατρούς) όσων δεν έχουν εμφανή συμπτώματα ασθένειας, διαμένουν στην Αθήνα και έχουν συνολικό χρέος άνω των 150 ευρώ.

   Ερώτημα προς SPARQL:

```
SELECT DISTINCT ?name ?surname ((?money1+?money2) AS ?debt1)
 ((?money1+?money3) AS ?debt2) ((?money1+?money2+?money3) AS ?debt3) {
   SERVICE <http://localhost:2030/sparql>{
      ?x mged:has_first_name ?name.
      ?x mged:has_last_name ?surname.
      ?x mged:has_disease_state ncit:Asymptomatic.
      ?x mged:unique_identifier ?id.
      ?x mged:has_address ?add1.
      ?x mged:has_nodes ?mo1.
      ?mo1 rdf:type ncit:Payment.
      ?mo1 mged:has_value ?money1.
   }
   OPTIONAL{
      SERVICE <http://localhost:2040/sparql>{
         ?y mged:unique_identifier ?id.
         ?y mged:has_address ?add2.
         ?y mged:has_nodes ?mo2.
         ?mo2 rdf:type ncit:Payment.
         ?mo2 mged:has_value ?money2.
      }
   }
   OPTIONAL{
      SERVICE <http://localhost:2050/sparql>{
         ?z mged:unique_identifier ?id.
         ?z mged:has_address ?add3.
         ?z mged:has_nodes ?mo3.
         ?mo3 rdf:type ncit:Payment.
         ?mo3 mged:has_value ?money3.
      }
   }
   FILTER (regex(str(?add1), "ATHINA")||regex(str(?add2), "ATHINA")||
      regex(str(?add3), "ATHINA"))
   FILTER ((?money1+?money3)>150||(?money1+?money2)>150||
```

```
      (?money1+?money2+?money3)>150)
}
ORDER BY DESC (?debt3) (?debt1) (?debt2)
```

Αποτέλεσμα:

```
-------------------------------------------------------------------------------------------------------
| name                    | surname                          | debt1   | debt2            | debt3 |
=======================================================================================================
| "NIKOLAOS"^^ *1 | "CHRISTODOULOU"^^ *1   |          | "155.0"^^ *2  |         |
| "ZAXARIAS"^^ *1 | "CHRISTOPOULOS"^^ *1   |          | "151.0"^^ *2  |         |
-------------------------------------------------------------------------------------------------------
```

όπου *2= ^^<http://www.w3.org/2001/XMLSchema#float>.
Το ερώτημα απαντήθηκε σε 32 ms.

Ερώτημα προς FedX:

```
SELECT DISTINCT ?name ?surname ((?money1+?money2) AS ?debt){
   ?x mged:has_first_name ?name.
   ?x mged:has_last_name ?surname.
   ?x mged:unique_identifier ?id.
   ?x mged:has_disease_state ncit:Asymptomatic.

   ?y mged:unique_identifier ?id.
   ?y mged:has_address ?add.
   FILTER regex(str(?add), "ATHINA").

   ?z mged:unique_identifier ?id.
   ?z mged:has_nodes ?mo1.
   ?mo1 rdf:type ncit:Payment.
   ?mo1 mged:has_value ?money1.
   FILTER EXISTS {?z mged:has_disease_state ncit:Asymptomatic.}

   ?w mged:unique_identifier ?id.
   ?w mged:has_nodes ?mo2.
   ?mo2 rdf:type ncit:Payment.
   ?mo2 mged:has_value ?money2.
   FILTER NOT EXISTS {?w mged:has_disease_state ncit:Asymptomatic.}

   FILTER((?money1+?money2)>150)
}
ORDER BY DESC (?debt)
```

Αποτέλεσμα:

```
[debt="155.0^^ *2; surname="CHRISTODOULOU"^^ *1; name="NIKOLAOS"^^ *1]
[debt="151.0^^ *2; surname="CHRISTOPOULOS"^^ *1;name="ZAXARIAS"^^ *1]
```

όπου *1, *2 όπως στα προηγούμενα.
Το ερώτημα απαντήθηκε σε 78ms.

Στο συγκεκριμένο ερώτημα δεν υπάρχει αναζήτηση σε δεδομένα προερχόμενα από διαδικασίες συλλογιστικής. Σκοπός είναι να γίνει επίδειξη της δυνατότητας που έχει η SPARQL να ορίζει νέες μεταβλητές επιστροφής στο χρήστη (?debt, ?debt1, κ.ο.κ.) μέσω μαθηματικών παραστάσεων που χρησιμοποιούν μεταβλητές που περιέχονται στα υποδείγματα γράφων του ερωτήματος.

Εδώ επιχειρείται για κάθε ασθενή που πληροί κάποια κριτήρια, να παρουσιαστεί το συνολικό του χρέος προς όλους τους ιατρούς. Όταν γίνεται χρήση των προτάσεων SERVICE είναι αναγκαστική τόσο η χρήση προτάσεων OPTIONAL, όσο και η ανάθεση διαφορετικών μεταβλητών για κάθε πιθανό άθροισμα χρεών που μπορεί να παρουσιαστεί. Αυτό γίνεται διότι οι μεταβλητές μιας παράστασης στην πρόταση SELECT πρέπει να είναι όλες δεσμευμένες για να αποδοθεί τιμή στην μεταβλητή που παρουσιάζεται στο χρήστη και υπάρχουν ασθενείς στο σύστημα που δεν έχουν επισκεφθεί και τους τρεις ιατρούς, οπότε δεν είναι δυνατόν να υπάρχουν πληροφορίες για οφειλόμενα ποσά σε γιατρούς τους οποίους δεν επισκέφθηκαν.

Στην περίπτωση της μηχανής FedX, η μη αναφορά σε τελικά σημεία διευκολύνει την αναζήτηση βρίσκοντας απλά τους A.M.K.A. των ζητούμενων ασθενών και ανακτώντας για κάθε ασθενή τα όποια οφειλόμενα ποσά υπάρχουν για τον παθολόγο και ξεχωριστά τα αντίστοιχα ποσά για τους άλλους ιατρούς. Ο διαχωρισμός γίνεται βάσει του αν ο RDF πόρος περιλαμβάνει το URI ncit:Asymptomatic στις ιδιότητές του (οπότε ανακτάται το οφειλόμενο ποσό προς τον παθολόγο) ή όχι (ανακτώνται οι υπόλοιπες οφειλές). Στο τέλος η ανάθεση του αθροίσματος των δύο ποσών σε μόνο μία μεταβλητή στην πρόταση SELECT είναι αρκετή.

Ένα άλλο σημείο διαφοράς μεταξύ των δύο παραλλαγών του ερωτήματος βρίσκεται στο γεγονός ότι όταν χρησιμοποιούνται προτάσεις SERVICE, αναγκαστικά σε κάθε υποερώτημα γίνεται ξεχωριστά αναζήτηση για τη διεύθυνση του ασθενή (με χρήση διαφορετικών μεταβλητών) για να καλυφθούν οι περιπτώσεις ασθενών που δήλωσαν διαφορετικές διευθύνσεις σε διαφορετικούς ιατρούς. Ο στόχος είναι να βρεθούν οι ασθενείς που έχουν δηλώσει σε οποιονδήποτε ιατρό ότι διαμένουν στην Αθήνα. Και πάλι η μηχανή FedX επιτρέπει την απλοποίηση του ερωτήματος καθώς απαιτεί μία μόνο ομάδα προτάσεων για την αναζήτηση των διευθύνσεων, η οποία υποβάλλεται σε όλα τα τελικά σημεία. Η σύνδεση μέσω του A.M.K.A. εξασφαλίζει ότι επιλέγονται όλοι οι ασθενείς που διαμένουν στην Αθήνα, όπου και αν το δήλωσαν. Η ίδια λογική ακολουθείται και στις προτάσεις FILTER που ακολουθούν τα υποδείγματα γράφων των ερωτημάτων. Στο ερώτημα με τις προτάσεις SERVICE χρησιμοποιείται ένα λογικό «Η» μεταξύ προτάσεων FILTER που ελέγχουν κάθε

μεταβλητή διεύθυνσης χωριστά για το αν περιέχει τη συμβολοσειρά «ATHINA», ενώ στην περίπτωση του FedX χρησιμοποιείται μόνο μία τέτοια πρόταση.

4. Ζητούνται από το σύστημα οι τιμές σακχάρου όλων των φαλακρών παχύσαρκων ατόμων που διαμένουν στη Λάρισα.

   Ερώτημα προς SPARQL:

```
SELECT DISTINCT ?bloodSugar{
   SERVICE <http://localhost:2030/sparql>{
      ?x mged:has_disease_state ncit:Hair_Loss.
      ?x mged:unique_identifier ?id.
      ?x mged:has_address ?add1.
      ?x mged:has_phone ?ph1.
   }
   SERVICE<http://localhost:2040/sparql>{
      {?y mged:unique_identifier ?id.
       ?y mged:has_disease_state ncit:Obesity.
       ?y mged:has_address ?add2.
       ?y mged:has_phone ?ph2.}
      UNION
      {?y mged:unique_identifier ?id.
       ?y rdfs:label "possible Obesity"^^xsd:string.
       ?y mged:has_address ?add2.
       ?y mged:has_phone ?ph2.}
   }
   SERVICE <http://localhost:2050/sparql>{
      ?z mged:unique_identifier ?id.
      ?z mged:has_address ?add3.
      ?z mged:has_phone ?ph3.
      ?z mged:has_measurement ?bm.
      ?bm rdf:type ncit:Fasting_Blood_Sugar_Measurement.
      ?bm mged:has_value ?bloodSugar.
   }
   FILTER (regex(str(?add1), "LARISA")||regex(str(?add2), "LARISA")||
         regex(str(?add3), "LARISA"))
   FILTER (!regex(str(?bloodSugar), "-0.0"))
}
ORDER BY DESC (?bloodsugar)
```

Αποτέλεσμα:

```
-------------------------------------------------------------------------
| bloodSugar                                                            |
=========================================================
| "168.0"^^<http://www.w3.org/2001/XMLSchema#float> |
| "115.0"^^<http://www.w3.org/2001/XMLSchema#float> |
-------------------------------------------------------------------------
```

Το ερώτημα απαντήθηκε σε 1ms.

Ερώτημα προς FedX:

```
SELECT DISTINCT ?bloodSugar{
   ?x1 mged:has_disease_state ncit:Hair_Loss.
   ?x1 mged:unique_identifier ?id.

   {?x2 mged:unique_identifier ?id.
    ?x2 mged:has_disease_state ncit:Obesity.
    ?x2 mged:has_address ?add2.
    ?x2 mged:has_phone ?ph2.}
   UNION
   {?x2 mged:unique_identifier ?id.
    ?x2 rdfs:label "possible Obesity"^^xsd:string.
    ?x2 mged:has_address ?add2.
    ?x2 mged:has_phone ?ph2.}

   ?x3 mged:unique_identifier ?id.
   ?x4 mged:has_measurement ?bm.
   ?bm rdf:type ncit:Fasting_Blood_Sugar_Measurement.
   ?bm mged:has_value ?bloodSugar.
   FILTER (!regex(str(?bloodSugar), "-0.0"))

   ?x4 mged:unique_identifier ?id.
   ?x4 mged:has_address ?add.
   FILTER regex(str(?add), "LARISA").
}
ORDER BY DESC (?bloodsugar)
```

Αποτέλεσμα:

```
[bloodSugar="168.0"^^<http://www.w3.org/2001/XMLSchema#float>]
[bloodSugar="115.0"^^<http://www.w3.org/2001/XMLSchema#float>]
```

Το ερώτημα απαντήθηκε σε 46ms.

Στο συγκεκριμένο ερώτημα ζητούνται ασθενείς των οποίων τα στοιχεία βρίσκονται και στους τρεις ιατρούς. Συγκεκριμένα από τον παθολόγο ζητούνται όλοι οι ασθενείς που πάσχουν από αλωπεκία. Τα σχετικά στοιχεία έχουν εισαχθεί από τον ίδιο τον ιατρό. Από αυτούς τους ασθενείς επιλέγονται όσοι είναι παχύσαρκοι. Αυτό γίνεται βάσει των προτάσεων που παράγονται από τους κανόνες του αρχείου DICOM_RULES.rules που αφορά τα δεδομένα του ακτινολόγου. Επειδή υπάρχουν δύο περιπτώσεις παχυσαρκίας (κανονική και ελαφράς μορφής) και οι κανόνες δημιουργούν διαφορετικές τριάδες για το καθένα, χρησιμοποιούνται δύο ξεχωριστές ομάδες προτάσεων για την αναζήτηση που περιέχουν την ίδια μεταβλητή – υποκείμενο. Οι ομάδες συνδέονται με ένα τελεστή UNION (αυτό γίνεται και στις δύο παραλλαγές). Τέλος από τον μικροβιολόγο ζητούνται οι τιμές του σακχάρου στο αίμα των ασθενών που ικανοποιούν τα κριτήρια που σχετίζονται με τους προηγούμενους ιατρούς.

Μετά τις αναζητήσεις χρησιμοποιούνται προτάσεις FILTER για την απομόνωση των ασθενών που μένουν στη Λάρισα με λογική όμοια με αυτή του προηγούμενου ερωτήματος. Επιπλέον χρησιμοποιείται άλλη μια πρόταση για να εξαιρεθούν από τα αποτελέσματα οι τιμές του σακχάρου που είναι «-0.0» καθώς η τιμή αυτή χρησιμοποιείται μόνο σαν δείκτης μη ύπαρξης τιμής (ο ιατρός δεν εκτέλεσε τη μέτρηση).

5. Ζητείται από το σύστημα όλο το ιστορικό (ημερομηνίες) των απεικονιστικών εξετάσεων των ατόμων με ακριβώς 5 εκατομμύρια ερυθρά αιμοσφαίρια.

Ερώτημα προς SPARQL:

```
SELECT distinct  ?id ?examdate{
   SERVICE <http://localhost:2050/sparql>{
      ?x mged:unique_identifier ?id.
      ?x mged:has_measurement ?mred.
      ?mred rdf:type ncit:Erythrocyte_Count.
      ?mred mged:has_value "5.0"^^<http://www.w3.org/2001/XMLSchema#float>.
   }
   SERVICE <http://localhost:2040/sparql>{
      ?y mged:unique_identifier ?id.
      ?y mged:has_nodes ?ned.
      ?ned rdf:type ncit:Physical_Examination_Date.
      ?ned mged:has_value ?examdate.
} }
ORDER BY (?id) (?examdate)
```

Αποτέλεσμα:

```
-----------------------------------------------------------------------------------------------------
| id                          | examdate                                                            |
=====================================================================
| "03068805833"^^ *1| "2011-07-16"^^<http://www.w3.org/2001/XMLSchema#date> |
| "03068805833"^^ *1| "2011-07-18"^^<http://www.w3.org/2001/XMLSchema#date> |
-----------------------------------------------------------------------------------------------------
```

Το ερώτημα απαντήθηκε σε 1ms.

Ερώτημα προς FedX:

```
SELECT distinct ?id ?examdate{
   ?x mged:unique_identifier ?id.
   ?x mged:has_measurement ?mred.
   ?mred rdf:type ncit:Erythrocyte_Count.
   ?mred mged:has_value "5.0"^^<http://www.w3.org/2001/XMLSchema#float>.

   ?y mged:unique_identifier ?id.
   ?y mged:has_nodes ?ned.
   ?ned rdf:type ncit:Physical_Examination_Date.
   ?ned mged:has_value ?examdate.
}
ORDER BY (?id) (?examdate)
```

Αποτέλεσμα:

```
[id="03068805833"^^ *1; examdate="2011-07-16"^^ *3]
[id="03068805833"^^ *1;examdate="2011-07-18"^^ *3]
```

όπου *3=<http://www.w3.org/2001/XMLSchema#date>.
Το ερώτημα απαντήθηκε σε 65ms.

6. Ζητούνται από το σύστημα οι τιμές των αιματολογικών στοιχείων όσων πάσχουν από αρρυθμία.

Ερώτημα προς SPARQL:

```
SELECT ?type ?value{
   SERVICE <http://localhost:2030/sparql>{
      ?x mged:unique_identifier ?id.
      ?x mged:has_disease_state ncit:Arrhythmia.
   }
   OPTIONAL{SERVICE <http://localhost:2050/sparql>{
      ?z mged:unique_identifier ?id.
```

```
      ?z mged:has_measurement ?mbe.
      ?mbe rdf:type ncit:Hematologic_Technique.
      ?mbe rdf:type ?type.
      ?type rdfs:subClassOf ncit:Hematologic_Technique.
      ?mbe mged:has_value ?value.
   }}
   FILTER (?type != ncit:Hematologic_Technique)
   FILTER (?type != ncit:Blood_Cell_Count)
   FILTER (?value != "-0.0"^^<http://www.w3.org/2001/XMLSchema#string>)
}
ORDER BY (?type)
```

Αποτέλεσμα:

```
-------------------------------------------------------------------------------------------------------------------
| type                                                                          |value                |
===================================================================================================================
| <http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Erythrocyte_Count>       | "5.0"^^ *2          |
| <http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Erythrocyte_Count>       | "4.8"^^ *2          |
| <http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Erythrocyte_Count>       | "5.5"^^ *2          |
| <http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Hematocrit>              | "39.0"^^ *2         |
| <http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Hematocrit>              | "44.0"^^ *2         |
| <http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Hemoglobin_Measurement> |"12.5"^^ *2|
| <http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Hemoglobin_Measurement> |"15.0"^^ *2|
| <http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Leukocyte_Count>         | "6.5"^^ *2          |
| <http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Leukocyte_Count>         | "7.0"^^ *2          |
| <http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Platelet_Count>          | "112.0"^^ *2        |
| <http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Platelet_Count>          | "128.0"^^ *2        |
-------------------------------------------------------------------------------------------------------------------
```

Το ερώτημα απαντήθηκε σε 1ms.

Ερώτημα προς FedX:

```
SELECT DISTINCT ?type ?value{
   ?x mged:unique_identifier ?id.
   ?x mged:has_disease_state ncit:Arrhythmia.
   ?z mged:unique_identifier ?id.
   ?z mged:has_measurement ?mbe.
   ?mbe rdf:type ncit:Hematologic_Technique.
   ?mbe rdf:type ?type.
   ?type rdfs:subClassOf ncit:Hematologic_Technique.
   ?mbe mged:has_value ?value.
   FILTER (?type != ncit:Hematologic_Technique)
   FILTER (?type != ncit:Blood_Cell_Count)
   FILTER (?value != "-0.0"^^<http://www.w3.org/2001/XMLSchema#string>)
}
ORDER BY (?type)
```

Αποτέλεσμα:

[type=http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Erythrocyte_Count;value="4.8"^^ *2]
[type=http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Erythrocyte_Count;value="5.5"^^ *2]
[type=http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Erythrocyte_Count;value="5.0"^^ *2]
[type=http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Hematocrit;value="44.0"^^ *2]
[type=http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Hematocrit;value="39.0"^^ *2]
[type=http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Hemoglobin_Measurement;value="12
.5"^^ *2]
[type=http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Hemoglobin_Measurement;value="15
.0"^^*2]
[type=http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Leukocyte_Count;value="7.0"^^<http
://www.w3.org/2001/XMLSchema#float>]
[type=http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Leukocyte_Count;value="6.5"^^ *2]
[type=http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Platelet_Count;value="112.0"^^ *2]
[type=http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Platelet_Count;value="128.0"^^ *2]

Το ερώτημα απαντήθηκε σε 32ms.

Στο παρόν ερώτημα αναζητούνται σύνολα στοιχείων για κάθε ασθενή (αιματολογικές μετρήσεις) τα οποία έχουν όλα μια κοινή υπερκλάση. Σύμφωνα με τους κανόνες συλλογιστικής RDFS εάν για έναν πόρο S ισχύει: S rdf:type C και D rdfs:subClassOf C, τότε ισχύει και S rdf:type D. Οι κατάλληλες προτάσεις σχηματίζονται από την συμπερίληψη της μηχανής συλλογιστικής RDFS της Jena στο αρχείο HL7_RULES.rules. Κατά συνέπεια, αντί να ζητείται κάθε αιματολογική μέτρηση χωριστά, ζητούνται όλες μαζί ως στιγμιότυπα των υποκλάσεων της κλάσης ncit: Hematologic_Technique. Επίσης επιστρέφεται και το όνομα της κλάσης της κάθε μέτρησης καθώς οι αριθμητικές τιμές από μόνες τους δεν έχουν καμία πρακτική σημασία. Για την αποφυγή επιστροφής των μετρήσεων ενός ατόμου ως στιγμιότυπα της κλάσης ncit: Hematologic_Technique ή μιας πιο γενικής υποκλάσης της όπως η ncit:Blood_Cell_Count (κάτι το οποίο έχει τόση χρησιμότητα, όση και οι αριθμητικές τιμές από μόνες τους) χρησιμοποιείται μια έκφραση FILTER που αποκλείει τις λύσεις με το συγκεκριμένο όνομα κλάσης. Ομοίως αποκλείονται και οι όποιες τιμές «-0.0» που σημαίνει ότι αυτή η μέτρηση δεν έγινε ποτέ.

ΚΕΦΑΛΑΙΟ 7: ΕΠΙΛΟΓΟΣ

7.1 Σύνοψη και συμπεράσματα

Στα πλαίσια της παρούσας διπλωματικής εργασίας ερευνήθηκε το πρόβλημα της ολοκλήρωσης δεδομένων, εξετάστηκε η κατηγορία λύσεων που βασίζεται σε οντολογίες και διαπιστώθηκε η επιρροή του παραπάνω συνόλου (πρόβλημα – λύση) στο πεδίο της ιατροφαρμακευτικής περίθαλψης.

Έχοντας τα παραπάνω υπόψιν δημιουργήθηκε η εφαρμογή IntegraHEALTH 1.0 με σκοπό να ολοκληρώσει δημογραφικά και ιατρικά δεδομένα ασθενών μιας σχεσιακής βάσης δεδομένων και αρχείων των προτύπων DICOM και HL7. Γίνεται χρήση των τεχνολογιών Σημασιολογικού Ιστού για τη μετατροπή των δεδομένων σε μια ενιαία μορφή και την υποβολή ερωτημάτων σε αυτά μέσω του εργαλείου ομόσπονδων ερωτημάτων FedX και των εσωτερικών μηχανισμών της γλώσσας SPARQL.

Από την πορεία υλοποίησης και τη μορφή των ερωτημάτων που δίνονται στο κεφάλαιο 6 προκύπτουν τα παρακάτω συμπεράσματα:

- Η ολοκλήρωση ιατρικών και δημογραφικών δεδομένων μπορεί να γίνει σε λιγότερα βήματα χωρίς πολλά ενδιάμεσα στάδια αποθήκευσης ή μετατροπής σε αντίθεση με άλλα συστήματα όπως αυτά που παρουσιάστηκαν στο κεφάλαιο 2.
- Η σημασιολογική ολοκλήρωση δεδομένων υγείας που είναι αποθηκευμένα πρωτογενώς σε ετερογενή μορφή επιτρέπει όχι μόνο το συνδυασμό αυτών των ετερογενών δεδομένων, αλλά και την παραγωγή νέας γνώσης από αυτά (μέσω της διαδικασίας συλλογιστικής που επιτελείται), καθώς και την απάντηση σύνθετων ερωτημάτων που δεν μπορούν να απαντηθούν με απευθείας ερώτηση των πρωτογενών δεδομένων.
- Η υιοθέτηση μιας στατικής προσέγγισης στην αντιστοιχία πηγών δεδομένων με το RDF μοντέλο προτιμήθηκε από μια δυναμική προσέγγιση μετατροπής SPARQL ερωτημάτων σε αντίστοιχα SQL ερωτήματα και κλήσεις προγραμματιστικών μεθόδων πρόσβασης σε δεδομένα που ακολουθούν πρότυπα υγείας (όπως των DICOM και HL7 που εξετάστηκαν στα πλαίσια της τρέχουσας διπλωματικής εργασίας). Αυτή η επιλογή έγινε κατ' αρχήν λόγω της αδυναμίας ενσωμάτωσης της διαδικασίας RDFS συλλογιστικής στη διαδικασία μετασχηματισμού (κάτι τέτοιο έχει γίνει δυνατό μόνο για ένα υποσύνολο της RDFS [3]), γεγονός που επιβάλλει την υλοποίηση (materialization) των παραγόμενων RDF γράφων. Ένας άλλος λόγος αφορά στην ιδιαιτερότητα των προτύπων υγείας, για τα οποία δεν έχει προταθεί ή προτυποποιηθεί κάποια εκφραστική γλώσσα ερωτημάτων, γεγονός το οποίο εμποδίζει ως ένα βαθμό την εφαρμογή μιας διαδικασίας μετάφρασης των εισερχόμενων SPARQL ερωτημάτων σε αντίστοιχα ερωτήματα επί των πηγών των πρωτογενών δεδομένων.

- Η μηχανή FedX επιτυγχάνει τους στόχους που τέθηκαν στην εισαγωγή, αφού πέρα από τη μορφή του γράφου που πρέπει να γνωρίζει ο χρήστης, δεν απαιτείται καμία άλλη εξωτερική πληροφορία. Η προκαθορισμένη μηχανή ερωτημάτων SPARQL επιτυγχάνει μόνο τον ένα από τους δύο στόχους καθώς αποκρύπτει την αρχική μορφή των δεδομένων, αλλά απαιτεί γνώση της αρχικής θέσης τους (προτάσεις SERVICE).
- Υπάρχει μια σημαντική διαφορά στην ταχύτητα απόκρισης των δύο μηχανών. Η μηχανή SPARQL μπορεί να δώσει αποτελέσματα κατά μέσο όρο 3 φορές πιο γρήγορα από την FedX. Μετά από μεγάλο διάστημα λειτουργίας τα ερωτήματα προς SPARQL απαντώνται ακαριαία (1ms) ενώ της FedX σε 30 – 70ms ανάλογα με το ερώτημα. Τα παραπάνω ευρήματα κρίνονται φυσιολογικά καθώς στην περίπτωση των προτάσεων SERVICE το σύστημα απαλλάσσεται από τη διαδικασία σχηματισμού υποερωτημάτων μιας και αυτό το πράττει ο χρήστης.
- Το εργαλείο D2RQ στερείται μιας σημαντικής δυνατότητας (αυτή της αυτόματης ενημέρωσης RDF προτάσεων του μοντέλου, σε περίπτωση ενημέρωσης των δεδομένων ενός πίνακα). Η αντιπρόταση των κατασκευαστών (επαναδημιουργία του μοντέλου) έχει αμελητέο κόστος σε μια μικρή βάση δεδομένων, αλλά είναι πιθανό να παρατηρηθούν σημαντικές καθυστερήσεις σε περιπτώσεις βάσεων με μερικές χιλιάδες εγγραφές.

7.2 Προτάσεις βελτίωσης - Επεκτάσεις

Η παρούσα έκδοση της IntegraHEALTH 1.0 είναι κατασκευασμένη ώστε να αναδεικνύει τις προτεινόμενες λύσεις για την ολοκλήρωση δεδομένων χωρίς να λαμβάνει υπόψιν τη μέγιστη απόδοση άλλων λειτουργιών. Έχοντας αυτό υπόψιν, προτείνεται ως μελλοντική βελτίωση της τρέχουσας έκδοσης του λογισμικού η καλύτερη διαχείριση των συναλλαγών (transactions), ώστε να γίνονται οι ενημερώσεις των μοντέλων σε πρώτο χρόνο (θεωρείται ότι η εφαρμογή θα βρίσκεται σε υπολογιστές με αρκετή ισχύ για να συμβαίνει κάτι τέτοιο).

Άλλες προτάσεις βελτίωσης περιλαμβάνουν:

- Τη δημιουργία ενός περισσότερο φιλικού προς τον τελικό χρήστη γραφικού περιβάλλοντος, το οποίο δεν θα προϋποθέτει γνώση SPARQL για την διατύπωση ερωτημάτων στην εφαρμογή.
- Τη δυνατότητα ορισμού και επεξεργασίας αντιστοιχιών με οντολογίες της επιλογής των παρόχων δεδομένων (ιατρών) μέσω γραφικού περιβάλλοντος. Στην τρέχουσα έκδοση, αυτές οι αντιστοιχίες είναι ενσωματωμένες στον κώδικα της εφαρμογής.
- Την εξέταση δυνατότητας ενσωμάτωσης μηχανισμών αυξητικής συλλογιστικής (incremental reasoning) [82] για βελτίωση της απόδοσης του συστήματος, μέσω της αποφυγής επανυπολογισμού όλων των εξαγόμενων συμπερασμάτων, όταν επέρχεται μια μεταβολή στον παραγόμενο RDF γράφο.

- Τη δοκιμή ενσωμάτωσης διαφορετικών μηχανών συλλογιστικής (reasoners) και αποθήκευσης των παραγόμενων RDF δεδομένων (triple stores).
- Τη διεξαγωγή περισσότερο εντατικών μετρήσεων με SPARQL ερωτήματα κλιμακούμενης πολυπλοκότητας σε RDF γράφους διαφόρων μεγεθών και πολυπλοκότητας.

ΒΙΒΛΙΟΓΡΑΦΙΑ – ΑΝΑΦΟΡΕΣ

[1]: AGFA. http://www.agfa.com/global/en/main/

[2]: Carlos Angulo, Pere Crespo, José A. Maldonado, David Moner, Daniel Pérez, Irene Abad, Jesús Mandingorra, Montserrat Robles, "Non-invasive lightweight integration engine for building EHR from autonomous distributed systems", *International Journal of Medical Informatics*, vol. 76, supplement 3, pp. S417-S424), December 2007.

[3]: Ashiq Anjum , Peter Bloodsworth , Andrew Branson , Tamas Hauer , Richard McClatchey , Kamran Munir , Dmitry Rogulin , Jetendr Shamdasani, "The Requirements for Ontologies in Medical Data Integration: A Case Study", *Proceedings of the 11th International Database Engineering and Applications Symposium*, pp.308-314, September 06-08, 2007 .

[4]: ANSI. http://www.ansi.org/

[5]: G. Antoniou, F. Van Harmelen, "A Semantic Web Primer", Cambridge, MA, USA, The MIT Press, 2004.

[6]: Apache Jena. http://incubator.apache.org/jena/index.html

[7]: Apache Tomcat. http://tomcat.apache.org/

[8]: ARQ 2.9.0 – incubating Java Documentation. http://incubator.apache.org/jena/documentation/javadoc/arq/index.html

[9]: James R. Baldwin, "The Data Warehouse: An Overview", Spring 1997, Retrieved from http://varietysoftworks.com/jbaldwin/Education/CS615-DataWarehouse.html

[10]: Andras Belokosztolszki (December 21, 2004), "Semantic Heterogeneity Spells Trouble", *SQL Server Pro*. Retrieved from http://www.sqlmag.com/article/tsql3/semantic-heterogeneity-spells-trouble

[11]: Prashant P. Bendale, Gaur Sunder, "Combining DICOM and HL7 into a Single Effective Protocol In Health Informatics – Opportunities and Challenges", *Telemedicon 2009.*

[12]: Tim Berners-Lee (July 7, 2006) "Linked Data". Retrieved from http://www.w3.org/DesignIssues/LinkedData.html

[13]: BizDharma.com, "What is Vertical and Horizontal integration?", Retrieved from http://bizdharma.com/blog/what-is-vertical-and-horizontal-integration/

[14]: Biohealthmatics.com (2010), Clinical Information Systems (CIS). http://www.biohealthmatics.com/technologies/his/cis.aspx

[15]: Biomedical Informatics Group, Biomedical Information Engineering, Pangea – LE. http://www.ibime.upv.es/bie/index.php?option=com_content&task=view&id=1&Itemid=6

[16]: Bioportal. http://bioportal.bioontology.org/

[17]: Olivier Bodenreider, "Ontologies and data integration in biomedicine: Success stories and challenging issues", *Data Integration in the Life Sciences*, vol. 5109 of *Lecture Notes in Computer Science*, pp. 1-4, 2008.

[18]: Darren D. Brennan, Paul F. Whelan, Kevin Robinson, Ovidiu Ghita, Julie M. O'Brien, Robert Sadleir and Stephen J. Eustace, "Rapid Automated Measurement of Body Fat Distribution from Whole-Body MRI", *Ajr American Journal Of Roentgenology*, vol. 185 no.2, pp. 418-423, 2005.
[19]: caCORE. https://cabig.nci.nih.gov/tools/concepts/caCORE_overview
[20]: CDISC. http://www.cdisc.org/
[21]: Center for Advanced Brain Imaging, The DICOM Standard. http://www.cabiatl.com/mricro/dicom/index.html
[22]: Edgar F. Codd, "A Relational Model of Data for Large Shared Data Banks", *Communications of the ACM*, vol.13, iss.6, pp. 377-387, June 1970.
[23]: Corerpoint Health (2010). The HL7 Evolution [White paper]. Retrieved from http://www.corepointhealth.com/sites/default/files/whitepapers/hl7-v2-v3-evolution.pdf
[24]: FaCT Reasoner. http://www.cs.man.ac.uk/~horrocks/FaCT/
[25]: Fulvio Corno, Laura Farinetti, "Rule-based reasoning in the Semantic Web". [Powerpoint Presentation]. Retrieved from http://elite.polito.it/files/courses/01LHV/2010/8-RuleBasedReasoning.pdf
[26]: DARQ – Federated Queries with SPARQL. http://darq.sourceforge.net/
[27]: DebugIT. http://www.debugit.eu/index.php
[28]: DICOM. http://medical.nema.org/
[29]: dcm4che. http://www.dcm4che.org/
[30]: EHR Vendor Comparison. http://www.ehr-software.net/comparison.htm
[31]: Fluid Operations, FedX. http://www.fluidops.com/fedx/
[32]: FOAF Vocabulary Specification 0.98 (August 9, 2010), http://xmlns.com/foaf/spec/
[33]: Foundational Model of Anatomy. http://sig.biostr.washington.edu/projects/fm/
[34]: Freie Universität Berlin, The D2RQ Platform - Treating Non-RDF Databases as Virtual RDF Graph. http://www4.wiwiss.fu-berlin.de/bizer/d2rq/
[35]: Freie Universität Berlin, (August 10, 2009) The D2RQ Platform v0.7 – Treating Non-RDF Relational Databases as Virtual RDF Graphs User Manual and Language Specification. http://www4.wiwiss.fu-berlin.de/bizer/d2rq/spec/
[36]: GALEN Ontology. http://www.co-ode.org/galen/index.php
[37]: Michel Gagnon, "Ontology–Based Integration of Data Sources", *10th International Conference on Information Fusion*, pp. 1-8, July 9-12, 2007.
[38]: Gene Ontology. http://www.geneontology.org/
[39]: Birte Glimm, Markus Krötzsch. "SPARQL Beyond Subgraph Matching", In *Proceedings of the 9th International Semantic Web Conference*, pp. 241–256. Springer 2010.
[40]: Peter Haase , Tobias Mathäß , Michael Ziller, "An evaluation of approaches to federated query processing over linked data", In *Proceedings of the 6th International Conference on Semantic Systems,* September 1-3, 2010, Graz, Austria
[41]: Health-e-Child. http://www.health-e-child.org
[42]: Health Level 7 Organisation. http://www.hl7.org/

[43]: Healthcare Information and Management Systems Society, Electronic Health Record (EHR). http://www.himss.org/ASP/topics_ehr.asp

[44]: HermiT reasoner. http://hermit-reasoner.com/

[45]: Alice Hertel, Jeen Broekstra, and Heiner Stuckenschmidt. "RDF Storage and Retrieval Systems". In *Handbook on Ontologies*, 2009.

[46]: Chun-Ju Hsiao, Esther Hing, Thomas C. Socey, Bill Cai (December 8, 2010), "Electronic Medical Record/Electronic Health Record Systems of Office-based Physicians: United States, 2009 and Preliminary 2010 State Estimates", National Center for Health Statistics. Retrieved from http://www.cdc.gov/nchs/data/hestat/emr_ehr_09/emr_ehr_09.htm

[47]: http://www.biology-online.org/dictionary/Macromolecule.

[48]: Richard Hull, "Managing Semantic Heterogeneity in Databases: A Theoretical Perspective", In *Proceedings of the 16th ACM Symposium on Principles of Database Systems (PODS'97)*, pp. 51-61, May 12-14, 1997.

[49]: IHE. http://www.ihe.net/

[50]: Interfaceware Chameleon. http://www.interfaceware.com/chameleon.html

[51]: Jena2 Database Interface – How To Create Persistent Models, http://jena.sourceforge.net/DB/creating-db-models.html

[52]: Ashish K. Jha, Catherine M. DesRoches, Eric G. Campbell, Karen Donelan, Sowmya R. Rao, Timothy G. Ferris, Alexandra Shields, Sara Rosenbaum, and David Blumenthal, "Use of Electronic Health Records in U.S. Hospitals", *The New England Journal of Medicine*, vol.360, pp.1628-1638 April 16, 2009.

[53]: Joseki – A SPARQL Server for Jena. http://www.joseki.org/

[54]: Joseki – Configuration. http://www.joseki.org/configuration.html

[55]: Joseki Quick Start. http://www.joseki.org/start.html

[56]: JPEG Committee, JPEG. http://www.jpeg.org/jpeg/index.html

[57]: JPEG Committee, Lossless JPEG. http://www.jpeg.org/jpeg/jpegls.html

[58]: Haim Kilov, "From semantic to object–oriented data modeling", *Proceedings of the first international conference on Systems integration*, pp.385-393, March 1990.

[59]: Maurizio Lenzerini, "Data Integration: A Theoretical Perspective", In *PODS '02: Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 233-246, June 3-5, 2002.

[60]: Paul Litwin, Fundamentals of Relational Database Design. Retrieved from http://www.deeptraining.com/litwin/dbdesign/FundamentalsOfRelationalDatabaseDesign.aspx

[61]: L. Martin, A. Anguita, V. Maojo, E. Bonsma, A.I.D. Bucur, J. Vrijnsen, M. Brochhausen, C. Cocos, H. Stenzhorn, M. Tsiknakis, M. Doerr, H. Kondylakis, "Ontology Based In-tegration of Distributed and Heterogeneous Data Sources in ACGT", In *Proceedings of the First International Conference on Health Informatics (HEALTHINF 2008)*, pp. 301-306, 2008.

[62]: Mediware Information Systems (May 5, 2010), Ascend HL7 Interface Specification. http://www.hosinc.com/Products/Interfaces/interface_documentation.htm#OBX

[63]: MGED Ontology. http://mged.sourceforge.net/ontologies/MGEDontology.php
[64]: Vincent F. Murphy (May 24th, 2003), "Measurement of Blood Flow". Retrieved from http://clinical.medicalengineer.co.uk/pages/clinical-engineering/measurement-of-blood-flow.html
[65]: MySQL 5.6 Rererence Manual, 10.4.3. The BLOB and TEXT Types. Retrieved from http://dev.mysql.com/doc/refman/5.6/en/blob.html
[66]: MySQL 5.6 Rererence Manual, 17.3. Using Triggers. Retrieved from http://dev.mysql.com/doc/refman/5.6/en/triggers.html
[67]: MySQL 5.6 Rererence Manual, 20.3.4.1. Driver/Datasource Class Names, URL Syntax and Configuration Properties for Connector/J. Retrieved from http://dev.mysql.com/doc/refman/5.6/en/connector-j-reference-configuration-properties.html
[68]: National Electrical Manufacturers Association, *Digital Imaging and Communications in Medicine (DICOM)*. [Brochure]. Retrieved from http://medical.nema.org/dicom/geninfo/Brochure.pdf
[69]: National Electrical Manufacturers Association, "Digital Imaging and Communications in Medicine (DICOM), Part 3: Information Object Definitions (PS 3.3)", 2011.
[70]: National Electrical Manufacturers Association, "Digital Imaging and Communications in Medicine (DICOM), Part 5: Data Structures and Encoding (PS 3.5)", 2011.
[71]: National Electrical Manufacturers Association, "Digital Imaging and Communications in Medicine (DICOM), Part 6: Data dictionary (PS 3.6)", 2011.
[72]: National Electrical Manufacturers Association, "Digital Imaging and Communications in Medicine (DICOM), Part 10: Media Storage and File Format for Media Interchange (PS 3.10)", 2011.
[73]: NCI Metathesaurus Ontology. http://ncicb.nci.nih.gov/support
[74]: NCI Thesaurus Ontology. https://cabig.nci.nih.gov/concepts/EVS/
[75]: NEMA. http://www.nema.org/
[76]: OBO Foundry Ontology. http://www.obofoundry.org/ro/
[77]: OpenEHR. http://www.openehr.org/home.html
[78]: openrdf.org...home of Sesame http://www.openrdf.org/
[79]: Open Source Initiative, Open Source Initiative OSI – The MIT License (MIT):Licensing, http://www.opensource.org/licenses/MIT
[80]: ORACLE. A Relational Database Overview. http://docs.oracle.com/javase/tutorial/jdbc/overview/database.html
[81]: ORACLE. Java SE Documentation, Java Image I/O Technology. http://docs.oracle.com/javase/6/docs/technotes/guides/imageio/index.html
[82]: Bijan Parsia, Christian HalaschekWiener,Evren Sirin, "Towards Incremental Reasoning Through Updates in OWL – DL", *WWW Conference (2006)*, pp. 1-6.
[83]: Pellet Reasoner. http://clarkparsia.com/pellet
[84]: RadiologyInfo.org, Bone Density Scan. Retrieved From http://www.radiologyinfo.org/en/info.cfm?pg=dexa

[85]: Run-Length Encoding. http://www.fileformat.info/mirror/egff/ch09_03.htm.
[86]: SearchSQLServer. Relational Database. Retrieved from: http://searchsqlserver.techtarget.com/definition/relational-database
[87]: Kevin Robinson, Paul F. Whelan, Ovidiu Ghita, Darren Brennan, "Measurement and Localisation of Body Fat in Whole Body MRI", In *3rd Annual IEI Biomedical Engineering Research Award*, Dublin, Ireland, 2005
[88]: Andreas Schwarte, Peter Haase, Katja Hose, Ralf Schenkel, and Michael Schmidt, "FedX: A Federation Layer for Distributed QueryProcessing on Linked Open Data", In *Proceedings of ISWC 2011*, Bonn (Germany).
[89]: Kenneth W. Scully, Robert D. Pates, George S. Desper, Alfred F. Connors, Frank E. Harrell Jr., Karen S. Pieper, Robert L. Hannan, Robert E. Reynolds, "Development of an enterprise-wide clinical data repository: merging multiple legacy databases.", *Proceedings of AMIA Annual Symposium 1997*, pp. 32-36, 1997.
[90]: A. Sheth, "Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics", In *Interoperating Geographic Information Systems*, pp. 5-30, 1998.
[91]: Amit P. Sheth , James A. Larson, "Federated database systems for managing distributed, heterogeneous, and autonomous databases", *ACM Computing Surveys (CSUR)*, vol.22 no.3, pp.183-236, September 1990.
[92]: Avi Silberschatz, Henry F. Korth, S. Sudarshan, "Συστήματα Βάσεων Δεδομένων – Η Πλήρης Θεωρία των Βάσεων Δεδομένων", Εκδόσεις Μ.Γκιούρδας, 4η Έκδοση, Αθήνα, 2004.
[93]: Sim-e-Child. http://www.sim-e-child.org/
[94]: SNOMED CT Ontology. http://www.ihtsdo.org/snomed-ct/
[95]: René Spronk (August 21, 2008), HL7 ADT Messages [White Paper]. Retrieved from http://www.ringholm.de/docs/00210_en_HL7_ADT_messages.htm
[96]: Nevena Stolba, Alexander Schanner, "eHealth Integrator – Clinical Data Integration in Lower Austria", *Third International Conference on Computational Intelligence in Medicine and Healthcare (CIMED 2007)*, July 2007.
[97]: Jonathan Strickland, "How Data Integration Works", HowStuffWorks.com, May 14, 2008, Retrieved from http://computer.howstuffworks.com/data-integration.htm
[98]: Talend. http://www.talend.com/index.php
[99]: Douglas Teodoro, Rémy Choquet, Emilie Pasche, Julien Gobeill, Christel Daniel, Patrick Ruch, Christian Lovis, "Biomedical Data Management: A Proposal Framework", *Medical Informatics in a United and Healthy Europe*, pp.175-179, 2009.
[100]:The EN13606 Association, The CEN/ISO EN13606 standard. http://www.en13606.org/the-ceniso-en13606-standard
[101]:Jeffrey D. Ulman, "Information integration using logical views", In *Proceedings of the 6th Int. Conference on Database Theory* (ICDT'97), vol. 1186 of *Lecture Notes in Computer Science*, pp. 19-40, 1997.

[102]:UMLS Ontology. http://www.nlm.nih.gov/research/umls/
[103]:University of Washington, Bone Densitometry. Retrieved From: http://courses.washington.edu/bonephys/opbmd.html
[104]:Panos Vassiliadis, "A Survey of Extract–Transform–Load Technology", *International Journal of Data Warehousing & Mining*, vol. 5, no. 3, pp. 1-27, July-September 2009.
[105]:Virtuoso. http://virtuoso.openlinksw.com/
[106]:W3C. (February 10, 2004) OWL Web Ontology Language Reference. http://www.w3.org/TR/owl-ref/
[107]:W3C. (October 27, 2009) OWL 2 Web Ontology Language Document Overview. http://www.w3.org/TR/owl-overview/
[108]:W3C. (March 25, 2005) RDF Data Access Use Cases and Requirements. http://www.w3.org/TR/rdf-dawg-uc/
[109]:W3C, (February 10, 2004) RDF Primer. http://www.w3.org/TR/rdf-primer/
[110]:W3C, (February 10, 2004) RDF Vocabulary Description Language 1.0: RDF Schema. http://www.w3.org/TR/rdf-schema/
[111]:W3C, (May 12, 2011) SPARQL 1.1 Query Language. http://www.w3.org/TR/rdf-sparql-query/
[112]:W3C, (August 09, 2011) Turtle – Terse RDF Triple Language. http://www.w3.org/TR/turtle/
[113]:W3C, (March 27, 2004) Using Qualified Names (QNames) as Identifiers in XML Content. http://www.w3.org/2001/tag/doc/qnameids
[114]:H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, S. Hübner, "Ontology-based Integration of Information – A Survey of Existing Approaches", *Proceedings of IJCAI-01 Workshop: Ontologies and Information Sharing*, Seattle, WA, 2001.
[115]:Jennifer E. Weiss, Norman T. Ilowite, "Juvenile Idiopathic Arthritis", *Pediatric Clinics of North America*, vol. 52, issue 2, pp 413-442, April 2005.
[116]:Fred Wood, "The CDISC Study Data Tabulation Model (SDTM): History, Perspective, and Basics.", *PharmaSUG Proceedings*, June 2008.
[117]:Yale Center for Medical Informatics, The EAV/CR Model of Data Representation. http://ycmi.med.yale.edu/nadkarni/eav_cr_frame.htm
[118]:Patrick Ziegler and Klaus R. Dittrich, "Three Decades of Data Integration – All Problems Solved?", In *Proceedings of IFIP 18th World Computer Congress (WCC 2004),* vol. 12, pp. 3-12, August 22-27, 2004.
[119]:Hubert Zimmermann, "OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection", *IEEE transactions on communications*, vol. 28, no. 4, pp. 425-432, April 1980.
[120]:Γεώργιος Ι. Ρόζος, «Αρχές Γενετικής (β΄ μέρος)». Retrieved from http://www.alfa-omega.gr/article.asp?AID=1650
[121]:Γεώργιος Στοΐλος, «Γλώσσες Αναπαράστασης Γνώσης στο Σημασιολογικό Ιστό», εισαγωγικές σημειώσεις, ΕΜΠ, 2007.
[122]:Ηλεκτρονική Συνταγογράφηση. http://www.e-syntagografisi.gr/

[123]:Αναστάσιος Τάγαρης, «Εισαγωγή στο πρότυπο HL7», Σεμινάριο Διαλειτουργικότητας, ΕΜΠ, 19-21 Σεπτεμβρίου 2005.

[124]:Φυσιολογικές τιμές Βιοχημικών και Αιματολογικών εξετάσεων. *http://www.ippokratios.com.* Retrieved from http://www.ippokratios.com/index.php?option=com_content&task=view &id=25&Itemid=44

[125]:Φυσιολογικές τιμές μικροβιολογικών εξετάσεων, γενικής, βιοχημικής, ανοσολογικής αίματος. Retrieved from http://www.uromed.gr/index.php?option=com_content&view=article&id=244:microbiologic-blood-test&catid=43:general-urology-topics&Itemid=103

[126]:Φυσιολογικές τιμές συνηθισμένων βιοχημικών εξετάσεων. Retrieved from http://pfy.gr/forum/index.php?topic=963.0;wap2

ΠΑΡΑΡΤΗΜΑ Α: ΠΑΡΑΔΕΙΓΜΑΤΑ ΕΠΙΚΕΦΑΛΙΔΩΝ ΑΡΧΕΙΩΝ DICOM

Οι παρακάτω εικόνες αποτελούν παραδείγματα κεφαλίδων αρχείων DICOM. Η εικόνα Α.1 παρουσιάζει ένα από τα αρχεία DICOM που κατασκευάστηκαν από τις κλάσεις της IntegraHEALTH 1.0. Η λήψη των στοιχείων έγινε κατά την εκτέλεση ενός στιγμιότυπου της κλάσης ListDicomHeader.java, η οποία για λόγους επαλήθευσης είναι ρυθμισμένη να τα παρουσιάζει στην κονσόλα εξόδου του eclipse SDK. Σε κάθε γραμμή απεικονίζεται κατά σειρά η ετικέτα του στοιχείου, η αναπαράσταση της τιμής του, το όνομά του και τέλος, η τιμή του.

(0008,0005) [CS] Specific Character Set [ISO_IR 100]
(0008,0008) [CS] Image Type [MRI]
(0008,0012) [DA] Instance Creation Date [20111107]
(0008,0013) [TM] Instance Creation Time [150541.109]
(0008,0018) [UI] SOP Instance UID [2.25.237948091496247999995076428383939984949]
(0008,1080) [LO] Admitting Diagnoses Description [KAKWSI GONATOU]
(0010,0020) [LO] Patient ID [23058504966]
(0010,0030) [DA] Patient's Birth Date [1985-05-23]
(0010,0050) [SQ] Patient's Insurance Plan Code Sequence [NONE]
(0010,1005) [PN] Patient's Birth Name [CHRISTODOULOU^NIKOLAOS^ANTONIOS]
(0010,1040) [LO] Patient's Address [ITEA KARDITSAS]
(0010,2154) [SH] Patient's Telephone Numbers [2310671390]
(0010,4000) [LT] Patient Comments [MERIKI RIKSI XIASTOU]
(0018,11A0) [DS] Body Part Thickness [-0.0]
(0020,000D) [UI] Study Instance UID [2.25.264310857274771369990752086712171101414]
(0020,000E) [UI] Series Instance UID [2.25.158131721030661728110920047243166241575]
(0020,1070) [IS] Other Study Numbers [-0.0|-0.0]
(0028,0002) [US] Samples per Pixel [3]
(0028,0004) [CS] Photometric Interpretation [YBR_FULL_422]
(0028,0010) [US] Rows [349]
(0028,0011) [US] Columns [550]
(0028,0100) [US] Bits Allocated [8]
(0028,0101) [US] Bits Stored [8]
(0028,0102) [US] High Bit [7]
(0028,0103) [US] Pixel Representation [0]
(3008,0250) [DA] Treatment Date [2009-04-03]

Εικόνα Α.1: Παράδειγμα επικεφαλίδας αρχείου DICOM κατασκευασμένο από την εφαρμογή IntegraHEALTH 1.0.

Η παραπάνω εικόνα παρουσιάζει μια επιλογή πληροφοριών που εξυπηρετεί τους σκοπούς της εφαρμογής IntegraHEALTH 1.0. Στις παρακάτω εικόνες δίνονται επιπλέον παραδείγματα (μαζί με τις πηγές τους) καθένα με το δικό του σύνολο πληροφοριών. Το πρότυπο επιτρέπει την αυθαίρετη επιλογή στοιχείων πληροφορίας για την επικεφαλίδα ενός αρχείου (υπενθυμίζεται ότι είναι δυνατή η κωδικοποίηση πάνω από 3000 στοιχείων στην κεφαλίδα ενός αρχείου).

Group Tag	Element Tag	Tag Description	Value
0008	1090	Manufacturer's Model Name	VJ-DR
0010	0010	Patient's Name	ELMO/CVC
0010	0011	Private Tag	FELINE
0010	0020	Patient ID	ELMO
0010	0030	Patient's Birth Date	00000000
0010	0040	Patient's Sex	Female
0010	1030	Patient's Weight	0.
0010	4000	Patient Comments	
0018	0015	Body Part Examined	no ade
0018	1000	Device Serial Number	00000000
0018	1020	Software Versions(s)	1.0.0.544
0018	1164	Imager Pixel Spacing	0.127\0.127
0018	5100	Patient Position	test
0018	7004	Detector Type	DIRECT
0018	7006	Detector Description	VJDR-35
0018	701A	Detector Binning	1.\1.
0020	000D	Study Instance UID	1.2.826.0.1.3680043.2.1330.1000001.3.0234804208.5305437979
0020	000E	Series Instance UID	1.2.826.0.1.3680043.2.1330.1000001.4.0234804208.5305418977
0020	0010	Study ID	Recheck
0020	0011	Series Number	1
0020	0013	Image Number	1832
0020	0020	Patient Orientation	\
0020	0060	Laterality	None
0020	4000	Image Comments	
0028	0002	Samples per Pixel	1
0028	0004	Photometric Interpretation	MONOCHROME2
0028	0010	Rows	3200
0028	0011	Columns	2304
0028	0100	Bits Allocated	16
0028	0101	Bits Stored	16
0028	0102	High Bit	15
0028	0103	Pixel Representation	0
0028	0301	Burned In Annotation	NO

Εικόνα Α.2 (http://www.codeproject.com/KB/graphics/dicomImageViewer.aspx).

First 128 bytes: unused by DICOM format
Followed by the characters 'D','I','C','M'
This preamble is followed by extra information e.g.:

0002,0000,File Meta Elements Group Len: 132
0002,0001,File Meta Info Version: 256
0002,0010,Transfer Syntax UID: 1.2.840.10008.1.2.1.
0008,0000,Identifying Group Length: 152
0008,0060,Modality: MR
0008,0070,Manufacturer: MRIcro
0018,0000,Acquisition Group Length: 28
0018,0050,Slice Thickness: 2.00
0018,1020,Software Version: 46\64\37
0028,0000,Image Presentation Group Length: 148
0028,0002,Samples Per Pixel: 1
0028,0004,Photometric Interpretation: MONOCHROME2.
0028,0008,Number of Frames: 2
0028,0010,Rows: 109
0028,0011,Columns: 91
0028,0030,Pixel Spacing: 2.00\2.00
0028,0100,Bits Allocated: 8
0028,0101,Bits Stored: 8
0028,0102,High Bit: 7
0028,0103,Pixel Representation: 0
0028,1052,Rescale Intercept: 0.00
0028,1053,Rescale Slope: 0.00392157
7FE0,0000,Pixel Data Group Length: 19850
7FE0,0010,Pixel Data: 19838

Εικόνα Α.3 (http://www.cabiatl.com/mricro/dicom/index.html).

ΠΑΡΑΡΤΗΜΑ Β:
ΠΙΝΑΚΕΣ HL7

Οι παρακάτω πίνακες είναι ενδεικτικοί και παρουσιάζουν μερικά από τα πιο δημοφιλή χαρακτηριστικά μηνυμάτων HL7 (τύπους μηνυμάτων, ονόματα τμημάτων, σύνθετους τύπους δεδομένων). Το πλήρες λεξιλόγιο δίνεται στην ιστοσελίδα του HL7 Organization [42] ενώ μια εφαρμογή η οποία περιέχει πληροφορίες για τη σύνταξη αρκετών μηνυμάτων και τμημάτων, ακόμα και στην δοκιμαστική (trial) έκδοσή της είναι το Chameleon της Interfaceware .

1. Πίνακας τύπων μηνυμάτων

Κωδικός	Λειτουργία Μηνύματος
ADTA01	Ειδοποίηση εισαγωγής/επίσκεψης (Admit/visit notification)
ADTA02	Μεταφορά ενός ασθενή (Transfer a patient)
ADTA03	Εξιτήριο/τέλος επίσκεψης (Discharge/end visit)
ADTA04	Εγγραφή ενός ασθενή (Register a patient)
ADTA08	Ενημέρωση στοιχείων ασθενή (Update patient information)
ADTA11	Ακύρωση εισαγωγής/επίσκεψης (Cancel admit/visit notification)
ADTA20	Ενημέρωση κατάστασης κλίνης (Bed staus update)
ADTA54	Αλλαγή θεράποντος ιατρού (Change attending doctor)
ADTA60	Ενημέρωση πληροφοριών αλλεργίας (Update allergy information)
BARP12	Ενημέρωση διάγνωσης/διαδικασίας (Update diagnosis/procedure)
BTSO31	Μετάγγιση/μετατόπιση παραγώγων αίματος (Blood product transfusion/disposition)
CRMC01	Εγγραφή ενός ασθενή σε μια κλινική δοκιμή (Register a patient on a clinical trial)
CSUC10	Ο ασθενής ολοκληρώνει την κλινική δοκιμή (Patient completes the clinical trial)
MDMT07	Ειδοποίηση επεξεργασίας εγγράφου (Document edit notification)
MDMT09	Ειδοποίηση αντικατάστασης εγγράφου (Document replacement notification)
OMBO27	Παραγγελία παραγώγου αίματος (Blood product order)
OMLO21	Παραγγελία εργαστηρίου (Laboratory order)
ORUR01	Αυτόκλητη μετάδοση ενός μηνύματος (ιατρικής) παρατήρησης (Unsolicited transmission of an observation message)
OSQQ06	Ερώτημα για την κατάσταση παραγγελίας (Query for order status)
OULR21	Αυτόκλητη παρατήρηση εργαστηρίου (Unsolicited laboratory observation)
PMUB01	Εισαγωγή καταχώρησης προσωπικού (Add personnel record)
PMUB02	Ενημέρωση καταχώρησης προσωπικού (Update personnel record)
PMUB03	Διαγραφή καταχώρησης προσωπικού (Delete personnel record)
QBPQ21	Λήψη δημογραφικών στοιχείων ενός ατόμου (Get person demographics)
QBPZ73	Πληροφορίες για τηλεφωνήματα (Information about Phone Calls)
QBPZ89	Ιστορικό εργαστηριακών εξετάσεων (Lab results history)
QRYA19	Ερώτημα ασθενούς (Patient Query)
RDEO11	Κωδικοποιημένη παραγγελία φαρμακείου/θεραπείας (Pharmacy/treatment encoded order)
SIUS12	Ειδοποίηση κλεισίματος ραντεβού (Notification of new appointment booking)
SIUS15	Ειδοποίηση ακύρωσης ραντεβού (Notification of appointment cancellation)
STIS30	Παραγγελία αντικειμένου (Request item)

2. Πίνακας ονομάτων τμημάτων

Όνομα	Λειτουργία Τμήματος
ACC	Ατύχημα (Accident)
AIS	Πληροφορίες ραντεβού (Appointment information)
AL1	Πληροφορίες αλλεργίας ασθενή (Patient allergy information)
ARQ	Παραγγελία ραντεβού (Appointment Request)
BLG	Κωδικός αίματος (Blood Code)
DG1	Διάγνωση (Diagnosis)
ERR	Σφάλμα (Error)
EVM	Τύπος συμβάντος (Event type)
FHS	Κεφαλίδα αρχείου (File header)
FT1	Οικονομική συναλλαγή (Financial transaction)
IN1	Ασφάλεια (Insurance)
MSA	Επιβεβαίωση μηνύματος (Message acknowledgement)
MSH	Κεφαλίδα μηνύματος (Message header)
NK1	Συγγενείς/συμβαλλόμενοι (Next of kin/Assosiated parties)
NPU	Ενημέρωση κατάστασης κλίνης (Bed status update)
NTE	Σημειώσεις και σχόλια (Notes and comments)
OBR	Παραγγελία ιατρικής παρακολούθησης (Observation request)
OBX	Αποτέλεσμα ιατρικής παρακολούθησης (Observation result)
ORC	Κοινή παραγγελία (Common order)
PD1	Πρόσθετα δημογραφικά στοιχεία ασθενή (Patient additional demographic)
PDA	Θάνατος ασθενή και αυτοψία (Patient death and autopsy)
PID	Στοιχεία ταυτότητας ασθενή (Patient identification)
PV1	Επίσκεψη ασθενούς (Patient visit)
RXO	Παραγγελία φαρμακείου/θεραπείας (Pharmacy/treatment order)

3. Πίνακας σύνθετων τύπων δεδομένων

Όνομα	Περιεχόμενο Τύπου Δεδομένων
AD	Διεύθυνση (Address)
CE	Κωδικοποιημένο στοιχείο (Coded element)
CF	Κωδικοποιημένο στοιχείο με μορφοποιημένες τιμές (Coded element with formatted values)
CQ	Σύνθετη ποσότητα με μονάδες μέτρησης (Composite quantity with units)
DTM	Ημερομηνία/Ωρα (Date/Time)
EI	Αναγνωριστικό οντότητας (Entity identifier)
FN	Οικογενειακό όνομα (Family name)
FT	Μορφοποιημένα δεδομένα κειμένου (Formatted text data)
ID	Κωδικοποιημένες τιμές για πίνακες HL7 (Coded Values for HL7 tables)
MO	Χρήματα (Money)
NM	Αριθμητικό (Numeric)
PL	Τοποθεσία ατόμου (Person Location)
SN	Δομημένο αριθμητικό (Structured Numeric)
TS	Χρονική στιγμή (Time stamp)
XCN	Επεκταμένη σύνθετη ταυτότητα και όνομα ατόμου (Extended Composite ID Number and Name for Persons)
XPN	Επεκταμένο ονομα ατόμου (Extended person name)

ΠΑΡΑΡΤΗΜΑ Γ: ΠΙΝΑΚΕΣ ΑΞΙΩΜΑΤΩΝ/ΚΑΝΟΝΩΝ RDF/RDFS/OWL

1. Πίνακες αξιωμάτων

1.1 RDF/RDFS

rdf:type rdfs:domain rdfs:Resource.
rdfs:domain rdfs:domain rdf:Property.
rdfs:range rdfs:domain rdf:Property.
rdfs:subPropertyOf rdfs:domain rdf:Property.
rdfs:subClassOf rdfs:domain rdfs:Class.
rdf:subject rdfs:domain rdf:Statement.
rdf:predicate rdfs:domain rdf:Statement.
rdf:object rdfs:domain rdf:Statement.
rdfs:member rdfs:domain rdfs:Resource.
rdf:first rdfs:domain rdf:List.
rdf:rest rdfs:domain rdf:List.
rdfs:seeAlso rdfs:domain rdfs:Resource.
rdfs:isDefinedBy rdfs:domain rdfs:Resource.
rdfs:comment rdfs:domain rdfs:Resource.
rdfs:label rdfs:domain rdfs:Resource.
rdf:value rdfs:domain rdfs:Resource.
rdf:type rdfs:range rdfs:Class.
rdfs:domain rdfs:range rdfs:Class.
rdfs:range rdfs:range rdfs:Class.
rdfs:subPropertyOf rdfs:range rdf:Property.
rdfs:subClassOf rdfs:range rdfs:Class.
rdf:subject rdfs:range rdfs:Resource.
rdf:predicate rdfs:range rdfs:Resource.
rdf:object rdfs:range rdfs:Resource.
rdfs:member rdfs:range rdfs:Resource.
rdf:first rdfs:range rdfs:Resource.
rdf:rest rdfs:range rdf:List.
rdfs:seeAlso rdfs:range rdfs:Resource.
rdfs:isDefinedBy rdfs:range rdfs:Resource.
rdfs:comment rdfs:range rdfs:Literal.
rdfs:label rdfs:range rdfs:Literal.
rdf:value rdfs:range rdfs:Resource.

rdf:Alt rdfs:subClassOf rdfs:Container.
rdf:Bag rdfs:subClassOf rdfs:Container.
rdf:Seq rdfs:subClassOf rdfs:Container.

rdf:XMLLiteral rdf:type rdfs:Datatype.
rdf:XMLLiteral rdfs:subClassOf rdfs:Literal.
rdfs:Datatype rdfs:subClassOf rdfs:Class.
rdfs:isDefinedBy rdfs:subPropertyOf rdfs:seeAlso.

```
rdf:_1 rdf:type rdfs:ContainerMembershipProperty.
rdf:_1 rdfs:domain rdfs:Resource.
rdf:_1 rdfs:range rdfs:Resource.
rdf:_2 rdf:type rdfs:ContainerMembershipProperty.
rdf:_2 rdfs:domain rdfs:Resource.
rdf:_2  rdfs:range rdfs:Resource .
…

rdfs:ContainerMembershipProperty rdfs:subClassOf rdf:Property.
rdfs:ContainerMembershipProperty rdf:type rdfs:Class.

rdfs:Resource rdf:type rdfs:Class.
rdfs:Class rdf:type rdfs:Class.
rdfs:Literal rdf:type rdfs:Class.
rdf:XMLLiteral rdf:type rdfs:Class.
rdfs:Datatype rdf:type rdfs:Class.
rdf:Seq rdf:type rdfs:Class.
rdf:Bag rdf:type rdfs:Class.
rdf:Alt rdf:type rdfs:Class.
rdfs:Container rdf:type rdfs:Class.
rdf:Property rdf:type rdfs:Class.
rdf:Statement rdf:type rdfs:Class.
rdf:List rdf:type rdfs:Class.
rdfs:domain rdf:type rdf:Property.
rdfs:range rdf:type rdf:Property.
rdfs:subPropertyOf rdf:type rdf:Property.
rdfs:subClassOf rdf:type rdf:Property.
rdfs:member rdf:type rdf:Property.
rdfs:seeAlso rdf:type rdf:Property.
rdfs:isDefinedBy rdf:type rdf:Property.
rdfs:comment rdf:type rdf:Property.
rdfs:label rdf:type rdf:Property .
```

1.2 OWL

1.2.1 Κλάσεις

Για τα παρακάτω στοιχεία ισχύει η τριάδα <X, rdf:type, owl:Class>:

```
owl:Class rdfs:subClassOf rdfs:Class.
owl:Class rdfs:subClassOf owl:Nothing.
owl:Thing rdfs:subClassOf owl:Class.
owl:Restriction rdfs:subClassOf owl:Class.
owl:ObjectProperty  rdfs:subClassOf rdf:Property.
owl:DatatypeProperty rdfs:subClassOf rdf:Property.
owl:FunctionalProperty rdfs:subClassOf rdf:Property.
owl:InverseFunctionalProperty rdfs:subClassOf owl:ObjectProperty.
owl:SymmetricProperty rdfs:subClassOf owl:ObjectProperty.
owl:TransitiveProperty rdfs:subClassOf owl:ObjectProperty.
```

1.2.2 Ιδιότητες

owl:sameAs rdfs:domain rdfs:Resource.
owl:sameAs rdfs:range rdfs:Resource.
owl:differentFrom rdfs:domain rdfs:Resource.
owl:differentFrom rdfs:range rdfs:Resource.
owl:equivalentClass rdfs:domain owl:Class.
owl:equivalentClass rdfs:range owl:Class.
owl:disjointWith rdfs:domain owl:Class.
owl:disjointWith rdfs:range owl:Class.
owl:complementOf rdfs:domain owl:Class.
owl:complementOf rdfs:range owl:Class.
owl:unionOf rdfs:domain owl:Class.
owl:unionOf rdfs:range rdf:List.
owl:intersectionOf rdfs:domain owl:Class.
owl:intersectionOf rdfs:range rdf:List.
owl:oneOf rdfs:domain rdfs:Class.
owl:oneOf rdfs:range rdf:List.
owl:equivalentProperty rdfs:domain rdf:Property.
owl:equivalentProperty rdfs:range rdf:Property.
owl:inverseOf rdfs:domain owl:ObjectProperty.
owl:inverseOf rdfs:range owl:ObjectProperty.
owl:onProperty rdfs:domain owl:Restriction.
owl:onProperty rdfs:range rdf:Property.
owl:allValuesFrom rdfs:domain owl:Restriction.
owl:allValuesFrom rdfs:range rdfs:Class.
owl:someValuesFrom rdfs:domain owl:Restriction.
owl:someValuesFrom rdfs:range rdfs:Class.
owl:hasValue rdfs:domain owl:Restriction.
owl:hasValue rdfs:range rdfs:Resource.
owl:cardinality rdfs:domain owl:Restriction.
owl:cardinality rdfs:range xsd:NonNegativeInteger.
owl:minCardinality rdfs:domain owl:Restriction.
owl:minCardinality rdfs:range xsd:NonNegativeInteger.
owl:maxCardinality rdfs:domain owl:Restriction.
owl:maxCardinality rdfs:range xsd:NonNegativeInteger.
owl:distinctMembers rdfs:domain owl:AllDifferent.
owl:distinctMembers rdfs:range owl:List.

2. Παραδείγματα κανόνων συλλογιστικής

Για τα παρακάτω παραδείγματα κανόνων θεωρούμε ένα σύνολο τριάδων E (ένα RDF έγγραφο)

2.1 RDF/RDFS

Περιεχόμενες τριάδες του E	Προστιθέμενες τριάδες στο E
(?u ?a ?v).	(?a rdf:type rdf:Property)
(?a rdfs:domain ?x). (?u ?a ?y).	(u? rdf:type x?)
(?a rdfs:range ?x). (?u ?a ?v).	(v? rdf:type x?)
(?u ?a ?x). όπου το ?x δεν είναι λεκτικό	(?u rdf:type rdfs:Resource) (?x rdf:type rdfs:Resource)
(?u rdfs:subPropertyOf ?v). (?v rdfs:subPropertyOf ?x).	(?u rdfs:subPropertyOf ?x).
(?u rdf:type rdf:Property).	(?u rdfs:subPropertyOf ?u).
(?a rdfs:subPropertyOf ?b). (?u ?a ?y).	(?u ?b ?y).
(?u rdf:type rdfs:Class).	(?u rdfs:subClassOf rdfs:Resource).
(?u rdfs:subClassOf ?x). (?v rdf:type ?u).	(?v rdf:type ?x) .
(?u rdfs:subClassOf ?x). (?v rdf:type ?u).	(?v rdf:type ?x).
(?u rdf:type rdfs:Class).	(?u rdfs:subClassOf ?u).
(?u rdfs:subClassOf ?v). (?v rdfs:subClassOf ?x).	(?u rdfs:subClassOf ?x).
(?u rdf:type rdfs:Datatype).	(?u rdfs:subClassOf rdfs:Literal).
(?u rdfs:domain ?v). (?v rdfs:subClassOf ?z).	(?u rdfs:domain ?z).
(?u rdfs:range ?v). (?v rdfs:subClassOf ?z).	(?u rdfs:range ?z).
(?u rdfs:domain ?v). (?w rdfs:subPropertyOf ?u).	(?w rdfs:domain ?v).
(?u rdfs:range ?v). (?w rdfs:subPropertyOf ?u).	(?w rdfs:range ?v).

2.2 OWL

Περιεχόμενες τριάδες του E	Προστιθέμενες τριάδες στο E
(?u rdf:type owl:TransitiveProperty). (?a ?u ?b). (?b ?u ?c).	(?a ?u ?c).
(?u rdf:type owl:SymmetricProperty). (?a ?u ?b).	(?b ?u ?a).
(?u owl:inverseOf ?v). (?a ?u ?b).	(?b ?v ?a).
(?a owl:disjointWith ?b). (?c rdf:subClassOf ?b).	(?a owl:disjointWith ?c).

(?a owl:complementOf ?b). (?c rdfs:subclassOf ?a).	(?c owl:disjointWith ?b).
(?u rdf:type owl:FunctionalProperty). (?a ?u ?b). (?a ?u ?c).	(?b owl:sameAs ?c).
(?u rdf:type owl:InverseFunctionalProperty). (?a ?u ?b). (?c ?u ?b).	(?a owl:sameAs ?c).
(?a owl:sameAs ?b). (?a rdf:type rdfs:Class). (?b rdf:type rdfs:Class).	(?a owl:equivalentClass ?b).
(?a rdfs:subPropertyOf ?b). (?b rdfs:subPropertyOf ?a).	(?a owl:equivalentProperty ?b).
(?r owl:onProperty ?p). όπου ?r πόρος που αναφέρεται σε περιορισμό (?r has:value ?v). (?a ?p ?v).	(?a rdf:type ?r).
(?r owl:onProperty ?p). όπου ?r πόρος που αναφέρεται σε περιορισμό (?r allValuesFrom ?v). (?a rdf:type ?r). (?a ?p ?b).	(?b rdf:type ?v)
(?r owl:onProperty ?p). όπου ?r πόρος που αναφέρεται σε περιορισμό (?r someValuesFrom ?v). (?a ?p ?b). (?b rdf:type ?r).	(?b rdf:type s?v)
(?a owl:inveseOf ?b). (?b rdf:type owl:TransitiveProperty).	(?a rdf:type owl:TransitiveProperty).
(?a owl:inveseOf ?b). (?b rdf:type owl:FunctionalProperty).	(?a rdf:type owl:InverseFunctionalProperty).
(?a owl:inveseOf ?b). (?b owl:inveseOf ?c).	(?a rdfs:subPropertyOf ?b).
(?a owl:complementOf ?b). (?b owl:equivalentClass owl:Nothing).	(?a owl:equivalentClass owl:Thing).
(?a rdf:type rdfs:Class).	(?a rdfs:subClassOf owl:Thing). (owl:Nothing rdfs:subClassOf ?a).
(?a owl:disjointWith ?b). (?x rdf:type ?a). (?y rdf:type ?b).	(?x owl:differentFrom ?y).

ΠΑΡΑΡΤΗΜΑ Δ:
ΠΙΝΑΚΕΣ ΜΟΝΤΕΛΩΝ ΟΝΤΟΛΟΓΙΩΝ ΚΑΙ ΕΙΔΙΚΩΝ ΠΡΟΤΑΣΕΩΝ ΚΑΝΟΝΩΝ (JENA)

1. Πίνακας ορισμάτων της ModelFactory.createOntologyModel (OntModelSpec);

Όλες οι παρακάτω τιμές αναφέρονται μόνο σε περιπτώσεις μοντέλων που αποθηκεύονται στη μνήμη του υπολογιστή (π.χ. δεν ισχύουν για τα παραμένοντα (persistent) μοντέλα).

Τιμή OntModelSpec	Προφίλ Γλώσσας	Μηχανή Συλλογιστικής
OWL_MEM	OWL Full	Χωρίς μηχανή
OWL_MEM_TRANS_INF	OWL Full	Μεταβατικής ιδιότητας κλάσεων – ιεραρχίας
OWL_MEM_RULE_INF	OWL Full	Βασισμένη σε κανόνες OWL
OWL_MEM_MICRO_RULE_INF	OWL Full	Βελτιστοποιημένη, βασισμένη σε κανόνες OWL
OWL_MEM_MINI_RULE_INF	OWL Full	Βασισμένη σε υποσυνόλο κανόνων OWL
OWL_DL_MEM	OWL DL	Χωρίς μηχανή
OWL_DL_MEM_RDFS_INF	OWL DL	Βασισμένη σε κανόνες συνεπαγωγής επιπέδου RDFS
OWL_DL_MEM_TRANS_INF	OWL DL	Μεταβατικής ιδιότητας κλάσεων – ιεραρχίας
OWL_DL_MEM_RULE_INF	OWL DL	Βασισμένη σε κανόνες OWL
OWL_LITE_MEM	OWL Lite	Χωρίς μηχανή
OWL_LITE_MEM_TRANS_INF	OWL Lite	Μεταβατικής ιδιότητας κλάσεων – ιεραρχίας
OWL_LITE_MEM_RDFS_INF	OWL Lite	Βασισμένη σε κανόνες συνεπαγωγής επιπέδου RDFS
OWL_LITE_MEM_RULES_INF	OWL Lite	Βασισμένη σε κανόνες OWL
DAML_MEM	DAML+OIL	Χωρίς μηχανή
DAML_MEM_TRANS_INF	DAML+OIL	Μεταβατικής ιδιότητας κλάσεων – ιεραρχίας
DAML_MEM_RDFS_INF	DAML+OIL	Βασισμένη σε κανόνες συνεπαγωγής επιπέδου RDFS
DAML_MEM_RULE_INF	DAML+OIL	Βασισμένη σε κανόνες DAML
RDFS_MEM	RDFS	Χωρίς μηχανή
RDFS_MEM_TRANS_INF	RDFS	Μεταβατικής ιδιότητας κλάσεων – ιεραρχίας
RDFS_MEM_RDFS_INF	RDFS	Βασισμένη σε κανόνες συνεπαγωγής επιπέδου RDFS

2. Πίνακας ειδικών προτάσεων μηχανής κανόνων Jena

Πρόταση	Λειτουργία
isLiteral(?x) notLiteral(?x)	Ελέγχει αν ένα όρισμα είναι ή δεν είναι ένα λεκτικό.
isFunctor(?x) notFunctor(?x)	Ελέγχει αν ένα όρισμα είναι ή δεν είναι ένα λεκτικό που λαμβάνει την τιμή του από μια συνάρτηση (functor-valued literal).
isBNode(?x) notBNode(?x)	Ελέγχει αν ένα όρισμα είναι ή δεν είναι ένας κενός κόμβος.
bound(?x...) unbound(?x..)	Ελέγχει αν τα ορίσματα είναι ή δεν είναι δεσμευμένες μεταβλητές.
equal(?x,?y) notEqual(?x,?y)	Ελέγχει αν x=y ή x!=y αντίστοιχα. Εξετάζεται η σημασιολογική ισότητα των ορισμάτων (π.χ. τα "1"^^xsd:integer και "1"^^xsd:decimal θεωρούνται ίσα).
lessThan(?x, ?y), greaterThan(?x, ?y) le(?x, ?y), ge(?x, ?y)	Ελέγχει αν x<y, x>y, x<=y, x>=y αντίστοιχα. Ισχύει μόνο αν τα x,y είναι και τα δύο αριθμοί ή χρονικές στιγμές (μπορούν να είναι ακέραιοι, κινητής υποδιαστολής ή XSDDateTime).
sum(?a, ?b, ?c) addOne(?a, ?c) difference(?a, ?b, ?c) min(?a, ?b, ?c) max(?a, ?b, ?c) product(?a, ?b, ?c) quotient(?a, ?b, ?c)	Θέτει το c ίσο με (a+b), (a+1), (a-b), min(a,b), max(a,b), (a*b), (a/b). Οι κανόνες δεν ισχύουν αντίστροφα, π.χ. αν στην sum το a και το c είναι δεσμευμένα και το b όχι, τότε η πρόταση δεν θέτει αυτομάτως b = c-a και έτσι δεν επαληθεύεται.
strConcat(?a1, .. ?an, ?t) uriConcat(?a1, .. ?an, ?t)	Συνενώνει (concatenates) τις λεξικολογικές μορφές (lexical forms) όλων των ορισμάτων πλήν του τελευταίου και μετά δεσμεύει την τελευταία μεταβλητή σε ένα απλό λεκτικό (strConcat) ή ένα κόμβο με URI την συνενωμένη λεξικολογική μορφή (uriConcat). Σε κάθε περίπτωση αν ένα όρισμα έχει τιμή URI, τότε χρησιμοποιείται ή λεξικολογική μορφή του.
regex(?t, ?p) regex(?t, ?p, ?m1, .. ?mn)	Αντιστοιχίζει τη λεξικολογική μορφή ενός λεκτικού (?t) στο υπόδειγμα μιας κανονικής έκφρασης (?p). Αν η αντιστοίχηση είναι επιτυχής και αν υπάρχουν πρόσθετα ορίσματα, τότε θα δεσμευθούν τα ορίσματα ?m1 έως ?mn στις πρώτες n ομάδες αντιστοίχησης (capture groups).
now(?x)	Δεσμεύει το ?x σε μια τιμή τύπου xsd:dateTime που αντιστοιχεί στην τρέχουσα ώρα.
makeTemp(?x)	Δεσμεύει το ?x σε ένα νεοδημιουργηθέν κενό κόμβο.
makeInstance(?x, ?p, ?v) makeInstance(?x, ?p, ?t, ?v)	Δεσμεύει το ?v σε έναν κενό κόμβο που θεωρείται ως η τιμή την ιδιότητας ?p για τον πόρο ?x και έχει (προαιρετικά) τύπο ?t. Πολλαπλές κλήσεις με τα ίδια ορίσματα επιστρέφουν τον ίδιο κόμβο.
makeSkolem(?x, ?v1, ... ?vn)	Δεσμεύει το ?x σε ένα κενό κόμβο. Ο κενός κόμβος δημιουργείται βάσει των τιμών των υπολοίπων ορισμάτων. Ίδιοι συνδυασμοί των ?v1 έως ?vn δημιουργούν τον ίδιο κόμβο.
noValue(?x, ?p) noValue(?x ?p ?v)	Αληθή αν δεν υπάρχει κάποια γνωστή τριάδα (x, p, *) ή (x, p, v) στο μοντέλο ή σε ρητές συνεπαγωγές κανονικής αλληλουχίας μέχρι τη στιγμή αποτίμησης της πρότασης.
remove(n, ...) drop(n, ...)	Αφαιρεί την πρόταση (τριάδα) που επαλήθευσε τον n-οστό όρο (προϋπόθεση) του κανόνα . Η remove θα μεταδώσει την αλλαγή σε επακόλουθους κανόνες συμπεριλαμβανομένου και του κανόνα που «πυροδοτεί» την αλλαγή (firing rule). Η drop θα αφαιρέσει «ήσυχα» τις τριάδες από το γράφο αλλά δεν θα «πυροδοτήσει» κανένα κανόνα.

isDType(?l, ?t) notDType(?l, ?t)	Ελέγχει αν το λεκτικό ?l είναι ή δεν είναι στιγμιότυπο του τύπου δεδομένων που ορίζεται από τον πόρο ?t.
print(?x, ...)	Τυπώνει (στο προκαθορισμένο ρεύμα εξόδου) μια αναπαράσταση του κάθε ορίσματος.
listContains(?l, ?x) listNotContains(?l, ?x)	Επαληθεύεται αν ?l είναι μια λίστα που περιέχει (ή, αντίστοιχα, δεν περιέχει)το στοιχείο ?x.
listEntry(?list, ?n, ?val)	Δεσμέυει το ?val στο n – οστό στοιχείο της RDF λίστας ?list. Αν δεν υπάρχει τέτοια καταχώριση η μεταβλητή θα μείνει αδέσμευτη και η συνάρτηση δεν θα επαληθευθει. Χρησιμοποιίται μόνο στο σώμα ενός κανόνα.
listLength(?l, ?len)	Δεσμεύει το ?len στο μήκος της λίστας l?.
listEqual(?la, ?lb) listNotEqual(?la, ?lb)	Η listEqual ελέγχει αν τα δύο ορίσματα είναι λίστες και περιέχουν τα ίδια στοιχεία. Ελέγχεται η σημασιολογική ισότητα των λεκτικών (sameValueAs) χωρίς να λαμβάνει υπόψη τυχόν ιδιότητες owl:sameAs. Η listNotEqual επαληθεύεται όταν αποτυγχάνει η listEqual.
listMapAsObject(?s, ?p ?l) listMapAsSubject(?l, ?p, ?o)	Μπορούν να χρησιμοποιηθούν μόνο στην κεφαλή ενός κανόνα. Συνάγουν λενα σύνολο τριάδων που παράγονται από το όρισμα-λίστα ?l. Η listMapAsObject παράγει τριάδες (?s ?p ?x) για κάθε ?x της λίστας ?l, ενώ η listMapAsSubject παράγει τριάδες (?x ?p ?o).
table(?p) tableAll()	Δηλώνει ότι όλοι οι στόχοι (goals) που εμπλέκουν την ιδιότητα ?p (ή όλοι οι στόχοι, αντίστοιχα) πρέπει να τοποθετηθούν σε πίνακα από την μηχανή ανάστροφής αλληλουχίας (backward engine).
hide(p)	Δηλώνει ότι οι προτάσεις που εμπλέκουν το κατηγόρημα p πρέπει να είναι κρυμμένες. Τυχόν ερωτήματα στο μοντέλο δεν πρέπει να τις εμφανίζουν.

ΠΑΡΑΡΤΗΜΑ Ε:
ΚΛΑΣΕΙΣ ΚΑΙ ΕΞΩΤΕΡΙΚΑ ΑΡΧΕΙΑ ΤΗΣ ΕΦΑΡΜΟΓΗΣ IntegraHEALTH 1.0

1. Πακέτο core.firstTime, κλάση FirstTimeInitialization.java

```
package core.firstTime;

import java.util.Date;

import com.hp.hpl.jena.db.IDBConnection;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;

import core.handlers.BasicFunctionsAndVariables;

public class FirstTimeInitialization {

//ΚΛΑΣΗ ΑΡΧΙΚΟΠΟΙΗΣΕΩΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ
//ΕΚΤΕΛΕΙΤΑΙ ΞΕΧΩΡΙΣΤΑ ΠΡΙΝ ΤΗΝ ΕΦΑΡΜΟΓΗ ΚΑΙ ΔΗΜΙΟΥΡΓΕΙ ΤΑ ΑΠΑΙΤΟΥΜΕΝΑ
PERSISTENT ΜΟΝΤΕΛΑ

    static Model InitModel = ModelFactory.createDefaultModel();

    public static void main (String[] arguments){
        initializeCommonModelPart();
        createPersistentModel("SQL");
        createPersistentModel("DICOM");
        createPersistentModel("HL7");
    }

    private static void initializeCommonModelPart (){

        //ΕΠΙΛΕΧΘΗΚΑΝ 3 ΟΝΤΟΛΟΓΙΕΣ ΑΠΟ ΤΙΣ ΟΠΟΙΕΣ ΠΡΟΕΡΧΟΝΤΑΙ ΤΑ ΛΕΞΙΛΟΓΙΑ
        ΟΛΩΝ ΤΩΝ ΙΑΤΡΩΝ
        //ΤΟΠΟΘΕΤΟΥΝΤΑΙ ΟΛΕΣ ΣΕ ΕΝΑ ΚΟΙΝΟ ΜΟΝΤΕΛΟ (InitModel) ΤΟ ΟΠΟΙΟ ΘΑ
        ΠΡΟΣΤΕΘΕΙ
        //ΣΕ ΚΑΘΕΝΑ ΑΠΟ ΤΑ ΕΠΙΜΕΡΟΥΣ PERSISTENT MODELS

        //ΑΡΧΙΚΑ ΟΡΙΖΟΝΤΑΙ ΤΑ ΠΡΟΘΕΜΑΤΑ (PREFIXES)
        InitModel.setNsPrefix("ncit", BasicFunctionsAndVariables.NCIThesaurusOntologyURI);
        InitModel.setNsPrefix("mged", BasicFunctionsAndVariables.MGEDOntologyURI);
        InitModel.setNsPrefix("foaf", BasicFunctionsAndVariables.FOAFOntologyURI);

        //ΣΤΗ ΣΥΝΕΧΕΙΑ ΦΟΡΤΩΝΟΝΤΑΙ ΟΙ ΟΝΤΟΛΟΓΙΕΣ ΣΤΟ ΜΟΝΤΕΛΟ
        InitModel.read("file:.\\ONTOLOGIES\\NCItPart.owl", "RDF/XML");
        InitModel.read("file:.\\ONTOLOGIES\\MGEDOntology_v1.3.1.1.owl", "RDF/XML");
        InitModel.read("file:.\\ONTOLOGIES\\foaf.rdf", "RDF/XML");
        System.out.println("Common Model Created");
    }
```

```
    public static void createPersistentModel(String ModelKeyWord) {

        //ΔΗΜΙΟΥΡΓΙΑ ΕΝΟΣ PERSISTENT MODEL

        //ΦΟΡΤΩΣΗ ΟΔΗΓΟΥ
        BasicFunctionsAndVariables.loadDriver();

        //ΣΥΝΔΕΣΗ ΣΤΗ ΒΑΣΗ ΠΟΥ ΠΕΡΙΕΧΕΙ ΤΟ ΜΟΝΤΕΛΟ
        IDBConnection Connection = BasicFunctionsAndVariables.connectToRDFBase
            ("jdbc:mysql://localhost/"+ModelKeyWord+"_rdf_model");

        //ΔΗΜΙΟΥΡΓΙΑ ΜΟΝΤΕΛΟΥ
        Model RDBModel = BasicFunctionsAndVariables.CreateModel(Connection,
        ModelKeyWord.toUpperCase()+"_Model");

        //ΠΡΟΣΘΗΚΗ ΤΟΥ ΚΟΙΝΟΥ ΜΟΝΤΕΛΟΥ
        RDBModel.add(InitModel);

        //ΑΠΟΣΥΝΔΕΣΗ ΑΠΟ ΤΗ ΒΑΣΗ
        BasicFunctionsAndVariables.disconnectFromRDFBase(Connection);

        System.out.println(ModelKeyWord+" Persistent Model Created");
        System.out.println(new Date());
    }
}
```

2. Πακέτο core.gui, κλάση StartUpFrame.java

```
package core.gui;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextArea;

import user.gui.UserFrame;
import core.handlers.UpdateModels;
import doctor1.gui.SQLFrame;
import doctor2.gui.DICOMFrame;
import doctor3.gui.HL7Frame;

public class StartUpFrame extends JFrame implements ActionListener{

    //ΚΛΑΣΗ ΑΡΧΙΚΟΥ GUI ΕΦΑΡΜΟΓΗΣ
```

```
private static final long serialVersionUID = 1L;

JTextArea Welcome = new JTextArea
    ("Καλως ορίσατε στο IntegraHEALTH 1.0, ένα πρόγραμμα διαχείρισης και ολοκλήρωσης
    δεδομένων ασθενών. " +
    "Παρακαλούμε επιλέξτε το πρότυπο αποθήκευσης των δεδομένων σας ή εισέλθετε ως
    χρήστης για να υποβάλετε " +
    "τις ερωτήσεις σας.", 2, 50);

JButton SQL = new JButton("Σχεσιακή Βάση Δεδομένων");
JButton DICOM = new JButton("Βάση Δεδομένων Αρχείων DICOM");
JButton HL7 = new JButton("Βάση Δεδομένων Αρχείων Μηνυμάτων HL7");
JButton USER = new JButton("Είσοδος ως Χρήστης/Υποβολή Ερωτημάτων");

public static void main (String[] arguments){
    new StartUpFrame();
}

private StartUpFrame(){

    //ΚΑΤΑΣΚΕΥΗ GUI
    super ("IntegraHealth 1.0 by Nikos Christodoulou - Καλώς Ορίσατε");
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    setBounds((screenSize.width-600)/2, (screenSize.height-150)/2, 600, 150);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JPanel WelcomePane = new JPanel();
    JPanel CommandPane = new JPanel();

    Welcome.setBackground(null);
    Welcome.setLineWrap(true);
    Welcome.setWrapStyleWord(true);
    WelcomePane.add(Welcome);

    FlowLayout flow = new FlowLayout();
    CommandPane.setLayout(flow);
    CommandPane.add(SQL);
    CommandPane.add(DICOM);
    CommandPane.add(HL7);
    CommandPane.add(USER);

    SQL.addActionListener(this);
    DICOM.addActionListener(this);
    HL7.addActionListener(this);
    USER.addActionListener(this);

    BorderLayout border = new BorderLayout();
    setLayout(border);
    add(WelcomePane, BorderLayout.NORTH);
    add(CommandPane, BorderLayout.CENTER);
    setVisible(true);
}
```

```
    public void actionPerformed(ActionEvent evt) {

        //ΑΝΑΛΟΓΑ ΜΕ ΤΟ ΚΟΥΜΠΙ ΠΟΥ ΘΑ ΠΑΤΗΘΕΙ
        //ΕΜΦΑΝΙΖΕΤΑΙ ΤΟ ΚΑΤΑΛΛΗΛΟ GUI

        Object source = evt.getSource();

        //1ΟΣ ΙΑΤΡΟΣ (ΑΠΛΗ ΣΧΕΣΙΑΚΗ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ)
        if (source==SQL){
            setVisible(false);
            SQLFrame sql = new SQLFrame();
            sql.setVisible(true);
        }

        //2ΟΣ ΙΑΤΡΟΣ (ΠΡΟΤΥΠΟ DICOM)
        if (source==DICOM){
            setVisible(false);
            DICOMFrame dicom = new DICOMFrame();
            dicom.setVisible(true);
        }

        //3ΟΣ ΙΑΤΡΟΣ (ΠΡΟΤΥΠΟ HL7)
        if (source==HL7){
            setVisible(false);
            HL7Frame hl7 = new HL7Frame();
            hl7.setVisible(true);
        }

        //ΧΡΗΣΤΗΣ-ΔΙΑΧΕΙΡΙΣΤΗΣ
        if (source==USER){
            setVisible(false);

            //ΕΝΗΜΕΡΩΣΗ PERSISTENT ΜΟΝΤΕΛΩΝ ΜΕ ΤΑ ΝΕΑ ΔΕΔΟΜΕΝΑ
            UpdateModels.updateSQLModel();
            UpdateModels.updateDicomModel();
            UpdateModels.updateHl7Model();

            //ΚΑΤΟΠΙΝ ΑΝΟΙΓΕΙ ΤΟ ΠΑΡΑΘΥΡΟ ΕΡΩΤΗΜΑΤΩΝ
            UserFrame user = new UserFrame();
            user.setVisible(true);
        }
    }
}
```

3. Πακέτο core.handlers

3.1 BasicFunctionsAndVariables.java

```
package core.handlers;

import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.SQLException;

import javax.swing.JOptionPane;

import com.hp.hpl.jena.db.DBConnection;
import com.hp.hpl.jena.db.IDBConnection;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.rdf.model.ModelMaker;

public class BasicFunctionsAndVariables {

    //ΚΛΑΣΗ ΟΡΙΣΜΟΥ ΒΑΣΙΚΩΝ ΜΕΘΟΔΩΝ ΚΑΙ ΜΕΤΑΒΛΗΤΩΝ ΠΟΥ ΘΑ ΧΡΕΙΑΣΤΟΥΝ
    ΣΕ ΟΛΟ ΤΟ ΠΡΟΓΡΑΜΜΑ
    //URIs ΟΝΤΟΛΟΓΙΩΝ
    public static String NCIThesaurusOntologyURI =
        "http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#";
    public static String MGEDOntologyURI =
        "http://mged.sourceforge.net/ontologies/MGEDOntology.owl#";
    public static String FOAFOntologyURI = "http://xmlns.com/foaf/0.1/";

    //ΦΟΡΤΩΣΗ ΚΛΑΣΗΣ ΟΔΗΓΟΥ MYSQL
    public static void loadDriver(){
        try {
            Class.forName("com.mysql.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            JOptionPane.showMessageDialog(null,
                "Η κλάση - οδηγός δεν μπορούσε να φορτωθεί"+'\n'+e.toString(),
                "Σφάλμα Φόρτωσης Οδηγού",
                JOptionPane.ERROR_MESSAGE);
        }
    }

    //ΣΥΝΔΕΣΗ ΣΕ ΒΑΣΗ ΠΟΥ ΠΕΡΙΕΧΕΙ PERSISTENT ΜΟΝΤΕΛΟ
    public static IDBConnection connectToRDFBase(String InDbURL){
        try {
            IDBConnection Conn = new DBConnection(InDbURL, "root", "", "MySQL");
            return Conn;
        }catch (Exception e) {
            JOptionPane.showMessageDialog(null,
                "Αδύνατη η σύνδεση με τη βάση"+'\n'+InDbURL+'\n'+e.toString(),
                "Σφάλμα Σύνδεσης σε Βάση Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
            return null;
        }
    }

    //ΔΗΜΙΟΥΡΓΙΑ PERISISTENT ΜΟΝΤΕΛΟΥ
    public static Model CreateModel (IDBConnection Conn, String ModelName){
        ModelMaker maker = ModelFactory.createModelRDBMaker(Conn);
        Model defModel = maker.createModel(ModelName, false);
```

```
        maker.close();
        return defModel;
    }

    //ΑΝΟΙΓΜΑ ΥΠΑΡΧΟΝΤΟΣ PERISISTENT ΜΟΝΤΕΛΟΥ
    public static Model OpenModel (IDBConnection Conn, String ModelName){
        ModelMaker maker = ModelFactory.createModelRDBMaker(Conn);
        Model defModel = maker.openModel(ModelName, false);
        maker.close();
        return defModel;
    }

    //ΑΠΟΣΥΝΔΕΣΗ ΑΠΟ ΒΑΣΗ ΠΟΥ ΠΕΡΙΕΧΕΙ PERSISTENT ΜΟΝΤΕΛΟ
    public static void disconnectFromRDFBase (IDBConnection Conn){
        try{
            Conn.close();
        }catch (SQLException e){
            JOptionPane.showMessageDialog(null,
                "Αδύνατη η αποσύνδεση από τη βάση"+'\n'+e.toString(),
                "Σφάλμα Αποσύνδεσης από Βάση Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
    }

    //ΣΥΝΔΕΣΗ ΣΕ ΒΑΣΗ ΠΟΥ ΠΕΡΙΕΧΕΙ ΙΑΤΡΙΚΑ ΔΕΔΟΜΕΝΑ
    public static Connection connectToBase(int DoctorCode){

        String URL = "jdbc:mysql://localhost/doctor"+DoctorCode;
        try {
            Connection Conn = DriverManager.getConnection(URL, "root", "");
            return Conn;
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null,
                "Αδύνατη η σύνδεση με τη βάση"+'\n'+URL+'\n'+e.toString(),
                "Σφάλμα Σύνδεσης σε Βάση Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
            return null;
        }
    }

    //ΑΠΟΣΥΝΔΕΣΗ ΑΠΟ ΒΑΣΗ ΠΟΥ ΠΕΡΙΕΧΕΙ ΙΑΤΡΙΚΑ ΔΕΔΟΜΕΝΑ
    public static void disconnectFromBase (Connection Conn){
        try{
            Conn.close();
        }catch (SQLException e){
            JOptionPane.showMessageDialog(null,
                "Αδύνατη η αποσύνδεση από τη βάση"+'\n'+e.toString(),
                "Σφάλμα Αποσύνδεσης από Βάση Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

3.2 DICOMData2RDF.java

```
package core.handlers;

import com.hp.hpl.jena.datatypes.xsd.XSDDatatype;
import com.hp.hpl.jena.db.IDBConnection;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.RDFNode;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.rdf.model.SimpleSelector;
import com.hp.hpl.jena.vocabulary.RDF;
import com.hp.hpl.jena.vocabulary.RDFS;

import doctor2.handlers.DICOMTuple;

public class DICOMData2RDF {

    //ΚΛΑΣΗ ΠΡΟΣΘΑΦΑΙΡΕΣΗΣ ΔΕΔΟΜΕΝΩΝ RDF ΣΤΟ ΜΟΝΤΕΛΟ ΤΟΥ 2ΟΥ ΙΑΤΡΟΥ

    public static void dicomTuple2RdfNode (DICOMTuple InDicomTuple){

        //ΔΗΜΙΟΥΡΓΙΑ ΝΕΟΥ ΚΟΜΒΟΥ

        //ΟΡΙΣΜΟΣ ΤΟΥ URI ΤΟΥ ΚΟΜΒΟΥ
        String DBURI = "http://example.org/doctor2/"+InDicomTuple.FileName;

        //ΦΟΡΤΩΣΗ ΣΧΕΤΙΚΟΥ PERSISTENT ΜΟΝΤΕΛΟΥ ΑΠΟ ΤΗ ΒΑΣΗ
        IDBConnection DICOMConnection = BasicFunctionsAndVariables.connectToRDFBase
            ("jdbc:mysql://localhost/dicom_rdf_model");
        Model DICOM_Model = BasicFunctionsAndVariables.OpenModel
            (DICOMConnection, "DICOM_Model");

        //ΔΗΜΙΟΥΡΓΙΑ ΤΟΥ ΚΟΜΒΟΥ
        Resource MainNode = DICOM_Model.createResource(DBURI);

        //ΟΡΙΖΕΤΑΙ ΤΟ CLASS TYPE ΤΟΥ ΚΟΜΒΟΥ (rdf:type)
        MainNode.addProperty(RDF.type, DICOM_Model.getResource(
            BasicFunctionsAndVariables.NCIThesaurusOntologyURI+"Health_Care_Visit"))

        //ΜΕ ΜΙΑ ΣΕΙΡΑ ΑΠΟ ΣΧΕΤΙΚΕΣ ΕΝΤΟΛΕΣ ΠΡΟΣΤΙΘΕΝΤΑΙ ΣΤΟΝ ΚΟΜΒΟ
        //ΟΛΑ ΤΑ ΕΠΙΜΕΡΟΥΣ ΣΤΟΙΧΕΙΑ ΥΠΟ ΤΗ ΜΟΡΦΗ ΙΔΙΟΤΗΤΩΝ (PROPERTIES)

        .addProperty(DICOM_Model.getProperty(
            BasicFunctionsAndVariables.MGEDOntologyURI+"has_first_name"),
            InDicomTuple.FName,

            //ΣΕ ΚΑΘΕ ΙΔΙΟΤΗΤΑ ΟΡΙΖΕΤΑΙ ΚΑΙ ΑΠΟ ΕΝΑΣ XML ΤΥΠΟΣ ΓΙΑ ΤΑ
            ΔΕΔΟΜΕΝΑ ΠΟΥ ΕΙΣΑΓΟΝΤΑΙ
            //ΔΗΜΙΟΥΡΓΩΝΤΑΣ ΤΥΠΟΠΟΙΗΜΕΝΑ ΛΕΚΤΙΚΑ
            XSDDatatype.XSDstring)
```

```
.addProperty(DICOM_Model.getProperty(
    BasicFunctionsAndVariables.MGEDOntologyURI+"has_last_name"),
    InDicomTuple.LName,  XSDDatatype.XSDstring)

.addProperty(DICOM_Model.getProperty(
    BasicFunctionsAndVariables.FOAFOntologyURI+"givenName"),
    InDicomTuple.FatherName,  XSDDatatype.XSDstring)

//Η ΣΥΓΚΕΚΡΙΜΕΝΗ ΙΔΙΟΤΗΤΑ ΟΡΙΖΕΤΑΙ ΑΠΟ ΤΗΝ ΟΝΤΟΛΟΓΙΑ ΠΟΥ ΤΗΝ
ΠΕΡΙΕΧΕΙ ΩΣ
//OBJECTPROPERTY ΕΠΟΜΕΝΩΣ ΔΗΜΙΟΥΡΓΟΥΜΕ ΕΝΑ ΝΕΟ NODE (ΟΥΣΙΑΣΤΙΚΑ
BLANK NODE)
//ΑΛΛΑ ΟΝΟΜΑΤΙΖΕΤΑΙ ΓΙΑ ΕΥΚΟΛΙΑ) ΚΑΙ ΣΕ ΑΥΤΟΝ ΣΥΝΔΕΟΝΤΑΙ ΤΑ
ΑΠΑΙΤΟΥΜΕΝΑ
//PROPERTIES. ΤΟ ΙΔΙΟ ΓΙΝΕΤΑΙ ΣΕ ΚΑΘΕ ΕΠΟΜΕΝΟ OBJECTPROPERTY
.addProperty(DICOM_Model.getProperty(
    BasicFunctionsAndVariables.MGEDOntologyURI+"has_nodes"),
    (DICOM_Model.createResource(DBURI+"/Birthday")
    .addProperty(RDF.type, DICOM_Model.getResource(
        BasicFunctionsAndVariables.NCIThesaurusOntologyURI+"Date_of_Birth"))
    .addProperty(DICOM_Model.getProperty(
        BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
        InDicomTuple.Birthday,  XSDDatatype.XSDdate)))

.addProperty(DICOM_Model.getProperty(
    BasicFunctionsAndVariables.MGEDOntologyURI+"unique_identifier"),
    InDicomTuple.Amka,  XSDDatatype.XSDstring)

.addProperty(DICOM_Model.getProperty(
    BasicFunctionsAndVariables.MGEDOntologyURI+"has_nodes"),
    (DICOM_Model.createResource(DBURI+"/SecurityOrg")
    .addProperty(RDF.type, DICOM_Model.getResource(
        BasicFunctionsAndVariables.NCIThesaurusOntologyURI+
        "Primary_Healthcare_Payer"))
    .addProperty(DICOM_Model.getProperty(
        BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
        InDicomTuple.SecurityOrganization, XSDDatatype.XSDstring)))

.addProperty(DICOM_Model.getProperty(
    BasicFunctionsAndVariables.MGEDOntologyURI+"has_address"),
    InDicomTuple.Address,  XSDDatatype.XSDstring)

.addProperty(DICOM_Model.getProperty(
    BasicFunctionsAndVariables.MGEDOntologyURI+"has_phone"),
    InDicomTuple.Phone,  XSDDatatype.XSDstring)

.addProperty(DICOM_Model.getProperty(
    BasicFunctionsAndVariables.MGEDOntologyURI+"has_nodes"),
    (DICOM_Model.createResource(DBURI+"/ExaminationDate")
    .addProperty(RDF.type, DICOM_Model.getResource(
        BasicFunctionsAndVariables.NCIThesaurusOntologyURI+
        "Physical_Examination_Date"))
```

```
        .addProperty(DICOM_Model.getProperty(
            BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
            InDicomTuple.ExaminationDate, XSDDatatype.XSDdate)))

    .addProperty(DICOM_Model.getProperty(
        BasicFunctionsAndVariables.MGEDOntologyURI+"has_disease_state"),
        (DICOM_Model.createResource(DBURI+"/Diagnosis")
        .addProperty(RDF.type, DICOM_Model.getResource(
            BasicFunctionsAndVariables.NCIThesaurusOntologyURI+"Diagnosis"))
        .addProperty(DICOM_Model.getProperty(
            BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
            InDicomTuple.Disease,  XSDDatatype.XSDstring)))

    .addProperty(DICOM_Model.getProperty(
        BasicFunctionsAndVariables.MGEDOntologyURI+"has_clinical_finding"),
        (DICOM_Model.createResource(DBURI+"/Findings")
        .addProperty(RDF.type, DICOM_Model.getResource(
            BasicFunctionsAndVariables.NCIThesaurusOntologyURI+"Finding"))
        .addProperty(DICOM_Model.getProperty(
            BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
            InDicomTuple.Remarks,  XSDDatatype.XSDstring)))

    .addProperty(DICOM_Model.getProperty(
        BasicFunctionsAndVariables.MGEDOntologyURI+"has_nodes"),
        (DICOM_Model.createResource(DBURI+"/Payment")
        .addProperty(RDF.type, DICOM_Model.getResource(
            BasicFunctionsAndVariables.NCIThesaurusOntologyURI+"Payment"))
        .addProperty(DICOM_Model.getProperty(
            BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
            Float.toString(InDicomTuple.Sum), XSDDatatype.XSDfloat)))

    .addProperty(DICOM_Model.getProperty(
        BasicFunctionsAndVariables.MGEDOntologyURI+"has_nodes"),
        (DICOM_Model.createResource(DBURI+"/FileName")
        .addProperty(RDF.type, DICOM_Model.getResource(
            BasicFunctionsAndVariables.NCIThesaurusOntologyURI+"External_Filename"))
        .addProperty(DICOM_Model.getProperty(
            BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
            InDicomTuple.FileName,  XSDDatatype.XSDstring)))

    .addProperty(DICOM_Model.getProperty(
        BasicFunctionsAndVariables.MGEDOntologyURI+"has_nodes"),
        (DICOM_Model.createResource(DBURI+"/ExaminationMethod")
        .addProperty(RDF.type, DICOM_Model.getResource(
            BasicFunctionsAndVariables.NCIThesaurusOntologyURI+"Examination_Method"))
        .addProperty(DICOM_Model.getProperty(
            BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
            InDicomTuple.ExaminationType,  XSDDatatype.XSDstring)))

    .addProperty(DICOM_Model.getProperty(
        BasicFunctionsAndVariables.MGEDOntologyURI+"has_measurement"),
        (DICOM_Model.createResource(DBURI+"/BloodFlow")
```

```
            .addProperty(RDF.type, DICOM_Model.getResource(
                BasicFunctionsAndVariables.NCIThesaurusOntologyURI+"Blood_Flow_Rate"))
            .addProperty(DICOM_Model.getProperty(
                BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
                InDicomTuple.BloodFlow.toString(), XSDDatatype.XSDfloat)))

        .addProperty(DICOM_Model.getProperty(
            BasicFunctionsAndVariables.MGEDOntologyURI+"has_measurement"),
            (DICOM_Model.createResource(DBURI+"/BodyFat")
            .addProperty(RDF.type, DICOM_Model.getResource(
                BasicFunctionsAndVariables.NCIThesaurusOntologyURI+"Adipose_Tissue"))
            .addProperty(DICOM_Model.getProperty(
                BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
                InDicomTuple.BodyFat.toString(), XSDDatatype.XSDfloat)))

        .addProperty(DICOM_Model.getProperty(
            BasicFunctionsAndVariables.MGEDOntologyURI+"has_measurement"),
            (DICOM_Model.createResource(DBURI+"/BoneDensity")
            .addProperty(RDF.type, DICOM_Model.getResource(
                BasicFunctionsAndVariables.NCIThesaurusOntologyURI+
                "Bone_Mineral_Density_Test"))
            .addProperty(DICOM_Model.getProperty(
                BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
                InDicomTuple.BoneDensity.toString(), XSDDatatype.XSDfloat)));

        BasicFunctionsAndVariables.disconnectFromRDFBase(DICOMConnection);
    }

    public static void deleteRDFNode(String InFileName){

        //ΔΙΑΓΡΑΦΗ RDF ΚΟΜΒΟΥ

        String DBURI = " http://example.org/doctor2/"+InFileName;

        //ΕΙΣΑΓΩΓΗ ΣΤΟ PERSISTENT ΜΟΝΤΕΛΟ
        BasicFunctionsAndVariables.loadDriver();
        IDBConnection DICOMConnection = BasicFunctionsAndVariables.connectToRDFBase(
            "jdbc:mysql://localhost/dicom_rdf_model");
        Model DICOM_Model = BasicFunctionsAndVariables.OpenModel(
            DICOMConnection, "DICOM_Model");

        //ΠΡΩΤΑ ΕΛΕΓΧΕΤΑΙ Η ΥΠΑΡΞΗ ΤΟΥ ΠΡΟΣ ΔΙΑΓΡΑΦΗ ΚΟΜΒΟΥ
        //ΑΝ ΕΠΙΒΕΒΑΙΩΘΕΙ ΔΙΑΓΡΑΦΕΤΑΙ Ο ΚΟΜΒΟΣ ΚΑΙ ΤΟ ΔΕΝΤΡΟ ΠΟΥ ΤΟΝ ΕΧΕΙ
        ΩΣ ΡΙΖΑ
        if (DICOM_Model.contains(DICOM_Model.getResource(DBURI), null, (RDFNode) null)){
            DICOM_Model.remove(DICOM_Model.listStatements(new SimpleSelector (
                DICOM_Model.getResource(DBURI), null, (RDFNode) null)));
            DICOM_Model.remove(DICOM_Model.listStatements(new SimpleSelector (
                DICOM_Model.getResource(InFileName+"/Diagnosis"), null, (RDFNode) null)));
            DICOM_Model.remove(DICOM_Model.listStatements(new SimpleSelector (
                DICOM_Model.getResource(InFileName+"/Findings"), null, (RDFNode) null)));
```

```
            DICOM_Model.remove(DICOM_Model.listStatements(new SimpleSelector (
                DICOM_Model.getResource(InFileName+"/Payment"), null, (RDFNode) null)));
            DICOM_Model.remove(DICOM_Model.listStatements(new SimpleSelector (
                DICOM_Model.getResource(InFileName+"/ExaminationDate"), null,
                (RDFNode) null)));
            DICOM_Model.remove(DICOM_Model.listStatements(new SimpleSelector (
                DICOM_Model.getResource(InFileName+"/FileName"), null, (RDFNode) null)));
            DICOM_Model.remove(DICOM_Model.listStatements(new SimpleSelector (
                DICOM_Model.getResource(InFileName+"/ExaminationMethod"), null,
                (RDFNode) null)));
            DICOM_Model.remove(DICOM_Model.listStatements(new SimpleSelector (
                DICOM_Model.getResource(InFileName+"/BloodFlow"), null, (RDFNode) null)));
            DICOM_Model.remove(DICOM_Model.listStatements(new SimpleSelector (
                DICOM_Model.getResource(InFileName+"/BodyFat"), null, (RDFNode) null)));
            DICOM_Model.remove(DICOM_Model.listStatements(new SimpleSelector (
                DICOM_Model.getResource(InFileName+"/BoneDensity"), null,
                (RDFNode) null)));
            DICOM_Model.remove(DICOM_Model.listStatements(new SimpleSelector (
                DICOM_Model.getResource(InFileName+"/SecurityOrg"), null, (RDFNode) null)));
        }
        BasicFunctionsAndVariables.disconnectFromRDFBase(DICOMConnection);
    }
}
```

3.3 HL7Data2RDF.java

```
package core.handlers;

import com.hp.hpl.jena.datatypes.xsd.XSDDatatype;
import com.hp.hpl.jena.db.IDBConnection;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.RDFNode;
import com.hp.hpl.jena.rdf.model.Resource;
import com.hp.hpl.jena.rdf.model.SimpleSelector;
import com.hp.hpl.jena.vocabulary.RDF;

import doctor3.handlers.HL7Tuple;

public class HL7Data2RDF {

    //ΚΛΑΣΗ ΠΡΟΣΘΑΦΑΙΡΕΣΗΣ ΔΕΔΟΜΕΝΩΝ RDF ΣΤΟ ΜΟΝΤΕΛΟ ΤΟΥ 3ΟΥ ΙΑΤΡΟΥ
    //ΤΟ ΣΚΕΠΤΙΚΟ ΚΑΙ Η ΔΙΑΔΙΚΑΣΙΑ ΕΙΝΑΙ ΠΑΝΟΜΟΙΟΤΥΠΗ ΜΕ ΑΥΤΗ ΤΗΣ ΚΛΑΣΗΣ
    DICOMDATA2RDF

    public static void hl7Tuple2RdfNode (HL7Tuple InHL7Tuple){
        String DBURI = " http://example.org/doctor3/"+InHL7Tuple.FileName;

        BasicFunctionsAndVariables.loadDriver();
        IDBConnection HL7Connection = BasicFunctionsAndVariables.connectToRDFBase(
        "jdbc:mysql://localhost/hl7_rdf_model");
        Model HL7_Model = BasicFunctionsAndVariables.CreateModel(
    HL7Connection, "HL7_Model");
```

```
XSDDatatype type = null;

Resource MainNode = HL7_Model.createResource(DBURI);

MainNode.addProperty(RDF.type, HL7_Model.getResource(
    BasicFunctionsAndVariables.NCIThesaurusOntologyURI+"Health_Care_Visit"))

.addProperty(HL7_Model.getProperty(
    BasicFunctionsAndVariables.MGEDOntologyURI+"has_first_name"),
    InHL7Tuple.FName,  XSDDatatype.XSDstring)

.addProperty(HL7_Model.getProperty(
    BasicFunctionsAndVariables.MGEDOntologyURI+"has_last_name"),
    InHL7Tuple.LName,  XSDDatatype.XSDstring)

.addProperty(HL7_Model.getProperty(
    BasicFunctionsAndVariables.FOAFOntologyURI+"givenName"),
    InHL7Tuple.FatherName,  XSDDatatype.XSDstring)

.addProperty(HL7_Model.getProperty(
    BasicFunctionsAndVariables.MGEDOntologyURI+"has_nodes"),
    (HL7_Model.createResource(DBURI+"/Birthday")
    .addProperty(RDF.type, HL7_Model.getResource(
        BasicFunctionsAndVariables.NCIThesaurusOntologyURI+"Date_of_Birth"))
    .addProperty(HL7_Model.getProperty(
        BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
        InHL7Tuple.Birthday,  XSDDatatype.XSDdate)))

.addProperty(HL7_Model.getProperty(
    BasicFunctionsAndVariables.MGEDOntologyURI+"unique_identifier"),
    InHL7Tuple.Amka,  XSDDatatype.XSDstring)

.addProperty(HL7_Model.getProperty(
    BasicFunctionsAndVariables.MGEDOntologyURI+"has_address"),
    InHL7Tuple.Address, XSDDatatype.XSDstring)

.addProperty(HL7_Model.getProperty(
    BasicFunctionsAndVariables.MGEDOntologyURI+"has_phone"),
    InHL7Tuple.Phone, XSDDatatype.XSDstring)

.addProperty(HL7_Model.getProperty(
    BasicFunctionsAndVariables.FOAFOntologyURI+"gender"),
    InHL7Tuple.Gender, XSDDatatype.XSDstring)

.addProperty(HL7_Model.getProperty(
    BasicFunctionsAndVariables.MGEDOntologyURI+"has_nodes"),
    (HL7_Model.createResource(DBURI+"/ExaminationDate")
    .addProperty(RDF.type, HL7_Model.getResource(
        BasicFunctionsAndVariables.NCIThesaurusOntologyURI+
        "Physical_Examination_Date"))
    .addProperty(HL7_Model.getProperty(
```

```
            BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
            InHL7Tuple.ExaminationDate, XSDDatatype.XSDdate)))

.addProperty(HL7_Model.getProperty(
      BasicFunctionsAndVariables.MGEDOntologyURI+"has_nodes"),
      (HL7_Model.createResource(DBURI+"/Payment")
      .addProperty(RDF.type, HL7_Model.getResource(
            BasicFunctionsAndVariables.NCIThesaurusOntologyURI+"Payment"))
      .addProperty(HL7_Model.getProperty(
            BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
            Float.toString(InHL7Tuple.Sum), XSDDatatype.XSDfloat)))

.addProperty(HL7_Model.getProperty(
      BasicFunctionsAndVariables.MGEDOntologyURI+"has_nodes"),
      (HL7_Model.createResource(DBURI+"/FileName")
      .addProperty(RDF.type, HL7_Model.getResource(
            BasicFunctionsAndVariables.NCIThesaurusOntologyURI+"External_Filename"))
      .addProperty(HL7_Model.getProperty(
            BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
            InHL7Tuple.FileName,  XSDDatatype.XSDstring)))

.addProperty(HL7_Model.getProperty(
      BasicFunctionsAndVariables.MGEDOntologyURI+"has_ID"),
      InHL7Tuple.PatientID,  XSDDatatype.XSDstring);

if (InHL7Tuple.TestData[0]!=Float.parseFloat("-0")){
      type = XSDDatatype.XSDfloat;
}
else{
      type = XSDDatatype.XSDstring;
}
MainNode.addProperty(HL7_Model.getProperty(
      BasicFunctionsAndVariables.MGEDOntologyURI+"has_measurement"),
      (HL7_Model.createResource(DBURI+"/Leukocytes")
      .addProperty(RDF.type, HL7_Model.getProperty(
            BasicFunctionsAndVariables.NCIThesaurusOntologyURI+"Leukocyte_Count"))
      .addProperty(HL7_Model.getProperty(
            BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
            InHL7Tuple.TestData[0].toString(), type)));

if (InHL7Tuple.TestData[1]!=Float.parseFloat("-0")){
      type = XSDDatatype.XSDfloat;
}
else{
      type = XSDDatatype.XSDstring;
}
MainNode.addProperty(HL7_Model.getProperty(
      BasicFunctionsAndVariables.MGEDOntologyURI+"has_measurement"),
      (HL7_Model.createResource(DBURI+"/Erythrocytes")
      .addProperty(RDF.type, HL7_Model.getProperty(
            BasicFunctionsAndVariables.NCIThesaurusOntologyURI+"Erythrocyte_Count"))
      .addProperty(HL7_Model.getProperty(
```

```
            BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
            InHL7Tuple.TestData[1].toString(), type)));

if (InHL7Tuple.TestData[2]!=Float.parseFloat("-0")){
    type = XSDDatatype.XSDfloat;
}
else{
    type = XSDDatatype.XSDstring;
}
MainNode.addProperty(HL7_Model.getProperty(
    BasicFunctionsAndVariables.MGEDOntologyURI+"has_measurement"),
    (HL7_Model.createResource(DBURI+"/Hemoglobin")
    .addProperty(RDF.type, HL7_Model.getProperty(
        BasicFunctionsAndVariables.NCIThesaurusOntologyURI+
        "Hemoglobin_Measurement"))
    .addProperty(HL7_Model.getProperty(
        BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
        InHL7Tuple.TestData[2].toString(), type)));

if (InHL7Tuple.TestData[3]!=Float.parseFloat("-0")){
    type = XSDDatatype.XSDfloat;
}
else{
    type = XSDDatatype.XSDstring;
}
MainNode.addProperty(HL7_Model.getProperty(
    BasicFunctionsAndVariables.MGEDOntologyURI+"has_measurement"),
    (HL7_Model.createResource(DBURI+"/Hematocrit")
    .addProperty(RDF.type, HL7_Model.getProperty(
        BasicFunctionsAndVariables.NCIThesaurusOntologyURI+"Hematocrit"))
    .addProperty(HL7_Model.getProperty(
        BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
        InHL7Tuple.TestData[3].toString(), type)));

if (InHL7Tuple.TestData[4]!=Float.parseFloat("-0")){
    type = XSDDatatype.XSDfloat;
}
else{
    type = XSDDatatype.XSDstring;
}
MainNode.addProperty(HL7_Model.getProperty(
    BasicFunctionsAndVariables.MGEDOntologyURI+"has_measurement"),
    (HL7_Model.createResource(DBURI+"/Platelets")
    .addProperty(RDF.type, HL7_Model.getProperty(
        BasicFunctionsAndVariables.NCIThesaurusOntologyURI+"Platelet_Count"))
    .addProperty(HL7_Model.getProperty(
        BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
        InHL7Tuple.TestData[4].toString(), type)));

if (InHL7Tuple.TestData[5]!=Float.parseFloat("-0")){
    type = XSDDatatype.XSDfloat;
}
```

```
else{
    type = XSDDatatype.XSDstring;
}
MainNode.addProperty(HL7_Model.getProperty(
    BasicFunctionsAndVariables.MGEDOntologyURI+"has_measurement"),
    (HL7_Model.createResource(DBURI+"/BloodSugar")
    .addProperty(RDF.type, HL7_Model.getProperty(
        BasicFunctionsAndVariables.NCIThesaurusOntologyURI+
        "Fasting_Blood_Sugar_Measurement"))
    .addProperty(HL7_Model.getProperty(
        BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
        InHL7Tuple.TestData[5].toString(), type)));

if (InHL7Tuple.TestData[6]!=Float.parseFloat("-0")){
    type = XSDDatatype.XSDfloat;
}
else{
    type = XSDDatatype.XSDstring;
}
MainNode.addProperty(HL7_Model.getProperty(
    BasicFunctionsAndVariables.MGEDOntologyURI+"has_measurement"),
    (HL7_Model.createResource(DBURI+"/Urea")
    .addProperty(RDF.type, HL7_Model.getProperty(
        BasicFunctionsAndVariables.NCIThesaurusOntologyURI+"Urea_Measurement"))
    .addProperty(HL7_Model.getProperty(
        BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
        InHL7Tuple.TestData[6].toString(), type)));

if (InHL7Tuple.TestData[7]!=Float.parseFloat("-0")){
    type = XSDDatatype.XSDfloat;
}
else{
    type = XSDDatatype.XSDstring;
}
MainNode.addProperty(HL7_Model.getProperty(
    BasicFunctionsAndVariables.MGEDOntologyURI+"has_measurement"),
    (HL7_Model.createResource(DBURI+"/Creatinine")
    .addProperty(RDF.type, HL7_Model.getProperty(
        BasicFunctionsAndVariables.NCIThesaurusOntologyURI+
        "Creatinine_Measurement"))
    .addProperty(HL7_Model.getProperty(
        BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
        InHL7Tuple.TestData[7].toString(), type)));

if (InHL7Tuple.TestData[8]!=Float.parseFloat("-0")){
    type = XSDDatatype.XSDfloat;
}
else{
    type = XSDDatatype.XSDstring;
}
MainNode.addProperty(HL7_Model.getProperty(
    BasicFunctionsAndVariables.MGEDOntologyURI+"has_measurement"),
```

```
        (HL7_Model.createResource(DBURI+"/UricAcid")
        .addProperty(RDF.type, HL7_Model.getProperty(
            BasicFunctionsAndVariables.NCIThesaurusOntologyURI+
            "Uric_Acid_Crystal_Measurement"))
        .addProperty(HL7_Model.getProperty(
            BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
            InHL7Tuple.TestData[8].toString(), type)));

    if (InHL7Tuple.TestData[9]!=Float.parseFloat("-0")){
        type = XSDDatatype.XSDfloat;
    }
    else{
        type = XSDDatatype.XSDstring;
    }
    MainNode.addProperty(HL7_Model.getProperty(
        BasicFunctionsAndVariables.MGEDOntologyURI+"has_measurement"),
        (HL7_Model.createResource(DBURI+"/Triglyceride")
        .addProperty(RDF.type, HL7_Model.getProperty(
            BasicFunctionsAndVariables.NCIThesaurusOntologyURI+
            "Triglyceride_Measurement"))
        .addProperty(HL7_Model.getProperty(
            BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
            InHL7Tuple.TestData[9].toString(), type)));

    if (InHL7Tuple.TestData[10]!=Float.parseFloat("-0")){
        type = XSDDatatype.XSDfloat;
    }
    else{
        type = XSDDatatype.XSDstring;
    }
    MainNode.addProperty(HL7_Model.getProperty(
        BasicFunctionsAndVariables.MGEDOntologyURI+"has_measurement"),
        (HL7_Model.createResource(DBURI+"/Lipids")
        .addProperty(RDF.type, HL7_Model.getProperty(
            BasicFunctionsAndVariables.NCIThesaurusOntologyURI+"Lipid_Measurement"))
        .addProperty(HL7_Model.getProperty(
            BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
            InHL7Tuple.TestData[10].toString(), type)));

    if (InHL7Tuple.TestData[11]!=Float.parseFloat("-0")){
        type = XSDDatatype.XSDfloat;
    }
    else{
        type = XSDDatatype.XSDstring;
    }
    MainNode.addProperty(HL7_Model.getProperty(
        BasicFunctionsAndVariables.MGEDOntologyURI+"has_measurement"),
        (HL7_Model.createResource(DBURI+"/LDL")
        .addProperty(RDF.type, HL7_Model.getProperty(
            BasicFunctionsAndVariables.NCIThesaurusOntologyURI+
            "Serum_LDL_Cholesterol_Measurement"))
        .addProperty(HL7_Model.getProperty(
```

```
                BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
                InHL7Tuple.TestData[11].toString(), type)));

        if (InHL7Tuple.TestData[12]!=Float.parseFloat("-0")){
            type = XSDDatatype.XSDfloat;
        }
        else{
            type = XSDDatatype.XSDstring;
        }
        MainNode.addProperty(HL7_Model.getProperty(
            BasicFunctionsAndVariables.MGEDOntologyURI+"has_measurement"),
            (HL7_Model.createResource(DBURI+"/HDL")
            .addProperty(RDF.type, HL7_Model.getProperty(
                BasicFunctionsAndVariables.NCIThesaurusOntologyURI+
                "Serum_HDL_Cholesterol_Measurement"))
            .addProperty(HL7_Model.getProperty(
                BasicFunctionsAndVariables.MGEDOntologyURI+"has_value"),
                InHL7Tuple.TestData[12].toString(), type)));

        BasicFunctionsAndVariables.disconnectFromRDFBase(HL7Connection);
    }

    public static void deleteRDFNode(String InFileName){
        String DBURI = " http://example.org/doctor3/"+InFileName;

        BasicFunctionsAndVariables.loadDriver();
        IDBConnection HL7Connection = BasicFunctionsAndVariables.connectToRDFBase(
            "jdbc:mysql://localhost/hl7_rdf_model");
        Model HL7_Model = BasicFunctionsAndVariables.CreateModel(
            HL7Connection, "HL7_Model");

        if (HL7_Model.contains(HL7_Model.getResource(DBURI), null, (RDFNode) null)){
            HL7_Model.remove(HL7_Model.listStatements(new SimpleSelector (
                HL7_Model.getResource(DBURI), null, (RDFNode) null)));
            HL7_Model.remove(HL7_Model.listStatements(new SimpleSelector (
                HL7_Model.getResource(InFileName+"/Birthday"), null, (RDFNode) null)));
            HL7_Model.remove(HL7_Model.listStatements(new SimpleSelector (
                HL7_Model.getResource(InFileName+"/ExaminationDate"), null,
                (RDFNode) null)));
            HL7_Model.remove(HL7_Model.listStatements(new SimpleSelector (
                HL7_Model.getResource(InFileName+"/Payment"), null, (RDFNode) null)));
            HL7_Model.remove(HL7_Model.listStatements(new SimpleSelector (
                HL7_Model.getResource(InFileName+"/FileName"), null, (RDFNode) null)));
            HL7_Model.remove(HL7_Model.listStatements(new SimpleSelector (
                HL7_Model.getResource(InFileName+"/Leukocytes"), null, (RDFNode) null)));
            HL7_Model.remove(HL7_Model.listStatements(new SimpleSelector (
                HL7_Model.getResource(InFileName+"/Erythrocytes"), null, (RDFNode) null)));
            HL7_Model.remove(HL7_Model.listStatements(new SimpleSelector (
                HL7_Model.getResource(InFileName+"/Hemoglobin"), null, (RDFNode) null)));
            HL7_Model.remove(HL7_Model.listStatements(new SimpleSelector (
                HL7_Model.getResource(InFileName+"/Hematocrit"), null, (RDFNode) null)));
```

```
                HL7_Model.remove(HL7_Model.listStatements(new SimpleSelector (
                    HL7_Model.getResource(InFileName+"/Platelets"), null, (RDFNode) null)));
                HL7_Model.remove(HL7_Model.listStatements(new SimpleSelector (
                    HL7_Model.getResource(InFileName+"/BloodSugar"), null, (RDFNode) null)));
                HL7_Model.remove(HL7_Model.listStatements(new SimpleSelector (
                    HL7_Model.getResource(InFileName+"/Urea"), null, (RDFNode) null)));
                HL7_Model.remove(HL7_Model.listStatements(new SimpleSelector (
                    HL7_Model.getResource(InFileName+"/Creatinine"), null, (RDFNode) null)));
                HL7_Model.remove(HL7_Model.listStatements(new SimpleSelector (
                    HL7_Model.getResource(InFileName+"/UricAcid"), null, (RDFNode) null)));
                HL7_Model.remove(HL7_Model.listStatements(new SimpleSelector (
                    HL7_Model.getResource(InFileName+"/Triglyceride"), null, (RDFNode) null)));
                HL7_Model.remove(HL7_Model.listStatements(new SimpleSelector (
                    HL7_Model.getResource(InFileName+"/Lipids"), null, (RDFNode) null)));
                HL7_Model.remove(HL7_Model.listStatements(new SimpleSelector (
                    HL7_Model.getResource(InFileName+"/LDL"), null, (RDFNode) null)));
                HL7_Model.remove(HL7_Model.listStatements(new SimpleSelector (
                    HL7_Model.getResource(InFileName+"/HDL"), null, (RDFNode) null)));
            }
            BasicFunctionsAndVariables.disconnectFromRDFBase(HL7Connection);
        }
}
```

3.4 Reasoning.java

```
import java.util.List;

import com.hp.hpl.jena.rdf.model.InfModel;
import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.ModelFactory;
import com.hp.hpl.jena.reasoner.ValidityReport;
import com.hp.hpl.jena.reasoner.ValidityReport.Report;
import com.hp.hpl.jena.reasoner.rulesys.GenericRuleReasoner;
import com.hp.hpl.jena.reasoner.rulesys.Rule;

public class Reasoning {

    //ΚΛΑΣΗ ΕΦΑΡΜΟΓΗΣ ΣΥΛΛΟΓΙΣΤΙΚΗΣ (REASONING) ΣΤΑ RDF ΜΟΝΤΕΛΑ

    public static InfModel NewInferenceModel (Model InModel, String RulesFileName){

        //ΟΡΙΣΜΟΣ ΚΑΝΟΝΩΝ ΑΠΟ ΣΧΕΤΙΚΑ ΑΡΧΕΙΑ
        List<Rule> rules = Rule.rulesFromURL(RulesFileName);

        //ΔΗΜΙΟΥΡΓΙΑ ΜΗΧΑΝΗΣ ΣΥΛΛΟΓΙΣΤΙΚΗΣ (REASONER) ΒΑΣΕΙ ΤΩΝ ΚΑΝΟΝΩΝ
        GenericRuleReasoner reasoner = new GenericRuleReasoner(rules);

        //ΔΗΜΙΟΥΡΓΙΑ ΜΟΝΤΕΛΟΥ ΜΕ ΤΙΣ ΝΕΕΣ ΤΡΙΑΔΕΣ ΛΟΓΩ ΣΥΜΠΕΡΑΣΜΟΥ
        InfModel OutModel = ModelFactory.createInfModel(reasoner, InModel);

        //ΕΠΙΣΤΡΟΦΗ ΤΟΥ ΝΕΟΥ ΜΟΝΤΕΛΟΥ ΣΑΝ ΕΞΟΔΟ
        return OutModel;
```

```
    }
}
```

3.5 SQLData2RDF.java

```
package core.handlers;

import com.hp.hpl.jena.rdf.model.Model;
import com.hp.hpl.jena.rdf.model.RDFNode;
import com.hp.hpl.jena.rdf.model.SimpleSelector;

public class SQLData2RDF {

    //ΚΛΑΣΗ ΠΡΟΣΘΑΦΑΙΡΕΣΗΣ ΔΕΔΟΜΕΝΩΝ RDF ΣΤΟ ΜΟΝΤΕΛΟ ΤΟΥ 1ΟΥ ΙΑΤΡΟΥ

    public static void deleteRDFNode(Model RDBModel, String DBkey){

        //ΔΙΑΓΡΑΦΗ RDF ΚΟΜΒΟΥ

        String DBURI = "http://example.org/doctor1/"+DBkey;

        //ΠΡΩΤΑ ΕΛΕΓΧΕΤΑΙ Η ΥΠΑΡΞΗ ΤΟΥ ΠΡΟΣ ΔΙΑΓΡΑΦΗ ΚΟΜΒΟΥ
        //ΑΝ ΕΠΙΒΕΒΑΙΩΘΕΙ ΔΙΑΓΡΑΦΕΤΑΙ Ο ΚΟΜΒΟΣ ΚΑΙ ΤΟ ΔΕΝΤΡΟ ΠΟΥ ΤΟΝ ΕΧΕΙ
        ΩΣ ΡΙΖΑ
        if (RDBModel.contains(RDBModel.getResource(DBURI), null, (RDFNode) null)){
            RDBModel.remove(RDBModel.listStatements(new SimpleSelector (
                RDBModel.getResource(DBURI), null, (RDFNode) null)));
            RDBModel.remove(RDBModel.listStatements(new SimpleSelector (
                RDBModel.getResource(DBURI+"/Birthday"), null, (RDFNode) null)));
            RDBModel.remove(RDBModel.listStatements(new SimpleSelector (
                RDBModel.getResource(DBURI+"/SecurityOrg"), null, (RDFNode) null)));
            RDBModel.remove(RDBModel.listStatements(new SimpleSelector (
                RDBModel.getResource(DBURI+"/ExaminationDate"), null, (RDFNode) null)));
            RDBModel.remove(RDBModel.listStatements(new SimpleSelector (
                RDBModel.getResource(DBURI+"/Findings"), null, (RDFNode) null)));
            RDBModel.remove(RDBModel.listStatements(new SimpleSelector (
                RDBModel.getResource(DBURI+"/Payment"), null, (RDFNode) null)));
        }
    }
}
```

3.6 UpdateModels.java

```
package core.handlers;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Date;
```

```
import javax.swing.JOptionPane;

import com.hp.hpl.jena.db.IDBConnection;
import com.hp.hpl.jena.rdf.model.InfModel;
import com.hp.hpl.jena.rdf.model.Model;

import de.fuberlin.wiwiss.d2rq.ModelD2RQ;
import doctor1.gui.SQLFrame;
import doctor2.handlers.DICOMDBDataFunctions;
import doctor2.handlers.DICOMTuple;
import doctor3.handlers.HL7DBDataFunctions;
import doctor3.handlers.HL7Tuple;

public class UpdateModels {

    //ΚΛΑΣΗ ΕΝΗΜΕΡΩΣΗΣ RDF ΜΟΝΤΕΛΩΝ ΑΠΟ ΤΙΣ ΑΛΛΑΓΕΣ ΣΤΙΣ ΒΑΣΕΙΣ ΤΩΝ
    ΙΑΤΡΩΝ

    public static void updateSQLModel() {

        //ΕΝΗΜΕΡΩΣΗ ΤΟΥ ΜΟΝΤΕΛΟΥ ΤΟΥ 1ΟΥ ΙΑΤΡΟΥ

        //ΦΟΡΤΩΣΗ PERSISTENT ΜΟΝΤΕΛΟΥ
        BasicFunctionsAndVariables.loadDriver();
        IDBConnection SQLConnection = BasicFunctionsAndVariables.connectToRDFBase(
            "jdbc:mysql://localhost/sql_rdf_model");
        Model SQL_Model = BasicFunctionsAndVariables.OpenModel(
            SQLConnection, "SQL_Model");

        //ΓΙΝΕΤΑΙ "ΧΕΙΡΟΚΙΝΗΤΗ" ΔΙΑΓΡΑΦΗ ΑΠΟ ΤΟ ΜΟΝΤΕΛΟ ΟΣΩΝ ΣΤΟΙΧΕΙΩΝ
        ΒΡΙΣΚΟΝΤΑΙ ΣΕ ΠΛΕΙΑΔΕΣ ΤΟΥ ΠΙΝΑΚΑ ELEMENTS_LOG
        //ΚΑΙ ΕΧΟΥΝ ΤΟ ΧΑΡΑΚΤΗΡΙΣΤΙΚΟ DELETE Η UPDATE
        Connection Conn = BasicFunctionsAndVariables.connectToBase(1);
        Statement St;
        try {
            St = Conn.createStatement();
            ResultSet ResSet = St.executeQuery(
                "SELECT amka, examinationDate " +
                "FROM elements_log " +
                "WHERE actionTaken = 'DELETE' "+
                "OR actionTaken = 'UPDATE'");
            while (ResSet.next()){
                SQLData2RDF.deleteRDFNode(
                    SQL_Model, ResSet.getString(1).concat("_").concat(ResSet.getString(2)));
            }
        }catch (SQLException e) {
            JOptionPane.showMessageDialog(null,
                "Αδύνατη η μετατροπή δεδομένων"+'\n'+e.toString(),
                "Σφάλμα Μετατροπής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
```

```
        //ΕΦΑΡΜΟΖΕΤΑΙ ΤΟ D2RQ MAPPING ΣΤΟΝ ΠΙΝΑΚΑ ELEMENTS_LOG ΚΑΙ
        ΠΡΟΚΥΠΤΕΙ ΤΟ ΜΟΝΤΕΛΟ ΤΩΝ ΝΕΩΝ ΔΕΔΟΜΕΝΩΝ
        Model Doctor_1_Update_Model = new ModelD2RQ(".\\n3\\Doctor1_Log_Map.n3");

        //ΠΡΟΣΤΙΘΕΤΑΙ ΣΤΟ ΑΡΧΙΚΟ ΜΟΝΤΕΛΟ
        SQL_Model.add(Doctor_1_Update_Model);

        //ΔΙΑΓΡΑΦΗ ΠΕΡΙΕΧΟΜΕΝΩΝ ΤΟΥ ΠΙΝΑΚΑ ELEMENTS_LOG
        try{
            St = Conn.createStatement();
            St.executeUpdate("DELETE FROM elements_log");

        }catch (SQLException e) {
            JOptionPane.showMessageDialog(null,
                "Αδύνατη η μετατροπή δεδομένων"+'\n'+e.toString(),
                "Σφάλμα Μετατροπής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }

        BasicFunctionsAndVariables.disconnectFromBase(Conn);
        System.out.println("SQL Model Updated");
        System.out.println(new Date());

        //ΕΦΑΡΜΟΓΗ REASONING
        InfModel SQL_Inf_Model = Reasoning.NewInferenceModel(
            SQL_Model, "file:.\\RULES\\SQL_RULES.rules");

        //ΑΝΤΙΚΑΤΑΣΤΑΣΗ ΤΟΥ ΥΠΑΡΧΟΝΤΟΣ ΜΟΝΤΕΛΟΥ ΑΠΟ ΤΟ ΝΕΟ
        SQL_Model.add(SQL_Inf_Model);
        System.out.println("apply SQL Reasoning OK");
        System.out.println(new Date());

        //ΑΠΟΣΥΝΔΕΣΗ ΑΠΟ ΤΗ ΒΑΣΗ
        BasicFunctionsAndVariables.disconnectFromRDFBase(SQLConnection);

    }

    public static void updateDicomModel() {

        //ΕΝΗΜΕΡΩΣΗ ΤΟΥ ΜΟΝΤΕΛΟΥ ΤΟΥ 2ΟΥ ΙΑΤΡΟΥ

        //ΣΥΝΔΕΣΗ ΣΤΗ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ ΤΟΥ ΙΑΤΡΟΥ
        BasicFunctionsAndVariables.loadDriver();
        Connection Conn = BasicFunctionsAndVariables.connectToBase(2);
        Statement St;

        //ΓΙΑ ΚΑΘΕ DELETE Η UPDATE ΠΛΕΙΑΔΑ ΤΟΥ ΠΙΝΑΚΑ ACTION LOG
        //ΔΙΑΓΡΑΦΟΝΤΑΙ ΑΠΟ ΤΟ ΜΟΝΤΕΛΟ ΟΙ ΑΝΤΙΣΤΟΙΧΟΙ ΚΟΜΒΟΙ
        //(ΑΥΤΟ ΓΙΝΕΤΑΙ ΣΤΟ DICOMDATA2RDF)
        try {
            St = Conn.createStatement();
```

```
            ResultSet ResSet = St.executeQuery(
                "SELECT oldFileName " +
                "FROM action_log " +
                "WHERE actionTaken = 'DELETE' " +
                "OR actionTaken = 'UPDATE'");
            while (ResSet.next()){
                DICOMData2RDF.deleteRDFNode(ResSet.getString(1));
            }

            //ΓΙΑ ΚΑΘΕ INSERT Η UPDATE ΠΛΕΙΑΔΑ ΤΟΥ ΠΙΝΑΚΑ ACTION LOG
            //ΔΗΜΙΟΥΡΓΟΥΝΤΑΙ ΚΑΙ ΠΡΟΣΤΙΘΕΝΤΑΙ ΣΤΟ ΜΟΝΤΕΛΟ ΟΙ ΑΝΤΙΣΤΟΙΧΟΙ
            ΚΟΜΒΟΙ
            //(ΑΥΤΟ ΓΙΝΕΤΑΙ ΣΤΟ DICOMDATA2RDF)
            ResSet = St.executeQuery(
                "SELECT newFileName " +
                "FROM action_log " +
                "WHERE actionTaken = 'INSERT' " +
                "OR actionTaken = 'UPDATE'");
            while (ResSet.next()){
                DICOMTuple WorkTuple = DICOMDBDataFunctions.fetchFromDicomBase(
                    ResSet.getString(1), "RDF");
                DICOMData2RDF.dicomTuple2RdfNode(WorkTuple);
            }
            ResSet.close();

            //ΜΕΤΑ ΤΗΝ ΕΝΗΜΕΡΩΣΗ ΔΙΑΓΡΑΦΕΤΑΙ ΤΟ LOG
            St.executeUpdate("DELETE FROM action_log");
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null,
                "Τα δεδομένα δεν μπορούν να μετατραπούν"+'\n'+e.toString(),
                "Σφάλμα Μετατροπής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
        BasicFunctionsAndVariables.disconnectFromBase(Conn);
        System.out.println("DICOM Model Updated");
        System.out.println(new Date());

        //ΦΟΡΤΩΣΗ RDF ΜΟΝΤΕΛΟΥ
        //Η ΜΕΤΕΠΕΙΤΑ ΔΙΑΔΙΚΑΣΙΑ ΕΙΝΑΙ ΙΔΙΑ ΜΕ ΤΗΝ ΠΕΡΙΠΤΩΣΗ ΤΟΥ 1ΟΥ ΙΑΤΡΟΥ
        IDBConnection DICOMConnection = BasicFunctionsAndVariables.connectToRDFBase(
            "jdbc:mysql://localhost/dicom_rdf_model");
        Model DICOM_Model = BasicFunctionsAndVariables.OpenModel(
            DICOMConnection,"DICOM_Model");

        //REASONING
        InfModel DICOM_Inf_Model = Reasoning.NewInferenceModel(
            DICOM_Model, "file:.\\RULES\\DICOM_RULES.rules");
        DICOM_Model.add(DICOM_Inf_Model);
        System.out.println("apply DICOM Reasoning OK");
        System.out.println(new Date());
        BasicFunctionsAndVariables.disconnectFromRDFBase(DICOMConnection);
    }
```

```
public static void updateHl7Model() {

    //ΕΝΗΜΕΡΩΣΗ ΤΟΥ ΜΟΝΤΕΛΟΥ ΤΟΥ 3ΟΥ ΙΑΤΡΟΥ
    //ΟΛΗ Η ΔΙΑΔΙΚΑΣΙΑ ΕΙΝΑΙ ΠΑΝΟΜΟΙΟΤΥΠΗ ΜΕ ΤΗΝ ΠΕΡΙΠΤΩΣΗ ΤΟΥ 2ΟΥ
    ΙΑΤΡΟΥ

    BasicFunctionsAndVariables.loadDriver();
    Connection Conn = BasicFunctionsAndVariables.connectToBase(3);
    Statement St;
    try {
        St = Conn.createStatement();
        ResultSet ResSet = St.executeQuery(
            "SELECT oldFileName " +
            "FROM action_log " +
            "WHERE actionTaken = 'DELETE' " +
            "OR actionTaken = 'UPDATE'");
        while (ResSet.next()){
            HL7Data2RDF.deleteRDFNode(ResSet.getString(1));
        }
        ResSet = St.executeQuery(
            "SELECT newFileName " +
            "FROM action_log " +
            "WHERE actionTaken = 'INSERT' " +
            "OR actionTaken = 'UPDATE'");
        while (ResSet.next()){
            HL7Tuple WorkTuple = HL7DBDataFunctions.fetchFromHL7Base(
                ResSet.getString(1), "RDF");
            HL7Data2RDF.hl7Tuple2RdfNode(WorkTuple);
        }
        ResSet.close();
        St.executeUpdate("DELETE FROM action_log");
    } S (SQLException e) {
        JOptionPane.showMessageDialog(null,
            "Τα δεδομένα δεν μπορούν να μετατραπούν"+'\n'+e.toString(),
            "Σφάλμα Μετατροπής Δεδομένων",
            JOptionPane.ERROR_MESSAGE);
    }
    BasicFunctionsAndVariables.disconnectFromBase(Conn);
    System.out.println("HL7 Model Updated");
    System.out.println(new Date());

    IDBConnection HL7Connection = BasicFunctionsAndVariables.connectToRDFBase(
        "jdbc:mysql://localhost/hl7_rdf_model");
    Model HL7_Model = BasicFunctionsAndVariables.OpenModel(
        HL7Connection,"HL7_Model");

    InfModel HL7_Inf_Model = Reasoning.NewInferenceModel(
        HL7_Model, "file:.\\RULES\\HL7_RULES.rules");
    HL7_Model.add(HL7_Inf_Model);
    System.out.println("apply HL7 Reasoning OK");
    System.out.println(new Date());
```

```
            BasicFunctionsAndVariables.disconnectFromRDFBase(HL7Connection);
        }
}
```

4. Πακέτο doctor1.gui, κλάση SQLFrame.java

```
package doctor1.gui;

import java.awt.Component;
import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.GridLayout;
import java.awt.Insets;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.StringTokenizer;

import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.ScrollPaneConstants;

import doctor1.handlers.DBTuple;
import doctor1.handlers.DataFunctions;

public class SQLFrame extends JFrame implements ActionListener{

    //ΚΛΑΣΗ GUI 1ΟΥ ΙΑΤΡΟΥ

    private static final long serialVersionUID = 1L;

    JButton Create = new JButton("Εισαγωγή");
    JButton Load = new JButton("Αναζήτηση");
    JButton Update = new JButton("Ενημέρωση");
    JButton Delete = new JButton("Διαγραφή");
    JButton ClearAll = new JButton("Καθαρισμός Πεδίων");

    JLabel FName = new JLabel("Όνομα:");
    JLabel LName = new JLabel("Επίθετο:");
    JLabel FatherName = new JLabel("Πατρώνυμο:");
    JLabel Birthday = new JLabel("Ημερομηνία Γέννησης:");
```

```
JLabel AMKA = new JLabel("A.M.K.A.:");
JLabel SecurityOrg = new JLabel ("Φορέας Ασφάλισης:");
JLabel Address = new JLabel("Διεύθυνση (Οδός/Αριθμός/Πόλη):");
JLabel Phone = new JLabel("Τηλέφωνο:");
JLabel ExaminationDate = new JLabel("Ημερομηνία Εξέτασης:");
JLabel Disease = new JLabel("Ασθένεια:");
JLabel Findings = new JLabel("Ευρήματα/Συμπτώματα/Παρατηρήσεις:");
JLabel Sum = new JLabel("Υπόλοιπο σε Ευρώ:");

JTextField FNameText = new JTextField(20);
JTextField LNameText = new JTextField(35);
JTextField FatherNameText = new JTextField(20);
JTextField BirthdayText = new JTextField(15);
JTextField AMKAText = new JTextField(20);
JTextField SecurityOrgText = new JTextField(20);
JTextField AddressText = new JTextField(35);
JTextField PhoneText = new JTextField(20);
JTextField ExaminationDateText = new JTextField(15);
JTextArea FindingsText = new JTextArea();
JTextField SumText = new JTextField(10);

//ΠΡΟΚΑΘΟΡΙΣΜΕΝΗ ΛΙΣΤΑ ΑΣΘΕΝΕΙΩΝ
String[] DiseaseList = {"Ασυμπτωματικός/Υγιής", "Κοινό Κρυολόγημα", "Άσθμα",
    "Αλλεργική Ρινίτιδα","Γρίπη", "Φαρυγγίτιδα", "Ωτίτιδα", "Δερματίτιδα", "Έρπης Ζωστήρας",
    "Γαστροοισοφαγική Παλινδρόμηση", "Υπέρταση", "Εμπύρετο", "Καρδιακή Αρρυθμία",
    "Αιμορροϊδες", "Γαστρεντερίτιδα", "Αλωπεκία"};

JComboBox DiseaseText = new JComboBox(DiseaseList);

JScrollPane Scroll = new JScrollPane(FindingsText,
ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED,
    ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);

GridBagLayout Gridbag = new GridBagLayout();
GridBagConstraints Constraints;

Insets Whitespace = new Insets (10,10,10,10);

DBTuple WorkTuple;

public SQLFrame(){

    //ΚΑΤΑΣΚΕΥΗ GUI

    super ("IntegraHEALTH 1.0 by Nikos Christodoulou –
        Καρτέλα Στοιχείων Ασθενούς/Εξέτασης");
    Dimension ScreenSize = Toolkit.getDefaultToolkit().getScreenSize();
    setBounds((ScreenSize.width-1280)/2, (ScreenSize.height-600)/2, 1280, 600);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLayout(Gridbag);

    Create.addActionListener(this);
```

```
Load.addActionListener(this);
Update.addActionListener(this);
Delete.addActionListener(this);
ClearAll.addActionListener(this);
Update.setEnabled(false);
Delete.setEnabled(false);

JPanel CommandPane = new JPanel();
CommandPane.setLayout(new GridLayout (1, 4, 10, 10));
CommandPane.add(Create);
CommandPane.add(Load);
CommandPane.add(Update);
CommandPane.add(Delete);
CommandPane.add(ClearAll);

FindingsText.setLineWrap(true);
FindingsText.setWrapStyleWord(true);

addComponent(FName, 0, 0, 1, 1, 10, 100, GridBagConstraints.NONE,
     GridBagConstraints.EAST);
addComponent(FNameText, 1, 0, 1, 1, 40, 100, GridBagConstraints.BOTH,
     GridBagConstraints.WEST);
addComponent(LName, 2, 0, 1, 1, 10, 100, GridBagConstraints.NONE,
     GridBagConstraints.EAST);
addComponent(LNameText, 3, 0, 1, 1, 40, 100, GridBagConstraints.BOTH,
     GridBagConstraints.WEST);
addComponent(FatherName, 0, 1, 1, 1, 10, 100, GridBagConstraints.NONE,
     GridBagConstraints.EAST);
addComponent(FatherNameText, 1, 1, 1, 1, 40, 100, GridBagConstraints.BOTH,
     GridBagConstraints.WEST);
addComponent(Birthday, 2, 1, 1, 1, 10, 100, GridBagConstraints.NONE,
     GridBagConstraints.EAST);
addComponent(BirthdayText, 3, 1, 1, 1, 40, 100, GridBagConstraints.BOTH,
     GridBagConstraints.WEST);
addComponent(AMKA, 0, 2, 1, 1, 10, 100, GridBagConstraints.NONE,
     GridBagConstraints.EAST);
addComponent(AMKAText, 1, 2, 1, 1, 40, 100, GridBagConstraints.BOTH,
     GridBagConstraints.WEST);
addComponent(SecurityOrg, 2, 2, 1, 1, 10, 100, GridBagConstraints.NONE,
     GridBagConstraints.EAST);
addComponent(SecurityOrgText, 3, 2, 1, 1, 40, 100, GridBagConstraints.BOTH,
     GridBagConstraints.WEST);
addComponent(Address, 0, 3, 1, 1, 10, 100, GridBagConstraints.NONE,
     GridBagConstraints.EAST);
addComponent(AddressText, 1, 3, 1, 1, 40, 100, GridBagConstraints.BOTH,
     GridBagConstraints.WEST);
addComponent(Phone, 2, 3, 1, 1, 10, 100, GridBagConstraints.NONE,
     GridBagConstraints.EAST);
addComponent(PhoneText, 3, 3, 1, 1, 40, 100, GridBagConstraints.BOTH,
     GridBagConstraints.WEST);
addComponent(ExaminationDate, 0, 4, 1, 1, 10, 100, GridBagConstraints.NONE,
     GridBagConstraints.EAST);
```

```
        addComponent(ExaminationDateText, 1, 4, 1, 1, 40, 100, GridBagConstraints.BOTH,
            GridBagConstraints.WEST);
        addComponent(Disease, 2, 4, 1, 1, 10, 100, GridBagConstraints.NONE,
            GridBagConstraints.EAST);
        addComponent(DiseaseText, 3, 4, 1, 1, 40, 100, GridBagConstraints.BOTH,
            GridBagConstraints.WEST);
        addComponent(Findings, 0, 5, 1, 1, 10, 100, GridBagConstraints.NONE,
            GridBagConstraints.EAST);
        addComponent(Scroll, 1, 5, 3, 1, 90, 100, GridBagConstraints.BOTH,
            GridBagConstraints.CENTER);
        addComponent(Sum, 0, 6, 1, 1, 10, 100, GridBagConstraints.NONE,
            GridBagConstraints.EAST);
        addComponent(SumText, 1, 6, 1, 1, 40, 100, GridBagConstraints.BOTH,
            GridBagConstraints.WEST);
        addComponent(CommandPane, 0, 7 ,4, 1, 100, 100, GridBagConstraints.NONE,
            GridBagConstraints.CENTER);
    }

    private void addComponent(Component Component, int GridX, int GridY, int GridWidth,
        int GridHeight, int WeightX, int WeightY, int Fill, int Anchor){
        GridBagConstraints Constraints = new GridBagConstraints();
        Constraints.gridx = GridX;
        Constraints.gridy = GridY;
        Constraints.gridwidth = GridWidth;
        Constraints.gridheight = GridHeight;
        Constraints.weightx = WeightX;
        Constraints.weighty = WeightY;
        Constraints.fill = Fill;
        Constraints.anchor = Anchor;
        Constraints.insets = Whitespace;
        Gridbag.setConstraints(Component, Constraints);
        add(Component);
    }

    public void actionPerformed(ActionEvent evt) {

        //ΑΝΑΛΟΓΑ ΜΕ ΤΟ ΚΟΥΜΠΙ ΠΟΥ ΕΠΙΛΕΓΕΤΑΙ,
        //ΕΝΕΡΓΟΠΟΙΕΙΤΑΙ Η ΚΑΤΑΛΛΗΛΗ ΔΙΑΔΙΚΑΣΙΑ ΑΛΛΑΓΗΣ ΣΤΗ ΒΑΣΗ
        //INSERT, UPDATE, DELETE

        Object Source = evt.getSource();

        //ΕΛΕΓΧΟΣ ΜΟΡΦΗΣ ΗΜΕΡΟΜΗΝΙΩΝ (ΑΠΑΙΤΟΥΜΕΝΗ ΜΟΡΦΗ: EEEE-MM-HH)
        boolean ExamineDateFormatConfirmed = confirmDateFormat(
            ExaminationDateText.getText());
        boolean BirthDateFormatConfirmed = confirmDateFormat(BirthdayText.getText());

        //ΕΙΣΑΓΩΓΗ ΝΕΩΝ ΣΤΟΙΧΕΙΩΝ
        if (Source==Create){

            //ΠΕΡΙΠΤΩΣΕΙΣ ΛΑΝΘΑΣΜΕΝΩΝ ΣΤΟΙΧΕΙΩΝ ΓΙΑ ΕΙΣΑΓΩΓΗ
            if (AMKAText.getText().isEmpty()
```

```
            |FNameText.getText().isEmpty()
            |LNameText.getText().isEmpty()){
            JOptionPane.showMessageDialog(null,
                "Το όνομα, το επίθετο, ή/και το Α.Μ.Κ.Α. είναι κενά!" +'\n'+
                "Συμπληρώστε τα και ξαναδοκιμάστε",
                "Σφάλμα Εισαγωγής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
        else if (ExaminationDateText.getText().contains(".")
            |ExaminationDateText.getText().contains("/")
            |BirthdayText.getText().contains(".")
            |BirthdayText.getText().contains("/")){
            JOptionPane.showMessageDialog(null,
                "Χρησιμοποιήσατε . ή / για την ημερομηνία!"+'\n'+
                "Επιλέξτε ένα άλλο σύμβολο και ξαναδοκιμάστε",
                "Σφάλμα Εισαγωγής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
        else if ((!BirthDateFormatConfirmed)|(!ExamineDateFormatConfirmed)){
            JOptionPane.showMessageDialog(null,
                "Οι ημερομηνίες πρέπει να είναι στη μορφή EEEE-MM-HH!" +'\n'+
                "Συμπληρώστε τες σωστά και ξαναδοκιμάστε",
                "Σφάλμα Εισαγωγής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
        else {

            //ΦΟΡΤΩΣΗ ΤΩΝ ΣΤΟΙΧΕΙΩΝ ΑΠΟ ΤΟ GUI
            WorkTuple = loadTuple();

            //ΑΠΟΘΗΚΕΥΣΗ ΣΤΗ ΒΑΣΗ
            DataFunctions.createInBase(WorkTuple);
        }
    }

    //ΑΝΑΚΤΗΣΗ ΣΤΟΙΧΕΙΩΝ ΣΤΟ GUI ΑΠΟ ΤΗ ΒΑΣΗ
    if (Source==Load){

        //ΠΕΡΙΠΤΩΣΕΙΣ ΣΤΟΙΧΕΙΩΝ ΓΙΑ ΑΝΑΖΗΤΗΣΗ

        //ΑΝΑΖΗΤΗΣΗ ΜΕ ΑΜΚΑ
        if (!AMKAText.getText().isEmpty()){

            //ΑΝΑΚΤΗΣΗ ΠΛΗΡΟΦΟΡΙΩΝ ΑΠΟ ΤΗ ΒΑΣΗ
            presentTuple(DataFunctions.fetchFromBase(AMKAText.getText(), ""));
            Create.setEnabled(false);
            Load.setEnabled(false);
            Update.setEnabled(true);
            Delete.setEnabled(true);
            AMKAText.setEnabled(false);
        }
        else {
```

```
            if (FNameText.getText().isEmpty()|LNameText.getText().isEmpty()){
                JOptionPane.showMessageDialog(null,
                    "Το όνομα, το επίθετο, ή/και το Α.Μ.Κ.Α. είναι κενά!" +'\n'+
                    "Συμπληρώστε τα και ξαναδοκιμάστε",
                    "Σφάλμα Εισαγωγής Δεδομένων",
                    JOptionPane.ERROR_MESSAGE);
            }

            //ΠΕΡΙΠΤΩΣΗ ΚΕΝΟΥ ΑΜΚΑ, ΑΝΑΖΗΤΗΣΗ ΜΕ ΟΝΟΜΑΤΕΠΩΝΥΜΟ
            else {

                //ΑΝΑΚΤΗΣΗ ΠΛΗΡΟΦΟΡΙΩΝ ΑΠΟ ΤΗ ΒΑΣΗ
                presentTuple(DataFunctions.fetchFromBase(
                    FNameText.getText(), LNameText.getText()));
                Create.setEnabled(false);
                Load.setEnabled(false);
                Update.setEnabled(true);
                Delete.setEnabled(true);
                AMKAText.setEnabled(false);
            }
        }
    }

    //ΕΝΗΜΕΡΩΣΗ ΣΤΟΙΧΕΙΩΝ ΣΤΗ ΒΑΣΗ
    if (Source==Update){
        if ((FNameText.getText().isEmpty())|(LNameText.getText().isEmpty())){
            JOptionPane.showMessageDialog(null,
                "Το όνομα ή/και το επίθετο είναι κενά!" +'\n'+
                "Συμπληρώστε τα και ξαναδοκιμάστε",
                "Σφάλμα Εισαγωγής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
        else{

            //ΦΟΡΤΩΣΗ ΤΩΝ ΣΤΟΙΧΕΙΩΝ ΑΠΟ ΤΟ GUI
            WorkTuple = loadTuple();

            //ΑΠΟΘΗΚΕΥΣΗ ΣΤΗ ΒΑΣΗ
            DataFunctions.updateInBase(WorkTuple);
        }
    }

    //ΔΙΑΓΡΑΦΗ ΣΤΟΙΧΕΙΩΝ ΑΠΟ ΤΗ ΒΑΣΗ
    if (Source==Delete){
        DataFunctions.deleteFromBase(AMKAText.getText());
        clearAllFields();
    }

    //ΑΠΑΛΟΙΦΗ ΟΛΩΝ ΤΩΝ ΠΕΡΙΕΧΟΜΕΝΩΝ ΤΟΥ GUI
    if (Source==ClearAll){
        clearAllFields();
    }
```

```
}

private void clearAllFields(){

    //ΑΠΑΛΟΙΦΗ ΟΛΩΝ ΤΩΝ ΠΕΡΙΕΧΟΜΕΝΩΝ ΤΟΥ GUI

    FNameText.setText("");
    LNameText.setText("");
    FatherNameText.setText("");
    BirthdayText.setText("");
    AMKAText.setText("");
    if (!AMKAText.isEnabled()){
        AMKAText.setEnabled(true);
    }
    SecurityOrgText.setText("");
    AddressText.setText("");
    PhoneText.setText("");
    ExaminationDateText.setText("");
    DiseaseText.setSelectedIndex(0);
    FindingsText.setText("");
    SumText.setText("");
    Create.setEnabled(true);
    Load.setEnabled(true);
    Update.setEnabled(false);
    Delete.setEnabled(false);
}

private boolean confirmDateFormat(String InDate) {

    //ΕΛΕΓΧΟΣ ΜΟΡΦΗΣ ΗΜΕΡΟΜΗΝΙΩΝ (ΕΕΕΕ-ΜΜ-ΗΗ)

    int Day, Month, Year;
    if(InDate.length()!=10){
        return false;
    }
    StringTokenizer stk = new StringTokenizer(InDate, InDate.substring(4, 5));
    if (stk.countTokens()!=3){
        return false;
    }
    else{
        Year = Integer.parseInt(stk.nextToken());
        Month = Integer.parseInt(stk.nextToken());
        Day = Integer.parseInt(stk.nextToken());
        DateFormat yearFormat = new SimpleDateFormat("yyyy");
        Date date = new Date();
        int CurrentYear = Integer.parseInt(yearFormat.format(date));
        if (Day>31|Day<1){
            return false;
        }
        else if (Month>12|Month<1){
            return false;
        }
```

```
            else if(((Month==2)|(Month==4)|(Month==6)|(Month==9)|(Month==11))&(Day==31)){
                return false;
            }
            else if((Month==2)&((Day==30)|(Day==31))){
                return false;
            }
            else if((Month==2)&(Year%4!=0)&(Day==29)){
                return false;
            }
            else if((Month==2)&(Year%4==0)&(Year%400!=0)&(Day==29)){
                return false;
            }
            else if ((Year<(CurrentYear-150))|(Year>CurrentYear)){
                return false;
            }
        }
        return true;
    }

    public DBTuple loadTuple(){

        //ΦΟΡΤΩΣΗ ΣΤΟΙΧΕΙΩΝ ΑΠΟ ΤΟ GUI ΣΕ ΜΙΑ ΠΛΕΙΑΔΑ

        DBTuple OutTuple = new DBTuple();
        OutTuple.FName = FNameText.getText();
        OutTuple.LName = LNameText.getText();
        OutTuple.FatherName = FatherNameText.getText();
        OutTuple.Birthday = BirthdayText.getText();
        OutTuple.Amka = AMKAText.getText();
        OutTuple.SecurityOrganization = SecurityOrgText.getText();
        OutTuple.Address = AddressText.getText();
        OutTuple.Phone = PhoneText.getText();
        OutTuple.ExaminationDate = ExaminationDateText.getText();
        OutTuple.Disease = DiseaseText.getSelectedIndex()+1;
        OutTuple.Remarks = FindingsText.getText();
        if (!SumText.getText().isEmpty()){
            OutTuple.Sum = Float.parseFloat(SumText.getText());
        }else{
            OutTuple.Sum = 0f;
        }
        return OutTuple;
    }
    private void presentTuple(DBTuple OutTuple){

        //ΠΑΡΟΥΣΙΑΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

        FNameText.setText(OutTuple.FName);
        LNameText.setText(OutTuple.LName);
        FatherNameText.setText(OutTuple.FatherName);
        BirthdayText.setText(OutTuple.Birthday);
        AMKAText.setText(OutTuple.Amka);
        SecurityOrgText.setText(OutTuple.SecurityOrganization);
```

```
            AddressText.setText(OutTuple.Address);
            PhoneText.setText(OutTuple.Phone);
            ExaminationDateText.setText(OutTuple.ExaminationDate);
            DiseaseText.setSelectedIndex(OutTuple.Disease-1);
            FindingsText.setText(OutTuple.Remarks);
            SumText.setText(OutTuple.Sum.toString());
        }
}
```

5. Πακέτο doctor1.handlers

5.1 BasicFunctions.java

```
package doctor1.handlers;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

import javax.swing.JOptionPane;

public class BasicFunctions {

    //ΚΛΑΣΗ ΒΑΣΙΚΩΝ ΣΥΝΑΡΤΗΣΕΩΝ ΕΙΣΟΔΟΥ ΚΑΙ ΕΞΟΔΟΥ ΑΠΟ ΤΗ ΒΑΣΗ ΤΟΥ 1ΟΥ
    ΙΑΤΡΟΥ

    public static void loadDriver(){

        //ΦΟΡΤΩΣΗ ΚΛΑΣΗΣ MYSQL DRIVER

        try {
            Class.forName("com.mysql.jdbc.Driver");
        } catch (ClassNotFoundException e) {
            JOptionPane.showMessageDialog(null,
                "Cannot Load Driver"+'\n'+e.toString(),
                "Driver Loading Error",
                JOptionPane.ERROR_MESSAGE);
        }
    }

    public static Connection connectToBase(){

        //ΣΥΝΔΕΣΗ ΣΤΗ ΒΑΣΗ ΤΟΥ 1ΟΥ ΙΑΤΡΟΥ

        try {
            String url = "jdbc:mysql://localhost/doctor1";
            Connection conn = DriverManager.getConnection(url, "root", "");
            return conn;
        } catch (SQLException e) {
            JOptionPane.showMessageDialog(null,
                "Cannot Connect to Base"+'\n'+e.toString(),
```

```
                "Database Connecting Error",
                JOptionPane.ERROR_MESSAGE);
            return null;
        }
    }

    public static void disconnectFromBase (Connection conn){

        //ΑΠΟΣΥΝΔΕΣΗ ΑΠΟ ΤΗ ΒΑΣΗ

        try{
            conn.close();
        }catch (SQLException e){
            JOptionPane.showMessageDialog(null,
                "Cannot Disconnect from Base"+'\n'+e.toString(),
                "Database Disconnecting Error",
                JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

5.2 DataFunctions.java

```
package doctor1.handlers;

import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import javax.swing.JOptionPane;

public class DataFunctions {

    //ΚΛΑΣΗ ΜΕΘΟΔΩΝ ΑΛΛΑΓΩΝ ΤΗΣ ΒΑΣΗΣ ΤΟΥ 1ΟΥ ΙΑΤΡΟΥ

    public static void createInBase(DBTuple InTuple){

        //ΕΙΣΑΓΩΓΗ ΝΕΩΝ ΣΤΟΙΧΕΙΩΝ

        BasicFunctions.loadDriver();
        Connection Conn = BasicFunctions.connectToBase();

        try{
            java.sql.PreparedStatement Prep = Conn.prepareStatement(
                "INSERT INTO "+
                "elements(fName, lName, fatherName, birthday, amka, " +
                "securityOrg, address, phone, examinationDate, disease, remarks, sum)" +
                "VALUES(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)");
            Prep.setString(1, InTuple.FName);
            Prep.setString(2, InTuple.LName);
            Prep.setString(3, InTuple.FatherName);
```

```
            Prep.setString(4, InTuple.Birthday);
            Prep.setString(5, InTuple.Amka);
            Prep.setString(6, InTuple.SecurityOrganization);
            Prep.setString(7, InTuple.Address);
            Prep.setString(8, InTuple.Phone);
            Prep.setString(9, InTuple.ExaminationDate);
            Prep.setInt(10, InTuple.Disease);
            Prep.setString(11, InTuple.Remarks);
            Prep.setFloat(12, InTuple.Sum);
            Prep.executeUpdate();
            Prep.close();
        }catch (SQLException e){
            JOptionPane.showMessageDialog(null,
                "Τα δεδομένα δεν γράφτηκαν στη βάση"+'\n'+e.toString(),
                "Σφάλμα Εισαγωγής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
        BasicFunctions.disconnectFromBase(Conn);
    }

    public static DBTuple fetchFromBase(String InElement1, String InElement2){

        //ΑΝΑΚΤΗΣΗ ΔΕΔΟΜΕΝΩΝ

        DBTuple ResultTuple = new DBTuple();
        int Response = 1;
        BasicFunctions.loadDriver();
        Connection Conn = BasicFunctions.connectToBase();
        try{
            Statement St = Conn.createStatement();
            if (InElement2==""){

                //ΑΝΑΖΗΤΗΣΗ ΒΑΣΕΙ ΑΜΚΑ
                ResultSet ResSet = St.executeQuery(
                    "SELECT * FROM elements WHERE (amka='" + InElement1 + "')");

                //ΒΡΟΧΟΣ ΓΙΑ ΤΗΝ ΕΠΙΛΟΓΗ ΤΗΣ ΕΠΙΘΥΜΗΤΗΣ ΗΜΕΡΟΜΗΝΙΑΣ
                ΕΞΕΤΑΣΗΣ
                while (ResSet.next() && (Response==1)){
                    Response = JOptionPane.showOptionDialog(null,
                        "Βρέθηκε ο ασθενής με ΑΜΚΑ: "+ResSet.getString(5)+'\n'+
                        "και ημερομηνία εξέτασης: "+ResSet.getString(9)+'\n'+
                        "Είναι αυτό το επιθυμητό αποτέλεσμα;",
                        "Εύρεση Πολλαπλών Αποτελεσμάτων",
                        JOptionPane.YES_NO_OPTION,
                        JOptionPane.QUESTION_MESSAGE,
                        null, null, null);

                    if (Response==0){
                        ResultTuple.FName = ResSet.getString(1);
                        ResultTuple.LName = ResSet.getString(2);
                        ResultTuple.FatherName = ResSet.getString(3);
```

```
                ResultTuple.Birthday = ResSet.getString(4);
                ResultTuple.Amka = ResSet.getString(5);
                ResultTuple.SecurityOrganization = ResSet.getString(6);
                ResultTuple.Address = ResSet.getString(7);
                ResultTuple.Phone = ResSet.getString(8);
                ResultTuple.ExaminationDate = ResSet.getString(9);
                ResultTuple.Disease = ResSet.getInt(10);
                ResultTuple.Remarks = ResSet.getString(11);
                ResultTuple.Sum = ResSet.getFloat(12);
                break;
            }
        }
    }
    else{

        //ΑΝΑΖΗΤΗΣΗ ΜΕ ΒΑΣΗ ΟΝΟΜΑΤΕΠΩΝΥΜΟ ΑΣΘΕΝΗ
        ResultSet ResSet = St.executeQuery(
            "SELECT * FROM elements WHERE (fName='" + InElement1 + "') " +
            "AND (lName='" + InElement2+ "')");

        //ΒΡΟΧΟΣ ΓΙΑ ΤΗΝ ΕΠΙΛΟΓΗ ΜΕΤΑΞΥ ΑΣΘΕΝΩΝ ΜΕ ΙΔΙΟ
        ΟΝΟΜΑΤΕΠΩΝΥΜΟ
        while (ResSet.next() && (Response==1)){
            Response = JOptionPane.showOptionDialog(null,
                "Βρέθηκε ο ασθενής: "+ResSet.getString(1)+"
                "+ResSet.getString(2)+'\n'+
                "με ΑΜΚΑ: "+ResSet.getString(5)+'\n'+
                "Είναι αυτό το επιθυμητό αποτέλεσμα;",
                "Εύρεση Πολλαπλών Αποτελεσμάτων",
                JOptionPane.YES_NO_OPTION,
                JOptionPane.QUESTION_MESSAGE,
                null, null, null);

            if (Response==0){
                ResultTuple.FName = ResSet.getString(1);
                ResultTuple.LName = ResSet.getString(2);
                ResultTuple.FatherName = ResSet.getString(3);
                ResultTuple.Birthday = ResSet.getString(4);
                ResultTuple.Amka = ResSet.getString(5);
                ResultTuple.SecurityOrganization = ResSet.getString(6);
                ResultTuple.Address = ResSet.getString(7);
                ResultTuple.Phone = ResSet.getString(8);
                ResultTuple.ExaminationDate = ResSet.getString(9);
                ResultTuple.Disease = ResSet.getInt(10);
                ResultTuple.Remarks = ResSet.getString(11);
                ResultTuple.Sum = ResSet.getFloat(12);
                break;
            }
        }
    }
    if (Response==1){
        ResultTuple.FName = "N/A";
```

```
                ResultTuple.LName = "N/A";
                ResultTuple.FatherName = "N/A";
                ResultTuple.Birthday = "N/A";
                ResultTuple.Amka = "N/A";
                ResultTuple.SecurityOrganization = "N/A";
                ResultTuple.Address = "N/A";
                ResultTuple.Phone = "N/A";
                ResultTuple.ExaminationDate = "N/A";
                ResultTuple.Disease = 1;
                ResultTuple.Remarks = "N/A";
                ResultTuple.Sum = 0f;
                JOptionPane.showMessageDialog(null, "Δεν βρέθηκε τίποτα");
            }
            St.close();
        }catch (Exception e){
            JOptionPane.showMessageDialog(null,
                "Τα δεδομένα δεν μπορούν να ανακτηθούν"+'\n'+e.toString(),
                "Σφάλμα Ανάκτησης Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
        BasicFunctions.disconnectFromBase(Conn);
        return ResultTuple;
    }

    public static void updateInBase(DBTuple InTuple){

        //ΕΝΗΜΕΡΩΣΗ ΔΕΔΟΜΕΝΩΝ

        BasicFunctions.loadDriver();
        Connection Conn = BasicFunctions.connectToBase();

        try{
            java.sql.PreparedStatement PrepSt = Conn.prepareStatement(
                "UPDATE elements " +
                "SET fName = '" + InTuple.FName + "', " +
                    "lName = '" + InTuple.LName + "', " +
                    "fatherName = '" + InTuple.FatherName + "', " +
                    "birthday = '" + InTuple.Birthday + "', " +
                    "securityOrg = '" + InTuple.SecurityOrganization + "', " +
                    "address = '" + InTuple.Address + "', " +
                    "phone = '" + InTuple.Phone + "', " +
                    "examinationDate = '" + InTuple.ExaminationDate + "', " +
                    "disease = '" + InTuple.Disease + "', " +
                    "remarks = '" + InTuple.Remarks + "', " +
                    "sum = '" + InTuple.Sum + "'" +
                "WHERE amka = '"+ InTuple.Amka +"'");
            PrepSt.executeUpdate();
            PrepSt.close();
     }catch (SQLException e){
        JOptionPane.showMessageDialog(null,
            "Τα δεδομένα δεν ενημερώθηκαν"+'\n'+e.toString(),
            "Σφάλμα Ενημέρωσης Δεδομένων",
```

```
                JOptionPane.ERROR_MESSAGE);
        }
            BasicFunctions.disconnectFromBase(Conn);
        }

        public static void deleteFromBase(String InAmka){

            //ΔΙΑΓΡΑΦΗ ΔΕΔΟΜΕΝΩΝ

            BasicFunctions.loadDriver();
            Connection Conn = BasicFunctions.connectToBase();
            try{
                java.sql.PreparedStatement PrepSt = Conn.prepareStatement(
                    "DELETE FROM elements WHERE amka = '"+ InAmka + "'");
                PrepSt.executeUpdate();
                PrepSt.close();
            }catch (SQLException e){
                JOptionPane.showMessageDialog(null,
                    "Τα δεδομένα δεν διαγράφθηκαν"+'\n'+e.toString(),
                    "Σφάλμα Διαγραφής Δεδομένων",
                    JOptionPane.ERROR_MESSAGE);
        }
            BasicFunctions.disconnectFromBase(Conn);
        }
}
```

5.3 DBTuple.java

```
package doctor1.handlers;

public class DBTuple {

    //ΚΛΑΣΗ ΠΛΕΙΑΔΑΣ ΔΕΔΟΜΕΝΩΝ ΓΙΑ ΤΗΝ ΒΑΣΗ ΤΟΥ 1ΟΥ ΙΑΤΡΟΥ

    public String FName;
    public String LName;
    public String FatherName;
    public String Birthday;
    public String Amka;
    public String SecurityOrganization;
    public String Address;
    public String Phone;
    public String ExaminationDate;
    public int Disease;
    public String Remarks;
    public Float Sum;
}
```

6. Πακέτο doctor2.gui, κλάση DICOMFrame.java

```
package doctor2.gui;

import java.awt.Component;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.GridLayout;
import java.awt.Insets;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.StringTokenizer;

import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.ScrollPaneConstants;

import doctor2.handlers.DICOMDBDataFunctions;
import doctor2.handlers.DICOMTuple;
import doctor2.handlers.JpegToDICOM;

public class DICOMFrame extends JFrame implements ActionListener{

    //ΚΛΑΣΗ GUI 2ΟΥ ΙΑΤΡΟΥ

    private static final long serialVersionUID = 1L;

    JButton Create = new JButton("Εισαγωγή");
    JButton Load = new JButton("Αναζήτηση");
    JButton Update = new JButton("Ενημέρωση");
    JButton Delete = new JButton("Διαγραφή");
    JButton ClearAll = new JButton("Καθαρισμός Πεδίων");
    JButton InsertImage = new JButton("Εισαγωγή Εικόνας Εξέτασης");

    JLabel FName = new JLabel("Όνομα:");
    JLabel LName = new JLabel("Επίθετο:");
    JLabel FatherName = new JLabel("Πατρώνυμο:");
    JLabel Birthday = new JLabel("Ημερομηνία Γέννησης:");
    JLabel AMKA = new JLabel("A.M.K.A.:");
```

```
JLabel SecurityOrg = new JLabel ("Φορέας Ασφάλισης:");
JLabel Address = new JLabel("Διεύθυνση (Οδός/Αριθμός/Πόλη):");
JLabel Phone = new JLabel("Τηλέφωνο:");
JLabel ExaminationDate = new JLabel("Ημερομηνία Εξέτασης:");
JLabel ExaminationType = new JLabel("Τύπος Εξέτασης:");
JLabel Disease = new JLabel("Ασθένεια:");
JLabel Findings = new JLabel("Ευρήματα/Συμπτώματα/Παρατηρήσεις:");
JLabel BoneDensity = new JLabel("Οστική Πυκνότητα (T-Score):");
JLabel BloodFlow = new JLabel("Ροή Αίματος (ml/sec):");
JLabel BodyFat = new JLabel("Ολικό Ποσοστό Λίπους (%):");
JLabel Sum = new JLabel("Υπόλοιπο σε Ευρώ:");
JLabel ImageFile = new JLabel("Αρχείο .jpg ή .dcm (Θέση στο Δίσκο):");

JTextField FNameText = new JTextField(20);
JTextField LNameText = new JTextField(35);
JTextField FatherNameText = new JTextField(20);
JTextField BirthdayText = new JTextField(15);
JTextField AMKAText = new JTextField(20);
JTextField SecurityOrgText = new JTextField(20);
JTextField AddressText = new JTextField(35);
JTextField PhoneText = new JTextField(20);
JTextField ExaminationDateText = new JTextField(10);
JTextField ExaminationTypeText = new JTextField(5);
JTextField DiseaseText = new JTextField(35);
JTextArea FindingsText = new JTextArea(2,35);
JTextField BoneDensityText = new JTextField(5);
JTextField BloodFlowText = new JTextField(5);
JTextField BodyFatText = new JTextField(5);
JTextField SumText = new JTextField(10);
JTextField ImageFileText = new JTextField(30);

JPanel ImagePane = null;
ImageIcon Icon = null;

JScrollPane Scroll = new JScrollPane(FindingsText,
    ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED,
    ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);

GridBagLayout Gridbag = new GridBagLayout();
Insets Whitespace = new Insets (10,10,10,10);

DICOMTuple DicomWorkTuple;
String[][] DicomHeadData = new String[23][3];
String CurrentJpegPicturePath = "";

public DICOMFrame(){

    //ΚΑΤΑΣΚΕΥΗ GUI

    super ("IntegraHEALTH 1.0 by Nikos Christodoulou –
        Καρτέλα Στοιχείων Ασθενούς/Εξέτασης");
    Dimension ScreenSize = Toolkit.getDefaultToolkit().getScreenSize();
```

```
setBounds(0, 0, ScreenSize.width, ScreenSize.height-40);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setLayout(Gridbag);

Create.addActionListener(this);
Load.addActionListener(this);
Update.addActionListener(this);
Delete.addActionListener(this);
ClearAll.addActionListener(this);
InsertImage.addActionListener(this);
Update.setEnabled(false);
Delete.setEnabled(false);

JPanel CommandPane = new JPanel();
CommandPane.setLayout(new GridLayout (1, 4, 10,10));
CommandPane.add(Create);
CommandPane.add(Load);
CommandPane.add(Update);
CommandPane.add(Delete);
CommandPane.add(ClearAll);

JPanel ImageCommandPane = new JPanel();
ImageCommandPane.setLayout(new GridLayout (3,1,10,10));
ImageCommandPane.add(ImageFile);
ImageCommandPane.add(ImageFileText);
ImageCommandPane.add(InsertImage);

//ΤΟ ΣΤΟΙΧΕΙΟ ΣΤΟ ΟΠΟΙΟ ΘΑ ΑΠΕΙΚΟΝΙΖΕΤΑΙ Η ΕΙΚΟΝΑ-ΑΚΤΙΝΟΓΡΑΦΙΑ ΤΟΥ
ΑΣΘΕΝΗ
ImagePane = new JPanel(){
    private static final long serialVersionUID = 1L;

    protected void paintComponent (Graphics g){
        if (Icon!=null){
            g.drawImage(Icon.getImage(),0,0,null);
            super.paintComponent(g);
        }
    }
};
ImagePane.setOpaque(false);

JPanel HelpingPane = new JPanel();
HelpingPane.setLayout(new GridLayout(2,2,10,10));
BloodFlow.setHorizontalAlignment(E_RESIZE_CURSOR);
BodyFat.setHorizontalAlignment(E_RESIZE_CURSOR);
HelpingPane.add(BloodFlow);
HelpingPane.add(BloodFlowText);
HelpingPane.add(BodyFat);
HelpingPane.add(BodyFatText);

FindingsText.setLineWrap(true);
FindingsText.setWrapStyleWord(true);
```

```
addComponent(FName, 0, 0, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(FNameText, 1, 0, 1, 1, 40, 100, GridBagConstraints.HORIZONTAL,
    GridBagConstraints.WEST);
addComponent(LName, 0, 1, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(LNameText, 1, 1, 1, 1, 40, 100, GridBagConstraints.HORIZONTAL,
    GridBagConstraints.WEST);
addComponent(FatherName, 0, 2, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(FatherNameText, 1, 2, 1, 1, 40, 100, GridBagConstraints.HORIZONTAL,
    GridBagConstraints.WEST);
addComponent(Birthday, 0, 3, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(BirthdayText, 1, 3, 1, 1, 40, 100, GridBagConstraints.HORIZONTAL,
    GridBagConstraints.WEST);
addComponent(AMKA, 0, 4, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(AMKAText, 1, 4, 1, 1, 40, 100, GridBagConstraints.HORIZONTAL,
    GridBagConstraints.WEST);
addComponent(SecurityOrg, 0, 5, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(SecurityOrgText, 1, 5, 1, 1, 40, 100, GridBagConstraints.HORIZONTAL,
    GridBagConstraints.WEST);
addComponent(Address, 0, 6, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(AddressText, 1, 6, 1, 1, 40, 100, GridBagConstraints.HORIZONTAL,
    GridBagConstraints.WEST);
addComponent(Phone, 0, 7, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(PhoneText, 1, 7, 1, 1, 40, 100, GridBagConstraints.HORIZONTAL,
    GridBagConstraints.WEST);
addComponent(ExaminationDate, 0, 8, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(ExaminationDateText, 1, 8, 1, 1, 40, 100,
    GridBagConstraints.HORIZONTAL, GridBagConstraints.WEST);
addComponent(ExaminationType, 0, 9, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(ExaminationTypeText, 1, 9, 1, 1, 40, 100,
    GridBagConstraints.HORIZONTAL, GridBagConstraints.WEST);
addComponent(Disease, 0, 10, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(DiseaseText, 1, 10, 1, 1, 40, 100, GridBagConstraints.HORIZONTAL,
    GridBagConstraints.WEST);
addComponent(Findings, 0, 11, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(Scroll, 1, 11, 1, 1, 40, 100, GridBagConstraints.BOTH,
    GridBagConstraints.WEST);
addComponent(Sum, 0, 12, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
```

```
        addComponent(SumText, 1, 12, 1, 1, 40, 100, GridBagConstraints.HORIZONTAL,
            GridBagConstraints.WEST);
        addComponent(BoneDensity, 0, 13, 1, 1, 10, 100, GridBagConstraints.NONE,
            GridBagConstraints.EAST);
        addComponent(BoneDensityText,1, 13, 1, 1, 40, 100, GridBagConstraints.HORIZONTAL,
            GridBagConstraints.WEST);
        addComponent(CommandPane, 0, 14, 4, 1, 30, 100,
            GridBagConstraints.NONE,GridBagConstraints.CENTER);
        addComponent(ImageCommandPane, 2, 0, 2, 2, 50, 100,
            GridBagConstraints.NONE,GridBagConstraints.CENTER);
        addComponent(ImagePane, 2, 2, 2, 10, 50, 100,
            GridBagConstraints.BOTH,GridBagConstraints.CENTER);
        addComponent(HelpingPane, 2, 12, 10, 2, 50, 100,
            GridBagConstraints.BOTH,GridBagConstraints.CENTER);
    }

    private void addComponent(Component Component, int GridX, int GridY, int GridWidth,
        int GridHeight, int WeightX, int WeightY, int Fill, int Anchor){
        GridBagConstraints Constraints = new GridBagConstraints();
        Constraints.gridx = GridX;
        Constraints.gridy = GridY;
        Constraints.gridwidth = GridWidth;
        Constraints.gridheight = GridHeight;
        Constraints.weightx = WeightX;
        Constraints.weighty = WeightY;
        Constraints.fill = Fill;
        Constraints.anchor = Anchor;
        Constraints.insets = Whitespace;
        Gridbag.setConstraints(Component, Constraints);
        add(Component);
    }

    public void actionPerformed(ActionEvent evt) {

        //ΑΝΑΛΟΓΑ ΜΕ ΤΟ ΚΟΥΜΠΙ ΠΟΥ ΕΠΙΛΕΓΕΤΑΙ,
        //ΕΝΕΡΓΟΠΟΙΕΙΤΑΙ Η ΚΑΤΑΛΛΗΛΗ ΔΙΑΔΙΚΑΣΙΑ ΑΛΛΑΓΗΣ ΣΤΗ ΒΑΣΗ ΤΩΝ
        ΑΡΧΕΙΩΝ DICOM
        //INSERT, UPDATE, DELETE

        Object Source = evt.getSource();

        //ΕΛΕΓΧΟΣ ΜΟΡΦΗΣ ΗΜΕΡΟΜΗΝΙΩΝ (ΑΠΑΙΤΟΥΜΕΝΗ ΜΟΡΦΗ: EEEE-MM-HH)
        boolean ExamineDateFormatConfirmed = confirmDateFormat(
            ExaminationDateText.getText());
        boolean BirthDateFormatConfirmed = confirmDateFormat(BirthdayText.getText());

        //ΕΙΣΑΓΩΓΗ ΝΕΩΝ ΣΤΟΙΧΕΙΩΝ
        if (Source==Create){

            //ΠΕΡΙΠΤΩΣΕΙΣ ΛΑΝΘΑΣΜΕΝΩΝ ΣΤΟΙΧΕΙΩΝ ΓΙΑ ΕΙΣΑΓΩΓΗ
            if ((AMKAText.getText().isEmpty())
                |(FNameText.getText().isEmpty())
```

```
            |(LNameText.getText().isEmpty())
            |(Icon==null)){
            JOptionPane.showMessageDialog(null,
                "Το όνομα, το επίθετο, ή/και το A.M.K.A. είναι κενά"+'\n'+
                "ή λείπει το όνομα του αρχείου εικόνας!"+'\n'+
                "Συμπληρώστε τα και ξαναδοκιμάστε",
                "Σφάλμα Εισαγωγής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
        else if (FatherNameText.getText().contains("^")
            |FNameText.getText().contains("^")
            |LNameText.getText().contains("^")){
            JOptionPane.showMessageDialog(null,
                "Σε κάποιο από τα πεδία: Όνομα, Επίθετο, Πατρώνυμο"+'\n'+
                "χρησιμοποιείται ο μη αποδεκτός χαρακτήρας '^' !"+'\n'+
                "Χρησιμοποιείστε έναν άλλο και ξαναδοκιμάστε",
                "Input Data Error",
                JOptionPane.ERROR_MESSAGE);
        }
        else if (ExaminationDateText.getText().contains(".")
            |ExaminationDateText.getText().contains("/")
            |BirthdayText.getText().contains(".")
            |BirthdayText.getText().contains("/")){
            JOptionPane.showMessageDialog(null,
                "Χρησιμοποιήσατε . ή / για την ημερομηνία!"+'\n'+
                "Επιλέξτε ένα άλλο σύμβολο και ξαναδοκιμάστε",
                "Σφάλμα Εισαγωγής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
        else if ((!BirthDateFormatConfirmed)|(!ExamineDateFormatConfirmed)){
            JOptionPane.showMessageDialog(null,
                "Οι ημερομηνίες πρέπει να είναι στη μορφή EEEE-MM-HH!" +'\n'+
                "Συμπληρώστε τες σωστά και ξαναδοκιμάστε",
                "Σφάλμα Εισαγωγής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
        else{
            //ΦΟΡΤΩΣΗ ΤΩΝ ΣΤΟΙΧΕΙΩΝ ΑΠΟ ΤΟ GUI
            DicomWorkTuple = loadDicomTuple();
            //ΔΗΜΙΟΥΡΓΙΑ ΕΝΟΣ ΑΡΧΕΙΟΥ DICOM
            JpegToDICOM ConversionToDicom = new JpegToDICOM();
            ConversionToDicom.jpg2dcm(DicomWorkTuple, "Create");
            DicomWorkTuple.FileName = ConversionToDicom.getDicomFileName();
            //ΑΠΟΘΗΚΕΥΣΗ ΣΤΟΙΧΕΙΩΝ ΣΤΗ ΒΑΣΗ
            DICOMDBDataFunctions.createInDicomBase(
                DicomWorkTuple.Amka,
                DicomWorkTuple.FileName,
                ConversionToDicom.getDicomFile(),
                DicomWorkTuple.Sum);
        }
    }
```

```
//ΑΝΑΚΤΗΣΗ ΣΤΟΙΧΕΙΩΝ ΣΤΟ GUI ΑΠΟ ΤΗ ΒΑΣΗ
if (Source==Load){

    //ΠΕΡΙΠΤΩΣΕΙΣ ΣΤΟΙΧΕΙΩΝ ΓΙΑ ΑΝΑΖΗΤΗΣΗ

    if (AMKAText.getText().isEmpty()&ImageFileText.getText().isEmpty()){
        JOptionPane.showMessageDialog(null,
            "Το όνομα του αρχείου ή.και το Α.Μ.Κ.Α. είναι κενά!" +'\n'+
            "Συμπληρώστε ένα απο αυτά και ξαναδοκιμάστε",
            "Σφάλμα Εισαγωγής Δεδομένων",
            JOptionPane.ERROR_MESSAGE);
    }
    else {

        //ΑΝΑΖΗΤΗΣΗ ΒΑΣΕΙ ΑΜΚΑ Η ΟΝΟΜΑΤΟΣ ΑΡΧΕΙΟΥ
        if(!AMKAText.getText().isEmpty()){
            presentDicomTuple(DICOMDBDataFunctions.fetchFromDicomBase(
                AMKAText.getText(), "store"));
        }
        else{
            presentDicomTuple(DICOMDBDataFunctions.fetchFromDicomBase(
                ImageFileText.getText(), "store"));
        }
        Update.setEnabled(true);
        Delete.setEnabled(true);
        CurrentJpegPicturePath = DICOMDBDataFunctions.getJpegFile().getPath();
        Icon = new ImageIcon(CurrentJpegPicturePath);
        ImagePane.repaint();
        AMKAText.setEnabled(false);
        Create.setEnabled(false);
        Load.setEnabled(false);
    }
}

//ΕΝΗΜΕΡΩΣΗ ΣΤΟΙΧΕΙΩΝ ΣΤΗ ΒΑΣΗ
if (Source==Update){

    //ΠΕΡΙΠΤΩΣΕΙΣ ΛΑΝΘΑΣΜΕΝΩΝ ΣΤΟΙΧΕΙΩΝ ΓΙΑ ΕΝΗΜΕΡΩΣΗ

    if ((AMKAText.getText().isEmpty())
        |(FNameText.getText().isEmpty())
        |(LNameText.getText().isEmpty())
        |(Icon==null)){
        JOptionPane.showMessageDialog(null,
            "Το όνομα, το επίθετο, το Α.Μ.Κ.Α."+'\n'+
            "ή/και το όνομα του αρχείου εικόνας είναι κενά!"+'\n'+
            "Συμπληρώστε τα και ξαναδοκιμάστε",
            "Σφάλμα Εισαγωγής Δεδομένων",
            JOptionPane.ERROR_MESSAGE);
    }
    else if (FatherNameText.getText().contains("^")
        |FNameText.getText().contains("^")
```

```
            |LNameText.getText().contains("^")){
            JOptionPane.showMessageDialog(null,
                "Σε κάποιο από τα πεδία: Όνομα, Επίθετο, Πατρώνυμο"+'\n'+
                "χρησιμοποιείται ο μη αποδεκτός χαρακτήρας '^' !"+'\n'+
                "Χρησιμοποιείστε έναν άλλο και ξαναδοκιμάστε",
                "Input Data Error",
                JOptionPane.ERROR_MESSAGE);
        }
        else if (ExaminationDateText.getText().contains(".")
            |ExaminationDateText.getText().contains("/")
            |BirthdayText.getText().contains(".")
            |BirthdayText.getText().contains("/")){
            JOptionPane.showMessageDialog(null,
                "Χρησιμοποιήσατε . ή / για την ημερομηνία!"+'\n'+
                "Επιλέξτε ένα άλλο σύμβολο και ξαναδοκιμάστε",
                "Σφάλμα Εισαγωγής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
        else if ((!BirthDateFormatConfirmed)|(!ExamineDateFormatConfirmed)){
            JOptionPane.showMessageDialog(null,
                "Οι ημερομηνίες πρέπει να είναι στη μορφή EEEE-MM-HH!" +'\n'+
                "Συμπληρώστε τες σωστά και ξαναδοκιμάστε",
                "Σφάλμα Εισαγωγής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
        else{

            //ΦΟΡΤΩΣΗ ΣΤΟΙΧΕΙΩΝ ΑΠΟ ΤΟ GUI
            DicomWorkTuple = loadDicomTuple();
            String OldDicomFileName = DicomWorkTuple.FileName;
            if (!DicomWorkTuple.FileName.endsWith(".jpg")){
                DicomWorkTuple.FileName = CurrentJpegPicturePath;
            }

            //ΔΗΜΙΟΥΡΓΙΑ ΕΝΟΣ ΕΝΗΜΕΡΩΜΕΝΟΥ ΑΡΧΕΙΟΥ DICOM
            JpegToDICOM ConversionToDicom = new JpegToDICOM();
            ConversionToDicom.jpg2dcm(DicomWorkTuple, "Update");
            DicomWorkTuple.FileName = ConversionToDicom.getDicomFileName();

            //ΑΠΟΘΗΚΕΥΣΗ ΣΤΗ ΒΑΣΗ
            DICOMDBDataFunctions.updateInDicomBase(
                DicomWorkTuple, OldDicomFileName, ConversionToDicom.getDicomFile());
        }
    }

    if (Source==Delete){
        //ΔΙΑΓΡΑΦΗ ΣΤΟΙΧΕΙΩΝ ΑΠΟ ΤΗ ΒΑΣΗ
        DICOMDBDataFunctions.deleteFromDicomBase(
            AMKAText.getText(),AMKAText.getText().concat("_").
            concat(ExaminationDateText.getText()).concat(".dcm"));
        ClearAllFields();
    }
```

```
        //ΕΙΣΑΓΩΓΗ ΑΡΧΕΙΟΥ ΕΙΚΟΝΑΣ ΣΤΟ ΣΧΕΤΙΚΟ ΠΑΝΕΛ
        if (Source==InsertImage){
            if (ImageFileText.getText().contains(".dcm")){
                JOptionPane.showMessageDialog(null,
                    "Η εφαρμογή δέχεται μόνο JPEG αρχεία!",
                    "Σφάλμα Εισαγωγής Δεδομένων",
                    JOptionPane.ERROR_MESSAGE);
            }
            else if (ImageFileText.getText().isEmpty()&Icon==null){
                JOptionPane.showMessageDialog(null,
                    "Δεν έχετε δώσει κάποιο όνομα αρχείου!",
                    "Σφάλμα Εισαγωγής Δεδομένων",
                    JOptionPane.ERROR_MESSAGE);
            }
            else if (!((ImageFileText.getText().endsWith(".jpg"))
                |(ImageFileText.getText().endsWith(".JPG")))){
                JOptionPane.showMessageDialog(null,
                    "Η εφαρμογή δέχεται μόνο JPEG αρχεία!",
                    "Σφάλμα Εισαγωγής Δεδομένων",
                    JOptionPane.ERROR_MESSAGE);
            }
            else{
                Icon = new ImageIcon(ImageFileText.getText());
                ImagePane.repaint();
            }
        }

        //ΑΠΑΛΟΙΦΗ ΟΛΩΝ ΤΩΝ ΠΕΡΙΕΧΟΜΕΝΩΝ ΤΟΥ GUI
        if (Source==ClearAll){
            ClearAllFields();
        }
    }

    private void ClearAllFields(){

        //ΑΠΑΛΟΙΦΗ ΟΛΩΝ ΤΩΝ ΠΕΡΙΕΧΟΜΕΝΩΝ ΤΟΥ GUI

        FNameText.setText("");
        LNameText.setText("");
        FatherNameText.setText("");
        BirthdayText.setText("");
        AMKAText.setText("");
        if (!AMKAText.isEnabled()){
            AMKAText.setEnabled(true);
        }
        SecurityOrgText.setText("");
        AddressText.setText("");
        PhoneText.setText("");
        ExaminationDateText.setText("");
        ExaminationTypeText.setText("");
        DiseaseText.setText("");
```

```
        FindingsText.setText("");
        BoneDensityText.setText("");
        BloodFlowText.setText("");
        BodyFatText.setText("");
        SumText.setText("");
        ImageFileText.setText("");
        Icon = null;
        ImagePane.repaint();
        Create.setEnabled(true);
        Load.setEnabled(true);
        Update.setEnabled(false);
        Delete.setEnabled(false);
        InsertImage.setEnabled(true);
    }

    private boolean confirmDateFormat(String InDate) {

        //ΕΛΕΓΧΟΣ ΜΟΡΦΗΣ ΗΜΕΡΟΜΗΝΙΩΝ (ΕΕΕΕ-ΜΜ-ΗΗ)

        int Day, Month, Year;
        if(InDate.length()!=10){
            return false;
        }
        StringTokenizer stk = new StringTokenizer(InDate, InDate.substring(4, 5));
        if (stk.countTokens()!=3){
            return false;
        }
        else{
            Year = Integer.parseInt(stk.nextToken());
            Month = Integer.parseInt(stk.nextToken());
            Day = Integer.parseInt(stk.nextToken());
            DateFormat yearFormat = new SimpleDateFormat("yyyy");
            Date date = new Date();
            int CurrentYear = Integer.parseInt(yearFormat.format(date));
            if (Day>31|Day<1){
                return false;
            }
            else if (Month>12|Month<1){
                return false;
            }
            else if(((Month==2)|(Month==4)|(Month==6)|(Month==9)|(Month==11))&(Day==31)){
                return false;
            }
            else if((Month==2)&((Day==30)|(Day==31))){
                return false;
            }
            else if((Month==2)&(Year%4!=0)&(Day==29)){
                return false;
            }
            else if((Month==2)&(Year%4==0)&(Year%400!=0)&(Day==29)){
                return false;
            }
```

```
            else if ((Year<(CurrentYear-150))|(Year>CurrentYear)){
                return false;
            }
        }
        return true;
    }

    private DICOMTuple loadDicomTuple(){

        //ΦΟΡΤΩΣΗ ΣΤΟΙΧΕΙΩΝ ΑΠΟ ΤΟ GUI ΣΕ ΜΙΑ ΠΛΕΙΑΔΑ

        DICOMTuple DicomOutTuple = new DICOMTuple();
        DicomOutTuple.FName = FNameText.getText();
        DicomOutTuple.LName = LNameText.getText();
        DicomOutTuple.FatherName = FatherNameText.getText();
        DicomOutTuple.Birthday = BirthdayText.getText();
        DicomOutTuple.Amka = AMKAText.getText();
        DicomOutTuple.SecurityOrganization = SecurityOrgText.getText();
        DicomOutTuple.Address = AddressText.getText();
        DicomOutTuple.Phone = PhoneText.getText();
        DicomOutTuple.ExaminationDate = ExaminationDateText.getText();
        DicomOutTuple.ExaminationType = ExaminationTypeText.getText();
        DicomOutTuple.Disease = DiseaseText.getText();
        DicomOutTuple.Remarks = FindingsText.getText();
        if (BoneDensityText.getText().isEmpty()){

            //ΠΡΟΚΑΘΟΡΙΣΜΕΝΗ ΤΙΜΗ -0 ΓΙΑ ΟΣΕΣ ΜΕΤΡΗΣΕΙΣ ΒΙΟΙΑΤΡΙΚΩΝ
            ΔΕΔΟΜΕΝΩΝ ΔΕΝ ΔΗΛΩΝΟΝΤΑΙ ΣΤΟ GUI
            DicomOutTuple.BoneDensity = Float.parseFloat("-0");
        }
        else{
            DicomOutTuple.BoneDensity = Float.parseFloat(BoneDensityText.getText());
        }
        if (BloodFlowText.getText().isEmpty()){
            DicomOutTuple.BloodFlow = Float.parseFloat("-0");
        }
        else{
            DicomOutTuple.BloodFlow = Float.parseFloat(BloodFlowText.getText());
        }
        if (BodyFatText.getText().isEmpty()){
            DicomOutTuple.BodyFat = Float.parseFloat("-0");
        }
        else{
            DicomOutTuple.BodyFat = Float.parseFloat(BodyFatText.getText());
        }
        if (!(SumText.getText().isEmpty())){
            DicomOutTuple.Sum = Float.parseFloat(SumText.getText());
        }else{
            DicomOutTuple.Sum = 0f;
        }
        DicomOutTuple.FileName = ImageFileText.getText();
        return DicomOutTuple;
```

```
    }

    private void presentDicomTuple(DICOMTuple DicomInTuple){

        //ΠΑΡΟΥΣΙΑΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

        DICOMTuple DicomWorkTuple = DicomInTuple;
        FNameText.setText(DicomWorkTuple.FName);
        LNameText.setText(DicomWorkTuple.LName);
        FatherNameText.setText(DicomWorkTuple.FatherName);
        BirthdayText.setText(DicomWorkTuple.Birthday);
        AMKAText.setText(DicomWorkTuple.Amka);
        SecurityOrgText.setText(DicomWorkTuple.SecurityOrganization);
        AddressText.setText(DicomWorkTuple.Address);
        PhoneText.setText(DicomWorkTuple.Phone);
        ExaminationDateText.setText(DicomWorkTuple.ExaminationDate);
        ExaminationTypeText.setText(DicomWorkTuple.ExaminationType);
        DiseaseText.setText(DicomWorkTuple.Disease);
        FindingsText.setText(DicomWorkTuple.Remarks);
        BoneDensityText.setText(DicomWorkTuple.BoneDensity.toString());
        BloodFlowText.setText(DicomWorkTuple.BloodFlow.toString());
        BodyFatText.setText(DicomWorkTuple.BodyFat.toString());
        SumText.setText(DicomWorkTuple.Sum.toString());
        ImageFileText.setText(DicomWorkTuple.FileName);
    }
}
```

7. Πακέτο doctor2.handlers

7.1 BasicFunctions.java

Η κλάση είναι ίδια με αυτήν του πακέτου doctor1.handlers. Αλλάζει όπου doctor1 ➔ doctor2.

7.2 DICOMDBDataFunctions.java

```
package doctor2.handlers;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.sql.Blob;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
```

```
import java.util.StringTokenizer;

import javax.swing.JOptionPane;

import org.dcm4che2.data.DicomObject;
import org.dcm4che2.io.DicomInputStream;

import doctor2.handlers.BasicFunctions;

public class DICOMDBDataFunctions {

    //ΚΛΑΣΗ ΜΕΘΟΔΩΝ ΑΛΛΑΓΩΝ ΤΗΣ ΒΑΣΗΣ ΤΟΥ 2ΟΥ ΙΑΤΡΟΥ

    static File IncomingJpgFile = null;

    public static void createInDicomBase(
        String InAmka, String InDicomFileName, File InDicomFile, Float InSum){

        //ΕΙΣΑΓΩΓΗ ΝΕΩΝ ΣΤΟΙΧΕΙΩΝ

        //ΦΟΡΤΩΣΗ ΤΟΥ ΑΡΧΕΙΟΥ DICOM ΣΕ ΕΝΑ ΡΕΥΜΑ ΕΙΣΟΔΟΥ
        InputStream StreamedDicomFile = null;
        int FileLength = (int) InDicomFile.length();
        try {
            StreamedDicomFile = new FileInputStream(InDicomFile);
        } catch (FileNotFoundException FNFe) {
            JOptionPane.showMessageDialog(null,
                "Δεν βρέθηκε αρχείο DICOM προς αποθήκευση"+'\n'+FNFe.toString(),
                "Σφάλμα Εισαγωγής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }

        //SQL STATEMENT
        BasicFunctions.loadDriver();
        Connection Conn = BasicFunctions.connectToDicomBase();
        try {
            java.sql.PreparedStatement PrepSt = Conn.prepareStatement(
                "INSERT INTO "+
                "dicom_elements(amka, dicomFileName, dicomFile, sum)" +
                "VALUES(?, ?, ?, ?)");
            PrepSt.setString(1, InAmka);
            PrepSt.setString(2, InDicomFileName);
            PrepSt.setBinaryStream(3, StreamedDicomFile, FileLength);
            PrepSt.setFloat(4, InSum);
            PrepSt.executeUpdate();
            PrepSt.close();
            StreamedDicomFile.close();
        } catch (SQLException SQLe){
            JOptionPane.showMessageDialog(null,
                "Τα δεδομένα δεν γράφτηκαν στη βάση"+'\n'+SQLe.toString(),
                "Σφάλμα Εισαγωγής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
```

```
        } catch (IOException IOe){
            JOptionPane.showMessageDialog(null,
                "Δεν ήταν δυνατή η ετοιμασία του αρχείου DICOM για τηβάση"+'\n'+IOe.toString(),
                "Σφάλμα Εισαγωγής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
        BasicFunctions.disconnectFromBase(Conn);
    }

    public static DICOMTuple fetchFromDicomBase(String InElement, String CaseFlag){

        //ΑΝΑΚΤΗΣΗ ΔΕΔΟΜΕΝΩΝ

        DICOMTuple ResultDicomTuple = new DICOMTuple();
        int response = 1;
        String [][] IncomingDicomHeadData = new String[26][3];

        BasicFunctions.loadDriver();
        Connection Conn = BasicFunctions.connectToDicomBase();
        try{
            Statement St = Conn.createStatement();

            //ΑΝΑΖΗΤΗΣΗ ΒΑΣΕΙ ΟΝΟΜΑΤΟΣ ΑΡΧΕΙΟΥ Η ΑΜΚΑ
            ResultSet ResSet = St.executeQuery(
                "SELECT * " +
                "FROM dicom_elements " +
                "WHERE dicomFileName LIKE '%"+ InElement + "%'" +
                "OR amka = '"+ InElement +"'");

            //ΒΡΟΧΟΣ ΓΙΑ ΤΗΝ ΕΠΙΛΟΓΗ ΤΟΥ ΕΠΙΘΥΜΗΤΟΥ ΑΡΧΕΙΟΥ
            while (ResSet.next() && (response == 1)){
                if (CaseFlag.equals("store")){
                    response = JOptionPane.showOptionDialog(null,
                        "Βρέθηκε το αρχείο: "+ResSet.getString(2)+'\n'+
                        "του ασθενή με ΑΜΚΑ: "+ResSet.getString(1)+'\n'+
                        "Είναι αυτό το επιθυμητό αποτέλεσμα;",
                        "Εύρεση Πολλαπλών Αποτελεσμάτων",
                        JOptionPane.YES_NO_OPTION,
                        JOptionPane.QUESTION_MESSAGE,
                        null, null, null);
                }
                else{
                    response = 0;
                }
                if (response==0){
                    ResultDicomTuple.Amka = ResSet.getString(1);
                    ResultDicomTuple.FileName = ResSet.getString(2);
                    ResultDicomTuple.Sum = ResSet.getFloat(4);

                    //ΦΟΡΤΩΣΗ ΤΟΥ ΑΡΧΕΙΟΥ DICOM ΑΠΟ ΤΗ ΜΟΡΦΗ BLOB
                    Blob DicomFileBlob = ResSet.getBlob(3);
```

```
                //ΑΝΑΔΗΜΙΟΥΡΓΙΑ ΤΟΥ ΑΡΧΕΙΟΥ DICOM
                File LoadedDicomFile =
                    new File(".\\DICOMFILES\\"+ResultDicomTuple.FileName);
                OutputStream DicomFileWriter = new FileOutputStream(LoadedDicomFile);
                readFromBlob(DicomFileBlob, DicomFileWriter);
                DicomFileWriter.close();

                //ΕΞΑΓΩΓΗ ΤΗΣ ΕΙΚΟΝΑΣ
                DICOMToJpeg ConversionToJpeg = new DICOMToJpeg();
                ConversionToJpeg.dcm2jpg(LoadedDicomFile);
                IncomingJpgFile = ConversionToJpeg.getJpegFile();

                //ΦΟΡΤΩΣΗ ΤΟΥ ΑΝΑΣΥΣΤΑΘΕΝΤΟΣ ΑΡΧΕΙΟΥ DICOM ΣΕ ΕΝΑ
                ΡΕΥΜΑ ΕΙΣΟΔΟΥ
                DicomObject Object = null;
                try {
                    DicomInputStream Dis = new DicomInputStream(LoadedDicomFile);
                    Object = Dis.readDicomObject();
                    Dis.close();
                } catch (Exception e) {
                    JOptionPane.showMessageDialog(null,
                        "Το αρχείο DICOM δεν μπορούσε να φορτωθεί"+'\n'+e.getMessage(),
                        "Σφάλμα Φόρτωσης Αρχείου DICOM",
                        JOptionPane.ERROR_MESSAGE);
                }

                //ΕΞΑΓΩΓΗ ΣΤΟΙΧΕΙΩΝ ΑΣΘΕΝΗ ΑΠΟ ΤΗΝ ΕΠΙΚΕΦΑΛΙΔΑ ΤΟΥ
                ΑΡΧΕΙΟΥ DICOM
                ListDICOMHeader list = new ListDICOMHeader();
                IncomingDicomHeadData = list.listHeader(Object);
                ResultDicomTuple.Disease = IncomingDicomHeadData[5][2];
                ResultDicomTuple.SecurityOrganization = IncomingDicomHeadData[8][2];
                StringTokenizer stk =  new StringTokenizer(
                    IncomingDicomHeadData[9][2],"^");
                ResultDicomTuple.LName = stk.nextToken();
                ResultDicomTuple.FName = stk.nextToken();
                ResultDicomTuple.FatherName = stk.nextToken();
                ResultDicomTuple.Birthday = IncomingDicomHeadData[7][2];
                ResultDicomTuple.Address = IncomingDicomHeadData[10][2];
                ResultDicomTuple.Phone = IncomingDicomHeadData[11][2];
                ResultDicomTuple.ExaminationType = IncomingDicomHeadData[1][2];
                ResultDicomTuple.Remarks = IncomingDicomHeadData[12][2];
                ResultDicomTuple.BoneDensity = Float.parseFloat(
                    IncomingDicomHeadData[13][2]);
                String [] OtherElements = IncomingDicomHeadData[16][2].split("\\|");
                ResultDicomTuple.BloodFlow = Float.parseFloat(OtherElements[0]);
                ResultDicomTuple.BodyFat = Float.parseFloat(OtherElements[1]);
                ResultDicomTuple.ExaminationDate = IncomingDicomHeadData[25][2];
                break;
            }
        }
        if (response==1){
```

```
                ResultDicomTuple.FName = "N/A";
                ResultDicomTuple.LName = "N/A";
                ResultDicomTuple.FatherName = "N/A";
                ResultDicomTuple.Birthday = "N/A";
                ResultDicomTuple.Amka = "N/A";
                ResultDicomTuple.SecurityOrganization = "N/A";
                ResultDicomTuple.ExaminationDate = "N/A";
                ResultDicomTuple.ExaminationType = "N/A";
                ResultDicomTuple.Disease = "N/A";
                ResultDicomTuple.Remarks = "N/A";
                ResultDicomTuple.BoneDensity = Float.parseFloat("-0");
                ResultDicomTuple.BloodFlow = Float.parseFloat("-0");
                ResultDicomTuple.BodyFat = Float.parseFloat("-0");
                ResultDicomTuple.Address = "N/A";
                ResultDicomTuple.Phone = "N/A";
                ResultDicomTuple.Sum = 0f;
                JOptionPane.showMessageDialog(null, "Δεν βρέθηκε τίποτα");
            }
            St.close();
        }catch (Exception e){
            JOptionPane.showMessageDialog(null,
                "Τα δεδομένα δεν μπορούν να ανακτηθούν"+'\n'+e.toString(),
                "Σφάλμα Ανάκτησης Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
        BasicFunctions.disconnectFromBase(Conn);
        return ResultDicomTuple;
    }

    public static void readFromBlob(Blob Blob, OutputStream Out){

        //ΑΝΑΓΝΩΣΗ ΕΝΟΣ ΑΝΤΙΚΕΙΜΕΝΟΥ BLOB (Binary Large OBject) ΑΠΟ ΜΙΑ ΒΑΣΗ

        try{
            InputStream In = Blob.getBinaryStream();
            int Length = -1;
            byte[] Buf = new byte[1024];
            while ((Length = In.read(Buf)) != -1) {
                Out.write(Buf, 0, Length);
            }
            In.close();
        } catch (SQLException SQLe){
            JOptionPane.showMessageDialog(null,
                "Τα δεδομένα δεν ανακτήθηκαν απο τη βάση"+'\n'+SQLe.toString(),
                "Σφάλμα Ανάκτησης Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        } catch (IOException IOe){
            JOptionPane.showMessageDialog(null,
                "Δεν υπάρχει έτοιμα αρχείο DICOM για να δεχτεί τα δεδομένα"+'\n'+IOe.toString(),
                "Σφάλμα Ανάκτησης Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
```

```
}

public static File getJpegFile(){

    //ΕΞΑΓΩΓΗ ΤΟΥ ΑΝΑΚΤΗΜΕΝΟΥ ΑΡΧΕΙΟΥ JPEG
    return IncomingJpgFile;
}

public static void updateInDicomBase(
    DICOMTuple InTuple, String OldDicomFileName, File IncomingDicomFile){

    //ΕΝΗΜΕΡΩΣΗ ΔΕΔΟΜΕΝΩΝ

    BasicFunctions.loadDriver();
    Connection Conn = BasicFunctions.connectToDicomBase();
    try{

        //ΦΟΡΤΩΣΗ ΤΟΥ ΑΡΧΕΙΟΥ DICOM ΣΕ ΕΝΑ ΡΕΥΜΑ ΕΙΣΟΔΟΥ
        InputStream StreamedDicomFile = null;
        int FileLength = (int) IncomingDicomFile.length();
        try {
            StreamedDicomFile = new FileInputStream(IncomingDicomFile);
        } catch (FileNotFoundException FNFe) {
            JOptionPane.showMessageDialog(null,
                "Δεν βρέθηκε αρχείο DICOM προς αποθήκευση"+'\n'+FNFe.toString(),
                "Σφάλμα Εισαγωγής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }

        //SQL STATEMENT
        java.sql.PreparedStatement PrepSt = Conn.prepareStatement(
            "UPDATE dicom_elements SET dicomFileName = '" + InTuple.FileName + "', " +
                "dicomFile = ?," +
                "sum = '" + InTuple.Sum + "' " +
            "WHERE amka LIKE '%"+ InTuple.Amka+"%'"+
            "AND dicomFileName = '"+ OldDicomFileName+"'");
        PrepSt.setBinaryStream(1, StreamedDicomFile, FileLength);
        PrepSt.executeUpdate();
    }catch (SQLException e){
        JOptionPane.showMessageDialog(null,
            "Τα δεδομένα δεν ενημερώθηκαν"+'\n'+e.toString(),
            "Σφάλμα Ενημέρωσης Δεδομένων",
            JOptionPane.ERROR_MESSAGE);
     }
    BasicFunctions.disconnectFromBase(Conn);
}

public static void deleteFromDicomBase(String InAmka, String InFileName){

    //ΔΙΑΓΡΑΦΗ ΔΕΔΟΜΕΝΩΝ

    BasicFunctions.loadDriver();
```

```
        Connection Conn = BasicFunctions.connectToDicomBase();
        try{
            java.sql.PreparedStatement PrepSt = Conn.prepareStatement(
                "DELETE FROM dicom_elements " +
                "WHERE(amka = '" + InAmka + "') " +
                "AND dicomFileName LIKE '%"+ InFileName +"%'");
            PrepSt.executeUpdate();
            PrepSt.close();
         }catch (SQLException e){
            JOptionPane.showMessageDialog(null,
                "Τα δεδομένα δεν διαγράφθηκαν"+'\n'+e.toString(),
                "Σφάλμα Διαγραφής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
    }
        BasicFunctions.disconnectFromBase(Conn);
    }
}
```

7.3 DICOMToJpeg.java

```
package doctor2.handlers;

import java.awt.image.BufferedImage;
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

import javax.imageio.ImageIO;
import javax.imageio.ImageReader;
import javax.imageio.stream.ImageInputStream;
import javax.swing.JOptionPane;

import org.dcm4che2.imageio.plugins.dcm.DicomImageReadParam;
import com.sun.image.codec.jpeg.JPEGCodec;
import com.sun.image.codec.jpeg.JPEGImageEncoder;

public class DICOMToJpeg {

    //ΕΞΑΓΩΓΗ ΕΙΚΟΝΑΣ JPEG ΑΠΟ ΕΝΑ ΑΡΧΕΙΟ DICOM

    static File jpgDestination = null;

    public void dcm2jpg(File InDicomFile){

        //ΔΗΜΙΟΥΡΓΙΑ ΕΝΟΣ ΑΡΧΕΙΟΥ JPEG

        BufferedImage MyJpegImage = null;

        //ΦΟΡΤΩΣΗ ΤΩΝ READER ΕΙΚΟΝΩΝ ΓΙΑ ΤΟ ΠΡΟΤΥΠΟ DICOM
        java.util.Iterator<ImageReader> Iter = ImageIO.getImageReadersByFormatName("DICOM");
```

```
        ImageReader Reader = (ImageReader) Iter.next();

        //ΦΟΡΤΩΣΗ ΤΩΝ ΠΑΡΑΜΕΤΡΩΝ ΤΩΝ READERS
        DicomImageReadParam Param = (DicomImageReadParam) Reader.getDefaultReadParam();
        try {

            //ΦΟΡΤΩΣΗ ΤΟΥ ΑΡΧΕΙΟΥ DICOM ΣΕ ΕΙΔΙΚΟ ΡΕΥΜΑ ΕΙΣΟΔΟΥ ΕΙΚΟΝΑΣ
            ImageInputStream Iis = ImageIO.createImageInputStream(InDicomFile);

            //ΤΟ ΡΕΥΜΑ ΕΙΣΟΔΟΥ ΔΙΝΕΤΑΙ ΣΤΟΝ IMAGE READER
            Reader.setInput(Iis, false);

            //ΕΞΑΓΩΓΗ ΤΗΣ JPEG ΕΙΚΟΝΑΣ
            MyJpegImage = Reader.read(0, Param);
            Iis.close();
            if (MyJpegImage == null) {
                JOptionPane.showMessageDialog(null,
                    "Η εικόνα του αρχείου DICOM δεν μπορούσε να αναγνωσθεί",
                    "Σφάλμα Ανάγνωσης Αρχείου DICOM",
                    JOptionPane.ERROR_MESSAGE);
                return;
            }

            //ΔΗΜΙΟΥΡΓΙΑ ΤΟΥ JPEG ΑΡΧΕΙΟΥ
            jpgDestination = new File(InDicomFile.getPath()
                    .replace("DICOMFILES", "JPEGHELPFILES")
                    .replace(".dcm", ".jpg"));

            //ΤΟΠΟΘΕΤΗΣΗ ΤΟΥ JPEG ΑΡΧΕΙΟΥ ΣΕ ΕΝΑ ΡΕΥΜΑ ΕΞΟΔΟΥ
            OutputStream output = new BufferedOutputStream(
                new FileOutputStream(jpgDestination));

            //ΚΩΔΙΚΟΠΟΙΗΤΗΣ JPEG ΠΟΥ ΣΥΝΔΥΑΖΕΤΑΙ ΜΕ ΤΟ ΡΕΥΜΑ ΕΞΟΔΟΥ
            JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(output);

            //Η JPEG ΕΙΚΟΝΑ "ΦΟΡΤΩΝΕΤΑΙ" ΣΤΟ JPEG ΑΡΧΕΙΟ
            encoder.encode(MyJpegImage);
            output.close();
        }
        catch(IOException e) {
            JOptionPane.showMessageDialog(null,
                "Το αρχείο DICOM δεν μπορούσε να αναγνωσθεί",
                "Σφάλμα Ανάγνωσης Αρχείου DICOM",
                JOptionPane.ERROR_MESSAGE);
        }
    }
    public File getJpegFile(){

        //ΕΞΑΓΩΓΗ ΤΟΥ JPEG ΑΡΧΕΙΟΥ ΠΡΟΣ ΑΛΛΕΣ ΚΛΑΣΕΙΣ

        return jpgDestination;
    }
```

```
}
```

7.4 DICOMTuple.java

```
package doctor2.handlers;

public class DICOMTuple {

    //ΚΛΑΣΗ ΠΛΕΙΑΔΑΣ ΓΙΑ ΤΗΝ ΜΕΤΑΦΟΡΑ ΔΕΔΟΜΕΝΩΝ ΑΠΟ ΚΑΙ ΠΡΟΣ ΑΡΧΕΙΑ
    DICOM

    public String FName;
    public String LName;
    public String FatherName;
    public String Birthday;
    public String Amka;
    public String SecurityOrganization;
    public String Address;
    public String Phone;
    public String ExaminationDate;
    public String ExaminationType;
    public String Disease;
    public String Remarks;
    public Float BoneDensity;
    public Float BloodFlow;
    public Float BodyFat;
    public Float Sum;
    public String FileName;
}
```

7.5 JpegToDICOM.java

```
package doctor2.handlers;

import java.awt.image.BufferedImage;
import java.io.BufferedInputStream;
import java.io.BufferedOutputStream;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.Date;
import java.util.StringTokenizer;

import javax.imageio.ImageIO;
import javax.swing.JOptionPane;

import org.dcm4che2.data.BasicDicomObject;
import org.dcm4che2.data.DicomObject;
import org.dcm4che2.data.Tag;
import org.dcm4che2.data.UID;
```

```
import org.dcm4che2.data.VR;
import org.dcm4che2.io.DicomOutputStream;
import org.dcm4che2.util.UIDUtils;

public class JpegToDICOM {

    //ΚΛΑΣΗ ΔΗΜΙΟΥΡΓΙΑΣ ΑΡΧΕΙΟΥ DICOM ΑΠΟ ΔΕΔΟΜΕΝΑ ΚΑΙ ΜΙΑ ΕΙΚΟΝΑ JPEG

    File DcmFile = null;
    String DcmFileName = "";
    StringTokenizer Dcmf = null;

    public void jpg2dcm (DICOMTuple DicomInTuple, String Case){

        //ΔΗΜΙΟΥΡΓΙΑ ΤΟΥ ΑΡΧΕΙΟΥ DICOM ΚΑΙ ΤΟΥ ΟΝΟΜΑΤΟΣ ΤΟΥ ΑΠΟ ΤΑ
        ΣΤΟΙΧΕΙΑ ΤΗΣ ΠΛΕΙΑΔΑΣ
        File JpgSource = new File(DicomInTuple.FileName);
        String LastToken = "";
        if (Case.equals("Update")){
            Dcmf = new StringTokenizer (DicomInTuple.FileName, "\\");
            while (Dcmf.hasMoreTokens()){
                LastToken = Dcmf.nextToken();
            }
            DcmFileName = DicomInTuple.ExaminationDate.concat("_").concat(
                LastToken.substring(0, LastToken.length()-4)).concat(".dcm");
        }
        if (Case.equals("Update")){
            Dcmf = new StringTokenizer (DicomInTuple.FileName, "_");
            while (Dcmf.hasMoreTokens()){
                LastToken = Dcmf.nextToken();
            }
            DcmFileName = DicomInTuple.ExaminationDate.concat("_").concat(
                LastToken.replace(".jpg", ".dcm"));
        }

        DcmFile = new File(".\\DICOMFILES\\"+DcmFileName);
        String PatientFullName = DicomInTuple.LName.concat("^").concat(
            DicomInTuple.FName).concat("^").concat(DicomInTuple.FatherName);

        try{

            //ΑΝΑΓΝΩΣΗ ΤΗΣ ΕΙΚΟΝΑΣ JPEG
            BufferedImage jpegImage = ImageIO.read(JpgSource);
            if (jpegImage == null){
                throw new Exception("Μη έγκυρο αρχείο");
            }

            //ΔΕΔΟΜΕΝΑ ΕΙΚΟΝΑΣ ΤΟΥ JPEG ΑΡΧΕΙΟΥ
            int ColorComponents = jpegImage.getColorModel().getNumColorComponents();
            int BitsPerPixel = jpegImage.getColorModel().getPixelSize();
            int BitsAllocated = (BitsPerPixel / ColorComponents);
            int SamplesPerPixel = ColorComponents;
```

```
//ΤΑ ΠΑΡΑΚΑΤΩ ΣΤΟΙΧΕΙΑ ΘΑ ΣΥΜΠΕΡΙΛΗΦΘΟΥΝ ΣΤΗΝ ΕΠΙΚΕΦΑΛΙΔΑ
ΤΟΥ ΑΡΧΕΙΟΥ
//ΠΕΡΙΛΑΜΒΑΝΟΝΤΑΙ ΚΑΙ ΤΑ ΣΤΟΙΧΕΙΑ ΕΙΚΟΝΑΣ
//ΚΑΙ ΑΥΤΑ ΤΟΥ ΑΣΘΕΝΗ ΠΟΥ ΥΠΗΡΧΑΝ ΣΤΗΝ ΠΛΕΙΑΔΑ - ΟΡΙΣΜΑ ΤΗΣ
ΜΕΘΟΔΟΥ

DicomObject Dicom = new BasicDicomObject();
Dicom.putString(Tag.SpecificCharacterSet, VR.CS, "ISO_IR 100");
Dicom.putString(Tag.PhotometricInterpretation,
    VR.CS, SamplesPerPixel == 3 ? "YBR_FULL_422" : "MONOCHROME2");
Dicom.putInt(Tag.SamplesPerPixel, VR.US, SamplesPerPixel);
Dicom.putInt(Tag.Rows, VR.US, jpegImage.getHeight());
Dicom.putInt(Tag.Columns, VR.US, jpegImage.getWidth());
Dicom.putInt(Tag.BitsAllocated, VR.US, BitsAllocated);
Dicom.putInt(Tag.BitsStored, VR.US, BitsAllocated);
Dicom.putInt(Tag.HighBit, VR.US, BitsAllocated-1);
Dicom.putInt(Tag.PixelRepresentation, VR.US, 0);
Dicom.putString(Tag.TreatmentDate, VR.DA, DicomInTuple.ExaminationDate);
Dicom.putString(Tag.PatientBirthName, VR.PN, PatientFullName);
Dicom.putString(Tag.PatientBirthDate, VR.DA, DicomInTuple.Birthday);
Dicom.putString(Tag.PatientID, VR.ST, DicomInTuple.Amka);
Dicom.putString(Tag.PatientInsurancePlanCodeSequence,
    VR.ST, DicomInTuple.SecurityOrganization);
Dicom.putString(Tag.PatientAddress, VR.ST, DicomInTuple.Address);
Dicom.putString(Tag.PatientTelephoneNumbers, VR.ST, DicomInTuple.Phone);
Dicom.putString(Tag.AdmittingDiagnosesDescription, VR.ST, DicomInTuple.Disease);
Dicom.putString(Tag.PatientComments, VR.LT, DicomInTuple.Remarks);
Dicom.putString(Tag.BodyPartThickness,
    VR.ST, DicomInTuple.BoneDensity.toString());
Dicom.putString(Tag.OtherStudyNumbers,
    VR.ST, (DicomInTuple.BloodFlow.toString()).concat("|").concat(
    DicomInTuple.BodyFat.toString()));
Dicom.putString(Tag.ImageType, VR.ST, DicomInTuple.ExaminationType);
Dicom.putDate(Tag.InstanceCreationDate, VR.DA, new Date());
Dicom.putDate(Tag.InstanceCreationTime, VR.TM, new Date());
Dicom.putString(Tag.StudyInstanceUID, VR.UI, UIDUtils.createUID());
Dicom.putString(Tag.SeriesInstanceUID, VR.UI, UIDUtils.createUID());
Dicom.putString(Tag.SOPInstanceUID, VR.UI, UIDUtils.createUID());
Dicom.initFileMetaInformation(UID.JPEGBaseline1);

//ΑΝΟΙΓΜΑ ΕΝΟΣ ΡΕΥΜΑΤΟΣ ΕΞΟΔΟΥ ΓΙΑ ΤΟ ΑΡΧΕΙΟ DICOM
//ΚΑΙ ΕΓΓΡΑΦΗ ΣΕ ΑΥΤΟ ΤΩΝ ΔΕΔΟΜΕΝΩΝ ΤΗΣ ΕΠΙΚΕΦΑΛΙΔΑΣ
FileOutputStream Fos = new FileOutputStream(DcmFile);
DicomOutputStream Dos = new DicomOutputStream(Fos);
Dos.writeDicomFile(Dicom);
Dos.writeHeader(Tag.PixelData, VR.OB, -1);
Dos.writeHeader(Tag.Item, null, 0);
int jpgLen = (int) JpgSource.length();
Dos.writeHeader(Tag.Item, null, (jpgLen+1)&~1);

//ΦΟΡΤΩΣΗ ΤΟΥ JPEG ΑΡΧΕΙΟΥ ΣΕ ΔΙΚΟ ΤΟΥ ΡΕΥΜΑ ΕΙΣΟΔΟΥ...
```

```
                FileInputStream Fis = new FileInputStream(JpgSource);
                DataInputStream Dis = new DataInputStream(Fis);

                //...ΚΑΙ ΕΓΓΡΑΦΗ ΤΟΥ ΣΤΟ ΡΕΥΜΑ ΕΞΟΔΟΥ ΤΟΥ ΑΡΧΕΙΟΥ DICOM
                byte[] buffer = new byte[65536];
                int b;
                while ((b = Dis.read(buffer)) > 0) {
                    Dos.write(buffer, 0, b);
                }

                //ΕΞΑΣΦΑΛΙΣΗ ΟΤΙ ΤΟ ΜΗΚΟΣ ΤΟΥ ΤΜΗΜΑΤΟΣ ΤΟΥ ΑΡΧΕΙΟΥ ΜΕ ΤΑ
                ΔΕΔΟΜΕΝΑ JPEG ΕΧΕΙ ΖΥΓΟ ΜΗΚΟΣ
                //ΓΙΝΕΤΑΙ ΚΑΙ ΣΥΜΠΛΗΡΩΣΗ ΜΕ ΜΗΔΕΝΙΚΑ (PADDING) ΑΝ ΧΡΕΙΑΣΤΕΙ
                if ((jpgLen&1) != 0) Dos.write(0);
                Dos.writeHeader(Tag.SequenceDelimitationItem, null, 0);

                //ΚΛΕΙΣΙΜΟ ΟΛΩΝ ΤΩΝ ΡΕΥΜΑΤΩΝ ΕΙΣΟΔΟΥ - ΕΞΟΔΟΥ
                Dis.close();
                Fis.close();
                Dos.close();
                Fos.close();
            } catch(Exception e) {
                JOptionPane.showMessageDialog(null,
                    "ERROR: "+ e.getMessage(),
                    "DICOM File Creating Error",
                    JOptionPane.ERROR_MESSAGE);
             }
        }

        public File getDicomFile(){

            //ΕΞΑΓΩΓΗ ΤΟΥ ΑΡΧΕΙΟΥ DICOM ΣΕ ΑΛΛΕΣ ΚΛΑΣΕΙΣ

            return DcmFile;
        }

        public String getDicomFileName(){

            //ΕΞΑΓΩΓΗ ΤΟΥ ΟΝΟΜΑΤΟΣ ΤΟΥ ΑΡΧΕΙΟΥ DICOM ΣΕ ΑΛΛΕΣ ΚΛΑΣΕΙΣ

            return DcmFileName;
        }
}
```

7.6 ListDICOMHeader.java

```
package doctor2.handlers;

import java.util.Iterator;

import org.dcm4che2.data.DicomElement;
import org.dcm4che2.data.DicomObject;
```

```
import org.dcm4che2.util.TagUtils;

public class ListDICOMHeader {

    //ΚΛΑΣΗ ΑΝΑΚΤΗΣΗΣ ΔΕΔΟΜΕΝΩΝ ΕΠΙΚΕΦΑΛΙΔΑΣ DICOM ΑΠΟ ΑΡΧΕΙΟ DICOM

    String[][] DICOMHeadData = new String[26][3];

    public String[][] listHeader(DicomObject Object) {

        //ΦΟΡΤΩΝΕΤΑΙ ΣΑΝ ΟΡΙΣΜΑ ΣΤΗ ΜΕΘΟΔΟ ΕΝΑ ΑΝΤΙΚΕΙΜΕΝΟ DICOM
        //ΚΑΙ ΟΡΙΖΕΤΑΙΕΝΑΣ ITERATOR ΠΟΥ ΨΑΧΝΕΙ ΓΙΑ ΔΕΔΟΜΕΝΑ ΕΠΙΚΕΦΑΛΙΔΑΣ
        Iterator<DicomElement> Iter = Object.datasetIterator();
        Boolean Circle = true;
        int i = 0;

        //ΒΡΟΧΟΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΚΑΘΕ ΔΕΔΟΜΕΝΟΥ ΕΠΙΚΕΦΑΛΙΔΑΣ ΠΟΥ ΠΡΟΚΥΠΤΕΙ
        //ΑΠΟ ΤΟΝ ΠΑΡΑΠΑΝΩ ITERATOR
        while(Iter.hasNext()&Circle) {
            DicomElement Element = Iter.next();

            //ΕΞΑΓΕΤΑΙ ΤΟ ΣΤΟΙΧΕΙΟ ΔΕΔΟΜΕΝΩΝ ΠΟΥ ΕΠΕΞΕΡΓΑΖΕΤΑΙ
            int Tag = Element.tag();
            try {

                //ΟΝΟΜΑ ΤΟΥ ΣΤΟΙΧΕΙΟΥ ΔΕΔΟΜΕΝΩΝ
                String TagName = Object.nameOf(Tag);

                //ΕΤΙΚΕΤΑ ΣΤΟΙΧΕΙΟΥ ΔΕΔΟΜΕΝΩΝ (2 4ΨΗΦΙΟΙ 16ΔΙΚΟΙ ΑΡΙΘΜΟΙ)
                String TagAddr = TagUtils.toString(Tag);

                //ΑΝΑΠΑΡΑΣΤΑΣΗ ΤΙΜΗΣ ΤΟΥ ΣΤΟΙΧΕΙΟΥ ΔΕΔΟΜΕΝΩΝ
                String TagVR = Object.vrOf(Tag).toString();

                //ΑΝ ΤΟ ΣΤΟΙΧΕΙΟ ΑΝΑΦΕΡΕΤΑΙ ΣΕ ΑΚΟΛΟΥΘΙΑ ΑΛΛΩΝ ΣΤΟΙΧΕΙΩΝ
                //SQ (Sequence of zero or more items)
                //ΤΟΤΕ ΑΠΛΑ ΓΙΝΕΤΑΙ ΑΝΑΦΟΡΑ ΣΤΗΝ ΥΠΑΡΞΗ ΤΟΥ
                //ΜΕ ΧΡΗΣΗ ΜΟΝΟ ΕΤΙΚΕΤΑΣ, ΑΝΑΠΑΡΑΣΤΑΣΗΣ ΤΙΜΗΣ ΚΑΙ
                ΟΝΟΜΑΤΟΣ ΣΤΟΙΧΕΙΟΥ
                //ΚΑΙ ΣΥΝΕΧΙΖΕΤΑΙ Η ΕΠΕΞΕΡΓΑΣΙΑ ΜΕ ΤΟ ΕΠΟΜΕΝΟ ΣΤΟΙΧΕΙΟ
                if (TagVR.equals("SQ")) {
                    if (Element.hasItems()) {

                        //ΠΡΟΑΙΡΕΤΙΚΟ ΜΗΝΥΜΑ ΠΡΟΒΟΛΗΣ ΤΟΥ ΣΤΟΙΧΕΙΟΥ
                        ΔΕΔΟΜΕΝΩΝ
                        System.out.println(TagAddr +" ["+  TagVR +"] "+ TagName);

                        DICOMHeadData [i][0] = TagAddr;
                        DICOMHeadData [i][1] = TagVR;
                        DICOMHeadData [i][2] = TagName;
                        i++;
                        listHeader(Element.getDicomObject());
```

```
                        continue;
                    }
                }

                //ΓΙΑ ΟΠΟΙΟΔΗΠΟΤΕ ΑΛΛΗ ΑΝΑΠΑΡΑΣΤΑΣΗ ΤΙΜΗΣ
                //ΛΑΜΒΑΝΟΝΤΑΙ ΚΑΝΟΝΙΚΑ Η ΕΤΙΚΕΤΑ, Η ΑΝΑΠΑΡΑΣΤΑΣΗ ΤΙΜΗΣ
                //ΤΟ ΟΝΟΜΑ ΚΑΙ Η ΤΙΜΗ ΤΟΥ ΣΤΟΙΧΕΙΟΥ ΔΕΔΟΜΕΝΩΝ
                String TagValue = Object.getString(Tag);

                //ΠΡΟΑΙΡΕΤΙΚΟ ΜΗΝΥΜΑ ΠΡΟΒΟΛΗΣ ΤΟΥ ΣΤΟΙΧΕΙΟΥ ΔΕΔΟΜΕΝΩΝ
                System.out.println(TagAddr +" ["+ TagVR +"] "+ TagName +" ["+ TagValue+"]");

                DICOMHeadData [i][0] = TagAddr;
                DICOMHeadData [i][1] = TagVR;
                DICOMHeadData [i][2] = TagValue;
                i++;

                //ΑΠΟ ΔΟΚΙΜΕΣ ΠΡΟΕΚΥΨΕ ΟΤΙ ΤΟ ΤΕΛΕΥΤΑΙΟ ΣΤΟΙΧΕΙΟ
                //ΤΗΣ ΑΚΟΛΟΥΘΙΑΣ ΤΩΝ ΕΠΙΜΕΡΟΥΣ ΣΤΟΙΧΕΙΩΝ
                //ΤΗΣ ΕΠΙΚΕΦΑΛΙΔΑΣ ΤΟΥ ΑΡΧΕΙΟΥ ΕΙΝΑΙ ΤΟ (3008, 0250)
                //ΚΑΤΟΠΙΝ Η ΜΕΘΟΔΟΣ ΠΡΟΣΠΑΘΕΙ ΝΑ ΔΙΑΒΑΣΕΙ ΤΟ ΕΠΟΜΕΝΟ
                ΣΤΟΙΧΕΙΟ
                //ΤΟ ΟΠΟΙΟ ΔΕΝ ΥΠΑΡΧΕΙ ΜΕ ΑΠΟΤΕΛΕΣΜΑ ΝΑ ΚΑΛΕΙΤΑΙ ΕΝΑ
                EXCEPTION
                //ΔΙΝΕΤΑΙ ΛΟΙΠΟΝ "ΧΕΙΡΟΚΙΝΗΤΑ" Η ΕΤΙΚΕΤΑ ΓΙΑ ΝΑ ΜΗΝ ΦΤΑΝΕΙ ΣΤΟ
                ΣΗΜΕΙΟ ΑΥΤΟ
                if (TagAddr.equals("(3008,0250)")) Circle=false;
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return DICOMHeadData;
    }
}
```

8. Πακέτο doctor3.gui, κλάση HL7Frame.java

```
package doctor3.gui;

import java.awt.Component;
import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.GridLayout;
import java.awt.Insets;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
```

```
import java.util.StringTokenizer;

import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JTextField;

import doctor3.handlers.HL7MessageCreator;
import doctor3.handlers.HL7DBDataFunctions;
import doctor3.handlers.HL7Tuple;

public class HL7Frame extends JFrame implements ActionListener{

    //ΚΛΑΣΗ GUI 3ΟΥ ΙΑΤΡΟΥ

    private static final long serialVersionUID = 1L;

    JButton Create = new JButton("Εισαγωγή Στοιχείων");
    JButton Load = new JButton("Αναζήτηση");
    JButton Update = new JButton("Ενημέρωση");
    JButton Delete = new JButton("Διαγραφή");
    JButton ClearAll = new JButton("Καθαρισμός Πεδίων");

    JLabel PatientData = new JLabel("Στοιχεία Ασθενή");
    JLabel FName = new JLabel("Όνομα:");
    JLabel LName = new JLabel("Επίθετο:");
    JLabel FatherName = new JLabel("Πατρώνυμο:");
    JLabel BirthDate = new JLabel("Ημερομηνία Γέννησης:");
    JLabel AMKA = new JLabel("A.M.K.A.:");
    JLabel Address = new JLabel("Διεύθυνση (Οδός/Αριθμός/Πόλη):");
    JLabel Phone = new JLabel("Τηλέφωνο:");
    JLabel ExamineDate = new JLabel("Ημερομηνία Εξέτασης:");
    JLabel Gender = new JLabel("Φύλο:");
    JLabel Sum = new JLabel("Υπόλοιπο σε Ευρώ:");
    JLabel PatientID = new JLabel("Κωδικός Ασθενούς:");
    JLabel HL7FileName = new JLabel("Όνομα σχετιζόμενου HL7 Αρχείου:");

    JLabel TestData = new JLabel("Στοιχεία Εξέτασης");
    JLabel Sakharo = new JLabel("Σάκχαρο (mg/dl):");
    JLabel Ouria = new JLabel("Ουρία (mg/dl):");
    JLabel Kreatinini = new JLabel("Κρεατινίνη (mg/dl):");
    JLabel OurikoOksy = new JLabel("Ουρικό Οξύ (mg/dl):");
    JLabel Triglykeridia = new JLabel("Τριγλυκερίδια (mg/dl):");
    JLabel OlikaLipidia = new JLabel("Ολικά Λιπίδια (mg/dl):");
    JLabel LDL = new JLabel("LDL-Χοληστερόλη (mg/dl):");
    JLabel HDL = new JLabel("HDL-Χοληστερόλη (mg/dl):");
    JLabel Leyka = new JLabel("Λευκά Αιμοσφαίρια (K/μl):");
    JLabel Erythra = new JLabel("Ερυθρά Αιμοσφαίρια (M/μl):");
    JLabel Aimosfairini = new JLabel("Αιμοσφαιρίνη (g/dl):");
```

```
JLabel Aimatokritis = new JLabel("Αιματοκρίτης (%):");
JLabel Aimopetalia = new JLabel("Αιμοπετάλια (K/μl):");

JTextField FNameText = new JTextField(20);
JTextField LNameText = new JTextField(35);
JTextField FatherNameText = new JTextField(20);
JTextField BirthDateText = new JTextField(15);
JTextField AMKAText = new JTextField(20);
JTextField AddressText = new JTextField(35);
JTextField PhoneText = new JTextField(20);
JTextField ExamineDateText = new JTextField(15);
JTextField SumText = new JTextField(10);
JTextField PatientIDText = new JTextField(10);
JTextField HL7FileNameText = new JTextField(35);

JTextField SakharoText = new JTextField(10);
JTextField OuriaText = new JTextField(10);
JTextField KreatininiText = new JTextField(10);
JTextField OurikoOksyText = new JTextField(10);
JTextField TriglykeridiaText = new JTextField(10);
JTextField OlikaLipidiaText = new JTextField(10);
JTextField LDLText = new JTextField(10);
JTextField HDLText = new JTextField(10);
JTextField LeykaText = new JTextField(10);
JTextField ErythraText = new JTextField(10);
JTextField AimosfairiniText = new JTextField(10);
JTextField AimatokritisText = new JTextField(10);
JTextField AimopetaliaText = new JTextField(10);

String[] GenderList = {"N/A", "Άνδρας", "Γυναίκα"};

JComboBox GenderText = new JComboBox(GenderList);

GridBagLayout Gridbag = new GridBagLayout();
GridBagConstraints Constraints;

Insets Whitespace = new Insets (10,10,10,10);

HL7Tuple HL7WorkTuple;

static Float[] TestResults = new Float[13];

public HL7Frame(){

    //ΚΑΤΑΣΚΕΥΗ GUI

    super ("IntegraHEALTH 1.0 by Nikos Christodoulou –
     Καρτέλα Στοιχείων Ασθενούς/Εξέτασης");
    Dimension ScreenSize = Toolkit.getDefaultToolkit().getScreenSize();
    setBounds((ScreenSize.width-1280)/2, (ScreenSize.height-680)/2, 1280, 680);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLayout(Gridbag);
```

```
Create.addActionListener(this);
Load.addActionListener(this);
Update.addActionListener(this);
Delete.addActionListener(this);
ClearAll.addActionListener(this);
Update.setEnabled(false);
Delete.setEnabled(false);

JPanel CommandPane = new JPanel();
CommandPane.setLayout(new GridLayout (1, 4, 10, 10));
CommandPane.add(Create);
CommandPane.add(Load);
CommandPane.add(Update);
CommandPane.add(Delete);
CommandPane.add(ClearAll);

addComponent(PatientData, 0, 0, 2, 1, 50, 100, GridBagConstraints.NONE,
    GridBagConstraints.CENTER);
addComponent(TestData, 2, 0, 2, 1, 50, 100, GridBagConstraints.NONE,
    GridBagConstraints.CENTER);
addComponent(PatientID, 0, 1, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(PatientIDText, 1, 1, 1, 1, 40, 100, GridBagConstraints.BOTH,
    GridBagConstraints.WEST);
addComponent(Leyka, 2, 1, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(LeykaText, 3, 1, 1, 1, 40, 100, GridBagConstraints.BOTH,
    GridBagConstraints.WEST);
addComponent(LName, 0, 2, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(LNameText, 1, 2, 1, 1, 40, 100, GridBagConstraints.BOTH,
    GridBagConstraints.WEST);
addComponent(Erythra, 2, 2, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(ErythraText, 3, 2, 1, 1, 40, 100, GridBagConstraints.BOTH,
    GridBagConstraints.WEST);
addComponent(FName, 0, 3, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(FNameText, 1, 3, 1, 1, 40, 100, GridBagConstraints.BOTH,
    GridBagConstraints.WEST);
addComponent(Aimosfairini, 2, 3, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(AimosfairiniText, 3, 3, 1, 1, 40, 100, GridBagConstraints.BOTH,
    GridBagConstraints.WEST);
addComponent(FatherName, 0, 4, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(FatherNameText, 1, 4, 1, 1, 40, 100, GridBagConstraints.BOTH,
    GridBagConstraints.WEST);
addComponent(Aimatokritis, 2, 4, 1, 1, 10, 100, GridBagConstraints.NONE,                \
    GridBagConstraints.EAST);
```

```
addComponent(AimatokritisText, 3, 4, 1, 1, 40, 100, GridBagConstraints.BOTH,
    GridBagConstraints.WEST);
addComponent(BirthDate, 0, 5, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(BirthDateText, 1, 5, 1, 1, 40, 100, GridBagConstraints.BOTH,
    GridBagConstraints.WEST);
addComponent(Aimopetalia, 2, 5, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(AimopetaliaText, 3, 5, 1, 1, 40, 100, GridBagConstraints.BOTH,
    GridBagConstraints.WEST);
addComponent(AMKA, 0, 6, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(AMKAText, 1, 6, 1, 1, 40, 100, GridBagConstraints.BOTH,
    GridBagConstraints.WEST);
addComponent(Sakharo, 2, 6, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(SakharoText, 3, 6, 1, 1, 40, 100, GridBagConstraints.BOTH,
    GridBagConstraints.WEST);
addComponent(Address, 0, 7, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(AddressText, 1, 7, 1, 1, 40, 100, GridBagConstraints.BOTH,
    GridBagConstraints.WEST);
addComponent(Ouria, 2, 7, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(OuriaText, 3, 7, 1, 1, 40, 100, GridBagConstraints.BOTH,
    GridBagConstraints.WEST);
addComponent(Phone, 0, 8, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(PhoneText, 1, 8, 1, 1, 40, 100, GridBagConstraints.BOTH,
    GridBagConstraints.WEST);
addComponent(Kreatinini, 2, 8, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(KreatininiText, 3, 8, 1, 1, 40, 100, GridBagConstraints.BOTH,
    GridBagConstraints.WEST);
addComponent(ExamineDate, 0, 9, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(ExamineDateText, 1, 9, 1, 1, 40, 100, GridBagConstraints.BOTH,
    GridBagConstraints.WEST);
addComponent(OurikoOksy, 2, 9, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(OurikoOksyText, 3, 9, 1, 1, 40, 100, GridBagConstraints.BOTH,
    GridBagConstraints.WEST);
addComponent(Gender, 0, 10, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(GenderText, 1, 10, 1, 1, 40, 100, GridBagConstraints.BOTH,
    GridBagConstraints.WEST);
addComponent(Triglykeridia, 2, 10, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
addComponent(TriglykeridiaText, 3, 10, 1, 1, 40, 100, GridBagConstraints.BOTH,
    GridBagConstraints.WEST);
addComponent(Sum, 0, 11, 1, 1, 10, 100, GridBagConstraints.NONE,
    GridBagConstraints.EAST);
```

```
        addComponent(SumText, 1, 11, 1, 1, 40, 100, GridBagConstraints.BOTH,
            GridBagConstraints.WEST);
        addComponent(OlikaLipidia, 2, 11, 1, 1, 10, 100, GridBagConstraints.NONE,
            GridBagConstraints.EAST);
        addComponent(OlikaLipidiaText, 3, 11, 1, 1, 40, 100, GridBagConstraints.BOTH,
            GridBagConstraints.WEST);
        addComponent(HL7FileName, 0, 12, 1, 1, 10, 100, GridBagConstraints.NONE,
            GridBagConstraints.EAST);
        addComponent(HL7FileNameText, 1, 12, 1, 1, 40, 100, GridBagConstraints.BOTH,
            GridBagConstraints.WEST);
        addComponent(LDL, 2, 12, 1, 1, 10, 100, GridBagConstraints.NONE,
            GridBagConstraints.EAST);
        addComponent(LDLText, 3, 12, 1, 1, 40, 100, GridBagConstraints.BOTH,
            GridBagConstraints.WEST);
        addComponent(HDL, 2, 13, 1, 1, 10, 100, GridBagConstraints.NONE,
            GridBagConstraints.EAST);
        addComponent(HDLText, 3, 13, 1, 1, 40, 100, GridBagConstraints.BOTH,
            GridBagConstraints.WEST);
        addComponent(CommandPane, 0, 14 ,4, 1, 100, 100,
            GridBagConstraints.NONE,GridBagConstraints.CENTER);
    }

    private void addComponent(Component Component, int GridX, int GridY, int GridWidth,
        int GridHeight, int WeightX, int WeightY, int Fill, int Anchor){
        GridBagConstraints Constraints = new GridBagConstraints();
        Constraints.gridx = GridX;
        Constraints.gridy = GridY;
        Constraints.gridwidth = GridWidth;
        Constraints.gridheight = GridHeight;
        Constraints.weightx = WeightX;
        Constraints.weighty = WeightY;
        Constraints.fill = Fill;
        Constraints.anchor = Anchor;
        Constraints.insets = Whitespace;
        Gridbag.setConstraints(Component, Constraints);
        add(Component);
    }

    public void actionPerformed(ActionEvent evt) {

        //ΑΝΑΛΟΓΑ ΜΕ ΤΟ ΚΟΥΜΠΙ ΠΟΥ ΕΠΙΛΕΓΕΤΑΙ,
        //ΕΝΕΡΓΟΠΟΙΕΙΤΑΙ Η ΚΑΤΑΛΛΗΛΗ ΔΙΑΔΙΚΑΣΙΑ ΑΛΛΑΓΗΣ ΣΤΗ ΒΑΣΗ ΤΩΝ
        ΑΡΧΕΙΩΝ DICOM
        //INSERT, UPDATE, DELETE

        Object Source = evt.getSource();

        //ΕΛΕΓΧΟΣ ΜΟΡΦΗΣ ΗΜΕΡΟΜΗΝΙΩΝ (ΑΠΑΙΤΟΥΜΕΝΗ ΜΟΡΦΗ: EEEE-MM-HH)
        boolean BirthDateFormatConfirmed = confirmDateFormat(BirthDateText.getText());
        boolean ExamineDateFormatConfirmed = confirmDateFormat(ExamineDateText.getText());
```

```
//ΕΙΣΑΓΩΓΗ ΝΕΩΝ ΣΤΟΙΧΕΙΩΝ
if (Source==Create){

    //ΠΕΡΙΠΤΩΣΕΙΣ ΛΑΝΘΑΣΜΕΝΩΝ ΣΤΟΙΧΕΙΩΝ ΓΙΑ ΕΙΣΑΓΩΓΗ
    if ((AMKAText.getText().isEmpty())
        |(FNameText.getText().isEmpty())
        |(LNameText.getText().isEmpty())
        |(PatientID.getText().isEmpty())){
        JOptionPane.showMessageDialog(null,
            "Το όνομα, το επίθετο, το A.M.K.A."+'\n'+
            "ή/και ο ειδικός κωδικός του ασθενή είναι κενά!"+'\n'+
            "Συμπληρώστε τα και ξαναδοκιμάστε",
            "Σφάλμα Εισαγωγής Δεδομένων",
            JOptionPane.ERROR_MESSAGE);
    }
    else if (GenderText.getSelectedIndex()==0){
        JOptionPane.showMessageDialog(null,
            "Παρακαλούμε επιλέξτε το φύλο του ασθενή",
            "Σφάλμα Εισαγωγής Δεδομένων",
            JOptionPane.ERROR_MESSAGE);
    }

    else if ((!BirthDateFormatConfirmed)|(!ExamineDateFormatConfirmed)){
        JOptionPane.showMessageDialog(null,
            "Οι ημερομηνίες πρέπει να είναι στη μορφή EEEE-MM-HH!" +'\n'+
            "Συμπληρώστε τες σωστά και ξαναδοκιμάστε",
            "Σφάλμα Εισαγωγής Δεδομένων",
            JOptionPane.ERROR_MESSAGE);
    }
    else {

        //ΦΟΡΤΩΣΗ ΣΤΟΙΧΕΙΩΝ ΑΠΟ ΤΟ GUI
        HL7WorkTuple = loadHL7Tuple();

        //ΔΗΜΙΟΥΡΓΙΑ ΜΗΝΥΜΑΤΟΣ HL7 ΤΥΠΟΥ ADT A04
        HL7MessageCreator MessageConstruct = new HL7MessageCreator();
        MessageConstruct.buildADTMessage(HL7WorkTuple, "ADTA04");

        //ΛΑΜΒΑΝΟΝΤΑΙ ΤΟ ΙΔΙΟ ΤΟ ΑΡΧΕΙΟ ΚΑΙ ΤΟ ΟΝΟΜΑ ΤΟΥ
        File IncomingHL7Message = MessageConstruct.getHL7MessageFile();
        HL7WorkTuple.FileName = MessageConstruct.getHL7MessageFileName();

        //ΑΠΟΘΗΚΕΥΣΗ ΣΤΗ ΒΑΣΗ
        HL7DBDataFunctions.createInHL7Base(
            HL7WorkTuple.PatientID, HL7WorkTuple.FileName, IncomingHL7Message);
    }
}

//ΑΝΑΚΤΗΣΗ ΣΤΟΙΧΕΙΩΝ ΣΤΟ GUI ΑΠΟ ΤΗ ΒΑΣΗ
if (Source==Load){
```

```
        //ΠΕΡΙΠΤΩΣΕΙΣ ΣΤΟΙΧΕΙΩΝ ΓΙΑ ΑΝΑΖΗΤΗΣΗ

        //ΑΝΑΖΗΤΗΣΗ ΜΕ ΒΑΣΗ ΤΟ ΟΝΟΜΑ ΤΟΥ ΑΡΧΕΙΟΥ
        if (!(HL7FileNameText.getText().isEmpty())){
            presentHL7Tuple(HL7DBDataFunctions.fetchFromHL7Base(
                HL7FileNameText.getText(), "store"));
            Create.setEnabled(false);
            Load.setEnabled(false);
            Update.setEnabled(true);
            Delete.setEnabled(true);
            AMKAText.setEnabled(false);
            PatientIDText.setEnabled(false);
            HL7FileNameText.setEnabled(false);
        }
        else {
            if (PatientIDText.getText().isEmpty()){
                JOptionPane.showMessageDialog(null,
                    "Το όνομα αρχείου ή/και ο ειδικός κωδικός του ασθενή είναι κενά!"+'\n'+
                    "Συμπληρώστε τα και ξαναδοκιμάστε",
                    "Σφάλμα Εισαγωγής Δεδομένων",
                    JOptionPane.ERROR_MESSAGE);
            }                                           else {

                //ΑΝΑΖΗΤΗΣΗ ΒΑΣΕΙ ΤΟΥ ΕΙΔΙΚΟΥ ΚΩΔΙΚΟΥ ΑΣΘΕΝΗ (ΟΧΙ ΑΜΚΑ)
                presentHL7Tuple(HL7DBDataFunctions.fetchFromHL7Base(
                    PatientIDText.getText(), "store"));
                Create.setEnabled(false);
                Load.setEnabled(false);
                Update.setEnabled(true);
                Delete.setEnabled(true);
                AMKAText.setEnabled(false);
                PatientIDText.setEnabled(false);
                HL7FileNameText.setEnabled(false);
            }
        }
    }

    //ΕΝΗΜΕΡΩΣΗ ΣΤΟΙΧΕΙΩΝ ΣΤΗ ΒΑΣΗ
    if (Source==Update){
        if ((FNameText.getText().isEmpty())|(LNameText.getText().isEmpty())){
            JOptionPane.showMessageDialog(null,
                "Το όνομα ή/και το επίθετο είναι κενά!"+'\n'+
                "Συμπληρώστε τα και ξαναδοκιμάστε",
                "Σφάλμα Εισαγωγής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
        else{

            //ΦΟΡΤΩΣΗ ΣΤΟΙΧΕΙΩΝ ΑΠΟ ΤΟ GUI
            HL7WorkTuple = loadHL7Tuple();
            String OldHL7FileName = HL7WorkTuple.FileName;
```

```
                //ΔΗΜΙΟΥΡΓΙΑ ΜΗΝΥΜΑΤΟΣ HL7 ΤΥΠΟΥ ADT A08
                HL7MessageCreator MessageConstruct = new HL7MessageCreator();
                MessageConstruct.buildADTMessage(HL7WorkTuple, "ADTA08");

                //ΛΑΜΒΑΝΟΝΤΑΙ ΤΟ ΙΔΙΟ ΤΟ ΑΡΧΕΙΟ ΚΑΙ ΤΟ ΟΝΟΜΑ ΤΟΥ
                File IncomingHL7Message = MessageConstruct.getHL7MessageFile();
                HL7WorkTuple.FileName = MessageConstruct.getHL7MessageFileName();

                //ΑΠΟΘΗΚΕΥΣΗ ΣΤΗ ΒΑΣΗ
                HL7DBDataFunctions.updateInHL7Base(HL7WorkTuple.PatientID,
                    OldHL7FileName, HL7WorkTuple.FileName, IncomingHL7Message);
            }
        }
        if (Source==Delete){

            //ΔΙΑΓΡΑΦΗ ΣΤΟΙΧΕΙΩΝ ΑΠΟ ΤΗ ΒΑΣΗ
            HL7DBDataFunctions.deleteFromHL7Base(HL7FileNameText.getText());
            clearAllFields();
        }

        if (Source==ClearAll){

            //ΑΠΑΛΟΙΦΗ ΟΛΩΝ ΤΩΝ ΠΕΡΙΕΧΟΜΕΝΩΝ ΤΟΥ GUI
            clearAllFields();
        }
    }

    private void clearAllFields(){

        //ΑΠΑΛΟΙΦΗ ΟΛΩΝ ΤΩΝ ΠΕΡΙΕΧΟΜΕΝΩΝ ΤΟΥ GUI

        PatientIDText.setText("");
        if (!PatientIDText.isEnabled()){
            PatientIDText.setEnabled(true);
        }
        LNameText.setText("");
        FNameText.setText("");
        FatherNameText.setText("");
        BirthDateText.setText("");
        AMKAText.setText("");
        if (!AMKAText.isEnabled()){
            AMKAText.setEnabled(true);
        }
        AddressText.setText("");
        PhoneText.setText("");
        ExamineDateText.setText("");
        GenderText.setSelectedIndex(0);
        SumText.setText("");
        HL7FileNameText.setText("");
        if (!HL7FileNameText.isEnabled()){
            HL7FileNameText.setEnabled(true);
        }
```

```
        SakharoText.setText("");
        OuriaText.setText("");
        KreatininiText.setText("");
        OurikoOksyText.setText("");
        TriglykeridiaText.setText("");
        OlikaLipidiaText.setText("");
        LDLText.setText("");
        HDLText.setText("");
        LeykaText.setText("");
        ErythraText.setText("");
        AimosfairiniText.setText("");
        AimatokritisText.setText("");
        AimopetaliaText.setText("");

        Create.setEnabled(true);
        Load.setEnabled(true);
        Update.setEnabled(false);
        Delete.setEnabled(false);
    }

    private boolean confirmDateFormat(String InDate) {

        //ΕΛΕΓΧΟΣ ΜΟΡΦΗΣ ΗΜΕΡΟΜΗΝΙΩΝ (ΕΕΕΕ-ΜΜ-ΗΗ)

        int Day, Month, Year;
        if(InDate.length()!=10){
            return false;
        }
        StringTokenizer stk = new StringTokenizer(InDate, InDate.substring(4, 5));
        if (stk.countTokens()!=3){
            return false;
        }
        else{
            Year = Integer.parseInt(stk.nextToken());
            Month = Integer.parseInt(stk.nextToken());
            Day = Integer.parseInt(stk.nextToken());
            DateFormat yearFormat = new SimpleDateFormat("yyyy");
            Date date = new Date();
            int CurrentYear = Integer.parseInt(yearFormat.format(date));
            if (Day>31|Day<1){
                return false;
            }
            else if (Month>12|Month<1){
                return false;
            }
            else if(((Month==2)|(Month==4)|(Month==6)|(Month==9)|(Month==11))&(Day==31)){
                return false;
            }
            else if((Month==2)&((Day==30)|(Day==31))){
                return false;
            }
            else if((Month==2)&(Year%4!=0)&(Day==29)){
```

```
                return false;
            }
            else if((Month==2)&(Year%4==0)&(Year%400!=0)&(Day==29)){
                return false;
            }
            else if ((Year<(CurrentYear-150))|(Year>CurrentYear)){
                return false;
            }
        }
        return true;
    }

    public HL7Tuple loadHL7Tuple(){

        //ΦΟΡΤΩΣΗ ΣΤΟΙΧΕΙΩΝ ΑΠΟ ΤΟ GUI ΣΕ ΜΙΑ ΠΛΕΙΑΔΑ
        //ΓΙΑ ΚΑΘΕ ΜΕΤΣΗΣΗ ΠΟΥ ΔΕΝ ΕΧΕΙ ΔΟΘΕΙ ΤΙΜΗ ΣΤΟ GUI
        //ΛΑΜΒΑΝΕΤΑΙ ΑΥΘΑΙΡΕΤΑ Η ΤΙΜΗ -0

        HL7Tuple OutTuple = new HL7Tuple();

        if (LeykaText.getText().isEmpty()){
            TestResults[0] = Float.parseFloat("-0");
        }
        else{
            TestResults[0] = Float.parseFloat(LeykaText.getText());
        }
        if (ErythraText.getText().isEmpty()){
            TestResults[1] = Float.parseFloat("-0");
        }
        else{
            TestResults[1] = Float.parseFloat(ErythraText.getText());
        }
        if (AimosfairiniText.getText().isEmpty()){
            TestResults[2] = Float.parseFloat("-0");
        }
        else{
            TestResults[2] = Float.parseFloat(AimosfairiniText.getText());
        }
        if (AimatokritisText.getText().isEmpty()){
            TestResults[3] = Float.parseFloat("-0");
        }
        else{
            TestResults[3] = Float.parseFloat(AimatokritisText.getText());
        }
        if (AimopetaliaText.getText().isEmpty()){
            TestResults[4] = Float.parseFloat("-0");
        }
        else{
            TestResults[4] = Float.parseFloat(AimopetaliaText.getText());
        }
        if (SakharoText.getText().isEmpty()){
            TestResults[5] = Float.parseFloat("-0");
```

```
}
else{
    TestResults[5] = Float.parseFloat(SakharoText.getText());
}
if (OuriaText.getText().isEmpty()){
    TestResults[6] = Float.parseFloat("-0");
}
else{
    TestResults[6] = Float.parseFloat(OuriaText.getText());
}
if (KreatininiText.getText().isEmpty()){
    TestResults[7] = Float.parseFloat("-0");
}
else{
    TestResults[7] = Float.parseFloat(KreatininiText.getText());
}
if (OurikoOksyText.getText().isEmpty()){
    TestResults[8] = Float.parseFloat("-0");
}
else{
    TestResults[8] = Float.parseFloat(OurikoOksyText.getText());
}
if (TriglykeridiaText.getText().isEmpty()){
    TestResults[9] = Float.parseFloat("-0");
}
else{
    TestResults[9] = Float.parseFloat(TriglykeridiaText.getText());
}
if (OlikaLipidiaText.getText().isEmpty()){
    TestResults[10] = Float.parseFloat("-0");
}
else{
    TestResults[10] = Float.parseFloat(OlikaLipidiaText.getText());
}
if (LDLText.getText().isEmpty()){
    TestResults[11] = Float.parseFloat("-0");
}
else{
    TestResults[11] = Float.parseFloat(LDLText.getText());
}
if (HDLText.getText().isEmpty()){
    TestResults[12] = Float.parseFloat("-0");
}
else{
    TestResults[12] = Float.parseFloat(HDLText.getText());
}

OutTuple.PatientID = PatientIDText.getText();
OutTuple.LName = LNameText.getText();
OutTuple.FName = FNameText.getText();
OutTuple.FatherName = FatherNameText.getText();
OutTuple.Birthday = BirthDateText.getText();
```

```
        OutTuple.Amka = AMKAText.getText();
        OutTuple.Address = AddressText.getText();
        OutTuple.Phone = PhoneText.getText();
        OutTuple.ExaminationDate = ExamineDateText.getText();
        OutTuple.Gender = GenderText.getSelectedItem().toString();
        if (!SumText.getText().isEmpty()){
            OutTuple.Sum = Float.parseFloat(SumText.getText());
        }else{
            OutTuple.Sum = 0f;
        }
        OutTuple.FileName = HL7FileNameText.getText();
        OutTuple.TestData = TestResults;
        return OutTuple;
    }

    public void presentHL7Tuple(HL7Tuple OutTuple){

        //ΠΑΡΟΥΣΙΑΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

        PatientIDText.setText(OutTuple.PatientID);
        LNameText.setText(OutTuple.LName);
        FNameText.setText(OutTuple.FName);
        FatherNameText.setText(OutTuple.FatherName);
        BirthDateText.setText(OutTuple.Birthday);
        AMKAText.setText(OutTuple.Amka);
        AddressText.setText(OutTuple.Address);
        PhoneText.setText(OutTuple.Phone);
        ExamineDateText.setText(OutTuple.ExaminationDate);
        if (OutTuple.Gender.equals("Female")){
            GenderText.setSelectedIndex(2);
        }
        if (OutTuple.Gender.equals("Male")){
            GenderText.setSelectedIndex(1);
        }
        else{
            GenderText.setSelectedIndex(0);
        }
        SumText.setText(OutTuple.Sum.toString());
        HL7FileNameText.setText(OutTuple.FileName);

        LeykaText.setText(OutTuple.TestData[0].toString());
        ErythraText.setText(OutTuple.TestData[1].toString());
        AimosfairiniText.setText(OutTuple.TestData[2].toString());
        AimatokritisText.setText(OutTuple.TestData[3].toString());
        AimopetaliaText.setText(OutTuple.TestData[4].toString());
        SakharoText.setText(OutTuple.TestData[5].toString());
        OuriaText.setText(OutTuple.TestData[6].toString());
        KreatininiText.setText(OutTuple.TestData[7].toString());
        OurikoOksyText.setText(OutTuple.TestData[8].toString());
        TriglykeridiaText.setText(OutTuple.TestData[9].toString());
        OlikaLipidiaText.setText(OutTuple.TestData[10].toString());
        LDLText.setText(OutTuple.TestData[11].toString());
```

```
            HDLText.setText(OutTuple.TestData[12].toString());
        }
}
```

9. Πακέτο doctor3.handlers

9.1 BasicFunctions.java

Η κλάση είναι ίδια με αυτήν του πακέτου doctor1.handlers. Αλλάζει όπου doctor1 ➔ doctor3.

9.2 HL7DBDataFunctions.java

```
package doctor3.handlers;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.sql.Blob;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

import javax.swing.JOptionPane;

public class HL7DBDataFunctions {

    //ΚΛΑΣΗ ΜΕΘΟΔΩΝ ΑΛΛΑΓΩΝ ΤΗΣ ΒΑΣΗΣ ΤΟΥ 2ΟΥ ΙΑΤΡΟΥ

    public static void createInHL7Base(String InPatientID, String InHL7FileName, File InHL7File){

        //ΕΙΣΑΓΩΓΗ ΝΕΩΝ ΣΤΟΙΧΕΙΩΝ
        InputStream StreamedHL7File = null;
        int FileLength = (int) InHL7File.length();
        try {
            StreamedHL7File = new FileInputStream(InHL7File);
        } catch (FileNotFoundException FNFe) {
            JOptionPane.showMessageDialog(null,
                "Δεν βρέθηκε αρχείο προς αποθήκευση"+'\n'+FNFe.toString(),
                "Σφάλμα Εισαγωγής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
        BasicFunctions.loadDriver();
        Connection Conn = BasicFunctions.connectToHL7Base();
        try {
```

```
            java.sql.PreparedStatement PrepSt = Conn.prepareStatement(
                "INSERT INTO "+
                "hl7_elements(patientID, hl7FileName, hl7File)" +
                "VALUES(?, ?, ?)");
            PrepSt.setString(1, InPatientID);
            PrepSt.setString(2, InHL7FileName);
            PrepSt.setBinaryStream(3, StreamedHL7File, FileLength);
            PrepSt.executeUpdate();
            PrepSt.close();
            StreamedHL7File.close();
        } catch (SQLException SQLe){
            JOptionPane.showMessageDialog(null,
                "Τα δεδομένα δεν γράφτηκαν στη βάση"+'\n'+SQLe.toString(),
                "Σφάλμα Εισαγωγής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        } catch (IOException IOe){
            JOptionPane.showMessageDialog(null,
                "Δεν μπορούσε να ετοιμαστει το αρχείο HL7 για τη βάση"+'\n'+IOe.toString(),
                "Σφάλμα Εισαγωγής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
        BasicFunctions.disconnectFromBase(Conn);
    }

    public static HL7Tuple fetchFromHL7Base(String InElement, String CaseFlag){

        //ΑΝΑΚΤΗΣΗ ΔΕΔΟΜΕΝΩΝ

        HL7Tuple ResultHL7Tuple = new HL7Tuple();
        int Response = 1;
        Blob HL7FileBlob = null;
        BasicFunctions.loadDriver();
        String Help1 = "";
        String Help2 = "";
        Connection conn = BasicFunctions.connectToHL7Base();
        try{
            Statement St = conn.createStatement();
            if (!(InElement.contains(".hl7"))){

                //ΑΝΑΖΗΤΗΣΗ ΒΑΣΕΙ ΤΟΥ ΕΙΔΙΚΟΥ ΚΩΔΙΚΟΥ ΑΣΘΕΝΗ
                ResultSet ResSet = St.executeQuery(
                    "SELECT * " +
                    "FROM hl7_elements " +
                    "WHERE (patientID ='" + InElement + "')");

                //ΒΡΟΧΟΣ ΧΕΙΡΣΜΟΥ ΠΟΛΛΑΠΛΩΝ ΑΠΟΤΕΛΕΣΜΑΤΩΝ
                while (ResSet.next() && (Response == 1)){
                    if (CaseFlag.equals("store")){
                        Response = JOptionPane.showOptionDialog(null,
                            "Βρέθηκε το αρχείο: "+ResSet.getString(2)+'\n'+
                            "του ασθενή με κωδικό: "+ResSet.getString(1)+'\n'+
                            "Είναι αυτό το επιθυμητό αποτέλεσμα;",
```

```
                    "Εύρεση Πολλαπλών Αποτελεσμάτων",
                    JOptionPane.YES_NO_OPTION,
                    JOptionPane.QUESTION_MESSAGE,
                    null, null, null);
            }
            else{
                Response = 0;
            }
            if (Response == 0){
                ResultHL7Tuple.PatientID = ResSet.getString(1);
                ResultHL7Tuple.FileName = ResSet.getString(2);
                Help1 = ResultHL7Tuple.PatientID;
                Help2 = ResultHL7Tuple.FileName;
                HL7FileBlob = ResSet.getBlob(3);
            }
        }
        ResSet.close();
    }
    else{

        //ΑΝΑΖΗΤΗΣΗ ΒΑΣΕΙ ΟΝΟΜΑΤΟΣ ΑΡΧΕΙΟΥ
        ResultSet ResSet = St.executeQuery(
            "SELECT * " +
            "FROM hl7_elements " +
            "WHERE (hl7FileName like '%" + InElement + "%')");
        while (ResSet.next()){
            Response = 0;
            ResultHL7Tuple.PatientID = ResSet.getString(1);
            ResultHL7Tuple.FileName = ResSet.getString(2);
            Help1 = ResultHL7Tuple.PatientID;
            Help2 = ResultHL7Tuple.FileName;
            HL7FileBlob = ResSet.getBlob(3);
            break;
        }
        ResSet.close();
    }

    //ΚΛΗΣΗ ΓΙΑ PARSING ΤΟΥ ΑΝΑΚΤΗΘΕΝΤΟΣ ΑΡΧΕΙΟΥ ΚΑΙ ΑΝΑΘΕΣΗ
    ΤΙΜΩΝ ΣΤΗΝ ΠΛΕΙΑΔΑ ΕΞΟΔΟΥ
    if (Response==0){
        File LoadedHL7File =
            new File(".\\HL7MESSAGES\\"+ResultHL7Tuple.FileName);
        OutputStream HL7FileWriter = new FileOutputStream(LoadedHL7File);
        readFromBlob(HL7FileBlob, HL7FileWriter);
        HL7FileWriter.close();
        HL7MessageParser HL7Parser = new HL7MessageParser();
        ResultHL7Tuple = HL7Parser.readHL7FileBySegment(LoadedHL7File);
        ResultHL7Tuple.PatientID = Help1;
        ResultHL7Tuple.FileName = Help2;
    }
    if (Response==1){
        ResultHL7Tuple.FName = "N/A";
```

```
                ResultHL7Tuple.LName = "N/A";
                ResultHL7Tuple.FatherName = "N/A";
                ResultHL7Tuple.Birthday = "N/A";
                ResultHL7Tuple.Amka = "N/A";
                ResultHL7Tuple.ExaminationDate = "N/A";
                ResultHL7Tuple.Gender = "N/A";
                ResultHL7Tuple.Address = "N/A";
                ResultHL7Tuple.FileName = "N/A";
                ResultHL7Tuple.Phone = "N/A";
                ResultHL7Tuple.PatientID = "N/A";
                ResultHL7Tuple.Sum = 0f;
                JOptionPane.showMessageDialog(null, "Nothing Found");
            }
            St.close();
        }catch (Exception e){
            JOptionPane.showMessageDialog(null,
                "Τα δεδομένα δεν μπορούν να ανακτηθούν"+'\n'+e.toString(),
                "Σφάλμα Ανάκτησης Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
        BasicFunctions.disconnectFromBase(conn);
        return ResultHL7Tuple;
    }

    public static void readFromBlob(Blob Blob, OutputStream Out){

        //ΑΝΑΓΝΩΣΗ ΕΝΟΣ ΑΝΤΙΚΕΙΜΕΝΟΥ BLOB (Binary Large OBject) ΑΠΟ ΜΙΑ ΒΑΣΗ

        try{
            InputStream In = Blob.getBinaryStream();
            int Length = -1;
            byte[] Buf = new byte[1024];
            while ((Length = In.read(Buf)) != -1) {
                Out.write(Buf, 0, Length);
            }
            In.close();
     } catch (SQLException SQLe){
        JOptionPane.showMessageDialog(null,
            "Τα δεδομένα δεν ανακτήθηκαν απο τη βάση"+'\n'+SQLe.toString(),
            "Σφάλμα Ανάκτησης Δεδομένων",
            JOptionPane.ERROR_MESSAGE);
     } catch (IOException IOe){
        JOptionPane.showMessageDialog(null,
            "Δεν υπάρχει έτοιμα αρχείο DICOM για να δεχτεί τα δεδομένα"+'\n'+IOe.toString(),
            "Σφάλμα Ανάκτησης Δεδομένων",
            JOptionPane.ERROR_MESSAGE);
        }
    }

    public static void updateInHL7Base(
        String InPatientID, String OldHL7FileName, String InHL7FileName, File InHL7File){
```

```
        //ΕΝΗΜΕΡΩΣΗ ΔΕΔΟΜΕΝΩΝ

        BasicFunctions.loadDriver();
        Connection Conn = BasicFunctions.connectToHL7Base();
        try{

            //ΦΟΡΤΩΣΗ ΤΟΥ ΑΡΧΕΙΟΥ DICOM ΣΕ ΕΝΑ ΡΕΥΜΑ ΕΙΣΟΔΟΥ
            InputStream StreamedHL7File = null;
            int fileLength = (int) InHL7File.length();
            try {
                StreamedHL7File = new FileInputStream(InHL7File);
            } catch (FileNotFoundException FNFe) {
                JOptionPane.showMessageDialog(null,
                    "Δεν βρέθηκε αρχείο DICOM προς αποθήκευση"+'\n'+FNFe.toString(),
                    "Σφάλμα Εισαγωγής Δεδομένων",
                    JOptionPane.ERROR_MESSAGE);
            }

            //SQL STATEMENT
            java.sql.PreparedStatement PrepSt = Conn.prepareStatement(
                "UPDATE hl7_elements SET hl7File = ?, " +
                    "hl7FileName = '" + InHL7FileName + "' " +
                "WHERE patientID = '"+ InPatientID +"' "+
                "AND hl7FileName = '"+ OldHL7FileName+"'");
            PrepSt.setBinaryStream(1, StreamedHL7File, fileLength);
            PrepSt.executeUpdate();
        }catch (SQLException e){
            JOptionPane.showMessageDialog(null,
                "Data not updated"+'\n'+e.toString(),
                "Data Updating Error",
                JOptionPane.ERROR_MESSAGE);
    }
        BasicFunctions.disconnectFromBase(Conn);
    }

    public static void deleteFromHL7Base(String InElement){

        //ΔΙΑΓΡΑΦΗ ΔΕΔΟΜΕΝΩΝ

        BasicFunctions.loadDriver();
        Connection Conn = BasicFunctions.connectToHL7Base();
        try{
            Statement St = Conn.createStatement();
                java.sql.PreparedStatement PrepSt = Conn.prepareStatement(
                    "DELETE " +
                    "FROM hl7_elements " +
                    "WHERE hl7FileName = '" + InElement +"'");
                PrepSt.executeUpdate();
            St.close();
        }catch (SQLException e){
            JOptionPane.showMessageDialog(null,
                "Τα δεδομένα δεν διαγράφθηκαν"+'\n'+e.toString(),
```

```
                "Σφάλμα Διαγραφής Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
            BasicFunctions.disconnectFromBase(Conn);
        }
}
```

9.3 HL7MessageCreator.java

```
package doctor3.handlers;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.StringTokenizer;

import javax.swing.JOptionPane;

public class HL7MessageCreator {

    //ΚΛΑΣΗ ΔΗΜΙΟΥΡΓΙΑΣ ΑΡΧΕΙΩΝ ΜΗΝΥΜΑΤΩΝ HL7

    private static File HL7Message = null;
    private static String HL7MessageFileName = "";

    public void buildADTMessage(HL7Tuple InHL7Tuple, String InEventCode){
        String[] OBX = new String[13];

        //ΚΛΗΣΕΙΣ ΜΕΘΟΔΩΝ ΚΑΤΑΣΚΕΥΗΣ ΤΜΗΜΑΤΩΝ HL7 (SEGMENTS)
        String MSH = buildMSHSegment(InEventCode);
        String EVN = buildEVNSegment(InEventCode);
        String PID = buildPIDSegment(InHL7Tuple);
        String PV1 = buildPV1Segment(
            InHL7Tuple.Sum,InEventCode, InHL7Tuple.ExaminationDate);

        //ΚΑΤΑΣΚΕΥΗ ΕΝΟΣ ΤΜΗΜΑΤΟΣ OBX ΓΙΑ ΚΑΘΕ ΜΕΤΡΗΣΗ ΠΟΥ ΕΧΕΙ ΓΙΝΕΙ
        int HasElement = 1;
        for (int i=0;i<13;i++){
            if (InHL7Tuple.TestData[i]!=Float.parseFloat("-0")){
                OBX[i] = buildOBXSegment(
                    InHL7Tuple.TestDataContext[i][0],
                    InHL7Tuple.TestData[i].toString(),
                    InHL7Tuple.TestDataContext[i][1],HasElement, i);
                HasElement++;
            }
            else{
                OBX[i] = "N/A";
```

```
            }
        }

        //ΔΗΜΙΟΥΡΓΙΑ ΟΝΟΜΑΤΟΣ ΑΡΧΕΙΟΥ
        HL7MessageFileName = InHL7Tuple.PatientID
        .concat("_"+InEventCode+"_")
        .concat(InHL7Tuple.ExaminationDate)
        .concat(".hl7");

        //ΔΗΜΙΟΥΡΓΙΑ ΑΡΧΕΙΟΥ
        HL7Message = new File(".\\HL7MESSAGES\\"+HL7MessageFileName);

        //ΤΟΠΟΘΕΤΗΣΗ ΑΡΧΕΙΟΥ ΣΕ ΕΝΑ ΡΕΥΜΑ ΕΞΟΔΟΥ
        FileOutputStream fos = null;
        try {
            fos = new FileOutputStream(HL7Message);
        } catch (FileNotFoundException e) {
            JOptionPane.showMessageDialog(null,
                "Το αρχείο δεν μπορούσε να δημιουργηθεί ή να φορτωθεί"+'\n'+e.toString(),
                "Σφάλμα Δημιουργίας/Φόρτωσης Αρχείου",
                JOptionPane.ERROR_MESSAGE);
        }

        //ΤΟΠΟΘΕΤΗΣΗ ΜΗΝΥΜΑΤΟΣ ΣΕ ΕΝΑΝ WRITER ΡΕΥΜΑΤΟΣ ΕΞΟΔΟΥ
        //ΠΟΥ ΣΥΝΔΕΕΤΑΙ ΜΕ ΤΟ ΡΕΥΜΑ ΕΞΟΔΟΥ ΤΟΥ ΑΡΧΕΙΟΥ
        OutputStreamWriter out = new OutputStreamWriter(fos);
        try {

            //ΕΓΓΡΑΦΗ ΤΟΥ ΜΗΝΥΜΑΤΟΣ ΣΤΟ ΑΡΧΕΙΟ
            String eol = System.getProperty("line.separator");
            out.write(MSH+eol);
            out.write(EVN+eol);
            out.write(PID+eol);
            out.write(PV1+eol);

            for(int i=0;i<OBX.length;i++){
                if (!OBX[i].equals("N/A")){
                    out.write(OBX[i]+eol);
                }
            }
            out.flush();
            out.close();
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null,
                    "Τα δεδομένα δεν γράφτηκαν"+'\n'+e.toString(),
                    "Σφάλμα Εγγραφής Δεδομένων",
                    JOptionPane.ERROR_MESSAGE);
        }
    }
```

```
//ΜΕΘΟΔΟΙ ΚΑΤΑΣΚΕΥΗΣ ΕΠΙΜΕΡΟΥΣ ΤΜΗΜΑΤΩΝ HL7

private String buildMSHSegment(String EventCode){
      String InEventCode = EventCode;
      String MSHSegment = "MSH|^~\\&|DOCTORS|HL7_FRAME|DOCTORS|doctors3|";
      MSHSegment = MSHSegment.concat(getDateTime());
      MSHSegment = MSHSegment.concat("||");
      MSHSegment = MSHSegment.concat(InEventCode);
      MSHSegment = MSHSegment.concat("^");
      MSHSegment = MSHSegment.concat(InEventCode.replace("^", "_"));
      MSHSegment = MSHSegment.concat("|CNTRL-3456|P^I|2.6 ");
      return MSHSegment;
}

private String buildEVNSegment(String EventCode){
      String InEventCode = EventCode;
      String EVNSegment = "EVN|";
      EVNSegment = EVNSegment.concat(InEventCode.substring(3, 6));
      EVNSegment = EVNSegment.concat("|");
      EVNSegment = EVNSegment.concat(getDateTime());
      EVNSegment = EVNSegment.concat("||");
      if (InEventCode == "ADT^A08"){
            EVNSegment = EVNSegment.concat("01");
      }
      else{
            EVNSegment = EVNSegment.concat("02");
      }
      EVNSegment = EVNSegment.concat("|01^DOC3|| ");
      return EVNSegment;
}

private String buildPIDSegment(HL7Tuple InTuple){
      String PIDSegment = "PID||";
      PIDSegment = PIDSegment.concat(InTuple.PatientID);
      PIDSegment = PIDSegment.concat("|");
      PIDSegment = PIDSegment.concat(InTuple.PatientID);
      PIDSegment = PIDSegment.concat("^^^DOCTORS^^HL7_FRAME||");
      PIDSegment = PIDSegment.concat(InTuple.LName);
      PIDSegment = PIDSegment.concat("^");
      PIDSegment = PIDSegment.concat(InTuple.FName);
      PIDSegment = PIDSegment.concat("^");
      PIDSegment = PIDSegment.concat(InTuple.FatherName);
      PIDSegment = PIDSegment.concat("||");
      PIDSegment = PIDSegment.concat(InTuple.Birthday.substring(0,4));
      PIDSegment = PIDSegment.concat(InTuple.Birthday.substring(5,7));
      PIDSegment = PIDSegment.concat(InTuple.Birthday.substring(8,10));
      PIDSegment = PIDSegment.concat("000000|");
      PIDSegment = PIDSegment.concat(InTuple.Gender);
      PIDSegment = PIDSegment.concat("|||");
      StringTokenizer stk = new StringTokenizer(InTuple.Address);
      int i = stk.countTokens();
      String[] AddressTokens = new String[i];
```

```
        for (int y=0;y<i;y++){
            AddressTokens[y] = stk.nextToken();
        }
        for (int x=0;x<i-1;x++){
            PIDSegment = PIDSegment.concat(AddressTokens[x]);
            PIDSegment = PIDSegment.concat(" ");
        }
        PIDSegment = PIDSegment.concat("^^");
        PIDSegment = PIDSegment.concat(AddressTokens[AddressTokens.length-1]);
        PIDSegment = PIDSegment.concat("^^^^H||");
        PIDSegment = PIDSegment.concat(InTuple.Phone);
        PIDSegment = PIDSegment.concat("||||||");
        PIDSegment = PIDSegment.concat(InTuple.Amka);
        return PIDSegment;
    }

    private String buildPV1Segment(Float InSum, String EventCode, String InExamineDate) {
        Float Sum = InSum;
        String PV1Segment = "PV1|1|O|DOCTORS_OFFICE|R|||^^DOCTOR3|||";
        PV1Segment = PV1Segment.concat(EventCode.substring(3, 6));
        PV1Segment =
            PV1Segment.concat("|||||||^^DOCTOR3||||||||||||||||||||DOCTORS_OFFICE||CR|||");
        PV1Segment = PV1Segment.concat(InExamineDate.substring(0,4));
        PV1Segment = PV1Segment.concat(InExamineDate.substring(5,7));
        PV1Segment = PV1Segment.concat(InExamineDate.substring(8,10));
        PV1Segment = PV1Segment.concat("||");
        PV1Segment = PV1Segment.concat(Sum.toString());
        PV1Segment = PV1Segment.concat("|||||| ");
        return PV1Segment;
    }

    private String buildOBXSegment(String TestElement, String Value, String Unit, int index, int
        ElementCode){
        int WorkIndex = index;
        int WorkElementCode = ElementCode;
        String OBXSegment = "OBX|";
        OBXSegment = OBXSegment.concat(Integer.toString(WorkIndex));
        OBXSegment = OBXSegment.concat("|ST|LCS_");
        OBXSegment = OBXSegment.concat(Integer.toString(WorkElementCode));
        OBXSegment = OBXSegment.concat("^");
        OBXSegment = OBXSegment.concat(TestElement);
        OBXSegment = OBXSegment.concat("^LCS||");
        OBXSegment = OBXSegment.concat(Value);
        OBXSegment = OBXSegment.concat("|");
        OBXSegment = OBXSegment.concat(Unit);
        OBXSegment = OBXSegment.concat("|||||F ");
        return OBXSegment;
    }
```

```
    private String getDateTime() {

        //ΛΗΨΗ ΤΡΕΧΟΥΣΑΣ ΗΜΕΡΟΜΗΝΙΑΣ ΣΕ ΜΟΡΦΗ ΚΑΤΑΛΛΗΛΗ ΓΙΑ ΤΟ HL7
        ΑΡΧΕΙΟ
        DateFormat dateFormat = new SimpleDateFormat("yyyyMMddHHmmss");
        Date date = new Date();
        return dateFormat.format(date);
    }

    public File getHL7MessageFile(){

        //ΑΠΟΣΤΟΛΗ ΑΡΧΕΙΟΥ ΣΕ ΑΛΛΗ ΚΛΑΣΗ
        return HL7Message;
    }

    public String getHL7MessageFileName(){

        //ΑΠΟΣΤΟΛΗ ΟΝΟΜΑΤΟΣ ΑΡΧΕΙΟΥ ΣΕ ΑΛΛΗ ΚΛΑΣΗ
        return HL7MessageFileName;
    }
}
```

9.4 HL7MessageParser.java

```
package doctor3.handlers;

import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;

import javax.swing.JOptionPane;

public class HL7MessageParser {

    //ΚΛΑΣΗ ΑΝΑΓΝΩΣΗΣ ΚΑΙ ΕΞΑΓΩΓΗΣ ΠΛΗΡΟΦΟΡΙΩΝ ΑΠΟ ΕΝΑ ΑΡΧΕΙΟ
    ΜΗΝΥΜΑΤΟΣ HL7

    public HL7Tuple readHL7FileBySegment(File InHL7File){

        //Η ΜΕΘΟΔΟΣ ΛΑΜΒΑΝΕΙ ΕΝΑ ΑΡΧΕΙΟ ΜΗΝΥΜΑΤΟΣ HL7
        //ΕΞΑΓΕΙ ΤΙΣ ΑΠΑΙΤΟΥΜΕΝΕΣ ΠΛΗΡΟΦΟΡΙΕΣ ΚΑΙ ΤΙΣ
        //ΑΠΟΘΗΚΕΥΕΙ ΣΕ ΜΙΑ ΠΛΕΙΑΔΑ
        HL7Tuple ReturningTuple = new HL7Tuple();
        String EventCode;
        try{

            //ΤΟ ΑΡΧΕΙΟ ΤΟΠΟΘΕΤΕΙΤΑΙ ΣΕ ΕΝΑ ΡΕΥΜΑ ΕΙΣΟΔΟΥ
            FileInputStream Fis = new FileInputStream(InHL7File);
```

```
InputStreamReader Isr = new InputStreamReader(Fis);
BufferedReader Br = new BufferedReader(Isr);
String StrLine;

while ((StrLine = Br.readLine()) != null){

    //ΑΝΑΛΟΓΩΣ ΜΕ ΤΟ ΕΙΔΟΣ ΤΟΥ ΤΜΗΜΑΤΟΣ, ΛΑΜΒΑΝΟΝΤΑΙ ΟΙ
    ΚΑΤΑΛΛΗΛΕΣ ΣΥΜΒΟΛΟΣΕΙΡΕΣ ΑΠΟ ΑΥΤΟ
    //ΚΑΙ ΑΝΤΙΣΤΟΙΧΙΖΟΝΤΑΙ ΣΤΑ ΣΤΟΙΧΕΙΑ ΤΗΣ ΠΛΕΙΑΔΑΣ ΠΟΥ ΘΑ ΔΟΘΕΙ
    ΣΑΝ ΕΞΟΔΟΣ
    if (StrLine.startsWith("MSH|^~\\&|")){
        String [] MSHTokens = StrLine.split("\\|");
        EventCode = MSHTokens[8].substring(0,6);
    }
    if (StrLine.startsWith("PID|")){
        String [] PIDTokens = StrLine.split("\\|");
        String [] PatientNameTokens = PIDTokens[5].split("\\^");
        ReturningTuple.LName = PatientNameTokens[0];
        ReturningTuple.FName = PatientNameTokens[1];
        ReturningTuple.FatherName = PatientNameTokens[2];
        ReturningTuple.Birthday = PIDTokens[7].substring(6, 8).concat("-")
            .concat(PIDTokens[7].substring(4, 6).concat("-"))
            .concat(PIDTokens[7].substring(0, 4));
        ReturningTuple.Gender = PIDTokens[8];
        String[] PatientAddressTokens = PIDTokens[11].split("\\^");
        ReturningTuple.Address = PatientAddressTokens[0].concat(" ")
            .concat(PatientAddressTokens[2]);
        ReturningTuple.Phone = PIDTokens[13];
        ReturningTuple.Amka = PIDTokens[19];
    }
    if (StrLine.startsWith("PV1|")){
        String [] PV1Tokens = StrLine.split("\\|");
        ReturningTuple.ExaminationDate = PV1Tokens[44].substring(6,8).concat("-");
        ReturningTuple.ExaminationDate = ReturningTuple.ExaminationDate.concat(
            PV1Tokens[44].substring(4, 6).concat("-"));
        ReturningTuple.ExaminationDate = ReturningTuple.ExaminationDate.concat(
            PV1Tokens[44].substring(0, 4));
        ReturningTuple.Sum = Float.parseFloat(PV1Tokens[46]);
    }

    if (StrLine.startsWith("OBX|")){
        String [] OBXTokens = StrLine.split("\\|");
        String [] OBXToken3 = OBXTokens[3].split("\\^");
        int index = Integer.parseInt(OBXToken3[0].substring(4));
        ReturningTuple.TestData[index] = Float.parseFloat(OBXTokens[5]);
    }
}
for (int i=0;i<13;i++){
    if (ReturningTuple.TestData[i] == null){

        //ΕΛΕΓΧΟΣ ΓΙΑ ΜΕΤΡΗΣΕΙΣ ΠΟΥ ΔΕΝ ΥΠΑΡΧΟΥΝ
        //ΤΟΤΕ ΔΙΝΕΤΑΙ Η ΑΥΘΑΙΡΕΤΗ ΤΙΜΗ -0
```

```
                    ReturningTuple.TestData[i] = Float.parseFloat("-0");
                }
            }
            Br.close();
            Isr.close();
            Fis.close();
        } catch (FileNotFoundException e) {
            JOptionPane.showMessageDialog(null,
                "Το αρχείο δεν μπορούσε να δημιουργηθεί ή να φορτωθεί"+'\n'+e.toString(),
                "Σφάλμα Δημιουργίας/Φόρτωσης Αρχείου",
                JOptionPane.ERROR_MESSAGE);
        } catch (IOException IOe){
            JOptionPane.showMessageDialog(null,
                "Το αρχείο δεν μπορούσε να προετοιμαστεί για εξαγωγή δεδομένων"+'\n'+
                    IOe.toString(),
                "Σφάλμα Ανάκτησης Δεδομένων",
                JOptionPane.ERROR_MESSAGE);
        }
        return ReturningTuple;
    }
}
```

9.5 HL7Tuple.java

```
package doctor3.handlers;

public class HL7Tuple {

    //ΚΛΑΣΗ ΠΛΕΙΑΔΑΣ ΓΙΑ ΤΗΝ ΜΕΤΑΦΟΡΑ ΔΕΔΟΜΕΝΩΝ ΑΠΟ ΚΑΙ ΠΡΟΣ ΑΡΧΕΙΑ
    ΜΗΝΥΜΑΤΩΝ HL7

    public String PatientID;
    public String FName;
    public String LName;
    public String FatherName;
    public String Birthday;
    public String Gender;
    public String Amka;
    public String Address;
    public String Phone;
    public String ExaminationDate;
    public Float Sum;
    public String FileName;

    //ΚΑΘΕ ΜΕΤΡΗΣΗ ΕΧΕΙ ΤΟ ΚΩΔΙΚΟ ΤΗΣ ΟΝΟΜΑ ΚΑΙ ΤΗ ΜΟΝΑΔΑ ΜΕΤΡΗΣΗΣ ΤΗΣ...
    public final String[][] TestDataContext = {
        {"Leyka_Aimosfairia", "K/μl"},
        {"Erythra_Aimosfairia", "M/μl"},
        {"Aimosfairini", "g/dl"},
        {"Aimatokritis", "%"},
        {"Aimopetalia", "K/μl"},
        {"Sakharo", "mg/dl"},
```

```
            {"Ouria", "mg/dl"},
            {"Kreatinini", "mg/dl"},
            {"Ouriko_Oksy", "mg/dl"},
            {"Triglykeridia", "mg/dl"},
            {"Olika_Lipidia", "mg/dl"},
            {"LDL", "mg/dl"},
            {"HDL", "mg/dl"}
        };

        //...ΚΑΙ ΤΗΝ ΤΙΜΗ ΤΗΣ
        public Float[] TestData = new Float[13];
}
```

10. Πακέτο user.gui, κλάση UserFrame.java

```
package user.gui;

import java.awt.Component;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Date;
import java.util.List;

import javax.swing.ButtonGroup;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JRadioButton;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.ScrollPaneConstants;

import org.openrdf.query.QueryLanguage;
import org.openrdf.query.TupleQuery;
import org.openrdf.query.TupleQueryResult;
import org.openrdf.repository.Repository;

import com.fluidops.fedx.Config;
import com.fluidops.fedx.FedXFactory;
import com.fluidops.fedx.FederationManager;
import com.fluidops.fedx.structures.Endpoint;
import com.hp.hpl.jena.query.DatasetFactory;
import com.hp.hpl.jena.query.QueryExecution;
import com.hp.hpl.jena.query.QueryExecutionFactory;
import com.hp.hpl.jena.query.ResultSet;
```

```
import com.hp.hpl.jena.query.ResultSetFormatter;
import com.hp.hpl.jena.query.Syntax;

import doctor2.handlers.DICOMTuple;
import doctor3.handlers.HL7Tuple;

public class UserFrame extends JFrame implements ActionListener{

    private static final long serialVersionUID = 1L;

    JButton Submit = new JButton("Εκτέλεση Ερωτήματος");
    JButton ClearAll = new JButton("Καθαρισμός Πεδίων");

    //ΚΟΥΜΠΙΑ ΕΠΙΛΟΓΗΣ ΜΗΧΑΝΗΣ ΟΜΟΣΠΟΝΔΩΝ ΕΡΩΤΗΜΑΤΩΝ (FEDERATED
    QUERY ENGINE)
    JRadioButton FedX = new JRadioButton("Χρήση FedX-1.1 Federation", false);
    JRadioButton Sparql = new JRadioButton("Χρήση Sparql SERVICE Federation", true);

    JLabel Greeting = new JLabel("Παρακαλούμε υποβάλετε το SPARQL
        ερώτημά σας προς την εφαρμογή.");
    JLabel Result = new JLabel("Αποτελέσματα:");

    JTextArea QueryText = new JTextArea();
    JTextArea ResultText = new JTextArea();

    JScrollPane ScrollReq = new JScrollPane(QueryText,
    ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED,
        ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED);
    JScrollPane ScrollRes = new JScrollPane(ResultText,
    ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED,
        ScrollPaneConstants.HORIZONTAL_SCROLLBAR_AS_NEEDED);

    Insets Whitespace = new Insets (10,10,10,10);

    GridBagLayout Gridbag = new GridBagLayout();
    GridBagConstraints Constraints;

    DICOMTuple ResultDicomTuple;
    HL7Tuple ResultHL7Tuple;

    List<Endpoint> Endpoints = null;
    Repository Repo = null;

    Long BeforeQuery = null;
    Long AfterQuery = null;

    public UserFrame(){

        //ΚΑΤΑΣΚΕΥΗ GUI

        super ("IntegraHEALTH 1.0 by Nikos Christodoulou - Καρτέλα Υποβολής Ερωτημάτων");
        Dimension ScreenSize = Toolkit.getDefaultToolkit().getScreenSize();
```

```
        setBounds(0, 0, ScreenSize.width, ScreenSize.height-40);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(Gridbag);

        Submit.addActionListener(this);
        ClearAll.addActionListener(this);

        ButtonGroup Feds = new ButtonGroup();
        Feds.add(FedX);
        Feds.add(Sparql);

        ResultText.setEditable(false);
        Font font = new Font("Courier New", Font.PLAIN, 14);
        QueryText.setFont(font);
        ResultText.setFont(font);

        addComponent(Greeting, 0, 0, 4, 1, 100, 10, GridBagConstraints.NONE,
            GridBagConstraints.CENTER);
        addComponent(ScrollReq, 0, 1, 4, 1, 100, 80, GridBagConstraints.BOTH,
            GridBagConstraints.CENTER);
        addComponent(Result, 0, 2, 4, 1, 100, 10, GridBagConstraints.NONE,
            GridBagConstraints.CENTER);
        addComponent(ScrollRes, 0, 3, 4, 1, 100, 80, GridBagConstraints.BOTH,
            GridBagConstraints.CENTER);
        addComponent(Submit, 0, 4 ,1, 1, 100, 10,
            GridBagConstraints.NONE,GridBagConstraints.CENTER);
        addComponent(ClearAll, 1, 4 ,1, 1, 100, 10,
            GridBagConstraints.NONE,GridBagConstraints.CENTER);
        addComponent(FedX, 2, 4 ,1, 1, 100, 10,
            GridBagConstraints.NONE,GridBagConstraints.CENTER);
        addComponent(Sparql, 3, 4 ,1, 1, 100, 10,
            GridBagConstraints.NONE,GridBagConstraints.CENTER);
    }

    private void addComponent(Component Component, int GridX, int GridY, int GridWidth,
        int GridHeight, int WeightX, int WeightY, int Fill, int Anchor){
        GridBagConstraints Constraints = new GridBagConstraints();
        Constraints.gridx = GridX;
        Constraints.gridy = GridY;
        Constraints.gridwidth = GridWidth;
        Constraints.gridheight = GridHeight;
        Constraints.weightx = WeightX;
        Constraints.weighty = WeightY;
        Constraints.fill = Fill;
        Constraints.anchor = Anchor;
        Constraints.insets = Whitespace;
        Gridbag.setConstraints(Component, Constraints);
        add(Component);
    }

    public void actionPerformed(ActionEvent evt) {
        Object source = evt.getSource();
```

```
if (source==Submit){

    //ΔΗΛΩΣΗ ΠΡΟΘΕΜΑΤΩΝ (PREFIXES) ΠΟΥ ΘΑ ΧΡΗΣΙΜΟΠΟΙΗΘΟΥΝ (ΔΕΝ
    ΕΜΦΑΝΙΖΟΝΤΑΙ ΣΤΟ ΠΑΡΑΘΥΡΟ)
    String Prefixes =
        "PREFIX ncit: <http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#>\n"+
        "PREFIX mged:<http://mged.sourceforge.net/ontologies/MGEDOntology.owl#>\n"+
        "PREFIX foaf: <http://xmlns.com/foaf/0.1/>\n"+
        "PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>\n"+
        "PREFIX owl: <http://www.w3.org/2002/07/owl#>\n"+
        "PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>\n"+
        "PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>\n";

    //ΚΕΙΜΕΝΟ ΤΟΥ ΕΡΩΤΗΜΑΤΟΣ
    String queryText = QueryText.getText();

    //ΑΝΑΛΟΓΑ ΜΕ ΤΟ ΕΠΙΛΕΓΜΕΝΟ ΚΟΥΜΠΙ ΧΡΗΣΙΜΟΠΟΙΕΙΤΑΙ Η
    ΚΑΤΑΛΛΗΛΗ ΜΗΧΑΝΗ ΟΜΟΣΠΟΝΔΩΝ ΕΡΩΤΗΜΑΤΩΝ
    if (Sparql.isSelected()){

        //ΧΡΗΣΗ ΜΟΝΟ ΤΗΣ SPARQL ΚΑΙ ΠΡΟΤΑΣΕΩΝ SERVICE
        System.out.println("Setting query...");
        System.out.println(new Date(System.currentTimeMillis()));
        QueryExecution x = QueryExecutionFactory.create(
                Prefixes.concat(queryText), Syntax.syntaxARQ, DatasetFactory.create());
        System.out.println("Running query...");
        BeforeQuery = System.currentTimeMillis();
        ResultSet results = x.execSelect();
        AfterQuery = System.currentTimeMillis();
        System.out.println("Done in "+ (AfterQuery-BeforeQuery) +" ms");
        JOptionPane.showMessageDialog(null,
            "Το ερώτημα απαντήθηκε σε"+ (AfterQuery-BeforeQuery) +" ms",
            "Χρόνος Απόκρισης",
            JOptionPane.INFORMATION_MESSAGE);

        //ΠΑΡΟΥΣΙΑΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ
        ResultText.setText(ResultSetFormatter.asText(results));
    }
    else{
        if (!FedX.isSelected()){
            JOptionPane.showMessageDialog(null,
                "Πρέπει να επιλέξετε μια μηχανή απάντησης ερωτημάτων!",
                "Σφάλμα Επιλογής Μηχανής Ερωτημάτων",
                JOptionPane.ERROR_MESSAGE);
        }

        //ΧΡΗΣΗ ΤΗΣ ΕΙΔΙΚΗΣ ΜΗΧΑΝΗΣ ΟΜΟΣΠΟΝΔΩΝ ΕΡΩΤΗΜΑΤΩΝ
        FEDX-1.1
        try{
```

```
                    //ΑΡΧΙΚΟΠΟΙΗΣΗ ΤΗΣ ΜΗΧΑΝΗΣ
                    Config.initialize();
                    Repo = FedXFactory.initializeFederation(".\\FEDERATION\\Doctor1-2-3.ttl");

                    //ΔΗΜΙΟΥΡΓΙΑ ΕΠΙΜΕΡΟΥΣ ΕΡΩΤΗΜΑΤΩΝ ΑΠΟ ΤΟ ΑΡΧΙΚΟ
                    System.out.println("Setting query...");
                    System.out.println(new Date(System.currentTimeMillis()));
                    TupleQuery query = Repo.getConnection().prepareTupleQuery(
                        QueryLanguage.SPARQL, Prefixes.concat(queryText));
                    System.out.println("Running query...");
                    BeforeQuery = System.currentTimeMillis();
                    TupleQueryResult res = query.evaluate();
                    AfterQuery = System.currentTimeMillis();
                    System.out.println("Done in "+ (AfterQuery-BeforeQuery) +" ms");
                    JOptionPane.showMessageDialog(null,
                        "Το ερώτημα απαντήθηκε σε"+ (AfterQuery-BeforeQuery) +" ms",
                        "Χρόνος Απόκρισης",
                        JOptionPane.INFORMATION_MESSAGE);

                    //ΠΑΡΟΥΣΙΑΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ
                    String results = "";
                    while (res.hasNext()) {
                        results=results+res.next().toString()+'\n';
                    }
                    ResultText.setText(results);

                    //"ΚΛΕΙΣΙΜΟ" ΤΗΣ ΜΗΧΑΝΗΣ
                    FederationManager.getInstance().shutDown();
                } catch (Exception e) {
                    JOptionPane.showMessageDialog(null,
                        "Exception:" + e.toString(),
                        "Σφάλμα Ερωτήματος",
                        JOptionPane.ERROR_MESSAGE);
                    e.printStackTrace();
                }
            }
        }
        if (source==ClearAll){
            ResultText.setText("");
            QueryText.setText("");
        }
    }
}
```

11. Doctor1_Log_Map.n3

```
@prefix map: <#> .
@prefix db: <jdbc:mysql://localhost/doctor1#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix d2rq: <http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#> .
```

```
@prefix jdbc: <http://d2rq.org/terms/jdbc/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ncit: <http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#>.
@prefix mged: <http://mged.sourceforge.net/ontologies/MGEDOntology.owl#>.

map:database a d2rq:Database;
      d2rq:jdbcDriver "com.mysql.jdbc.Driver";
      d2rq:jdbcDSN "jdbc:mysql://localhost/doctor1";
      d2rq:username "root";
      d2rq:password "";
      jdbc:autoReconnect "true";
      jdbc:zeroDateTimeBehavior "convertToNull";
      .

# Table elements_log
map:elements_log a d2rq:ClassMap;
       d2rq:condition "elements_log.actionTaken <> 'DELETE'";
      d2rq:dataStorage map:database;
      d2rq:uriPattern "http://example.org/doctor1/@@elements_log.amka
            |urlify@@_@@elements_log.examinationDate|urlify@@";
      d2rq:class ncit:Health_Care_Visit;
      .
map:elements_log_fName a d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:elements_log;
      d2rq:property mged:has_first_name;
      d2rq:column "elements_log.fName";
      d2rq:datatype xsd:string;
      .
map:elements_log_lName a d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:elements_log;
      d2rq:property mged:has_last_name;
      d2rq:column "elements_log.lName";
      d2rq:datatype xsd:string;
      .
map:elements_log_fatherName a d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:elements_log;
      d2rq:property foaf:givenName;
      d2rq:column "elements_log.fatherName";
      d2rq:datatype xsd:string;
      .
map:elements_log_birthday a d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:elements_log;
      d2rq:property mged:has_nodes;
      d2rq:refersToClassMap map:birthday;
      .
map:elements_log_amka a d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:elements_log;
      d2rq:property mged:unique_identifier;
      d2rq:column "elements_log.amka";
      d2rq:datatype xsd:string;
      .
map:elements_log_securityOrg a d2rq:PropertyBridge;
```

```
      d2rq:belongsToClassMap map:elements_log;
      d2rq:property mged:has_nodes;
      d2rq:refersToClassMap map:securityOrg;
      .
map:elements_log_address a d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:elements_log;
      d2rq:property mged:has_address;
      d2rq:column "elements_log.address";
      d2rq:datatype xsd:string;
      .
map:elements_log_phone a d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:elements_log;
      d2rq:property mged:has_phone;
      d2rq:column "elements_log.phone";
      d2rq:datatype xsd:string;
      .
map:elements_log_examinationDate a d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:elements_log;
      d2rq:property mged:has_nodes;
      d2rq:refersToClassMap map:examinationDate;
      .
map:elements_log_disease a d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:elements_log;
      d2rq:property mged:has_disease_state;
      d2rq:join "elements_log.disease = diseases.diseaseID";
      d2rq:uriColumn "diseases.diseaseURI";
      .
map:elements_log_remarks a d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:elements_log;
      d2rq:property mged:has_clinical_finding;
      d2rq:refersToClassMap map:findings;
      .
map:elements_log_sum a d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:elements_log;
      d2rq:property mged:has_nodes;
      d2rq:refersToClassMap map:payment;
       .

map:birthday a d2rq:ClassMap;
       d2rq:condition "elements_log.actionTaken <> 'DELETE'";
       d2rq:dataStorage map:database;
      d2rq:uriPattern ""http://example.org/doctor1/@@elements_log.amka
            |urlify@@_@@elements_log.examinationDate|urlify@@/Birthday";
      .
map:birthday_Type a d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:birthday;
      d2rq:property rdf:type;
       d2rq:uriPattern "http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Date_of_Birth";
       .
map:birthday_Value a d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:birthday;
      d2rq:property mged:has_value;
```

```
      d2rq:column "elements_log.birthday";
      d2rq:datatype xsd:date;
       .

map:securityOrg a d2rq:ClassMap;
       d2rq:condition "elements_log.actionTaken <> 'DELETE'";
       d2rq:dataStorage map:database;
      d2rq:uriPattern ""http://example.org/doctor1/@@elements_log.amka
           |urlify@@_@@elements_log.examinationDate|urlify@@/SecurityOrg";
      .
map:securityOrg_Type a d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:securityOrg;
      d2rq:property rdf:type;
       d2rq:uriPattern "http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl
           #Primary_Healthcare_Payer";
       .
map:securityOrg_Value a d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:securityOrg;
      d2rq:property mged:has_value;
      d2rq:column "elements_log.securityOrg";
      d2rq:datatype xsd:string;
      .

map:examinationDate a d2rq:ClassMap;
       d2rq:condition "elements_log.actionTaken <> 'DELETE'";
       d2rq:dataStorage map:database;
      d2rq:uriPattern ""http://example.org/doctor1/@@elements_log.amka
           |urlify@@_@@elements_log.examinationDate|urlify@@/ExaminationDate";
      .
map:examinationDate_Type a d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:examinationDate;
      d2rq:property rdf:type;
       d2rq:uriPattern "http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl
#Physical_Examination_Date";
       .
map:examinationDate_Value a d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:examinationDate;
      d2rq:property mged:has_value;
      d2rq:column "elements_log.examinationDate";
      d2rq:datatype xsd:date;
      .

map:findings a d2rq:ClassMap;
       d2rq:condition "elements_log.actionTaken <> 'DELETE'";
       d2rq:dataStorage map:database;
      d2rq:uriPattern ""http://example.org/doctor1/@@elements_log.amka
           |urlify@@_@@elements_log.examinationDate|urlify@@/Findings";
      .

map:findings_Type a d2rq:PropertyBridge;
      d2rq:belongsToClassMap map:findings;
      d2rq:property rdf:type;
```

```
        d2rq:uriPattern "http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Finding";
        .
map:findings_Value a d2rq:PropertyBridge;
        d2rq:belongsToClassMap map:findings;
        d2rq:property mged:has_value;
        d2rq:column "elements_log.remarks";
        d2rq:datatype xsd:string;
        .

map:payment a d2rq:ClassMap;
        d2rq:condition "elements_log.actionTaken <> 'DELETE'";
        d2rq:dataStorage map:database;
        d2rq:uriPattern ""http://example.org/doctor1/@@elements_log.amka
              |urlify@@_@@elements_log.examinationDate|urlify@@/Payment";
        .
map:payment_Type a d2rq:PropertyBridge;
        d2rq:belongsToClassMap map:payment;
        d2rq:property rdf:type;
        d2rq:uriPattern "http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#Payment";
        .
map:payment_Value a d2rq:PropertyBridge;
        d2rq:belongsToClassMap map:payment;
        d2rq:property mged:has_value;
        d2rq:column "elements_log.sum";
        d2rq:datatype xsd:float;
        .
```

12. SQL.ttl, DICOM.ttl, HL7.ttl

```
@prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd:    <http://www.w3.org/2001/XMLSchema#> .
@prefix module: <http://joseki.org/2003/06/module#> .
@prefix joseki: <http://joseki.org/2005/06/configuration#> .
@prefix ja:     <http://jena.hpl.hp.com/2005/11/Assembler#> .

<> rdfs:label "Joseki Configuration File" .

<#server> a joseki:Server;
   joseki:initialization
       [ module:implementation
          [ module:className <java:org.joseki.util.ServiceInitSimple>;
            rdfs:label "Example initializer" ]
       ] .

<#fixed_service> a joseki:Service;
   rdfs:label         "service point";
   joseki:serviceRef   "sparql";
   joseki:dataset      <#doctor1>;
   joseki:processor    <#ProcessorSPARQL_FixedDS> .


<#ProcessorSPARQL_FixedDS> a joseki:Processor;
```

```
   rdfs:label "SPARQL processor for fixed datasets";
   module:implementation <#ImplSPARQL>;
   # This processor does not accept queries with FROM/FROM NAMED
   joseki:allowExplicitDataset       "false"^^xsd:boolean;
   joseki:allowWebLoading            "false"^^xsd:boolean;
   joseki:lockingPolicy              joseki:lockingPolicyMRSW .

<#ImplSPARQL> a joseki:ServiceImpl;
   module:className <java:org.joseki.processors.SPARQL> .

<#doctor1> a ja:RDFDataset;
   rdfs:label "doctor1 Dataset";
   ja:defaultGraph <#sqlDBModel>.

<#sqlDBModel> a ja:RDBModel;
   ja:connection
   [
      ja:dbType        "MySQL" ;
      ja:dbURL         "jdbc:mysql://localhost/sql_rdf_model" ;
      ja:dbUser        "root" ;
      ja:dbPassword    "" ;
      ja:dbClass       "com.mysql.jdbc.Driver" ;
   ] ;
   ja:modelName "SQL_Model".
```

Το παραπάνω κείμενο είναι αυτό του αρχείου SQL.ttl. Για τα άλλα δύο αρχεία πρέπει να γίνουν οι αντικαταστάσεις doctor1 ➔ doctor2, sql_rdf_model ➔ dicom_rdf_model, SQL_Model ➔ DICOM_Model (DICOM.ttl) και doctor1 ➔ doctor3, sql_rdf_model ➔ hl7_rdf_model, SQL_Model ➔ HL7_Model (HL7.ttl)

13. SQL.bat, DICOM.bat, HL7.bat

```
@echo off
cd\
cd ptyxiakh\help_programs\joseki-3.4.4
bin\rdfserver --port 2030 SQL.ttl
```

Κατά την ίδια λογική που χρησιμοποιήθηκε στην προηγούμενη ενότητα, παραπάνω παρουσιάζεται μόνο το SQL.bat. Με τις αντικαταστάσεις 2030 ➔ 2040 και SQL.ttl ➔ DICOM.ttl, και 2030 ➔ 2050 και SQL.ttl ➔ HL7.ttl λαμβάνονται τα DICOM.bat, HL7.bat, αντίστοιχα.

14. Doctor1-2-3.ttl

```
@prefix fluid: <http://fluidops.org/config#>.

<http://SQL> fluid:store "SPARQLEndpoint";
fluid:SPARQLEndpoint "http://localhost:2030/sparql".
```

<http://DICOM> fluid:store "SPARQLEndpoint";
fluid:SPARQLEndpoint "http://localhost:2040/sparql".

<http://HL7> fluid:store "SPARQLEndpoint";
fluid:SPARQLEndpoint "http://localhost:2050/sparql".

15. SQL_RULES.rules

@prefix pre: <http://jena.hpl.hp.com/prefix#>.
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ncit: <http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#>.
@prefix mged: <http://mged.sourceforge.net/ontologies/MGEDOntology.owl#>.
@include <RDFS>.

[SameIDrule: (?g owl:sameAs ?f) <- (?f mged:unique_identifier ?a) (?g mged:unique_identifier ?a) (?f mged:nas_nodes ?b1) (?b1 rdf:type ncit:Physical_Examination_Date) (?b1 mged:has_value ?c) (?g mged:nas_nodes ?b2) (?b2 rdf:type ncit:Physical_Examination_Date) (?b2 mged:has_value ?c)]

16. DICOM_RULES.rules

@prefix pre: <http://jena.hpl.hp.com/prefix#>.
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ncit: <http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#>.
@prefix mged: <http://mged.sourceforge.net/ontologies/MGEDOntology.owl#>.

@include <RDFS>.

[sameID: (?u owl:sameAs ?f) <- (?f mged:unique_identifier ?a) (?u mged:unique_identifier ?a) (?f mged:has_nodes ?n1) (?n1 rdf:type ncit:Physical_Examination_Date) (?n1 mged:has_value ?b) (?u mged:has_nodes ?n2) (?n2 rdf:type ncit:Physical_Examination_Date) (?n2 mged:has_value ?b)]

[Osteoporosis: (?m mged:has_disease_state ncit:Osteoporosis) <- (?m mged:has_measurement ?n) (?n rdf:type ncit:Bone_Mineral_Density_Test) (?n mged:has_value ?v) isLiteral(?v) lessThan(?v "-2.5"^^xsd:float)]

[LowBoneDensity: (?m rdfs:label "possible Osteoporosis"^^xsd:string) <- (?m mged:has_measurement ?n) (?n rdf:type ncit:Bone_Mineral_Density_Test) (?n mged:has_value ?v) isLiteral(?v) lessThan(?v "-1"^^xsd:float) ge(?v "-2.5"^^xsd:float)]

[Obesity: (?m mged:has_disease_state ncit:Obesity) <- (?m mged:has_measurement ?n) (?n rdf:type ncit:Adipose_Tissue) (?n mged:has_value ?v) isLiteral(?v) greaterThan(?v "32"^^xsd:float)]

[Obesity2: (?m rdfs:label "possible Obesity"^^xsd:string) <- (?m mged:has_measurement ?n) (?n rdf:type ncit:Adipose_Tissue) (?n mged:has_value ?v) isLiteral(?v) lessThan(?v "32"^^xsd:float) ge(?v "25"^^xsd:float)]

17. HL7_RULES.rules

```
@prefix pre: <http://jena.hpl.hp.com/prefix#>.
@prefix ncit: <http://ncicb.nci.nih.gov/xml/owl/EVS/Thesaurus.owl#>.
@prefix mged: <http://mged.sourceforge.net/ontologies/MGEDOntology.owl#>.
@prefix foaf: <http://xmlns.com/foaf/0.1/>.

@include <RDFS>.

[SameID: (?u owl:sameAs ?f) <- (?f mged:unique_identifier ?a) (?u mged:unique_identifier ?a)
(?f mged:has_nodes ?nA) (?nA rdf:type ncit:Physical_Examination_Date) (?nA mged:has_value ?b)
(?u mged:has_nodes ?nB) (?nB rdf:type ncit:Physical_Examination_Date) (?nB mged:has_value ?b)]

[SameID2: (?u owl:sameAs ?f) <- (?f mged:has_ID ?a) (?u mged:has_ID ?a) (?f mged:has_nodes ?nA)
(?nA rdf:type ncit:Physical_Examination_Date) (?nA mged:has_value ?b) (?u mged:has_nodes ?nB)
(?nB rdf:type ncit:Physical_Examination_Date) (?nB mged:has_value ?b)]

[MaleAnemia1: (?m mged:has_disease_state ncit:Anemia) <- (?m foaf:gender "Male"^^xsd:string)
(?m mged:has_measurement ?n) (?n rdf:type ncit:Hematocrit) (?n mged:has_value ?v) isLiteral(?v)
lessThan(?v "40"^^xsd:float)]

[MaleAnemia2: (?m mged:has_disease_state ncit:Anemia) <- (?m foaf:gender "Male"^^xsd:string)
(?m mged:has_measurement ?n) (?n rdf:type ncit:Hemoglobin_Measurement) (?n mged:has_value ?v)
isLiteral(?v) lessThan(?v "13"^^xsd:float)]

[FemaleAnemia1: (?m mged:has_disease_state ncit:Anemia) <- (?m foaf:gender "Female"^^xsd:string)
(?m mged:has_measurement ?n) (?n rdf:type ncit:Hematocrit) (?n mged:has_value ?v) isLiteral(?v)
lessThan(?v "37"^^xsd:float)]

[FemaleAnemia2: (?m mged:has_disease_state ncit:Anemia) <- (?m foaf:gender "Female"^^xsd:string)
(?m mged:has_measurement ?n) (?n rdf:type ncit:Hemoglobin_Measurement) (?n mged:has_value ?v)
isLiteral(?v) lessThan(?v "11.6"^^xsd:float)]

[Thrombocytopenia: (?m mged:has_disease_state ncit:Thrombocytopenia) <-
(?m mged:has_measurement ?n) (?n rdf:type ncit:Platelet_Count) (?n mged:has_value ?v) isLiteral(?v)
lessThan(?v "150"^^xsd:float)]

[HighLeukocyte: (?m mged:has_disease_state ncit:Infectious_Disorder) <-
(?m mged:has_measurement ?n) (?n rdf:type ncit:Leukocyte_Count) (?n mged:has_value ?v)
isLiteral(?v) greaterThan(?v "11"^^xsd:float)]

[HighLDL: (?m mged:has_disease_state ncit:Type_II_Hyperlipidemia) <- (?m mged:has_measurement
?n) (?n rdf:type ncit:Serum_LDL_Cholesterol_Measurement) (?n mged:has_value ?v) isLiteral(?v)
greaterThan(?v "130"^^xsd:float)]

[HighTriglycerides: (?m mged:has_disease_state ncit:Hypertriglyceridemia) <-
(?m mged:has_measurement ?n) (?n rdf:type ncit:Triglyceride_Measurement) (?n mged:has_value ?v)
isLiteral(?v) greaterThan(?v "150"^^xsd:float)]

[Nephropathy: (?m mged:has_disease_state ncit:Nephropathy) <- (?m mged:has_measurement ?n) (?n
rdf:type ncit:Creatinine_Measurement) (?n mged:has_value ?v) isLiteral(?v)
greaterThan(?v "1.5"^^xsd:float)]
```

```
[Hyperlipidemia: (?m mged:has_disease_state ncit:Hyperlipidemia) <- (?m mged:has_measurement ?n)
(?n rdf:type ncit:Lipid_Measurement) (?n mged:has_value ?v) isLiteral(?v)
greaterThan(?v "1000"^^xsd:float)]

[FemaleHyperuricemia: (?m mged:has_disease_state ncit:Hyperuricemia) <- (?m foaf:gender
"Female"^^xsd:string) (?m mged:has_measurement ?n) (?n rdf:type
ncit:Uric_Acid_Crystal_Measurement) (?n mged:has_value ?v) isLiteral(?v)
greaterThan(?v "6.2"^^xsd:float)]

[MaleHyperuricemia: (?m mged:has_disease_state ncit:Hyperuricemia) <- (?m foaf:gender
"Male"^^xsd:string) (?m mged:has_measurement ?n) (?n rdf:type
ncit:Uric_Acid_Crystal_Measurement) (?n mged:has_value ?v) isLiteral(?v)
greaterThan(?v "7.5"^^xsd:float)]

[Hyperglycemia: (?m mged:has_disease_state ncit:Hyperglycemia) <- (?m mged:has_measurement ?n)
(?n rdf:type ncit:Fasting_Blood_Sugar_Measurement) (?n mged:has_value ?v) isLiteral(?v)
greaterThan(?v "110"^^xsd:float)]

[Hypoglycemia: (?m mged:has_disease_state ncit:Hypoglycemia) <- (?m mged:has_measurement ?n)
(?n rdf:type ncit:Fasting_Blood_Sugar_Measurement) (?n mged:has_value ?v) isLiteral(?v)
lessThan(?v "60"^^xsd:float)]
```

18. Αποθηκευμένες διαδικασίες (σκανδάλες, triggers) των βάσεων

18.1 Βάση doctor1

```
-- Triggers `elements`
--
DROP TRIGGER IF EXISTS `after_insert`;
DELIMITER //
CREATE TRIGGER `after_insert` AFTER INSERT ON `elements`
    FOR EACH ROW BEGIN
        INSERT INTO ELEMENTS_LOG
            SET actionTaken = 'INSERT',
            fName = NEW.fName,
            lName = NEW.lName,
            fatherName = NEW.fatherName,
            birthday = NEW.birthday,
            amka = NEW.amka,
            securityOrg = NEW.securityOrg,
            address = NEW.address,
            phone = NEW.phone,
            examinationDate = NEW.examinationDate,
            disease = NEW.disease,
            remarks = NEW.remarks,
            sum = NEW.sum;
    END
//
DELIMITER ;


DROP TRIGGER IF EXISTS `after_update`;
DELIMITER //
CREATE TRIGGER `after_update` AFTER UPDATE ON `elements`
```

```
    FOR EACH ROW BEGIN
        DELETE FROM ELEMENTS_LOG
        WHERE amka = OLD.amka
        AND examinationDate = OLD.examinationDate;
        INSERT INTO ELEMENTS_LOG
            SET actionTaken = 'UPDATE',
            fName = NEW.fName,
            lName = NEW.lName,
            fatherName = NEW.fatherName,
            birthday = NEW.birthday,
            amka = NEW.amka,
            securityOrg = NEW.securityOrg,
            address = NEW.address,
            phone = NEW.phone,
            examinationDate = NEW.examinationDate,
            disease = NEW.disease,
            remarks = NEW.remarks,
            sum = NEW.sum;
    END
//
DELIMITER ;


DROP TRIGGER IF EXISTS `after_delete`;
DELIMITER //
CREATE TRIGGER `after_delete` AFTER DELETE ON `elements`
    FOR EACH ROW BEGIN
        DELETE FROM ELEMENTS_LOG
        WHERE amka = OLD.amka
        AND examinationDate = OLD.examinationDate;
        INSERT INTO ELEMENTS_LOG
            SET actionTaken = 'DELETE',
            fName = OLD.fName,
            lName = OLD.lName,
            fatherName = OLD.fatherName,
            birthday = OLD.birthday,
            amka = OLD.amka,
            securityOrg = OLD.securityOrg,
            address = OLD.address,
            phone = OLD.phone,
            examinationDate = OLD.examinationDate,
            disease = OLD.disease,
            remarks = OLD.remarks,
            sum = OLD.sum;
    END
//
DELIMITER ;
```

18.2 Βάση doctor2

```
-- Triggers `dicom_elements`
--
DROP TRIGGER IF EXISTS `after_insert`;
DELIMITER //
CREATE TRIGGER `after_insert` AFTER INSERT ON `dicom_elements`
    FOR EACH ROW BEGIN
        INSERT INTO ACTION_LOG
            SET actionTaken = 'INSERT',
            oldFileName = "",
```

```
            newFileName = NEW.dicomFileName ;
    END
//
DELIMITER ;


DROP TRIGGER IF EXISTS `after_update`;
DELIMITER //
CREATE TRIGGER `after_update` AFTER UPDATE ON `dicom_elements`
    FOR EACH ROW BEGIN
        DELETE FROM ACTION_LOG
        WHERE newFileName = OLD.dicomFileName;
        INSERT INTO ACTION_LOG
            SET actionTaken = 'UPDATE',
            oldFileName = OLD.dicomFileName,
            newFileName = NEW.dicomFileName ;
    END
//
DELIMITER ;


DROP TRIGGER IF EXISTS `after_delete`;
DELIMITER //
CREATE TRIGGER `after_delete` AFTER DELETE ON `dicom_elements`
    FOR EACH ROW BEGIN
        DELETE FROM ACTION_LOG
        WHERE newFileName = OLD.dicomFileName;
        INSERT INTO ACTION_LOG
            SET actionTaken = 'DELETE',
            oldFileName = OLD.dicomFileName,
            newFileName = "" ;
    END
//
DELIMITER ;
```

18.3 Βάση doctor3

```
-- Triggers `hl7_elements`
--
DROP TRIGGER IF EXISTS `after_insert`;
DELIMITER //
CREATE TRIGGER `after_insert` AFTER INSERT ON `hl7_elements`
    FOR EACH ROW BEGIN
        INSERT INTO ACTION_LOG
            SET actionTaken = 'INSERT',
            oldFileName = "",
            newFileName = NEW.hl7FileName ;
    END
//
DELIMITER ;


DROP TRIGGER IF EXISTS `after_update`;
DELIMITER //
CREATE TRIGGER `after_update` AFTER UPDATE ON `hl7_elements`
    FOR EACH ROW BEGIN
        DELETE FROM ACTION_LOG
        WHERE newFileName = OLD.hl7FileName;
        INSERT INTO ACTION_LOG
```

```
            SET actionTaken = 'UPDATE',
            oldFileName = OLD.hl7FileName,
            newFileName = NEW.hl7FileName ;
****    END
//
DELIMITER ;


DROP TRIGGER IF EXISTS `after_delete`;
DELIMITER //
CREATE TRIGGER `after_delete` AFTER DELETE ON `hl7_elements`
    FOR EACH ROW BEGIN
        DELETE FROM ACTION_LOG
        WHERE newFileName = OLD.hl7FileName;
        INSERT INTO ACTION_LOG
            SET actionTaken = 'DELETE',
            oldFileName = OLD.hl7FileName,
            newFileName = "" ;
    END
//
DELIMITER ;
```