ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Δημιουργία Web εφαρμογής για τη μετατροπή μίας Αντικειμενοσχεσιακής Βάσης Δεδομένων σε linked data

Του φοιτητη

Κόλιας Ζήσης
Αρ. Μητρώου: 10/3608

Επιβλέπων καθηγητής

Κεραμόπουλος Ευκλείδης

Θεσσαλονίκη 2014

ΠΕΡΙΛΗΨΗ

Σε ένα κόσμο ο οποίος εξελίσσεται με γρήγορους ρυθμούς, όπου η τεχνολογία παίζει καταλυτικό ρόλο σε αυτό. Σε ένα κόσμο όπου το διαδίκτυο μπαίνει στην καθημερινότητα μας όλο και πιο πολύ. Σε ένα κόσμο όπου τα δεδομένα και η αποθήκευση τους είναι σημαντική όσο και η προσπέλαση τους με διάφορους τρόπους και με διάφορες μορφές.

Η παρούσα πτυχιακή παρουσιάζει κάποιους τρόπους αποθήκευσης και προσπέλασης των δεδομένων σε μια συγκεκριμένη μορφή. Πιο συγκεκριμένα θα δούμε τι είναι οι Αντικειμενο-σχεσιακές Βάσεις Δεδομένων, θα δούμε τις τεχνολογίες JDBC και JSF, καθώς επίσης θα κάνουμε μια εισαγωγή στον Σημασιολογικό Ιστό και τα πρότυπα RDF, RDF Schema και RDF Turtle. Επίσης θα δούμε πως μετατρέπονται τα δεδομένα μιας σχεσιακής βάσης σε μορφή RDF Turtle σύμφωνα με αλγόριθμους που υπάρχουν, καθώς επίσης θα δούμε και τον αλγόριθμο μετατροπής που δημιουργήσαμε σε μορφή ψευδοκώδικα.

Θα υλοποιήσουμε μια διαδικτυακή εφαρμογή με την τεχνολογία JSF η οποία θα μετατρέπει μια Αντικειμενο-σχεσιακή Βάση Δεδομένων σε RDF Turtle μορφή. Για την υλοποίηση της Αντικειμενο-σχεσιακής Βάσης χρησιμοποιήθηκε η IBM DB2, για την JSF εφαρμογή το πρόγραμμα Netbeans και η γλώσσα προγραμματισμού Java.

Στα κεφάλαια που ακολουθούν περιγράφονται αναλυτικά οι τεχνολογίες που χρησιμοποιήθηκαν μαζί με αποσπάσματα κώδικα και τις αντίστοιχες επεξηγήσεις τους και παρατίθεται ένας οδηγός χρήσης της εφαρμογής.

ABSTRACT

In a world that is evolving rapidly, where technology plays a major role in this. In a world where the Internet comes everyday in our life more and more. In a world where data and the storage of them is important as the access in different ways and in different forms.

This thesis presents some ways to store and access data in a specific format. In particular we will see what are Object-Relational Databases, we will see the JDBC and JSF technologies, as well we will make an introduction to the Semantic Web and RDF, RDF Schema and RDF Turtle standards. Also we will see how data from a relational database converted to RDF Turtle format according to algorithms that already exist, as well as we will see the conversion algorithm we have created in the form of pseudocode.

We will implement a web application with JSF technology which converts an object-relational database in RDF Turtle form. For the implementation of the Object-Relational Database we used the IBM DB2, for the JSF application the Netbeans program and the programming language Java.

In the following chapters there is detailed description of the technologies which has been used with excerpts codes. Also there is a guide of how to use the application.

ΕΥΧΑΡΙΣΤΙΕΣ

Πρώτα από όλους θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου Δρ. Κεραμόπουλο Ευκλείδη για την υποστήριξη του και την ευκαιρία που μου έδωσε να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα.

Επίσης θα ήθελα να ευχαριστήσω τους καθηγητές του τμήματος Πληροφορικής του ΑΤΕΙ Θεσσαλονίκης, είτε βρίσκονται είτε όχι ακόμα στο τμήμα, με τους οποίους συνεργάστηκα και πήρα σημαντικά εφόδια για την συνέχεια της καριέρας μου.

Τέλος θέλω να ευχαριστήσω την οικογένεια μου που με στήριξε όλα αυτά τα χρόνια με κάθε τρόπο και μου έδωσε την ευκαιρία να φτάσω στην ολοκλήρωση των σπουδών μου.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ

ΕΙΣΑΓΩΓΗ

Μέχρι στιγμής υπάρχουν αρκετοί οδηγοί και παραδείγματα στο διαδίκτυο, αναφορικά με τους κανόνες μετατροπής των εγγράφων από απλές σχεσιακές βάσεις δεδομένων σε μορφή αποδεκτή για Διασυνδεδεμένα Δεδομένα, όπως είναι για παράδειγμα το πρότυπο RDF 1.1 Turtle. Ο σκοπός της παρούσας πτυχιακής είναι η δημιουργία μιας διαδικτυακής εφαρμογής που θα μετατρέπει μια Αντικειμενο-σχεσιακή βάση δεδομένων σε ένα αρχείο Turtle έτοιμο για επεξεργασία στον Σημασιολογικό Ιστό.

Αρχικά θα κάνουμε μια μικρή εισαγωγή στις βάσεις δεδομένων και στα συστήματα διαχείρισης βάσεων δεδομένων.

Στην συνέχεια θα δούμε τι είναι οι Αντικειμενο-σχεσιακές βάσεις δεδομένων, τις ιδιαιτερότητες τους και θα δημιουργήσουμε ένα παράδειγμα στην IBM DB2.

Έπειτα θα δούμε την τεχνολογία JDBC και τα χαρακτηρίστηκα της, καθώς επίσης θα δούμε τον τρόπο και τον λόγο που την χρησιμοποιήσαμε.

Στην συνέχεια θα κάνουμε μια εισαγωγή στον Σημασιολογικό Ιστό και τα συνδεδεμένα δεδομένα, καθώς επίσης θα δούμε και κάποιες λεπτομέρειες σχετικά με τα RDF, RDF Schema και RDF Turtle πρότυπα.

Μετά από αυτά θα δούμε την τεχνολογία JSF που χρησιμοποιήσαμε για την δημιουργία της διαδικτυακής μας εφαρμογής.

Τέλος θα δούμε ένα οδηγό χρήσης της εφαρμογής για την καλύτερη κατανόηση της.

ΚΕΦΑΛΑΙΟ 1 – Εισαγωγή στις Βάσεις Δεδομένων

ΕΙΣΑΓΩΓΗ

Οι Βάσεις Δεδομένων αποτελούν σήμερα ένα πολύ σημαντικό κομμάτι για κάθε επιχείρηση καθώς χρησιμοποιούνται για την διατήρηση διαφόρων δεδομένων, για διάφορους λόγους και σκοπούς. Στο κεφάλαιο αυτό θα κάνουμε μια μικρή εισαγωγή στις βάσεις δεδομένων και θα δούμε τις βασικότερες κατηγορίες αυτών.

1.1 Τι είναι μία Βάση Δεδομένων (DataBase)

Μια Βάση Δεδομένων (DataBase) είναι ένας οργανωμένος τρόπος αποθήκευσης πληροφοριών, σε οργανωμένες διακριτές συλλογές σχετιζόμενων ηλεκτρονικά και ψηφιακά αποθηκευμένων δεδομένων.

Για να μπορέσουμε να διαχειριστούμε μια βάση δεδομένων χρησιμοποιούμε προγράμματα τα οποία ονομάζονται *Συστήματα Διαχείρισης Βάσεων Δεδομένων (*DBMS, DataBase Management System*)* και με την βοήθεια τους μπορούμε να αποθηκεύσουμε, να προσθέσουμε, να τροποποιήσουμε, να εμφανίσουμε ή και να διαγράψουμε τα αποθηκευμένα δεδομένα.

Οι βασικότερες κατηγορίες βάσεων δεδομένων περικλείονται στις εξής:

- ✓ Σχεσιακές Βάσεις Δεδομένων
- ✓ Αντικειμενοστρεφείς Βάσεις Δεδομένων
- ✓ Αντικειμενο-σχεσιακές Βάσεις Δεδομένων

1.1.1 Σχεσιακές Βάσεις Δεδομένων

Στις Σχεσιακές *(Relational)* Βάσεις δεδομένων, τα δεδομένα συνδέονται μεταξύ τους με σχέσεις *(relations)*, οι οποίες προκύπτουν από τα κοινά πεδία που υπάρχουν σε διαφορετικά αρχεία. Τα αρχεία αποκαλούνται *πίνακες (tables)*, οι εγγραφές ονομάζονται *γραμμές (rows)* και τα πεδία *στήλες (columns)*. Η ύπαρξη μιας κοινής τιμής στα πεδία δύο αρχείων καθορίζει και μια σχέση μεταξύ των γραμμών διαφορετικών πινάκων.

Οι σχεσιακές βάσεις δεδομένων έχουν το πλεονέκτημα ότι είναι λογικά κατανοητές και πολύ ευέλικτες και δεκτικές σε αλλαγές για αυτό και είναι ο κυρίαρχος τύπος βάσεων δεδομένων που χρησιμοποιούνται σήμερα.

1.1.2 Αντικειμενοστρεφείς Βάσεις Δεδομένων

Οι Σχεσιακές Βάσεις Δεδομένων ωστόσο είχαν και κάποια μειονεκτήματα και ένα από αυτά είναι ότι δεν μπορούν να υποστηρίξουν τις αρχές του αντικειμενοστρεφούς προγραμματισμού. Έτσι λοιπόν δημιουργήθηκαν οι Αντικειμενοστρεφείς Βάσεις Δεδομένων.

Οι Αντικειμενοστρεφείς Βάσεις Δεδομένων είναι συστήματα βάσεων δεδομένων που κρατούν τα δεδομένα τους σαν αντικείμενα, κατά τις αρχές του αντικειμενοστρεφούς προγραμματισμού και προσφέρουν επιπλέον δυνατότητες σε σχέση με τις Σχεσιακές Βάσεις Δεδομένων. Μερικές από τις δυνατότητες αυτές είναι :

- ✓ Δεν έχουν περιορισμούς στους τύπους δεδομένων
- ✓ Δεν έχουν περιορισμούς στις γλώσσες αιτημάτων
- ✓ Είναι ευέλικτες και προσφέρουν στο σχεδιαστή τη δυνατότητα να προσδιορίσει τόσο τη δομή πολύπλοκων αντικειμένων, όσο και τις συναρτήσεις που μπορούν να εφαρμοστούν στα αντικείμενα αυτά

1.1.3 Αντικειμενο-σχεσιακές Βάσεις Δεδομένων

Ωστόσο η εξέλιξη της τεχνολογίας έφερε νέες ανάγκες τις οποίες δεν μπορούσαν να καλύψουν τα δύο προαναφερθέντα μοντέλα. Έτσι δημιουργήθηκαν οι Αντικειμενο-σχεσιακές Βάσεις Δεδομένων. Οι Αντικειμενο-σχεσιακές Βάσεις Δεδομένων συνδυάζουν στοιχεία από τις Σχεσιακές και τις Αντικειμενοστρεφείς Βάσεις Δεδομένων και ουσιαστικά αποτελούν μια γέφυρα μεταξύ των σχεσιακών και των αντικειμενοστρεφών προτύπων αφού συνδυάζουν στοιχεία και από τις Σχεσιακές και τις Αντικειμενοστρεφείς Βάσεις Δεδομένων.

Στην παρούσα πτυχιακή θα ασχοληθούμε με τις Αντικειμενο-σχεσιακές Βάσεις Δεδομένων και τα Αντικειμενο-σχεσιακά Συστήματα Διαχείρισης Βάσεων Δεδομένων (ORDBMS) και συγκεκριμένα με την IBM DB2 όπου παρακάτω παρουσιάζονται τα χαρακτηριστικά αυτών.

ΕΠΙΛΟΓΟΣ

Οι βάσεις δεδομένων αποτελούν ένα σημαντικό γρανάζι της τεχνολογίας και βοηθούν σε πολλούς τομείς. Σε αυτό το κεφάλαιο είδαμε κάποια βασικά πράγματα για τις βάσεις δεδομένων και τις κατηγορίες αυτών.

Στο επόμενο κεφάλαιο θα δούμε τις Αντικειμενο-σχεσιακές Βάσεις Δεδομένων με τις οποίες θα ασχοληθούμε μιας και ο στόχος της παρούσας πτυχιακής έχει άμεση σχέση με αυτές.

ΚΕΦΑΛΑΙΟ 2 – Αντικειμενο-σχεσιακές Βάσεις Δεδομένων

ΕΙΣΑΓΩΓΗ

Κατά την διάρκεια της δεκαετίας του 1990, τα Αντικειμενοστρεφή Συστήματα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ) έκαναν μια περιορισμένη εισβολή η οποία σιγά-σιγά έσβησε. Έτσι αντί να γίνει μια μετανάστευση από τα σχεσιακά ΣΔΒΔ στα αντικειμενοστρεφή, δημιουργήθηκαν τα Αντικειμενο-σχεσιακά ΣΔΒΔ τα οποία ενσωματώνουν πολλές από τις δυνατότητες των αντικειμενοστρεφή ΣΔΒΔ.

Σε αυτό το κεφάλαιο θα δούμε τις Αντικειμενο-σχεσιακές Βάσεις Δεδομένων και τα χαρακτηριστικά αυτών με την βοήθεια παραδειγμάτων. Θα δούμε πως δημιουργούνται οι τύποι από τον χρήστη και τα χαρακτηριστικά τους. Θα δούμε πως δημιουργούνται και εκτελούνται οι μέθοδοι και οι συναρτήσεις.

2.1 Αντικειμενοσχεσιακές Βάσεις Δεδομένων

Το Σχεσιακό μοντέλο και το Αντικειμενοστρεφές μοντέλο, αποτελούσαν για πολύ καιρό την βασική επιλογή για ένα Σύστημα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ), ωστόσο η εξέλιξη της τεχνολογίας και των εφαρμογών δημιούργησαν την ανάγκη για την δημιουργία ενός νέου μοντέλου. Έτσι δημιουργήθηκε το Αντικειμενο-σχεσιακό μοντέλο το οποίο ακολουθεί μία διαφορετική προσέγγιση για την υποστήριξη των σύγχρονων εφαρμογών σε σχέση με το Αντικειμενοστρεφή μοντέλο.

Το Αντικειμενο-σχεσιακό μοντέλο υποστηρίζει πλήρως το Σχεσιακό μοντέλο δεδομένων, ενώ έχει εμπλουτισθεί με επιπλέον δυνατότητες με στόχο την υποστήριξη πολύπλοκων αντικειμένων. Θα λέγαμε πως προέρχεται από τον εμπλουτισμό του Σχεσιακού μοντέλου με αντικειμενοστρεφείς ιδιότητες.

Το μεγαλύτερο πλεονέκτημα των Αντικειμενο-σχεσιακών βάσεων δεδομένων είναι η ευκολία χρήσης τους και η υποστήριξή τους από μεγάλες εταιρείες κατασκευής σχεσιακών βάσεων δεδομένων, αυτός είναι και ο λόγος που έχει συμβάλει στη μεγάλη αποδοχή τους από το αγοραστικό κοινό. Παρόλα αυτά όσο και να εξελιχθεί η τεχνολογία αυτή ποτέ δε θα καταφέρει να αποδώσει, όπως μια καθαρά αντικειμενοστρεφή λύση.

Ωστόσο το μεγαλύτερο μειονέκτημα των Αντικειμενο-σχεσιακών βάσεων δεδομένων βρίσκεται στην ταχύτητα απόκρισής τους, η οποία ειδικά σε πολύπλοκες δομές δεδομένων, εμφανίζεται πολύ πιο μικρή από την αντίστοιχη μιας αμιγώς αντικειμενοστρεφούς βάσης δεδομένων.

Απαιτήσεις της Βάσης Δεδομένων	Αντικειμενοστραφείς Βάσεις Δεδομένων	Σχεσιακές Βάσεις Δεδομένων	Αντικειμενοσχεσιακές (Σχεσιακές με αντικειμενοστραφή διασύνδεση) Βάσεις Δεδομένων
Πολυπλοκότητα μοντέλου δεδομένων	Μεγάλη	Μικρή	Ανάλογα με τον κατασκευαστή
Τύποι δεδομένων καθορισμένοι από τον χρήστη	Ναι	Όχι	Ναι
Διαχείριση διαφορετικών εκδόσεων των ίδιων δεδομένων	Ναι	Όχι	Όχι
Εργαλεία για την ανάπτυξη της εφαρμογής	Αντικειμενοστραφείς γλώσσες προγραμματισμού	Γλώσσες προγραμματισμού 4ης γενιάς	Γλώσσες προγραμματισμού 3ης και 4ης γενιάς
Κληρονομικότητα	Ναι	Όχι	Ανάλογα με τον κατασκευαστή
Πολυμορφισμός	Ναι	Όχι	Ανάλογα με τον κατασκευαστή

Σχήμα 1 – Σύγκριση Μοντέλων

Το Αντικειμενο-σχεσιακό μοντέλο προσφέρει κάποια επιπλέον χαρακτηριστικά σε σχέση με το σχεσιακό. Προσφέρει την δυνατότητα στον χρήστη να δημιουργήσει νέους τύπους, τύπους οι οποίοι είτε θα βασίζονται στους ενσωματωμένους τύπους του συστήματος (απλοί τύποι), είτε θα είναι εξ ολοκλήρου ορισμένοι από τον χρήστη (σύνθετοι τύποι).

Επίσης το Αντικειμενο-σχεσιακό μοντέλο μας προσφέρει την δυνατότητα για τον σχηματισμό ιεραρχία, εξειδίκευσης μεταξύ των νέων τύπων μέσω της απλής κληρονομικότητας, όπου ένας τύπος μπορεί να κληρονομηθεί από άλλους τύπους κληρονομώντας τα χαρακτηριστικά και τις μεθόδους του.

Επιπλέον σε μια Αντικειμενο-σχεσιακή βάση δεδομένων χρησιμοποιείται η ταυτότητα αντικειμένων. Έτσι ώστε κάθε αντικείμενο να μπορεί να ξεχωρίσει από άλλα αντικείμενα.

Τέλος για τον ορισμό συσχετίσεων μεταξύ γραμμών το Αντικειμενο-σχεσιακό μοντέλο επιτρέπει τη χρήση του τύπου αναφοράς για να κάνει αναφορά σε κάποιο άλλο τύπο.

Στην συνέχεια του κεφαλαίου θα παρουσιαστούν πιο αναλυτικά τα χαρακτηρίστηκα αυτά.

2.2 Απλοί Τύποι (distinct types)

Οι απλοί τύποι βασίζονται στους ενσωματωμένους τύπους του συστήματος (integer,varchar,double…) και ουσιαστικά θα λέγαμε ότι αποτελούν μετονομασία ενός βασικού τύπου. Με την χρήση τους μπορούμε να διαχειριστούμε καλύτερα τη σημασιολογία των δεδομένων και να δημιουργήσουμε δικούς μας τύπους. Ωστόσο οι απλοί τύποι έχουν κάποιους περιορισμούς όπως το ότι δεν μπορούν να χρησιμοποιηθούν για να δηλωθούν πάνω τους άλλοι απλοί τύποι.

Για να δημιουργήσουμε έναν απλό τύπο το κάνουμε με την εντολή Create Distinct Type με την εξής σειρά.

Create Distinct Type όνομα AS τύπο WITH COMPARISONS;

Όπου η παράμετρος WITH COMPARISONS χρησιμοποιείται για να ενεργοποιηθεί η δυνατότητα σύγκρισης μεταξύ δύο απλών τύπων, οι οποίοι έχουν δηλωθεί βασιζόμενοι σε ενσωματωμένο τύπο της db2 που μπορεί να συγκριθεί.

Για παράδειγμα για την δημιουργία ενός τύπου βασιζόμενο στον τύπο varchar θα το δηλώσουμε κάπως έτσι :

Create Distinct Type typos_diplomatos AS varchar(30) WITH COMPARISONS;

Εδώ δημιουργούμε τον τύπο με όνομα typos_diplomatos ο οποίος είναι τύπου varchar.

2.3 Σύνθετοι Τύποι (Structured types)

Οι σύνθετοι τύποι σε αντίθεση με τους απλούς τύπους διαθέτουν κάποια επιπλέον χαρακτηριστικά τα οποία είναι : Ταυτότητα Αντικειμένου, Ιδιότητες ,Κληρονομικότητα, Αναφορά. Χρησιμοποιώντας σύνθετους τύπους μπορούμε να αναπαραστήσουμε σύνθετες δομές. Στην συνέχεια περιγράφονται τα χαρακτηριστικά των σύνθετων τύπων.

2.3.1 Ταυτότητα Αντικειμένου (OID – Object Identity)

Η ταυτότητα αντικειμένου είναι η ιδιότητα ενός αντικειμένου που ξεχωρίζει ένα αντικείμενο από κάποιο άλλο. Η ταυτότητα αντικειμένου δημιουργείται αυτόματα από το σύστημα είναι μοναδική, αμετάβλητη και ανεξάρτητη από τις ιδιότητες του αντικειμένου και παραμένει ζωντανή για όλη την διάρκεια του αντικειμένου ανεξάρτητα από τις αλλαγές που γίνονται στο αντικείμενο αυτό. Επίσης από την στιγμή που εισαχθεί η ταυτότητα τότε αυτή δεν μπορεί να αλλάξει. Η ταυτότητα δηλώνεται όταν δημιουργούμε τον πίνακα πάνω σε ένα τύπο και δηλώνεται ως εξής

Create table όνομα πίνακα OF σύνθετος τύπος (REF is OID USER GENERATED);

Και η τιμή της εισάγεται με την εκτέλεση ενός INSERT INTO ερωτήματος ως εξής

INSERT INTO όνομα πίνακα (Oid, άλλες στήλες) VALUES (σύνθετος τύπος(τιμή), άλλες τιμες);

2.3.2 Ιδιότητες

Η κατάσταση (state) των αντικειμένων χαρακτηρίζεται από τις τιμές που έχουν αυτά για κάποιες ιδιότητες (properties). Οι ιδιότητες μπορεί να είναι είτε χαρακτηριστικά (attributes) - κατάσταση του αντικειμένου, είτε αναφορές (relationships) ανάμεσα στο αντικείμενο και κάποια άλλα αντικείμενα, είτε μέθοδοι – συμπεριφορές. Η κατάσταση περιγράφει τις ιδιότητες του αντικειμένου οι οποίες μπορεί να είναι Απλού τύπου, Σύνθετου τύπου, Τύπου Συλλογής ή Αναφορά σε ένα άλλου τύπου αντικείμενο. Γενικά, οι τιμές των ιδιοτήτων ενός αντικειμένου μπορούν να αλλάξουν στο χρόνο.

2.3.3 Κληρονομικότητα

Οι σύνθετοι τύποι σε αντίθεση με τους απλούς τύπους μπορούν να κληρονομηθούν από άλλους σύνθετους τύπους μέσω της απλής κληρονομικότητας.

Κάθε σύνθετος τύπος μπορεί να έχει τα δικά του χαρακτηρίστηκα και επιπλέον να κληρονομεί και τα χαρακτηριστικά ενός άλλου σύνθετου τύπου.

Ο τύπος από τον οποίο μια υποκατηγορία κληρονομεί χαρακτηριστικά είναι γνωστός ως υπερτύπος (supertype) και οι υποκατηγορίες που τον κληρονομούν

ονομάζονται υποτύποι (subtypes). Για να ορίσουμε την κληρονομικότητα γράφουμε ως εξής

Create type όνομα υποτύπου under όνομα υπερτύπου AS (χαρακτηριστικά υποτύπου) mode db2sql;

Και για την δήλωση του πίνακα ως εξής

Create table όνομα πίνακα υποτάξης OF όνομα υποτύπου under όνομα πίνακα υπερτάξης INHERIT SELECT PRIVILEGES;

2.3.4 Αναφορά REF

Τύπος Αναφοράς ονομάζεται ο τύπος πάνω στον οποίο μπορεί να χτιστεί η ταυτότητα ενός νέου τύπου και δηλώνεται κατά την δημιουργία ενός τύπου με την CREATE TYPE.

Αν δεν ορίσουμε τον τύπο αναφοράς τότε η ταυτότητα ορίζεται πάνω στον τύπο VARCHAR(16), ο οποίος είναι ο προεπιλεγμένος τύπος.

Μπορούμε να ορίσουμε τύπο αναφοράς μόνο για τον υπερτύπο μιας ιεραρχίας, καθώς για τους υποτύπους του, θα δημιουργηθούν ταυτότητες βάση του τύπου αναφοράς του υπερτύπου τους.

Για να ορίσουμε τύπους γράφουμε ως εξής

Create type όνομα τύπου AS (ιδιότητες)

Ref using int mode db2sql;

Το γνώρισμα τύπου αναφοράς, δηλώνει ότι ένα γνώρισμα είναι αναφορά σε αντικείμενο ενός άλλου τύπου.

Για να ορίσουμε γνώρισμα τύπου αναφοράς γράφουμε ως εξής

Create type όνομα τύπου AS (όνομα γνωρίσματος αναφοράς REF(όνομα τύπου))

2.3.5 INSTANTIABLE και NOT FINAL

Στο Αντικειμενο-σχεσιακό μοντέλο μπορούμε να δημιουργήσουμε πίνακες – αντικείμενα με βάση κάποιον τύπο που έχει δημιουργήσει ο χρήστης. Για να το

επιτύχουμε αυτό θα πρέπει να βάλουμε το INSTANTIABLE όταν θα ορίσουμε τον τύπο αυτόν. Αν δεν θέλουμε να δημιουργήσουμε πίνακες – αντικείμενα τότε βάζουμε το NOT INSTANTIABLE το οποίο αντιστοιχεί στις αφηρημένες κλάσεις(abstract class). Ενώ αν δεν βάλουμε τίποτα από τα δύο τότε είναι σαν να έχουμε βάλει το INSTANTIABLE, καθώς είναι η προεπιλεγμένη επιλογή.

Παρακάτω να δηλώσουμε τα προαναφερθέντα γράφουμε ως εξής.

CREATE TYPE όνομα τύπου AS (ιδιότητες)
INSTANTIABLE

NOT FINAL

Ref using int mode db2sql;

Η χρήση του NOT FINAL δηλώνει ότι επιτρέπεται να δημιουργήσουμε νέους τύπους χρησιμοποιώντας τον τύπο αυτό στην ιεραρχία.

2.4 Δημιουργία Πινάκων

Για να δημιουργήσουμε ένα πίνακα σε μια αντικειμενο-σχεσιακή βάση δεδομένων με βάση ένα σύνθετο τύπο το κάνουμε με την εντολή CREATE TABLE και γράφουμε με τον εξής τρόπο.

CREATE TABLE όνομα πίνακα OF όνομα τύπου (REF is OID USER GENERATED);

Και για να δημιουργήσουμε ένα πίνακα ο οποίος θα κληρονομεί κάποιον ως εξής

CREATE TABLE όνομα πίνακα υποτάξης OF όνομα υποτύπου under όνομα πίνακα υπερτάξης INHERIT SELECT PRIVILEGES;

2.5 Μέθοδοι

Σε ένα Αντικειμενο-σχεσιακό ΣΔΒΔ μπορούμε να ορίσουμε μεθόδους και να εφαρμόσουμε ειδικές πράξεις σε τιμές με τύπους τους οποίους έχει ορίσει ο χρήστης, καθώς επίσης και να συμπεριλάβουμε μεθόδους στην δημιουργία ενός σύνθετου τύπου. Με αυτόν τον τρόπο μπορούμε να εκτελέσουμε συγκεκριμένες λειτουργίες χωρίς να γράφουμε τον ίδιο κώδικα ξανά και ξανά.

2.5.1 Δήλωση Μεθόδου

Για να δηλώσουμε μια μέθοδο πρώτα δηλώνουμε τον τύπο και μετά υλοποιείται το σώμα της μεθόδου. Η δήλωση μίας μεθόδου γίνεται είτε με την εντολή CREATE TYPE, είτε με την εντολή ALTER TYPE.

ALTER TYPE όνομα τύπου ADD METHOD όνομα μεθόδου()

RETURNS τύπο που επιστρέφει

LANGUAGE SQL;

Παρακάτω παρατίθεται ένα παράδειγμα δήλωσης μεθόδου στον τύπο Ypallilos_t με όνομα getAge() και ο τύπος που επιστρέφει είναι integer.

ALTER TYPE Ypallilos_t

ADD METHOD getAge()

RETURNS integer

LANGUAGE SQL;

Για την σύνταξη του σώματος της μεθόδου χρησιμοποιείται η εντολή CREATE METHOD. Το σώμα της μεθόδου μπορεί να δημιουργηθεί είτε εξωτερικά με μία γλώσσα προγραμματισμού, ή εσωτερικά με εντολές SQL.

CREATE METHOD όνομα_μεθόδου (όνομα_παραμέτρου1 τύπος1, …)

RETURNS τύπος_που_επιστρέφει **FOR** σύνθετος_τύπος_που_δηλώθηκε

RETURN SQL-command

Στο αποτέλεσμα που θέλουμε να μας επιστρέφει μπορούμε να χρησιμοποιήσουμε είτε SELECT ερωτήματα και με τη λέξη self.. όνομα στήλης να αναφερθούμε στην στήλη που θέλουμε, είτε να χρησιμοποιήσουμε συνθήκες με την εντολή CASE.

Παρακάτω παρατίθεται ένα παράδειγμα σύνταξης της μεθόδου getAge() που αναφέρθηκε παραπάνω, η οποία επιστρέφει 1 αν η ηλικία είναι μεγαλύτερη από 40 και 0 αν δεν είναι.

```
CREATE METHOD getAge()
RETURNS integer
FOR Ypallilos_t
 RETURN
      (CASE
        WHEN (self..hlikia > 40)
          THEN 1
        ELSE 0
      END);
```

Παρακάτω παρατίθεται ένα παράδειγμα σύνταξης της μεθόδου get ari8mos_dromologiwn() η οποία επιστρέφει τον αριθμό των δρομολογίων του πίνακα Odigos.

```
CREATE METHOD get ari8mos_dromologiwn()
RETURNS integer
FOR Odigos_t
RETURN SELECT SELF.. ari8mos_dromologiwn FROM Odigos;
```

2.5.2 Διαγραφή Μεθόδου

Για να διαγράψουμε μία μέθοδο θα πρέπει να διαγράψουμε τόσο το σώμα της καθώς επίσης θα πρέπει να την διαγράψουμε και από την δήλωση αν υπάρχει σε κάποιο τύπο.

Για να διαγράψουμε το σώμα μίας μεθόδου και να σβηστεί η μέθοδος αυτή χρησιμοποιούμε την εντολή DROP με τον παρακάτω τρόπο:

DROP όνομα μεθόδου FOR όνομα τύπου;

Παραδείγματος χάρη για να διαγράψουμε την getAge() μέθοδο.

DROP getAge() FOR Ypallilos_t;

Για να διαγράψουμε μία μέθοδο από έναν τύπο θα πρέπει πρώτα να διαγράψουμε το σώμα της μεθόδου και μετά να αφαιρέσουμε τη μέθοδο από τον τύπο καθώς στη σύνταξη του τύπου που δηλώθηκε η μέθοδος υπάρχει ακόμα. Για να την διαγράψουμε το κάνουμε όπως φαίνεται παρακάτω :

ALTER TYPE όνομα τύπου DROP όνομα μεθόδου;

Παραδείγματος χάρη για να αφαιρέσουμε την μέθοδο getAge() από τον τύπο Ypallilos_t .

ALTER TYPE Ypallilos_t DROP getAge();

2.5.3 Κλήση Μεθόδου

Για να καλέσουμε μια μέθοδο υπάρχουν δύο τρόποι είτε με τον συνδυασμό των συμβόλων της παύλας και του μεγαλύτερου -> ή με την DEREF και την διπλή τελεία όπως φαίνεται παρακάτω.

SELECT πεδίο1, DEREF(oid)..όνομα μεθόδου()

FROM όνομα πίνακα;

Ή

SELECT πεδίο1, oid -> όνομα μεθόδου()

FROM όνομα πίνακα;

Παραδείγματος χάρη για να καλέσουμε την μέθοδο getAge() γράφουμε ως εξής :

SELECT oid->getAge () FROM Ypallilos;

2.6 Συναρτήσεις

Στα Αντικειμενο-σχεσιακά Συστήματα Βάσεων Δεδομένων υπάρχει η δυνατότητα ο χρήστης να δημιουργήσει νέες συναρτήσεις και να τις χρησιμοποιήσει σε αιτήματα. Στόχος της δημιουργίας των νέων συναρτήσεων είναι η επεξεργασία των δεδομένων και η παραγωγή αποτελεσμάτων ανάλογα με τις ανάγκες και τις απαιτήσεις του χρήστη.

Τα συστατικά στοιχεία τα οποία αποτελούν μια συνάρτηση είναι δύο η υπογραφή της και το σώμα της. Η υπογραφή μίας συνάρτησης χωρίζεται σε άλλα τρία επιμέρους στοιχεία, αποτελείται από το όνομα της , τις παραμέτρους της και τον τύπο των παραμέτρων αυτής.

Στα Αντικειμενο-σχεσιακά Συστήματα Βάσεων Δεδομένων υπάρχουν τριών ειδών συναρτήσεις οι Εξωτερικές , οι SQL και οι Sourced συναρτήσεις.

Οι εξωτερικές συναρτήσεις,είναι συναρτήσεις που γράφονται σε μια γλώσσα προγραμματισμού όπως η java , η C κτλ, και για να δημιουργήσουμε μια εξωτερική συνάρτηση πρέπει να δηλώσουμε το όνομα της, τις παραμέτρους της, τον τύπο που επιστρέφει με το RETURNS. Επίσης θα πρέπει να δηλώσουμε ότι δεν θα χρησιμοποιηθεί SQL καθώς επίσης να δηλώσουμε και την γλώσσα που θα γραφεί, το στυλ που θα δηλωθούν οι παράμετροι και τέλος το εξωτερικό όνομα της συνάρτησης.

Οι SQL Συναρτήσεις σε αντίθεση με τις εξωτερικές κάνουν χρήση μόνο της SQL return εντολής και μπορούν να περιέχουν ένα Select αίτημα ή άλλες συναρτήσεις του συστήματος. Για να δημιουργήσουμε τέτοιου είδους συναρτήσεις τις δημιουργούμε με την εντολή create function.

Οι SOURCE συναρτήσεις αποτελούνται από τις έτοιμες συναρτήσεις του συστήματος και τις χρησιμοποιούμε όταν θέλουμε να κάνουμε χρήση των έτοιμων συναρτήσεων του συστήματος.

2.7 Εναύσματα (Triggers)

Σε μια Αντικειμενο-σχεσιακή βάση δεδομένων μπορούμε να δημιουργήσουμε εναύσματα για την καλύτερη διαχείριση των δεδομένων της βάσης μας. Τα εναύσματα λοιπόν είναι μια διαδικασία την οποία δηλώνει ο διαχειριστής της βάσης δεδομένων και η οποία ενεργοποιείται αυτόματα από το Σύστημα Διαχείρισης Βάσης Δεδομένων κάθε φορά που συμβαίνει κάποια μεταβολή ορισμένου τύπου στα δεδομένα που υπάρχουν στην βάση δεδομένων.

Κάθε έναυσμα το οποίο υπάρχει στο σχήμα της βάση δεδομένων πρέπει να έχει μοναδικό όνομα και να μην υπάρχει άλλο έναυσμα με το ίδιο όνομα.

Τα συστατικά στοιχεία ενός εναύσματος περιγράφονται σε τρία διακριτά μέρη, το γεγονός, την συνθήκη και την ενέργεια.

Το γεγονός έχει να κάνει με την ενεργοποίηση του εναύσματος με βάση κάποια μεταβολή που γίνεται στην βάση δεδομένων. Μεταβολή όπως εισαγωγή , διαγραφή ή ενημέρωση (INSERT,DELETE,UPDATE) κάποιων δεδομένων ή δημιουργία, διαγραφή ή ενημέρωση (CREATE,DROP,ALTER) κάποιου πίνακα.

Η συνθήκη αποτελείται είτε από ένα αίτημα, είτε από έναν έλεγχο ο οποίος εκτελείται με την ενεργοποίηση του εναύσματος.

Η ενέργεια είναι μία διαδικασία η οποία εκτελείται όταν ισχύει η συνθήκη και ενεργοποιείται το έναυσμα.

Ένα έναυσμα εκτελείται χρονικά είτε πριν (Before) συμβεί κάποια μεταβολή στην βάση δεδομένων, είτε αφότου (After) συμβεί κάποια μεταβολή.

Η σύνταξη ενός εναύσματος έχει την εξής μορφή :

CREATE TRIGGER όνομα-εναύσματος
{ INSERT | DELETE | UPDATE } [OF όνομα_στήλης [, όνομα_στήλης]*] ON όνομα_πίνακα
REFERENCING
FOR EACH ROW / FOR EACH STATEMENT
MODE DB2ROW / DB2SQL
WHEN (συνθήκη)
BEGIN
Σώμα του εναύσματος
END

Παραδείγματος χάρη δημιουργούμε το έναυσμα με όνομα DROMOLOGIO1 το οποίο εκτελείτε όταν γίνεται μια εισαγωγή στον πίνακα DROMOLOGIO και ενημερώνει την στήλη WRA δίνοντας της την τρέχουσα ώρα του συστήματος.

Για την συγκεκριμένη εισαγωγή θα γράψουμε ως εξής.

CREATE TRIGGER DROMOLOGIO1

AFTER INSERT ON DROMOLOGIO

REFERENCING NEW AS N

FOR EACH ROW MODE DB2SQL

UPDATE DROMOLOGIO SET WRA = CURRENT TIMESTAMP

WHERE OID = N.OID

ΕΠΙΛΟΓΟΣ

Τα Αντικειμενο-σχεσιακά Συστήματα Διαχείρισης Δεδομένων δημιουργήθηκαν από τις ανάγκες και την εξέλιξη της τεχνολογίας και θα λέγαμε ότι λειτουργούν ως μια ενδιάμεση λύση ανάμεσα στα Σχεσιακά και Αντικειμενοστρεφή ΣΔΒΔ. Σε αυτό το κεφάλαιο είδαμε τα χαρακτηριστικά των Αντικειμενο-σχεσιακών ΣΔΒΔ και τα αναλύσαμε με την χρήση παραδειγμάτων.

Στο επόμενο κεφάλαιο θα κάνουμε μια εισαγωγή στην JDBC τεχνολογία, την οποία χρησιμοποιούμε για να έχουμε πρόσβαση στα δεδομένα και τα μεταδεδομένα μιας Αντικειμενο-σχεσιακής Βάσης Δεδομένων.

ΚΕΦΑΛΑΙΟ 3 – Ανάκτηση δεδομένων με την JDBC

ΕΙΣΑΓΩΓΗ

Για να είμαστε σε θέση να ανακτήσουμε τα δεδομένα και τα μεταδεδομένα, που είναι αποθηκευμένα στην βάση δεδομένων, θα χρησιμοποιήσουμε την JDBC τεχνολογία, η οποία μας δίνει την δυνατότητα αυτή.

Σε αυτό το κεφάλαιο θα εξετάσουμε την JDBC τεχνολογία και τις μεθοδολογίες που χρησιμοποιεί για την ανάκτηση των δεδομένων και των μεταδεδομένων από την βάση δεδομένων, με την χρήση παραδειγμάτων και Java κώδικα.

3.1 Τεχνολογία JDBC (Java Database Connectivity)

Η τεχνολογία JDBC είναι ένα interface (API) για την Java που ορίζει πώς ένας χρήστης μπορεί να έχει πρόσβαση σε μια βάση δεδομένων. Παρέχει συναρτήσεις για εξαγωγή, πρόσθεση, ανανέωση ή διαγραφή δεδομένων σε μια βάση. Με την JDBC το ίδιο εκτελέσιμο αποκτά πρόσβαση σε πολλά περιβάλλοντα DBMS, χωρίς να χρειάζεται να επαναμεταγλωτίζεται. Επίσης μπορεί να εκτελεστεί και εκτός του Διακομιστή της Βάσης Δεδομένων δίνοντας τη δυνατότητα ταυτόχρονης σύνδεσης με πολλούς και διαφορετικούς Διακομιστές Βάσεων Δεδομένων.

3.2 JDBC - Υλοποίηση και σύνδεση στην IBM DB2

Για να μπορέσουμε να ανακτήσουμε τα δεδομένα και τα μεταδεδομένα για αρχή θα πρέπει να μπορέσουμε να συνδεθούμε στην βάση δεδομένων που είναι αποθηκευμένα αυτά που θέλουμε να ανακτήσουμε. Παρακάτω φαίνονται τα βήματα για να μπορέσουμε να συνδεθούμε στην βάση δεδομένων με την JDBC και συγκεκριμένα να συνδεθούμε στην IBM DB2 .

1.Φόρτωση του προγράμματος οδήγησης JDBC με χρήση της μεθόδου Class.forName.

2.Ορισμός του URL για την σύνδεση, όπου ορίζεται η θέση του διακομιστή της Βάσης Δεδομένων, η θύρα και το όνομα της Βάσης Δεδομένων.

3.Δημιουργία της δικτυακής σύνδεσης με τη Βάση Δεδομένων με χρήση του URL και γνωρίζοντας το Username και Password για την πρόσβαση στη Βάση χρησιμοποιώντας την κλάση Connection.

```
Private String driverClassName = "com.ibm.db2.jcc.DB2Driver";
static Connection dbConnection = null;
public void connect() {
      try {
         Class.forName(driverClassName);  (1)
           String url = "jdbc:db2://" + ip + ":" + port + "/" + dbname;  (2)
     dbConnection = DriverManager.getConnection(url, username, password); (3)

         msg = "Συνδεθήκατε στην βάση δεδομένων με επιτυχία";

      } catch (Exception e) {
         e.printStackTrace();
         msg = "Κάποιο πρόβλημα προέκυψε δεν καταφέρατε να συνδεθείτε δοκιμάστε
ξανά!!!!";
      }
   }
```

3.3 ResultSet

Ένα ResultSet είναι μια δομή στην οποία μπορούμε να αποθηκεύσουμε τα δεδομένα που παίρνουμε από την βάση δεδομένων μας . Ένα ResultSet είναι ένα σύνολο γραμμών που περιέχει διάφορα δεδομένα από μια βάση δεδομένων και διαβάζοντας αυτό παίρνουμε τα δεδομένα που είναι αποθηκευμένα σε αυτά. Στην εφαρμογή μας τα ResultSet μας βοηθούν ώστε να ανακτήσουμε τα δεδομένα που μας ενδιαφέρουν.

3.4 JDBC Εκτέλεση Ερωτημάτων

Για να μπορούμε να διαχειριστούμε και να έχουμε πρόσβαση στα δεδομένα μια βάσης δεδομένων, θα πρέπει να μπορούμε να εκτελέσουμε διάφορα ερωτήματα σε αυτήν. Ερωτήματα όπως ερωτήματα ανάκτησης και ενημέρωσης δεδομένων καθώς επίσης και ερωτήματα τροποποίησης σχημάτων και πινάκων.

Αφού καταφέρουμε και συνδεθούμε με την βάση δεδομένων η JDBC τεχνολογία μας προσφέρει τις κλάσεις Statement, PreparedStatement και CallableStatement με τις οποίες μπορούμε να εκτελούμε τέτοιου είδους ερωτήματα.

3.4.1 Statement

Η κλάση Statement μας προσφέρει την δυνατότητα για αποστολή SQL ερωτημάτων στην βάση δεδομένων μας και χρησιμοποιείται για πρόσβαση γενικής χρήσης και για αποστολή στατικών SQL ερωτημάτων. Η κλάση Statement δεν μπορεί να πάρει παραμέτρους και το αποτέλεσμα της αποθηκεύεται σε ένα ResultSet. Επίσης υπάρχει η αναλογία ένα ResultSet αντικείμενο ανά ένα Statement αντικείμενο, δηλαδή ότι ένα αντικείμενο Statement μπορεί να τρέχει κάθε φορά και οπότε μπορούμε να έχουμε πρόσβαση σε αυτό μέχρι να το κλείσουμε. Για να εκτελέσουμε ένα δεύτερο ερώτημα με το ίδιο Statement θα πρέπει να κλείσουμε το τρέχον ή να δημιουργήσουμε νέο Statement.

Προτού χρησιμοποιήσουμε ένα αντικείμενο Statement και εκτελέσουμε ερωτήματα θα πρέπει πρώτα να το δημιουργήσουμε χρησιμοποιώντας το αντικείμενο τύπου Connection και την μέθοδο createStatement().

```
static Statement statement = null;
try {
      statement = dbConnection.createStatement();
   . . .
}
catch (SQLException e) {
   . . .
}
```

Έπειτα για να εκτελέσουμε ερωτήματα χρησιμοποιούμε τις μεθόδους, executeQuery για ερωτήματα ανάκτησης δεδομένων (SELECT) και την executeUpdate για ερωτήματα ενημέρωσης, δημιουργίας και διαγραφής (Create, Alter, Drop, Update, Delete). Το αποτέλεσμα των μεθόδων αυτών αποθηκεύεται σε ένα ResultSet αντικείμενο.

```
ResultSet query = statement.executeQuery("select * from
TableName");
```

Εδώ παίρνουμε όλα τα δεδομένα από ένα πίνακα και τα αποθηκεύουμε σε ένα ResultSet εν ονόματι query.

3.4.2 PreparedStatement

Η PreparedStatement είναι υποκλάση της κλάσης Statement και κληρονομεί όλη την λειτουργικότητα της κλάσης Statement. Χρησιμοποιείται όταν θέλουμε να εκτελέσουμε το ίδιο ή παρόμοιο ερώτημα πολλές φορές καθώς είναι πιο γρήγορη μιας και είναι προμεταγλωτισμένη.

Η κλάση PreparedStatement λειτουργεί παρόμοια με την Statement καθώς χρησιμοποιεί τις ίδιες μεθόδους executeQuery και executeUpdate με την διαφορά ότι χρησιμοποιούνται αποκλειστικά χωρίς παραμέτρους. Επίσης με την PreparedStatement μπορούμε να χρησιμοποιήσουμε παραμέτρους κατά την δημιουργία αντικειμένου τύπου PreparedStatement.

Για να δημιουργήσουμε ένα PreparedStatement αντικείμενο το κάνουμε χρησιμοποιώντας το αντικείμενο τύπου Connection και την μέθοδο prepareStatement() .

```
PreparedStatement pstmt = null;
try {
   String SQL = "Update Employees SET age = ? WHERE id = ?";
   pstmt = dbConnection.prepareStatement(SQL);
   . . .
}
catch (SQLException e) {
   . . .
}
```

Όλες οι παράμετροι δηλώνονται με το σύμβολο ?, το οποίο είναι γνωστό ως δείκτης παραμέτρου και πρέπει να γίνει εισαγωγή τιμών για κάθε παράμετρο πριν από την εκτέλεση του SQL Statement. Αυτό γίνεται με την μέθοδο setXXX() όπου XXX είναι ο τύπος της παραμέτρου.

3.4.3 CallableStatement

Η CallableStatement χρησιμοποιείται για να έχουμε πρόσβαση σε αποθηκευμένες διαδικασίες (Store Procedures) που υπάρχουν σε μια βάση δεδομένων και μπορεί να συνταχθεί και με παραμέτρους.

```
CallableStatement cstmt = null;
try {
   String SQL = "{call getEmpName (?, ?)}";
   cstmt = dbConnection.prepareCall (SQL);
   . . .
}
catch (SQLException e) {
   . . .
}
```

Όπου εδώ δημιουργούμε ένα αντικείμενο CallableStatement και καλούμε την αποθηκευμένη διαδικασία με το όνομα getEmpName.

3.5 Μεταδεδομένα (metadata)

Τα μεταδεδομένα είναι δεδομένα για τα δεδομένα (data about data) και περιγράφουν τη δομή μίας βάσης ή ενός συγκεκριμένου τύπου δεδομένων όπως ενός αντικειμένου, ενός πίνακα ή ακόμα και της ίδιας της βάσης.

Υπάρχουν τρεις τύποι μεταδεδομένων στην JDBC και είναι οι εξής :

- ResultSetMetaData προσφέρει πληροφορίες σχετικά με τις στήλες ενός αντικειμένου ResultSet
- DatabaseMetaData προσφέρει πληροφορίες σχετικά με τη δομή μιας βάσης δεδομένων.
- ParameterMetaData προσφέρει πληροφορίες σχετικά με τις παραμέτρους ενός αντικειμένου PreparedStatement

Στην παρούσα πτυχιακή θα ασχοληθούμε κυρίως με τα DatabaseMetaData και τα ResultSetMetaData που σε συνεργασία με τα ResultSet θα μας βοηθήσουν να έχουμε πρόσβαση στις πληροφορίες, τα μεταδεδομένα και τα δεδομένα που θέλουμε. Ωστόσο θα δούμε και την ParameterMetaData εν συντομία.

3.5.1 DatabaseMetaData

Το DatabaseMetaData χρησιμοποιήθηκε για να έχουμε πρόσβαση στα ονόματα και τους τύπους των πινάκων της βάσης. Τον τύπο τον χρειαζόμαστε για να μπορούμε να κάνουμε διάκριση ανάμεσα στους πίνακες που υπάρχουν στην

βάση και να πάρουμε μόνο αυτούς που είναι φτιαγμένοι με βάση το αντικειμενο-σχεσιακό μοντέλο. Τα ονόματα των πινάκων θα τα χρειαστούμε για να διαβάσουμε στην συνέχεια τα δεδομένα που έχουν καθώς επίσης και για να φτιάξουμε το τελικό μας αρχείο.

Παρακάτω βλέπουμε πως χρησιμοποιείται ένα DatabaseMetaData.

1. Δημιουργία αντικειμένου DatabaseMetaData
2. Ανάκτηση δεδομένων και αποθήκευση σε ένα ResultSet
3. Διάβασμα ResultSet και ανάκτηση συγκεκριμένων πληροφοριών όπως το όνομα και ο τύπος των πινάκων.

```
 static DatabaseMetaData meta = null;   (1)
 meta = dbConnection.getMetaData(); (1)

ResultSet rs = meta.getTables(null, schemaname, null, null); (2)

while (rs.next()) {
        String TableName = rs.getString("TABLE_NAME");
        String tableType = rs.getString("TABLE_TYPE");
} (3)
```

Το DatabaseMetaData μας παρέχει μια πληθώρα πληροφοριών που έχουν να κάνουν με μια βάση δεδομένων. Εδώ θα παρουσιάσουμε κάποιες βασικές δυνατότητες και κάποιες βασικές μεθόδους.

getTables()

Με την μέθοδο getTables παίρνουμε πληροφορίες που έχουν να κάνουν με τους πίνακες που υπάρχουν στην βάση δεδομένων.

getUDTs

Με την μέθοδο getUDTs παίρνουμε πληροφορίες που έχουν να κάνουν με τους τύπους που έχει ορίσει ο χρήστης σύνθετους αλλά και απλούς.

getPrimaryKeys

Με την μέθοδο getPrimaryKeys παίρνουμε πληροφορίες που έχουν να κάνουν με το κύριο κλειδί κάποιου πίνακα.

getImportedKeys

Με την μέθοδο getImportedKeys παίρνουμε πληροφορίες που έχουν να κάνουν με το ξένο κλειδί κάποιου πίνακα.

getProcedures

Με την getProcedures παίρνουμε πληροφορίες που έχουν να κάνουν με τις αποθηκευμένες διαδικασίες σε μια βάση.

3.5.2 ResultSetMetaData

Το ResultSetMetaData μας δίνει πληροφορίες που έχουν να κάνουν με τις στήλες ενός πίνακα. Το ResultSetMetaData χρησιμοποιήθηκε για να έχουμε πρόσβαση στα δεδομένα και τα μεταδεδομένα των πινάκων που είναι αποθηκευμένα στην βάση δεδομένων. Για να το υλοποιήσουμε αυτό εκτελούμε ένα SQL SELECT ερώτημα για κάθε πίνακα με την χρήση μιας επανάληψης για να πάρουμε όλους τους πίνακες και με την χρήση του ResultSetMetaData παιρνούμε στοιχεία όπως τον αριθμό των στηλών για κάθε πίνακα, τον τύπο κάθε στήλης και τα δεδομένα που έχει κάθε στήλη ανάλογα τον τύπο της στήλης.

Παρακάτω βλέπουμε με την χρήση κώδικα πως χρησιμοποιήσαμε τα ResultSetMetaData και τι ακριβώς καταφέραμε με αυτό.

1. Δημιουργία SQL SELECT ερωτήματος και αποθήκευση αυτού σε ένα ResultSet
2. Δημιουργία αντικειμένου ResultSetMetaData για το ResultSet του SELECT ερωτήματος
3. Επανάληψη για να πάρουμε όλες τις στήλες
4. Αποθήκευση του τύπου κάθε στήλης
5. Αναλόγως το τύπο της στήλης παίρνουμε και τα δεδομένα με την getColumnName() και τα αποθηκεύουμε σε ένα Vector σε μορφή Turtle(παρακάτω παρατίθεται ένα δείγμα από τον κώδικα και ένας πίνακας με τους τύπους και την αντίστοιχη μέθοδο που παίρνουμε για τον κάθε τύπο)

```
ResultSet query = statement.executeQuery("select * from " + TableName); (1)
ResultSetMetaData rsmd = query.getMetaData(); (2)
for (int i = 2; i <= rsmd.getColumnCount(); i++) { (3)
        int jdbcType = rsmd.getColumnType(i);  (4)
switch (jdbcType) {
                case -1://longvarchar
                  String slgch = "\t:" + rsmd.getColumnName(i) + " \"" +
query.getString(rsmd.getColumnName(i) + "\"" + end);
                  vec.addElement(slgch);
                  break;
                case 4://intenger or number
```

```
                    int n = query.getInt(rsmd.getColumnName(i));
                    String num = "\t:" + rsmd.getColumnName(i) + " " + n + " " + end;
                    //  System.out.println(n);
                    vec.addElement(num);
                    break;
                  case 2006://Ref
                    Object ref = query.getObject(rsmd.getColumnName(i));
                    vec.addElement("\trel:" + rsmd.getColumnName(i) + " <" +
rsmd.getColumnTypeName(i) + ":" + ref + ">" + end);
                    break;

                } (5)
}
```

	TINYINT	SMALLINT	INTEGER	BIGINT	REAL	FLOAT	DOUBLE	DECIMAL	NUMERIC	BIT	CHAR	VARCHAR	LONGVARCHAR	BINARY	VARBINARY	LONGVARBINARY	DATE	TIME	TIMESTAMP	CLOB	BLOB	ARRAY	REF	STRUCT	JAVA OBJECT
getByte	X	x	x	x	x	x	x	x	x	x	x	x	x												
getShort	x	X	x	x	x	x	x	x	x	x	x	x	x												
getInt	x	x	X	x	x	x	x	x	x	x	x	x	x												
getLong	x	x	x	X	x	x	x	x	x	x	x	x	x												
getFloat	x	x	x	x	X	x	x	x	x	x	x	x	x												
getDouble	x	x	x	x	x	X	X	x	x	x	x	x	x												
getBigDecimal	x	x	x	x	x	x	x	X	X	x	x	x	x												
getBoolean	x	x	x	x	x	x	x	x	x	X	x	x	x												
getString	x	x	x	x	x	x	x	x	x	x	X	X	x	x	x	x	x	x	x						
getBytes														X	X	x									
getDate											x	x	x				X		x						
getTime											x	x	x					X	x						
getTimestamp											x	x	x				x	x	X						
getAsciiStream											x	x	X	x	x	x									
getBinaryStream														x	x	X									
getCharacterStream											x	x	X	x	x	x									
getClob																				X					
getBlob																					X				
getArray																						X			
getRef																							X		
getObject	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	X	X

Σχήμα 2 – Πίνακας με τύπους και μεθόδους προσπέλασης τους

Μερικές από τις πιο σημαντικές μεθόδους των ResultSetMetaData, όπου κάποιες από αυτές χρησιμοποιηθήκαν παρουσιάζονται παρακάτω

getColumnCount()

Η μέθοδος getColumnCount() επιστρέφει το πλήθος των στηλών ενός πίνακα.

getColumnName(int column)

Η μέθοδος getColumnName(int column) επιστρέφει το όνομα της στήλης.

getColumnType(int column)

Η μέθοδος getColumnType(int column) επιστρέφει τον τύπο της στήλης.

getTableName(int column)

Η μέθοδος getTableName(int column) επιστρέφει το όνομα του πίνακα.

3.5.3 ParameterMetaData

Η ParameterMetaData χρησιμοποιείται για να πάρει πληροφορίες σχετικά με τους τύπους και τις ιδιότητες για κάθε παράμετρο δείκτη σε ένα αντικείμενο PreparedStatement. Ωστόσο κάποια ερωτήματα θα πρέπει να έχουν εκτελεστεί έτσι ώστε να μπορεί να επιστρέψει αποτελέσματα και να έχει πρόσβαση στις πληροφορίες που θέλει.

Μερικές από τις πιο σημαντικές μεθόδους παρουσιάζονται παρακάτω

getParameterCount()

Η μέθοδος getParameterCount() επιστρέφει το αριθμό των παραμέτρων του PreparedStatement στο οποίο αναφέρεται.

getParameterType(int param)

Η μέθοδος getParameterType(int param) επιστρέφει τον τύπο της παραμέτρου.

getParameterTypeName(int param)

Η μέθοδος getParameterTypeName(int param) επιστρέφει το όνομα της παραμέτρου.

3.6 Σχέσεις Κληρονομικότητας

Σε μια αντικειμενο-σχεσιακή βάση δεδομένων μπορούμε να έχουμε κληρονομικότητα όπως είδαμε παραπάνω. Πως όμως θα καταφέρουμε να βρούμε πληροφορίες που έχουν να κάνουν με την κληρονομικότητα στην βάση μας. Η JDBC τεχνολογία μαζί με την βοήθεια ενός SQL SELECT ερωτήματος μας δίνει την λύση και μας προσφέρει τις πληροφορίες που θέλουμε και μας ενδιαφέρουν. Θα δούμε λοιπόν ποιοι είναι οι υποτύποι και οι υπερτύποι στην βάση μας ως εξής.

1) Θα εκτελέσουμε ένα SQL SELECT ερώτημα που μας δίνει τις πληροφορίες που θέλουμε. Το ερώτημα θα απευθύνεται στον πίνακα SYSIBM.SYShierarchies που έχει τις πληροφορίες αυτές και θα ψάξουμε για το σχήμα που θέλει ο χρήστης (schemaname)
2) Θα εμφανίσουμε τις πληροφορίες αυτές με την βοήθεια ενός ResultSet

```
ResultSet query = statement.executeQuery("Select SUB_NAME as sname,
SUPER_NAME as suname from SYSIBM.SYShierarchies WHERE ROOT_SCHEMA ='" +
schemaname + "'");   (1)

        String result = null;
        while (query.next()) {
            result = query.getString("sname") + "    INHERITS   " +
query.getString("suname");

            classes.addElement(result);  (2)
        }
```

Με αυτόν τον τρόπο βρίσκουμε τους υποτύπους και υπερτύπους στην βάση μας.

ΕΠΙΛΟΓΟΣ

Η JDBC τεχνολογία μας προσφέρει πολλές δυνατότητες οι οποίες μας δίνουν την ευκαιρία, να διαχειριστούμε και να ανακτήσουμε τα δεδομένα και τα μεταδεδομένα από μια βάση δεδομένων. Είδαμε μερικές από αυτές τις δυνατότητες οι οποίες χρησιμοποιηθήκαν στην παρούσα πτυχιακή με την χρήση δειγμάτων κώδικα μέσα από την εφαρμογή.

Μέχρι στιγμής έχουμε δημιουργήσει μία Αντικειμενο-σχεσιακή Βάση Δεδομένων, έχουμε πρόσβαση σε αυτήν, μέσο της java και της JDBC τεχνολογίας.

Στο επόμενο κεφάλαιο θα κάνουμε μια εισαγωγή στον Σημασιολογικό Ιστό και τα RDF πρότυπα και θα δούμε την RDF Turtle μορφή που θα έχουν τα δεδομένα μας.

ΚΕΦΑΛΑΙΟ 4 – Σημασιολογικός Ιστός και RDF πρότυπα

ΕΙΣΑΓΩΓΗ

Καθώς το διαδίκτυο μεγαλώνει και επεκτείνεται γεννιούνται καθημερινά νέες τεχνολογίες και εφευρίσκονται νέοι τρόποι παρουσίασης και επεξεργασίας δεδομένων. Ο Σημασιολογικός Ιστός (Semantic Web) είναι μία επέκταση του ήδη υπάρχοντα ιστού ο οποίος βασίζει την λειτουργία του στην σημασία των δεδομένων.

Σε αυτό το κεφάλαιο θα παρουσιάσουμε τι είναι ο Σημασιολογικός Ιστός, τα Συνδεδεμένα Δεδομένα, τι είναι τα IRIs, URIs και Literals καθώς επίσης και τα RDF, RDFS και RDF Turtle πρότυπα έπειτα θα δούμε και την μορφή που έχουν τα βασικά στοιχεία του σχεσιακού και αντικειμενο-σχεσιακού μοντέλου σε μορφή Turtle. Τέλος θα δούμε τον αλγόριθμο μετατροπής των δεδομένων που δημιουργήσαμε με την μορφή ψευδοκώδικα.

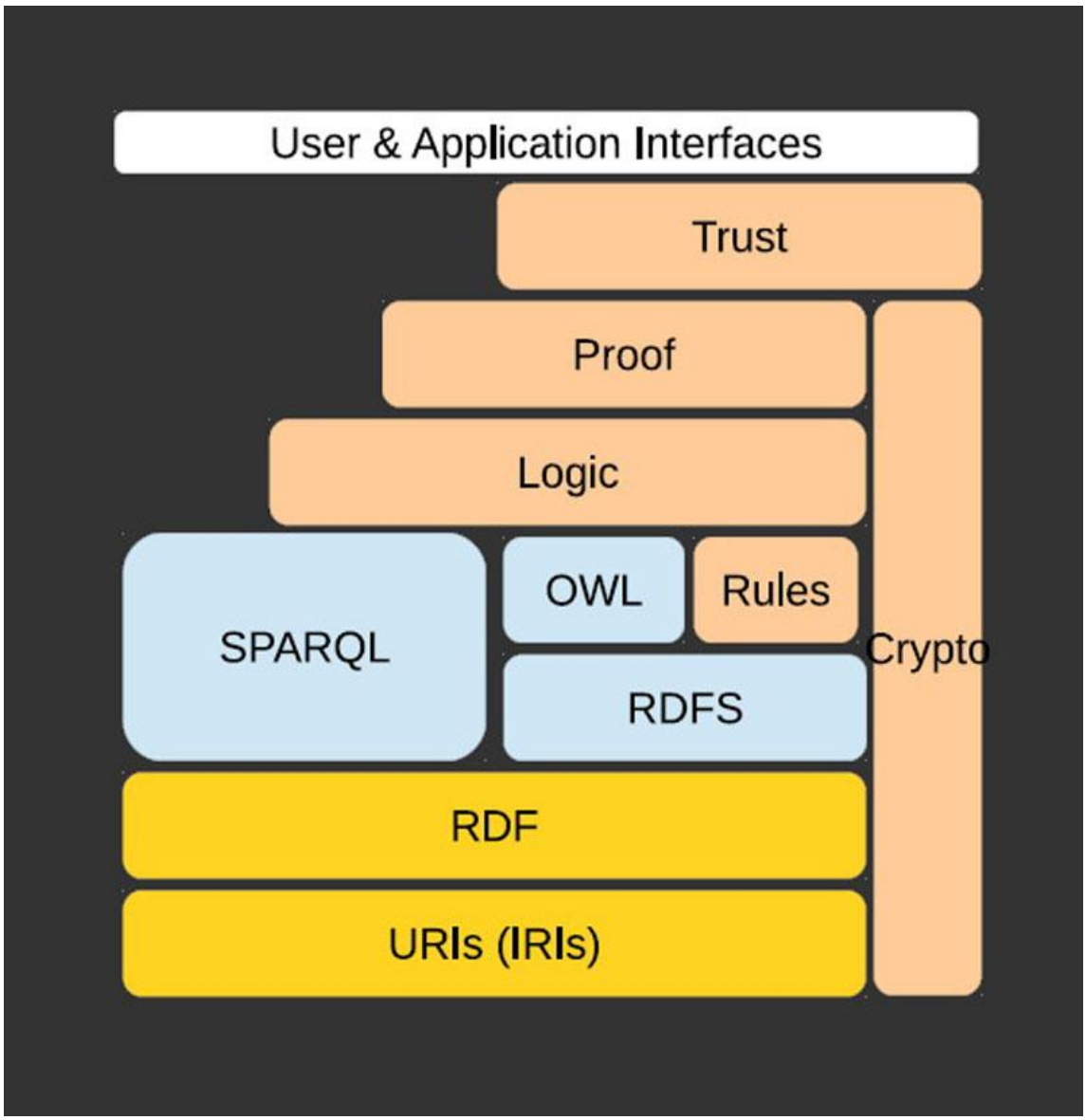
4.1 Σημασιολογικός Ιστός

Ο Σημασιολογικός Ιστός είναι «ένα πλέγμα διασυνδεδεμένων δεδομένων». Τα δεδομένα είναι δομημένα με κοινή μορφή και συνδέονται, έτσι ώστε να μπορούν να υποβάλλονται σε επεξεργασία άμεσα και έμμεσα από μηχανές. Τα δεδομένα μπορούν να διαμοιρασθούν και να επαναχρησιμοποιηθούν από διαφορετικές εφαρμογές, οργανισμούς και κοινότητες.

Οι άνθρωποι μπορούν να δημιουργήσουν αποθήκες δεδομένων στο Σημασιολογικό Ιστό, να φτιάξουν λεξιλόγια, και να γράψουν κανόνες για τη διαχείριση τους. Εφαρμογές της γενικής ιδέας του Σημασιολογικού Ιστού στην έρευνα της βιομηχανίας, της βιολογίας και των επιστημών του ανθρώπου έχουν ήδη αποδείξει την εγκυρότητα και τη χρησιμότητά της.

Ο Σημασιολογικός Ιστός βασίζεται σε τεχνολογίες που ήδη υπάρχουν (URI και XML) αλλά και σε νέες τεχνολογίες (RDF, RDFS, OWL, κα.), οι οποίες αναπτύσσονται με την βοήθεια της κοινότητας. Το κίνημα του Σημασιολογικού Ιστού καθοδηγείται από το διεθνή οργανισμό τυποποίησης World Wide Web Consortium (W3C).

Ο Σημασιολογικός Ιστός διαθέτει κάποια επίπεδα που φαίνονται στο παρακάτω σχήμα (σχ.3) και αναλύονται κάποια από αυτά στην συνέχεια του κεφαλαίου.



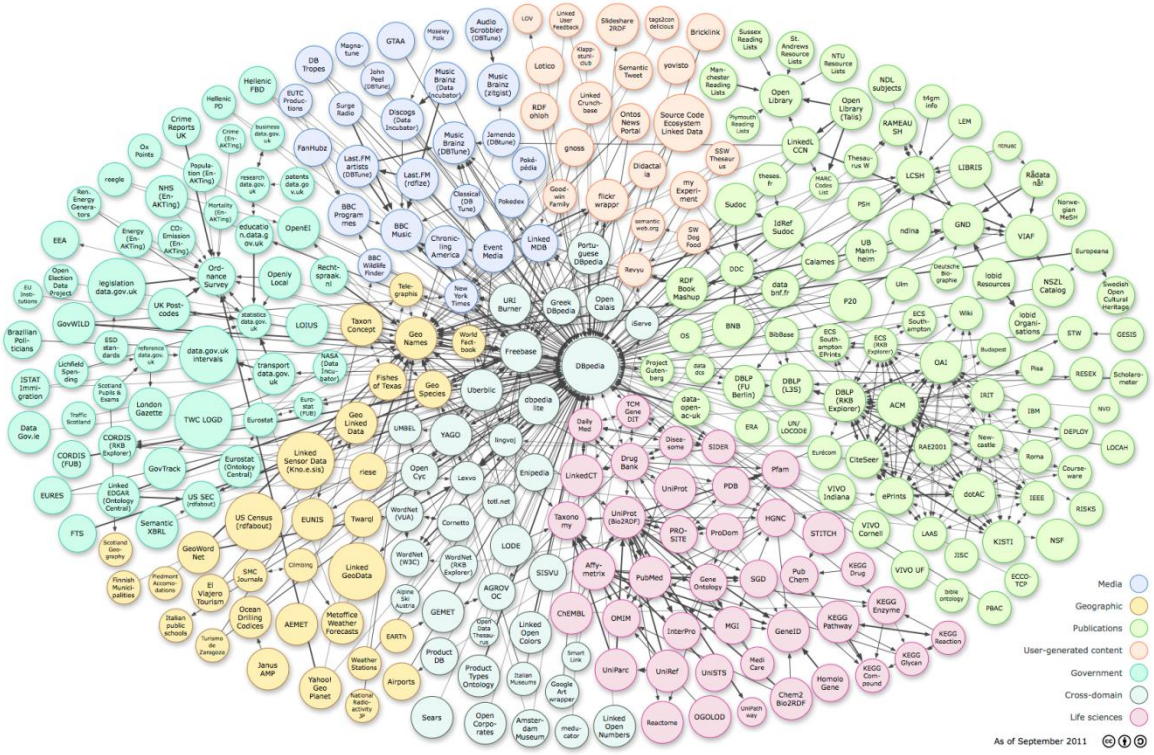
Σχήμα 3 - Επίπεδα Σημασιολογικού Ιστού

4.2 Συνδεδεμένα Δεδομένα (Linked Data)

Τα Συνδεδεμένα Δεδομένα είναι μια μεθοδολογία συσχέτισης πραγμάτων (δομημένων δεδομένων, εννοιών και εγγράφων) στον παγκόσμιο ιστό (web), έτσι ώστε αυτά να μπορούν να συνδέονται μεταξύ τους και να γίνουν πιο χρήσιμα. Αυτά τα πράγματα και οι σχέσεις τους προσδιορίζονται (ονομάζονται) με Uniform Resource Identifiers (URIs), περιγράφονται από το Resource Description Framework (RDF) και δημοσιεύονται (γίνονται προσβάσιμα) μέσω του Hypertext Transfer (or Transport) Protocol (HTTP), κατά τέτοιο τρόπο ώστε να μπορούν να ερμηνεύονται και να χρησιμοποιούνται από ανθρώπους αλλά και από υπολογιστές.

Παρακάτω βλέπουμε τις τέσσερις αρχές των Linked Data οι οποίες παρουσιάστηκαν από τον Tim Berners-Lee το 2006:

1. Χρησιμοποιήστε URIs ως ονόματα για τα πράγματα
2. Χρησιμοποιήστε HTTP URIs έτσι ώστε οι άνθρωποι να μπορούν να αναζητούν αυτά τα ονόματα.
3. Όταν κάποιος αναζητά ένα URI, παράσχετε χρήσιμες πληροφορίες, χρησιμοποιώντας τα πρότυπα (RDF, SPARQL)
4. Συμπεριλάβετε συνδέσμους προς άλλα URIs, έτσι ώστε να μπορούν να ανακαλύψουν περισσότερα πράγματα



Σχήμα 4 – Συνδεδεμένα Δεδομένα

4.3 URI (Universal Resource Identifier)

Το **URI (Universal Resource Identifier**) μια γενική μορφή αναγνώρισης των αρχείων, των πόρων, των δεδομένων και των πληροφοριών που συνδέονται με τον ιστό.

Το URI χρησιμοποιείται από το RDF ως μοναδικό αναγνωριστικό για το υποκείμενο (subject), την ιδιότητα (property ή predicate) και το αντικειμένο (object) μιας RDF πρότασης (statement). Με τη χρήση των URI επιτυγχάνεται μία μοναδική παγκόσμια ονομασία για τα δεδομένα και λύνει το πρόβλημα της «ομωνυμίας» στην αναπαράσταση των κατανεμημένων δεδομένων.

Το URI μπορεί να έχουν τις εξής μορφές:

- Ένα URL όπως για παράδειγμα http://www.it.teithe.gr/~sw
- Ή ένα μοναδικό αναγνωριστικό όπως για παράδειγμα #SemanticWeb

4.4 IRI (Internationalized Resource Identifier)

Τα IRIs είναι μια γενίκευση των URIs η οποία επιτρέπει την χρήση χαρακτήρων Unicode. Κάθε απόλυτο URI και η διεύθυνση URL είναι ένα IRI, ωστόσο κάθε IRI δεν είναι ένα URI. Όταν τα IRIs χρησιμοποιηθούν σε πράξεις που ορίζονται μόνο για URIs, θα πρέπει πρώτα να μετατραπούν σύμφωνα με την χαρτογράφηση [RFC3987].

4.5 Literal

Τα Literal είναι είτε συμβολοσειρές, είτε αριθμοί είτε ημερομηνίες που γράφονται σε περίπτωση που δεν έχουμε κάποιο πόρο, ή όταν ο πόρος δεν έχει νόημα να χρησιμοποιηθεί στη συγκεκριμένη περίπτωση. Παραδείγματος χάρη η τιμή της ηλικίας κάποιου (που είναι πάντα ένας απλός ακέραιος αριθμός) ή όταν περιγράφουμε ένα αντικείμενο (το κείμενο θα είναι διαφορετικό κάθε φορά). Τα literal όπως και οι πόροι χρησιμοποιούνται για να δώσουν τιμή σε κάποια ιδιότητα ενός υποκείμενου (subject) αλλά δεν προσφέρουν περαιτέρω πληροφορίες για τον εαυτό τους, παρά μόνο ότι μπορούν να καταλάβουν οι χρήστες.

4.6 RDF (Resource Description Framework) πρότυπο

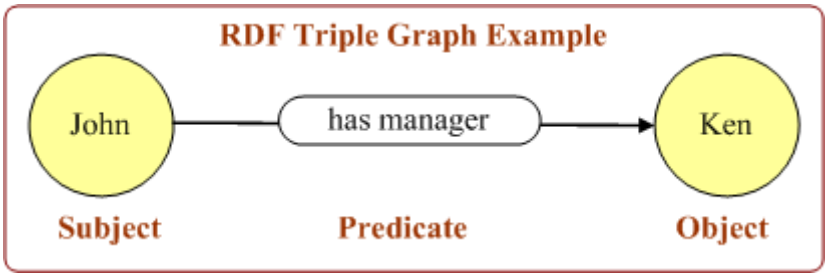
RDF (Resource Description Framework) : Είναι ένα πρότυπο W3C, στο πρώτο επίπεδο του Σημασιολογικού Ιστού το οποίο δημιουργήθηκε αρχικά σαν ένα μοντέλο μετα-δεδομένων για να “αποθηκεύει” τη σημασία της πληροφορίας. Είναι ένα μοντέλο το οποίο μας βοηθάει να μοντελοποιήσουμε σε μορφή γράφου δεδομένα και τις συσχετίσεις τους.

Το RDF βασίζεται στην ιδέα ότι οι πόροι οι οποίοι πρέπει να περιγραφούν έχουν ιδιότητες (properties) οι οποίες έχουν συγκεκριμένη τιμή. Έτσι λοιπόν μια μετα-πληροφορία για έναν πόρο ο οποίος αποτελείται από μία ιδιότητα και την τιμή που έχει ο πόρος για την ιδιότητα αυτή. Μια έκφραση για έναν πόρο ονομάζεται RDF πρόταση(statement) και αποτελείται από τρία μέρη *(triple)* ενός υποκειμένου (subject), μιας ιδιότητας (property ή predicate) και ενός αντικειμένου

(object). Τη θέση του υποκειμένου καταλαμβάνει ο πόρος, τη θέση της ιδιότητας η ιδιότητα που του αποδίδουμε, ενώ τη θέση του αντικειμένου η τιμή που έχει ο πόρος αυτός για την ιδιότητα. Η τιμή αυτή μπορεί να είναι κάποιος άλλος πόρος ή κάποια τιμή δεδομένων. Επίσης το υποκείμενο και η ιδιότητα πρέπει να είναι URIs ενώ το αντικείμενο μπορεί να είναι είτε URI είτε Literal.

Σχήμα 5 – RDF παράσταση ως τριάδα.

Ένα σύνολο από τριάδες RDF μπορούμε να το αντιληφθούμε και ως έναν γράφο. Σε αυτό το γράφο τα αντικείμενα και τα υποκείμενα παίζουν το ρόλο των κόμβων ενώ οι ιδιότητες παίζουν το ρόλο των ακμών που τους συνδέουν(σχ.6).



Σχήμα 6 – RDF τριάδα σε σχήμα γράφου.

4.7 RDF Schema(RDFS)

Το RDF Schema ή RDFS είναι μια γλώσσα που μας επιτρέπει να περιγράφουμε resources χρησιμοποιώντας το δικό μας λεξιλόγιο. Κυρίως σκοπός του δεν είναι να παρέχει προκαθορισμένες κλάσεις και ιδιότητες, αλλά να προσφέρει τα μέσα για να τις φτιάξουμε, καθώς επίσης μας επιτρέπει να δείξουμε ποιες κλάσεις και ιδιότητες αναμένεται να χρησιμοποιούνται μαζί. Το RDFS χρησιμοποιεί τη σύνταξη RDF, βασίζεται στο RDF vocabulary και το επεκτείνει σημασιολογικά.

Η διαφορά του RDF με το RDFS είναι ότι στο RDF μιλάμε για οντότητες/αντικείμενα, ενώ στο RDFS μιλάμε για κλάσεις και συσχετίσεις μεταξύ

των ιδιοτήτων (properties). Οι RDFS τάξεις (classes) και ιδιότητες (properties) ορίζονται με παρόμοιο τρόπο με τις αντικειμενοστρεφείς γλώσσες προγραμματισμού με την διαφορά ότι στα αντικειμενοστρεφή συστήματα οι τάξεις ορίζονται με βάση τις ιδιότητες που θα έχουν τα στιγμιότυπα τους, ενώ στο RDFS οι τάξεις περιγράφουν τις ιδιότητες τους με βάση τις τάξεις πόρους που θα εφαρμοστούν. Αυτό υλοποιείται με χρήση των πεδίων ορισμού (domain) και εμβέλειας (range).

Ας πάρουμε για παράδειγμα την τάξη Βιβλίο με ιδιότητα Συγγραφέας τύπου Άτομο.

- Στον αντικειμενοστρεφή προγραμματισμό: Ορίζεται η τάξη Βιβλίο με ιδιότητα Συγγραφέας τύπου Άτομο
- Ενώ στο RDFS: ορίζεται η ιδιότητα Συγγραφέας με domain Έγγραφο (Βιβλίο) και range Άτομο.

Με την RDF προσέγγιση μπορεί μεταγενέστερα να οριστεί νέα ιδιότητα με domain το Έγγραφο (βιβλίο) και range το Άτομο χωρίς καμία αλλαγή στην περιγραφή των κλάσεων.

Επίσης το RDF Schema προσφέρει την ιδιότητα της κληρονομικότητας. Αυτό το κάνει με την ιδιότητα “subClassOf” για να δείξει ότι ένα resource είναι υποτάξη κάποιου άλλου αντικειμένου και παρόμοια με το “subPropertyOf” για τα predicates. Αυτό μοιάζει πάρα πολύ με το αντικειμενοστραφή προγραμματισμό όπου δηλώνουμε τάξεις οι οποίες μπορούν να κληρονομηθούν από άλλες τάξεις.

Παρακάτω παρουσιάζονται μερικές από τις κυριότερες ιδιότητες του RDFS

1. rdfs:Resource. Είναι η κλάση για όλα τα resources.
2. rdfs:Class. Είναι η κλάση όλων των κλάσεων.
3. rdfs:Literal. Είναι η κλάση για όλα τα literals.
4. rdf:Property. Είναι η κλάση για όλες τις ιδιότητες.
5. rdf:Statement. Είναι η κλάση για όλα τα statement.
6. rdf:type. Συνδέει κάθε resource με την κλάση του.
7. rdfs:subClassOf. Συνδέει την κλάση στην οποία δηλώνεται με μια από τις υπερ-κλάσεις του.

8. rdfs:subPropertyOf. Συνδέει την ιδιότητα στην οποία δηλώνεται με μια υπερ-ιδιότητά του.

9. rdfs:domain. Καθορίζει την κλάση όλων των resource που μπορούν να είναι subject ενός triple.

10. rdfs:range. Καθορίζει την κλάση όλων των resource που μπορούν να είναι object ενός triple.

11. rdfs:ConstraintResource.Είναι η κλάση όλον των περιορισμών.

12. rdfs:ConstraintProperty. Είναι υποκλαση του ConstraintResource και Property έχει δυο στιγμιότυπα το domain και range.

4.8 RDF Turtle(Terse RDF Triple Language)

Η Turtle (Terse RDF Triple Language) είναι ένα πρότυπο μορφοποίησης και παρουσίασης δεδομένων σε μορφή απλού κειμένου η οποία χρησιμοποιείται για την αναπαράσταση RDF γράφων. Ένα RDF γράφημα αποτελείται από μία τριάδα (triple) ενός υποκειμένου (subject), μιας ιδιότητας (property ή predicate) και ενός αντικειμένου (object). Με το πρότυπο Turtle μπορούμε να αναπαραστήσουμε μια συλλογή από τέτοιες RDF τριάδες οι οποίες έχουν σχέση μεταξύ τους, προσφέροντας έτσι τρόπους συντόμευσης των προτάσεων όταν υπάρχουν πολλαπλές ιδιότητες για το ίδιο υποκείμενο.

Κάθε δήλωση τελειώνει με μια περίοδο, και κάθε στοιχείο της τριάδας είναι ένα URI εκτός από το αντικείμενο, το οποίο μπορεί να είναι ένα κομμάτι κειμένου όπως ένα "string" ή ένας αριθμός της μορφής 1234.

4.9 Κανόνες σύνταξης Turtle

Το πρότυπο Turtle έχει κάποιους κανόνες σύνταξης οι οποίοι αναλύονται σε αυτό το κεφάλαιο.

Μια πρόταση σε Turtle

Η πιο απλή triple πρόταση (statement) στην Turtle είναι μια ακολουθία από υποκείμενο - ιδιότητα - αντικείμενο (subject, predicate, object) τα οποία χωρίζονται μεταξύ τους με κενό και στο τέλος κάθε τέτοιας τριάδας τοποθετείται τελεία.

Για παράδειγμα

```
<http://e.gr/#sp> <http://www.e.gr/relationship/enemyOf>
<http://e.gr/#goblin> .
```

Πολλές Ιδιότητες

Επειδή πολύ συχνά πολλές ιδιότητες(predicate) αναφέρονται στο ίδιο υποκείμενο(subject) μπορούμε να τις συντάξουμε σε μία πρόταση με την χρήση του ερωτηματικού **;** στο τέλος κάθε ιδιότητας και στην τελευταία βάζουμε την τελεία.

Πολλά Αντικείμενα

Επειδή πολύ συχνά πολλά αντικείμενα αναφέρονται στην ίδια ιδιότητα και στο ίδιο υποκείμενο μπορούμε να γράψουμε τα αντικείμενα αυτά σε μια πρόταση με την χρήση του συμβόλου της κόμμα **,**

Για παράδειγμα
```
<http://e.org/#spi> <http://xmlns.com/foaf/0.1/name> "Spider", "man".
```

IRIs

Τα IRIs μπορούν να γραφούν σχετικά ή απόλυτα ή σαν προκαθορισμένα ονόματα (prefix). Τα σχετικά και τα απόλυτα IRIs περικλείονται από τα σύμβολα του μικρότερου και μεγαλύτερου **<>** και μπορούν να περιέχουν αριθμητικές ακολουθίες.Για να δηλώσουμε ένα νέο IRI το κάνουμε χρησιμοποιώντας το @base.

Ένα παράδειγμα απόλυτου IRI είναι κάπως έτσι

```
<http://example.org/#green-goblin>.
```

Ενώ ένα σχετικό κάπως έτσι `<#green-goblin>`

Προκαθορισμένα ονόματα (prefix)

Τα προκαθορισμένα ονόματα (prefix) διαχωρίζονται με την χρήση του συμβόλου της άνω κάτω τελείας **:** . Ένα prefix δηλώνεται με την χρήση του @prefix όνομα άνω κάτω τελεία < IRI ή URI > και τελεία.

Literals

Τα Literals στην Turtle μπορούν να γραφούν με διάφορους τρόπους ανάλογα την κατηγορία που ανήκουν. Έχουμε τρεις κατηγορίες

- Quoted Literals
- Numbers Literals
- Booleans Literals

Quoted Literals

Τα Quoted Literals έχουν λεξιλογική μορφή ακολουθούνται από μία ετικέτα που δηλώνει την γλώσσα ή μια ετικέτα που δηλώνει τον τύπο δεδομένων της ή χωρίς κάποιο από τα δύο. Quoted Literals είναι τύποι δεδομένων string, time, date.

1. "That Seventies Show"^^xsd:string .
2. "That Seventies how"^^<http://www.w3.org/2001/XMLSchema#string>
3. "That Seventies Show" .

Και τα τρία παραδείγματα γίνονται αποδεκτά από την Turtle και έχουν την ίδια σημασία.

Numbers Literals

Τα Numbers Literals είναι οι αριθμητικές τιμές αριθμητικών τύπων όπως integer,float, double κτλ. Η Turtle μας δίνει δυνατότητα να γράψουμε τις τιμές αυτές χωρίς να βάλουμε τον τύπο (`^^xsd:`) από δίπλα. Σε περίπτωση που γράψουμε τον τύπο τότε η τιμή μπαίνει σε διπλά εισαγωγικά αλλιώς δεν μπαίνει.

1. ”-5”^^xsd:integer
2. -5

Booleans Literals

Τα Booleans Literals είναι οι λογικές τιμές true και false και η Turtle μας δίνει την ίδια δυνατότητα με τα Numbers Literals όπου ισχύουν τα ίδια πράματα.

Literal πολλών γραμμών

Όταν ένα Literal έχει παραπάνω από μία γραμμές τότε μπαίνει σε τριπλά εισαγωγικά.

```
"""The first line
The second line
  more""" .
```

Το γράμμα a αντί του rdf:type.

Το γράμμα a χρησιμοποιείται από την Turtle για να αντικαταστήσει το IRI http://www.w3.org/1999/02/22-rdf-syntax-ns#type . το οποίο δηλώνει τον τύπο (rdf:type).

Σχόλια

Τα σχόλια ξεκινούν με το σύμβολο # και ένα κενό και συνεχίζουν ως το τέλος της γραμμής.

Ανώνυμα αντικείμενα (objects)

Όταν δεν θέλουμε να ορίσουμε μια διεύθυνση URL για κάποιο αντικείμενο, τότε μπορούμε να χρησιμοποιήσουμε ένα ανώνυμο αντικείμενο στην περιοχή του αντικειμένου. Τα ανώνυμα αντικείμενα ορίζονται με την χρήση των συμβόλων [ ] και μέσα σε αυτά μπαίνουν οι σχέσεις οι οποίες διαχωρίζονται με ερωτηματικά ; .

Συλλογές

Μια συλλογή στην Turtle γράφεται ανάμεσα σε παρενθέσεις και οι τιμές μέσα στις παρενθέσεις χωρίζονται με το κενό.

Empty prefix

Ένα empty prefix γράφεται @prefix : <uri>. Και μπορούμε να αναφερθούμε σε αυτό χρησιμοποιώντας την άνω κάτω τελεία μόνο.

4.10 Αποτελέσμα σε RDF Turtle μορφή

Το παρακάτω απόσπασμα είναι ένα δείγμα από τα αποτελέσματα σε μορφή RDF Turtle.

@prefix rel : <http://it.teithe.gr/Rdf/Turtle/relations/>.

@prefix : <http://it.teithe.gr/Rdf/Turtle/elements/>.

@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 sc: < http://it.teithe.gr/Rdf/Turtle/schema/>.

```
@prefix t: <http:// it.teithe.gr/Rdf/Turtle /triggers/>.
@prefix DROMOLOGIO : <http://it.teithe.gr/Rdf/Turtle/DROMOLOGIO/>.
@prefix LEOFOREIO : <http://it.teithe.gr/Rdf/Turtle/LEOFOREIO/>.
DROMOLOGIO:1
        rel:AFETIRIA POLI:10;
        rel:PROORISMOS POLI:20;
        :HMERA "Deutera" ;
        :WRA "10:00:00";
        rel:LEWFOREIO LEOFOREIO:10;
        :TIMH_EISHTHRIOU 50.0 .
DROMOLOGIO:2
        rel:AFETIRIA POLI:10;
        rel:PROORISMOS POLI:20;
        :HMERA "Triti" ;
        :WRA "10:00:00";
        rel:LEWFOREIO LEOFOREIO:10;
        :TIMH_EISHTHRIOU 50.0 .
DROMOLOGIO:3
        rel:AFETIRIA POLI:10;
        rel:PROORISMOS POLI:20;
        :HMERA "Tetarti" ;
        :WRA "10:00:00";
        rel:LEWFOREIO LEOFOREIO:10;
        :TIMH_EISHTHRIOU 50.0 .
LEOFOREIO:10
        :ARITHMOS_KUKLOFORIAS "NHE1010" ;
        rel:EDRA POLI:10;
```

```
        :HMEROMHNIA_KATASKEUHS "2007-05-01";
        rel:PLIROMA PLIROMA:1;
        :ARI8MOS_XILIOMETRWN 10000 .
LEOFOREIO:20
        :ARITHMOS_KUKLOFORIAS "NEE9990" ;
        rel:EDRA POLI:10;
        :HMEROMHNIA_KATASKEUHS "2006-06-01";
        rel:PLIROMA PLIROMA:2;
        :ARI8MOS_XILIOMETRWN 50000 .

sc:AFETIRIA
         a rdfs:Property;
         rdfs:domain sc:DROMOLOGIO;
         rdfs:range sc:POLI.

sc:YPALLILOS a rdfs:Class.
sc:ODIGOS a rdfs:Class.
sc:ODIGOS rdfs:subClassOf sc:YPALLILOS.
```

Επεξήγηση

Το prefix rel αναφέρεται στην σχέσεις ανάμεσα στους πίνακες (τις αναφορές).

Το prefix : είναι το empty prefix και αναφέρεται στις στήλες του κάθε πίνακα.

Το prefix rdf αναφέρεται στην γλώσσα που είναι γραμμένο το αρχείο.

Το prefix rdfs αναφέρεται στο πρότυπο RDF Schema.

Το prefix xsd: αναφέρεται στον XML τύπο.

Το prefix sc αναφέρεται στο σχήμα της βάσης.

Το prefix t αναφέρεται στα εναύσματα.

Το DROMOLOGIO:1 αναφέρεται στο URI <http://it.teithe.gr/Rdf/Turtle/DROMOLOGIO/#1>. Και τα υπόλοιπα αντίστοιχα.

Τα υπόλοιπα είναι τα πεδία και οι τιμές κάθε πίνακα.

Το sc:AFETIRIA αναφέρεται στην στήλη AFETIRIA και μας δείχνει σε ποιους πίνακες αναφέρεται.

Το sc:YPALLILOS a rdfs:Class. αναφέρεται στον πίνακα YPALLILOS και μας δείχνει ότι είναι κλάση.

Το sc:ODIGOS rdfs:subClassOf sc:YPALLILOS αναφέρεται στον πίνακα ODIGOS και μας δείχνει ότι είναι υπόκλαση της κλάσης YPALLILOS.

Ένα πλήρες αρχείο Turtle υπάρχει στα παραρτήματα όπου φαίνονται επίσης και οι εντολές δημιουργίας του σχήματος που φαίνεται στο Turtle αρχείο.

4.11 Σχεσιακό μοντέλο σε RDF Turtle μορφή

Μέχρι στιγμής υπάρχουν αρκετοί τρόποι και διάφορα παραδείγματα στο διαδίκτυο, αναφορικά με την μετατροπή των εγγραφών από απλές σχεσιακές βάσεις δεδομένων σε μορφή RDF Turtle. Σε αυτό το υποκεφάλαιο θα παρουσιάσουμε την μορφή των εγγράφων από μια σχεσιακή βάση σε Turtle μορφή σύμφωνα με τον αλγόριθμο R2RML [16] που δημιουργήθηκε από World Wide Web Consortium (W3C) .

Για αρχή θα παρουσιάσουμε ένα παράδειγμα το οποίο θα αποτελείται από δύο πίνακες οι οποίοι θα έχουν κάποιες στήλες και κάποια δεδομένα.

Έστω ότι έχουμε τον πίνακα εργαζόμενος και τον πίνακα τμήμα που φαίνονται παρακάτω.

Πίνακας Εργαζόμενος			
Κωδικός(κύριο κλειδί) integer	Όνομα - varchar	Δουλειά-varchar	Κωδικός Τμήματος(ξένο κλειδί)integer
12345	Νίκος	Προϊστάμενος	10

Πίνακας Τμήμα		
Κωδικός Τμήματος(κύριο κλειδί)-integer	Όνομα τμήματος- varchar	Τοποθεσία - varchar
10	Λογιστήριο	Θεσσαλονίκη

Έστω ότι έχουμε τα εξής URI για τους πίνακες μας
<http://data.example.com/εργαζόμενος> για τον εργαζόμενο
<http://data.example.com/τμήμα> για το τμήμα

Μια απλή RDF Turtle πρόταση

<http://data.example.com/εργαζόμενος/12345> ex:όνομα “Νίκος”.

Με αυτήν την πρόταση παρουσιάσουμε την στήλη όνομα για τον πίνακα εργαζόμενος για τον εργαζόμενο με κωδικό 12345 που έχει τιμή Νίκος.

Χρήση Προκαθορισμένων ονομάτων (prefix)

Έστω ότι δημιουργούμε τα εξής προκαθορισμένα ονόματα για τα εξής URI

@prefix emp : <http://data.example.com/εργαζόμενος/>.

@prefix dep : <http://data.example.com/τμήμα/>.

@prefix ex: <http://data.example.com/elements/>.

Με αυτόν τον τρόπο μπορούμε να αναφερόμαστε σε αυτά τα URI με πιο σύντομο όνομα.

Άρα η προηγούμενη πρόταση μπορεί να γίνει κάπως έτσι και να έχει το ίδιο νόημα

emp:12345 ex:όνομα “Νίκος”.

Το prefix ex αναφέρεται στις στήλες του κάθε πίνακα.

Στήλες

Οι στήλες κάθε πίνακα στην Turtle αποτελούν την ιδιότητα σε μια τριάδα και παρουσιάζονται με την βοήθεια κάποιου προκαθορισμένου ονόματος x άνω κάτω τελεία και το όνομα της στήλης.

emp:12345 x:Όνομα “Νίκος”.

Με το x:Όνομα αναφερόμαστε στην στήλη Όνομα.

Κύριο Κλειδί

Το κύριο κλειδί χρησιμοποιείται για τον διαχωρισμό των οντοτήτων σε ένα πίνακα.

Στην Turtle την τιμή του κύριου κλειδιού την γράφουμε στο υποικείμενο της τριάδας είτε σε μορφή URI είτε σε μορφή prefix.

<http://data.example.com/εργαζόμενος/12345>

emp:12345

Ξένο Κλειδί

Το ξένο κλειδί δηλώνει την αναφορά ενός πίνακα σε έναν άλλο.

Στην Turtle την τιμή του ξένου κλειδιού την γράφουμε στο αντικείμενο της τριάδας είτε σε μορφή URI είτε σε μορφή prefix.

<http://data.example.com/τμήμα/10>.

dep:10

Πλήρες Turtle μορφή

Εδώ βλέπουμε την πλήρη Turtle μορφή που θα έχει ο πίνακας εργαζόμενος.

@prefix emp : <http://data.example.com/εργαζόμενος/>.

@prefix dep : <http://data.example.com/τμήμα/>.

@prefix ex: <http://data.example.com/elements/>.

emp:12345 ex:Όνομα “Νίκος”;

ex:Κωδικός Τμήματος dep:10.

Και Εδώ βλέπουμε την πλήρη Turtle μορφή που θα έχει ο πίνακας τμήμα.

dep:10 ex: Όνομα τμήματος “Λογιστήριο”;

ex:Τοποθεσία “Θεσσαλονίκη”.

Υπάρχουν επίσης και άλλοι τρόποι για την μετατροπή μιας σχεσιακής βάσης δεδομένων σε RDF μορφή όπως η Direct Mapping [22] η οποία δημιουργήθηκε

από την World Wide Web Consortium (W3C), καθώς επίσης υπάρχει και η D2RQ [5] πλατφόρμα η οποία δημιουργήθηκε από την Apache license και η οποία είναι λογισμικό ανοιχτού κώδικα.

4.12 Αντικείμενο-Σχεσιακό μοντέλο σε RDF Turtle μορφή

Παραπάνω είδαμε τους κανόνες σύνταξης της Turtle καθώς επίσης και το πώς μετατρέπονται τα στοιχεία μιας σχεσιακής βάσης δεδομένων σε Turtle μορφή. Σε αυτό το υποκεφάλαιο θα δούμε πως μια αντικειμενο-σχεσιακή βάση μετατρέπεται σε Turtle μορφή σύμφωνα με την εφαρμογή που έχουμε δημιουργήσει όπου με την βοήθεια της JDBC τεχνολογίας διαβάζουμε τις τιμές όλων των στηλών που υπάρχουν στο κάθε πίνακα για κάθε εγγραφή και τις αποθηκεύουμε σε ένα Vector σε κατάλληλη μορφή σύμφωνα με τους κανόνες του πρότυπου Turtle.

Ταυτότητα Αντικειμένου (OID)

Στην Turtle η ταυτότητα αντικειμένου γράφεται στο υποκείμενο της τριάδας είτε σε μορφή URI είτε σε μορφή prefix.

<http://localhost/DROMOLOGIO/1>
DROMOLOGIO:1

Επίσης γράφεται όταν έχουμε αναφορά σε άλλο πίνακα.

DROMOLOGIO:1 rel:AFETIRIA POLI:10.

Αναφορά

Στην Turtle η αναφορά γράφεται με το @prefix rel : <http://localhost/relations/>.

DROMOLOGIO:1 rel:AFETIRIA POLI:10 .

Ο πίνακας δρομολόγιο με τύπο αναφοράς στον πίνακα πόλη και πιο συγκεκριμένα το δρομολόγιο με ταυτότητα αντικειμένου 1 έχει αφετηρία την πόλη με ταυτότητα αντικειμένου 10.

Επίσης κάθε στήλη που είναι τύπος αναφοράς γράφεται σύμφωνα με το πρότυπο RDF Schema όπου μας δείχνει ποιους πίνακες συνδέει ως έξης.

sc:AFETIRIA

a rdfs:Property;

rdfs:domain sc:DROMOLOGIO;

rdfs:range sc:POLI.

Εδώ φαίνεται η στήλη AFETIRIA η οποία έχει domain το DROMOLOGIO και range την POLI πράμα που σημαίνει ότι είναι ο συνδετικός κρίκος ανάμεσα στους δύο πίνακες.

Στήλες

Οι στήλες κάθε πίνακα στην Turtle αποτελούν το χαρακτηριστικό σε μια τριάδα και παρουσιάζονται με την βοήθεια κάποιου προκαθορισμένου ονόματος x άνω κάτω τελεία και το όνομα της στήλης ή με την βοήθεια του ανώνυμου αντικείμενου όπου γράφουμε σκέτο άνω κάτω τελεία και το όνομα της στήλης.

Με προκαθορισμένο όνομα

@prefix χ: <http://localhost/elements/>.

DROMOLOGIO:1 χ:HMERA "Deutera" .

Ανώνυμο Αντικείμενο

@prefix : <http://localhost/elements/>.

DROMOLOGIO:1 :HMERA "Deutera" .

Κληρονομικότητα

Η κληρονομικότητα στο τελικό αρχείο θα γραφτεί με την βοήθεια το πρότυπου RSD Schema και την βοήθεια του rdfs:subClassOf.

Παραδείγματος χάρη η κλάση ODIGOS είναι υποκλάση της κλάσης YPALLILOS.

sc:YPALLILOS a rdfs:Class.

sc:ODIGOS a rdfs:Class.

sc:ODIGOS rdfs:subClassOf sc:YPALLILOS.

Εναύσματα

Στο τελικό αρχείο Turtle υπάρχουν πληροφορίες για τα εναύσματα που υπάρχουν στο συγκεκριμένο σχήμα της βάσης δεδομένων. Οι πληροφορίες αυτές τις

παίρνουμε με την εκτέλεση ενός SELECT ερωτήματος στον πίνακα SYSCAT.Triggers ο οποίος έχει πληροφορίες για τα εναύσματα. Παρακάτω φαίνεται σε μορφή Turtle κάποια από τα εναύσματα.

@prefix t: <http://localhost/triggers/>.

t:DROMOLOGIO1 t:Trigtime "After event";

t:Trigevent "Insert";

t:Valid "Valid".

t:LEOFOREIO1 t:Trigtime "After event";

t:Trigevent "Insert";

t:Valid "Valid".

4.13 Αλγόριθμος Μετατροπής Αντικειμενο-σχεσιακής βάση σε RDF Turtle

Σε αυτό το κεφάλαιο παρατίθεται ο αλγόριθμος σε μορφή ψευδοκώδικα μετατροπής των δεδομένων της βάσης δεδομένων σε RDF Turtle μορφή.

Μετατροπή δεδομένων πινάκων

```
Δεδομένα
rs,query : ResultSet;
rsmd : ResultSetMetaData ;
TableName,TableType : String;
jdbcType : int;
vec : Vector;

rs := πληροφορίες_πινάκων ;
ΟΣΟ ( rs έχει στοιχεία ) ΕΠΑΝΈΛΑΒΕ
        TableName := rs.όνομα πίνακα;
        TableType := rs.τύπος πίνακα;
        ΕΑΝ (TableType = TYPED TABLE) ΤΌΤΕ
                query := πάρε όλα τα στοιχεία για τον πίνακα TableName ;
                Rsmd := πάρε πληροφορίες από το query;
                ΌΣΟ (query έχει στοιχεία και δεν είναι άδειο)ΕΠΑΝΈΛΑΒΕ
                        ΓΙΑ (i = 1; έως αριθμός στηλών) ΕΠΑΝΈΛΑΒΕ
                                jdbcType = τύπος στήλης I ;
                                ΕΠΙΛΕΞΕ (jdbcType)
                                        ΠΕΡΙΠΤΩΣΗ -7 :
                                                Vec = τιμή στήλης σε μορφή Turtle ;
                                        ΠΕΡΙΠΤΩΣΗ -6 :
                                                Vec = τιμή στήλης σε μορφή Turtle ;
                                        …
```

```
                                        …
                                        ΠΕΡΙΠΤΩΣΗ n :
                                                Vec = τιμή στήλης σε μορφή Turtle ;
                                        ΤΕΛΟΣ ΕΠΙΛΟΓΩΝ


                                ΤΕΛΟΣ ΓΙΑ
                        ΤΕΛΟΣ  ΟΣΟ
                TELOS AN
ΤΕΛΟΣ  ΟΣΟ
```

Μετατροπή Πληροφοριών Κληρονομικότητας

```
Δεδομένα
result: String;
query : ResultSet;
classesInherits: Vector;

query:= πάρε τα στοιχεία της στήλης SUB_NAME και SUPER_NAME από τον πίνακα
SYSIBM.SYShierarchies όπου η στήλη ROOT_SCHEMA = όνομα του σχήματος ;

ΟΣΟ (query έχει στοιχεία) ΕΠΑΝΕΛΑΒΕ
        result := "sc:" + query. SUB_NAME + " rdfs:subClassOf sc:" + query. SUPER_NAME + ".";
        classesInherits := πρόσθεσε result;
ΤΕΛΟΣ  ΟΣΟ
```

Μετατροπή Εναυσμάτων

```
Δεδομένα
Type, event, text : String;
Rs : ResultSet;
VecTrigger : Vector;

Rs := πάρε όλα τα στοιχεία για τον πίνακα SYSCAT.Triggers όπου η στήλη Tabschema = όνομα του
σχήματος ;

ΟΣΟ (rs έχει στοιχεία) ΕΠΑΝΕΛΑΒΕ
        Text := “t:”  rs. Όνομα trigger;

        Type := rs.Trigtime;
        ΑΝ (type= ‘Α’) ΤΟΤΕ
                Text := Text + "t:Trigtime \"After event\";";
        ΑΛΛΙΩΣ ΑΝ (type= ‘Β’)ΤΌΤΕ
                Text := Text + "t:Trigtime \"Before event\";";
```

```
        ΑΛΛΙΩΣ
                Text := Text + "t:Trigtime \" Instead of event\";";
        ΤΕΛΟΣ ΑΝ

        event := rs. Trigevent;
        ΑΝ (event = 'I')ΤΌΤΕ
                text = text + "\t t:Trigevent \"Insert\";";
        ΑΛΛΙΩΣ (event = 'D') ΑΝ ΤΌΤΕ
                text = text + "\t t:Trigevent \"Delete\";";
        ΑΛΛΙΏΣ
                text = text + "\t t:Trigevent \"Update\";";
        ΤΕΛΟΣ ΑΝ

        ΑΝ (rs.valid = 'Y')ΤΌΤΕ
                text = text + "\t t:Valid \"Valid\".";
        ΑΛΛΙΏΣ
                text = text + "\t t:Valid \" Inoperative\".";
        ΤΕΛΟΣ ΑΝ
        VecTrigger := πρόσθεσε text;
ΤΕΛΟΣ  ΟΣΟ
```

ΕΠΙΛΟΓΟΣ

Ο Σημασιολογικός Ιστός (Web 3.0) είναι μια επέκταση του σημερινού Ιστού, που φέρει δομή στο ουσιαστικό περιεχόμενο των ιστοσελίδων και αναμένεται να παίξει μεγάλο ρόλο στη ζωή μας τα επόμενα χρόνια. Το RDF είναι το κύριο πρότυπο του Σημασιολογικού Ιστού στο οποίο βασίζονται και άλλα πρότυπα.

Στο κεφάλαιο αυτό είδαμε τι είναι ο Σημασιολογικός Ιστός καθώς και τα μέρει που τον αποτελούν. Είδαμε τα πρότυπα RDF,RDF Schema και RDF Turtle το οποίο και το αναλύσαμε μιας και είναι και το ζητούμενο της πτυχιακής αυτής, καθώς τα αποτελέσματα μας θα πρέπει να είναι σε αυτήν την μορφή.

Μέχρι στιγμής έχουμε δημιουργήσει μιας Αντικειμενοσχεσιακή Βάση Δεδομένων, έχουμε πρόσβαση σε αυτήν, μέσο της java και της JDBC τεχνολογίας και έχουμε δει τον τρόπο γραφής του τελικού αποτελέσματος σε RDF Turtle μορφή.

Στο επόμενο κεφάλαιο θα κάνουμε μια εισαγωγή στην JSF τεχνολογία και θα δούμε τις τεχνικές που χρησιμοποιήσαμε για την δημιουργία της διαδικτυακής μας εφαρμογής.

ΚΕΦΑΛΑΙΟ 5 – Τεχνολογία JSF (JavaServer Faces)

ΕΙΣΑΓΩΓΗ

Η εξέλιξη στην τεχνολογία και πόσο μάλλον στον χώρο της πληροφορικής και του διαδικτύου έφερε την ανάγκη για την δημιουργία νέων τεχνολογιών στον χώρο τον διαδικτυακών εφαρμογών. Σκοπός της παρούσας πτυχιακής είναι η δημιουργία μια διαδικτυακής εφαρμογής με την χρήση της τεχνολογίας JSF(JavaServer Faces).

Σε αυτό το κεφάλαιο θα δούμε τι είναι αυτή η τεχνολογία και θα παρουσιάσουμε τις μεθόδους που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής αυτής.

5.1 Εισαγωγή στην JSF

JavaServer Faces (JSF) είναι μια προδιαγραφή Java για τη δημιουργία διεπαφών χρήστη (UI) για server-side δικτυακές εφαρμογές. Αναπτύχθηκε μέσα από την Java κοινοτική διαδικασία βάσει του JSR - 314, σχεδιασμένη έτσι ώστε να μπορεί να αξιοποιηθεί από εργαλεία που θα κάνουν την ανάπτυξη μιας web εφαρμογής ακόμα πιο εύκολη. Έχει σχεδιαστεί για να διευκολύνει σημαντικά τον τρόπο γραφής κώδικα και τη διατήρηση εφαρμογών που τρέχουν σε ένα διακομιστή εφαρμογών Java.

Η JavaServer Faces τεχνολογία είναι ευέλικτη και αξιοποιεί τις υπάρχουσες τυποποιημένες έννοιες UI και web-tier χωρίς να περιορίζει την ανάπτυξη με μια συγκεκριμένη markup language, ή πρωτόκολλο, ή συσκευή πελάτη.

Χρησιμοποιεί XML αρχεία τα οποία ονομάζονται view templates ή Facelets views τα οποία φορτώνονται από τον FacesServlet μετά από την επεξεργασία κάποιου αιτήματος. Αφού ο FacesServlet φορτώσει το κατάλληλο view template χτίζει ένα συστατικό δέντρο επεξεργάζεται τα δεδομένα και επιστρέφει την απάντηση συνήθως σε HTML μορφή.

Η JavaServer Faces τεχνολογία περιλαμβάνει ένα σύνολο από APIs για την αναπαράσταση UI components και τη διαχείριση της κατάστασης τους, με το

χειρισμό των συμβάντων και την επικύρωση των εισροών, που καθορίζει η σελίδα πλοήγησης.

5.2 Facelets

Τα Facelets είναι μια γλώσσα δήλωσης που χρησιμοποιείται για τη κατασκευή JSF σελίδων χρησιμοποιώντας τα πρότυπα μορφοποίησης της HTML.

Χρησιμοποιούν την XHTML για την δημιουργία των ιστοσελίδων και τις Facelets tag, JavaServer Faces ,JSTL tag βιβλιοθήκες. Παρακάτω παρουσιάζεται σε μορφή πίνακα οι βιβλιοθήκες που χρησιμοποιούνται.

Tag Library	URI	Prefix	Example	Contents
JavaServer Faces Facelets Tag Library	http://java.sun.com /jsf/facelets	ui:	ui:component ui:insert	Tags for templating
JavaServer Faces HTML Tag Library	http://java.sun.com/jsf/html	h:	h:head h:body h:outputText h:inputText	JavaServer Faces component tags for all UIComponent objects
JavaServer Faces Core Tag Library	http://java.sun.com/jsf/core	f:	f:actionListener f:attribute	Tags for JavaServer Faces custom actions that are independent of any particular render kit
JSTL Core Tag Library	http://java.sun.com /jsp/jstl/core	c:	c:forEach c:catch	JSTL 1.2 Core Tags
JSTL Functions Tag Library	http://java.sun.com/jsp/jstl /functions	fn:	fn:toUpperCase fn:toLowerCase	JSTL 1.2 Functions Tags

Σχήμα 7 – Facelets βιβλιοθήκες

5.3 JSF Managed Beans

Τα Managed Beans χρησιμοποιούνται από την JSF για να διαχειρίζεται τις κλάσεις που δημιουργούνται από τον χρήστη. Με τα Managed Beans μπορούμε να έχουμε πρόσβαση στις μεθόδους και τις μεταβλητές μιας κλάσης μέσο μιας JSF σελίδας κάνοντας ανάγνωση και εγγραφή σε αυτές.

Για την δημιουργία ενός Managed Bean θα πρέπει να εισάγουμε στην αρχή της κλάσης @ManagedBean και ανάλογα τον σκοπό του Bean τα @RequestScoped, ή @ApplicationScoped, ή @SessionScoped ή αν δεν θέλουμε να αποθηκεύονται οι αλλαγές στην κλάση δεν εισάγουμε τίποτα από αυτά. Τέλος εισάγουμε και τις αντίστοιχες βιβλιοθήκες.

- import javax.faces.bean.ManagedBean;
- import javax.faces.bean. RequestScoped;
- import javax.faces.bean.ApplicationScoped;
- import javax.faces.bean. SessionScoped;

Έπειτα θα πρέπει να δηλώσουμε στο Faces-config.xml αρχείο σε ποια κλάση αναφέρεται το Bean, σε ποιο πακέτο βρίσκεται, με ποιο όνομα θα μπορούμε να έχουμε πρόσβαση σε αυτό και τον σκοπό του Bean αυτού.

Παράδειγμα

```
<managed-bean>
        <managed-bean-name>
              connectToDB
        </managed-bean-name>
        <managed-bean-class>
              myapp. connectToDB
        </managed-bean-class>
        <managed-bean-scope>
              application
        </managed-bean-scope>
</managed-bean>
```

Όπου **connectToDB** είναι το όνομα με το οποίο θα μπορούμε να έχουμε πρόσβαση σε αυτό, **myapp. connectToDB** είναι το όνομα του πακέτου και της κλάσης, **application** είναι ο σκοπός του Bean αυτού.

Στην συνέχεια για να έχουμε πρόσβαση στο bean μέσο μιας JSF σελίδας γράφουμε με τον εξής τρόπο #{όνομα_του_bean.όνομα μεθόδου ()} ή #{όνομα_του_bean.όνομα μεταβλητής)}

Παράδειγμα

** #{connectToDB.getMsg()} **

Όπου η μέθοδος **getMsg()** είναι μια μέθοδος που επιστρέφει την τιμή μιας μεταβλητής και βρίσκεται στην κλάση **connectToDB** το αποτέλεσμα της μεθόδου αυτής θα εμφανιστεί μέσα σε ένα div**.**

5.4 Τύποι Managed Beans

Τα Managed Beans έχουν διαφορετικούς τρόπους για να χειρίζονται τα HTTP αιτήματα (requests) και αυτό δηλώνεται στην κλάση ανάλογα με τον σκοπό του Bean. Αναλόγως τον τύπο που θα δηλώσουμε η κλάση θα έχει και την αντίστοιχη διάρκεια ζωής και επομένως οι μεταβλητές μας θα έχουν και την αντίστοιχη διάρκεια αποθήκευσης των τιμών. Οι τύποι των Managed Beans είναι :

Request

Ένα Bean που έχει δηλωθεί ως Request Scoped έχει διάρκεια ζωής για μια μόνο HTTP αίτηση και στην επόμενη αίτηση ξανά δημιουργείται το Bean. Δημιουργείτε όταν γίνεται ένα HTTP αίτημα και καταστρέφεται όταν η απόκριση του HTTP αιτήματος έχει τελειώσει. Ένα τέτοιο Bean δηλώνεται ως εξής: @RequestScoped

Application

Ένα Bean που έχει δηλωθεί ως Application Scoped έχει διάρκεια ζωής ίση με την διάρκεια ζωής που έχει και η web εφαρμογή. Δημιουργείται μετά την πρώτη HTTP αίτηση που αναφέρεται στο Bean αυτό ή όταν ξεκινά η web εφαρμογή και έχει δηλωθεί στο Managed Bean το eager = true. Ένα τέτοιο Bean καταστρέφεται όταν η web εφαρμογή τερματίζει και δηλώνεται ως εξής: @ ApplicationScoped. Ένα παράδειγμα που χρησιμοποιείται αυτός ο τύπος είναι σε JDBC datasource object.

Session

Ένα Bean που έχει δηλωθεί ως Session Scoped έχει διάρκεια ζωής ίση με την διάρκεια ζωής ενός Session και παραμένει ζωντανό για πολλαπλά αιτήματα καθώς τα αντικείμενα του αποθηκεύονται στο session. Δημιουργείται με την πρώτη HTTP αίτηση που αναφέρεται στο bean κατά την διάρκεια ενός session και καταστρέφεται όταν τελειώνει το session ή ο χρόνος του session αυτού. Ένα παράδειγμα που χρησιμοποιείται είναι για καλάθια αγορών και δηλώνεται ως εξής: @ SessionScoped.

View

Ένα Bean που έχει δηλωθεί ως View Scoped έχει διάρκεια ζωής για όσο ο χρήστης αλληλεπιδρά με την ίδια JSF σελίδα στο παράθυρο ή την καρτέλα του φυλλομετρητή. Δημιουργείται με την πρώτη HTTP αίτηση που αναφέρεται στο bean αυτό και τελειώνει όταν ο χρήστης κλείσει την καρτέλα ή το παράθυρο του φυλλομετρητή. Δηλώνεται ως εξής: @ ViewScoped

None

Ένα τέτοιο bean αυτού του τύπου δεν αποθηκεύεται πουθενά και δημιουργείται όποτε είναι αναγκαίο. Δηλώνεται ως εξής: @NoneScoped

5.5 JSF Ετικέτες

Η τεχνολογία JSF παρέχει στον χρήστη διάφορες κατηγορίες ετικετών οι οποίες διευκολύνουν την ανάπτυξη μια διαδικτυακής εφαρμογής. Οι κατηγορίες των ετικετών αυτών παρουσιάζονται στην συνέχεια του κεφαλαίου.

Βασικές JSF Ετικέτες

Η JSF προσφέρει στον χρήστη ετικέτες οι οποίες παρέχουν τις βασικές HTML ετικέτες, αυτές οι ετικέτες αντιστοιχούν στην έξοδο των αντίστοιχων HTML ετικετών.

Για να τις χρησιμοποιήσουμε θα πρέπει να εισάγουμε τα εξής URI στην αρχή της σελίδας μέσα σε html αγκύλες.

< html

xmlns="http://www.w3.org/1999/xhtml"
xmlns:h="http://java.sun.com/jsf/html"

>

JSF Facelets Ετικέτες

Η JSF τεχνολογία παρέχει ειδικές ετικέτες για την δημιουργία και την σχεδίαση μιας διαδικτυακής εφαρμογής οι οποίες ονομάζονται Facelets ετικέτες. Οι ετικέτες αυτές παρέχουν ευελιξία στον χρήστη όσων αφορά την διαχείριση κοινών σημείων μεταξύ πολλαπλών σελίδων σε μια διαδικτυακή εφαρμογή.

Για να τις χρησιμοποιήσουμε θα πρέπει να εισάγουμε τα εξής URI στην αρχή της σελίδας μέσα σε html αγκύλες.

< html

xmlns="http://www.w3.org/1999/xhtml"

xmlns:ui="http://java.sun.com/jsf/facelets"

>

JSF Convertor Ετικέτες

Η JSF τεχνολογία παρέχει ετικέτες οι οποίες μετατρέπουν τα δεδομένα από ένα στοιχείο διεπαφής χρήστη σε αντικείμενο το οποίο χρησιμοποιείται από ένα managed bean. Για παράδειγμα αυτές οι ετικέτες μπορούν να μετατρέψουν ένα απλό κείμενο σε ημερομηνία, καθώς επίσης και να καθορίσουν την μορφή του.

Για να τις χρησιμοποιήσουμε θα πρέπει να εισάγουμε τα εξής URI στην αρχή της σελίδας μέσα σε html αγκύλες.

< html

xmlns="http://www.w3.org/1999/xhtml"

xmlns:f="http://java.sun.com/jsf/core"

>

JSF Validator Ετικέτες

Οι JSF Validator ετικέτες χρησιμοποιούνται για την επικύρωση και τον έλεγχο της ορθότητας των τιμών που εισάγονται από τους χρήστες της διαδικτυακής εφαρμογής. Οι ετικέτες αυτές μπορούν να ελέγξουν και να επικυρώσουν το μήκος ενός πεδίου ή το είδος της τιμής που εισάγεται σε κάποιο πεδίο. Αυτές οι ετικέτες είναι πολύ χρήσιμες στην συμπλήρωση φορμών εισαγωγής στοιχείων και γενικότερα σε σημεία όπου πρέπει ο χρήστης να συμπληρώσει κάποια στοιχεία και οι τιμές αυτών πρέπει να έχουν κάποια συγκεκριμένη μορφή.

Για να τις χρησιμοποιήσουμε θα πρέπει να εισάγουμε τα εξής URI στην αρχή της σελίδας μέσα σε html αγκύλες.

```
< html

    xmlns="http://www.w3.org/1999/xhtml"

    xmlns:f="http://java.sun.com/jsf/core"

>
```

ΕΠΙΛΟΓΟΣ

Η JSF είναι μια τεχνολογία με την οποία μπορούμε να δημιουργήσουμε διαδικτυακές εφαρμογές χρησιμοποιώντας την γλώσσα προγραμματισμού JAVA και μας προσφέρει διάφορες δυνατότητες. Σε αυτό κεφάλαιο παρουσιάσαμε πως υλοποιούνται αυτές οι δυνατότητες της JSF και ποιες τεχνικές χρησιμοποιηθήκαν για την υλοποίηση της εφαρμογής.

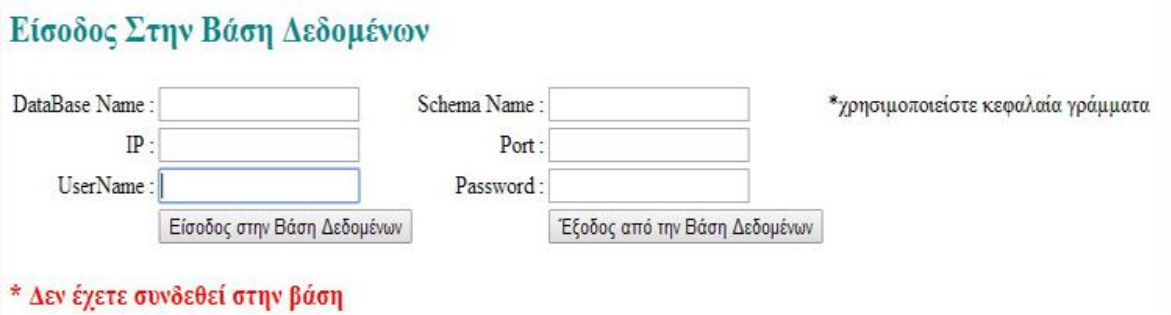
Σε αυτό το σημείο έχουμε ολοκληρώσει το ζητούμενο της πτυχιακής αυτής, που είναι η δημιουργία μιας JSF διαδικτυακής εφαρμογής, που θα διαβάζει μια Αντικειμενο-σχεσιακή Βάση Δεδομένων και θα μετατρέπει σε RDF Turtle μορφή.

Στο επόμενο κεφάλαιο παρουσιάζεται ένας ενδεικτικός οδηγός χρήσης.

ΚΕΦΑΛΑΙΟ 6 – Οδηγός χρήσης της εφαρμογής

Στόχος της παρούσας πτυχιακής ήταν η δημιουργία μιας διαδικτυακής εφαρμογής με την JSF τεχνολογία η οποία θα διαβάζει μια Αντικειμενο-σχεσιακή βάση δεδομένων και θα την μετατρέπει σε RDF Turtle μορφή ώστε να μπορεί να χρησιμοποιηθεί από τον σημασιολογικό ιστό.

Ξεκινώντας λοιπόν και αφού έχουμε φορτώσει την ιστοσελίδα μας θα πάμε από το βασικό μενού στο «Είσοδος στην Βάση» και θα συμπληρώσουμε τα απαιτούμενα στοιχεία για να κάνουμε είσοδο στην βάση. Συμπληρώνουμε τα στοιχεία και πατάμε το κουμπί «Είσοδος στην Βάση Δεδομένων» το οποίο καλεί την μέθοδο execute() η οποία δίνει στις μεταβλητές τις τιμές από την φόρμα συμπλήρωσης των στοιχείων και καλεί την μέθοδο connect(). Η μέθοδος connect() δημιουργεί την σύνδεση με την βάση δεδομένων και καλεί τις μεθόδους getData(), setUDTS(), setRDFsSubClasses(), setTriggers() και createprefixs() και μεταφέρεται στην σελίδα export.xhtml.



Η μέθοδος getData() διαβάζει τα δεδομένα από τους πίνακες τις βάσεις και τα αποθηκεύει σε RDF Turtle μορφή σε ένα Vector.

Η μέθοδος setUDTS() αποθηκεύει σε ένα Vector το ονόματα των τύπων που έχει δημιουργήσει ο χρήστης.

Η μέθοδος createprefixs() δημιουργεί τα prefixs για το Turtle αρχείο σύμφωνα με την ip διεύθυνση που έχει δώσει ο χρήστης και τα ονόματα των πινάκων που υπάρχουν στην βάση.

Η μέθοδος setRDFsSubClasses() αποθηκεύει σε ένα Vector πληροφορίες για την κληρονομικότητα.

Αν όλα πάνε καλά τότε θα μεταφερθούμε σε μια σελίδα όπου θα μας εμφανίζονται τα δεδομένα της βάσης σε RDF Turtle μορφή και πατώντας το κουμπί «Κατέβασμα Αρχείου» θα μπορέσουμε να κατεβάσουμε το αρχείο σε ttl μορφή. Το κουμπί αυτό καλεί την μέθοδο getTTLFile() η οποία δημιουργεί και κατεβάζει ένα αρχείο με τα αποτελέσματα σε RDF Turtle μορφή.

Επίσης εφόσον κάνουμε σύνδεση στην βάση πατώντας από το βασικό μενού στο «Πληροφορίες Βάσης» μπορούμε να δούμε τα ονόματα των πινάκων, τους τύπους που έχει δημιουργήσει ο χρήστης καθώς επίσης και πληροφορίες που έχουν να κάνουν με την κληρονομικότητα των τύπων και των πινάκων.

Ονόματα Πινάκων:

DOKIMI_DECIMAL_TABLE
DROMOLOGIO
LEOFOREIO
ODIGOS
PLIROMA
POLI
SUNODOS
YPALLILOS

Ονόματα Τύπων του Χρήστη:

TUPOS_DEMICLA
TYPOS_DIPLOMATOS
VATHMOS_SUNODOU
DOKIMI_DECIMAL
DROMOLOGIO_T
LEOFOREIO_T
ODIGOS_T
PLIROMA_T
POLI_T
SUNODOS_T
YPALLILOS_T

Κληρομονικότητα:

ODIGOS_T INHERITS YPALLILOS_T
SUNODOS_T INHERITS YPALLILOS_T
ODIGOS INHERITS YPALLILOS
SUNODOS INHERITS YPALLILOS

Για να αποσυνδεθούμε από την βάση πατάμε από το βασικό μενού στο «Είσοδος στην Βάση» και πατάμε το κουμπί «Έξοδος από την Βάση Δεδομένων» το οποίο καλεί την μέθοδο closeCon() η οποία κλείνει την σύνδεση με την βάση και διαγράφει τους πίνακες με τα δεδεμένα.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Έχουμε δημιουργήσει μια διαδικτυακή εφαρμογή η οποία συνδέεται στην IBM DB2 και διαβάζει μια Αντικειμενο-σχεσιακή βάση δεδομένων την οποία την μετατρέπει σε μορφή RDF Turtle κατάλληλη για χρήση στο Σημασιολογικό Ιστό. Μια εφαρμογή η οποία είναι έτοιμη για χρήση για μια οποιαδήποτε Αντικειμενο-σχεσιακή βάση δεδομένων που βρίσκεται αποθηκευμένη στην IBM DB2.

Επίσης είδαμε την λειτουργία των JDBC και JSF τεχνολογιών, τα πρότυπα RDF, RDFS και RDF Turtle, καθώς επίσης και τις ιδιαιτερότητες των Αντικειμενο-σχεσιακών βάσεων δεδομένων.

Χαίρομαι πάρα πολύ που μου δόθηκε η δυνατότητα να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα το οποίο θα μας απασχολήσει ακόμα πιο έντονα τα επόμενα χρόνια.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης - Μονάδα Σημασιολογικού Ιστού - http://www.swu.auth.gr/el/glossary - 2014
2. Κεραμόπουλος Ευκλείδης, Τμήμα Μηχανικών Πληροφορικής του Αλεξάνδρειου ΤΕΙ Θεσσαλονίκης – Διαφάνειες Θεώριας Τεχνολογία Βάσεων Δεδομένων - 2014
3. Κεραμόπουλος Ευκλείδης, Τμήμα Μηχανικών Πληροφορικής του Αλεξάνδρειου ΤΕΙ Θεσσαλονίκης - Διαφάνειες Εργαστηρίων (Linked Data, RDF, RDFS) - 2014
4. Πολυμεσικές βάσεις δεδομένων και διαδίκτυο - http://www.ipet.gr/digitech2/index.php?option=com_content&task=view&id=88&Itemid=57- 2014
5. Apache - D2RQ - http://d2rq.org/ - 2014
6. Jeffrey D. Ullman, Jennifer Widom. Βασικές Αρχές για τα Συστήματα Βάσεων Δεδομένων Δεύτερη Αμερικάνικη Έκδοση
7. Facelets - JavaServer Faces View Definition Framework - https://facelets.java.net/nonav/docs/dev/docbook.html- 2014
8. Grigoris Antoniou, Frank van Harmelen. A Semantic Web Primer, 2nd edition. The MIT Press, 2008.
9. IBM Knowledge Center - http://www-01.ibm.com/support/knowledgecenter- 2014
10. Introduction to JavaServer Faces 2.x - https://netbeans.org/kb/docs/web/jsf20-intro.html- 2014
11. Introduction to Java Server Faces(JSF) - http://web.princeton.edu/sites/isapps/jasig/2004summerWestminster/Presentations/java%20server%20faces.pdf- 2014
12. JavaServer Faces Technology Overview - http://www.oracle.com/technetwork/java/javaee/overview-140548.html- 2014
13. JSF Managed Beans - http://www.tutorialspoint.com/jsf/jsf_managed_beans.htm- 2014
14. JavaServer Faces (JSF) Tutorial - http://www.tutorialspoint.com/jsf - 2014

15. Lesson: JDBC Basics - http://docs.oracle.com/javase/tutorial/jdbc/basics/- 2014
16. Lightning intro to Turtle - http://wikitravel.org/en/Wikitravel:Turtle_RDF- 2014
17. R2RML RDB to RDF Mapping Language - http://www.w3.org/TR/2012/PR-r2rml-20120814/ - 2014
18. RDF 1.1 Turtle - http://www.w3.org/TR/turtle/- 2014
19. RDF Schema 1.1 - http://www.w3.org/TR/rdf-schema/- 2014
20. Resource Description Framework (RDF) - http://www.w3.org/RDF/- 2014
21. The JavaServer Faces Managed Bean Facility - http://www.oracle.com/technetwork/topics/index-101145.html- 2014
22. Terse RDF Triple Language-https://dvcs.w3.org/hg/rdf/raw-file/tip/rdf-turtle/index.html – 2014
23. W3C - A Direct Mapping of Relational Data to RDF - http://www.w3.org/TR/rdb-direct-mapping/ - 2014

ΠΑΡΑΡΤΗΜΑΤΑ

Δημιουργία τύπων και πινάκων

```
Create Distinct Type typos_diplomatos AS varchar(30) WITH COMPARISONS;

Create Distinct Type vathmos_sunodou AS varchar(30) WITH COMPARISONS;

Create type Ypallilos_t AS (onoma varchar(20),epwnumo varchar(30), fulo
varchar(1), hlikia integer, dieu8unsh_katoikias
varchar(40),hmeromhnia_proslhpshs Date)

Ref using int mode db2sql;

Create type dokimi_decimal  AS (dokimiDec tupos_demicla)mode db2sql;

Create table dokimi_decimal_table OF dokimi_decimal (REF is OID USER
GENERATED);

Create type Odigos_t under Ypallilos_t AS (typos_diplomatos
typos_diplomatos,ari8mos_dromologiwn integer)

mode db2sql;

Create type Sunodos_t under Ypallilos_t AS (vathmos_sunodou
vathmos_sunodou)

mode db2sql;

Create type Pliroma_t AS (Odigos REF(Odigos_t),Synodos REF(Sunodos_t))

REF USING INT MODE DB2SQL;

Create type Poli_t AS (onoma varchar(20),xwra varchar(30))
```

Ref using int mode db2sql;

Create type leoforeio_t AS (arithmos_kukloforias varchar(30), edra REF(Poli_t), hmeromhnia_kataskeuhs Date, pliroma REF(Pliroma_t), ari8mos_xiliometrwn integer)

REF USING INT MODE DB2SQL;

Create type dromologio_t AS (Afetiria REF(Poli_t), proorismos REF(Poli_t), hmera varchar(10), wra Time, lewforeio REF(leoforeio_t), timh_eishthriou double)

REF USING INT MODE DB2SQL;

Create table Ypallilos OF Ypallilos_t (REF is OID USER GENERATED);

Create table Odigos OF Odigos_t under Ypallilos INHERIT SELECT PRIVILEGES;

Create table Sunodos OF Sunodos_t under Ypallilos INHERIT SELECT PRIVILEGES;

Create table Pliroma OF Pliroma_t (REF is OID USER GENERATED, Odigos with options scope Odigos, Synodos with options scope Sunodos);

Create table Poli OF Poli_t (REF is OID USER GENERATED);

Create table leoforeio OF leoforeio_t (REF is OID USER GENERATED, edra with options scope Poli, pliroma with options scope Pliroma);

Create table dromologio OF dromologio_t (REF is OID USER GENERATED, Afetiria with options scope Poli, proorismos with options scope Poli, lewforeio with options scope leoforeio);

Αρχείο σε RDF Turtle μορφή

@prefix DROMOLOGIO : <http://localhost/DROMOLOGIO/>.

@prefix LEOFOREIO : <http://localhost/LEOFOREIO/>.

```
@prefix ODIGOS : <http://localhost/ODIGOS/>.
@prefix PLIROMA : <http://localhost/PLIROMA/>.
@prefix POLI : <http://localhost/POLI/>.
@prefix SUNODOS : <http://localhost/SUNODOS/>.
@prefix YPALLILOS : <http://localhost/YPALLILOS/>.
@prefix rel : <http://localhost/relations/>.
@prefix : <http://localhost/elements/>.
@prefix sc: <http://localhost/schema/>.
@prefix t: <http://localhost/triggers/>.
@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#> .
DROMOLOGIO:1
        rel:AFETIRIA POLI:10 ;
        rel:PROORISMOS POLI:20 ;
        :HMERA "Deutera" ;
        :WRA "10:00:00" ;
        rel:LEWFOREIO LEOFOREIO:10 ;
        :TIMH_EISHTHRIOU 50.0 .


DROMOLOGIO:2
        rel:AFETIRIA POLI:10 ;
        rel:PROORISMOS POLI:20 ;
        :HMERA "Triti" ;
        :WRA "10:00:00" ;
        rel:LEWFOREIO LEOFOREIO:10 ;
```

:TIMH_EISHTHRIOU 50.0 .

DROMOLOGIO:3

rel:AFETIRIA POLI:10 ;

rel:PROORISMOS POLI:20 ;

:HMERA "Tetarti" ;

:WRA "10:00:00" ;

rel:LEWFOREIO LEOFOREIO:10 ;

:TIMH_EISHTHRIOU 50.0 .

DROMOLOGIO:4

rel:AFETIRIA POLI:10 ;

rel:PROORISMOS POLI:20 ;

:HMERA "Pempti" ;

:WRA "10:00:00" ;

rel:LEWFOREIO LEOFOREIO:10 ;

:TIMH_EISHTHRIOU 50.0 .

DROMOLOGIO:5

rel:AFETIRIA POLI:10 ;

rel:PROORISMOS POLI:20 ;

:HMERA "Paraskeui" ;

:WRA "10:00:00" ;

rel:LEWFOREIO LEOFOREIO:10 ;

:TIMH_EISHTHRIOU 50.0 .

DROMOLOGIO:6

rel:AFETIRIA POLI:10 ;

rel:PROORISMOS POLI:20 ;

:HMERA "Sabbato" ;

:WRA "10:00:00" ;

rel:LEWFOREIO LEOFOREIO:10 ;

:TIMH_EISHTHRIOU 50.0 .

DROMOLOGIO:7

rel:AFETIRIA POLI:10 ;

rel:PROORISMOS POLI:20 ;

:HMERA "Kuriaki" ;

:WRA "10:00:00" ;

rel:LEWFOREIO LEOFOREIO:10 ;

:TIMH_EISHTHRIOU 50.0 .

DROMOLOGIO:11

rel:AFETIRIA POLI:20 ;

rel:PROORISMOS POLI:10 ;

:HMERA "Deutera" ;

:WRA "08:00:00" ;

rel:LEWFOREIO LEOFOREIO:10 ;

:TIMH_EISHTHRIOU 50.0 .

DROMOLOGIO:12

rel:AFETIRIA POLI:20 ;

rel:PROORISMOS POLI:10 ;

:HMERA "Triti" ;

:WRA "08:00:00" ;

rel:LEWFOREIO LEOFOREIO:10 ;

:TIMH_EISHTHRIOU 50.0 .

DROMOLOGIO:13

rel:AFETIRIA POLI:20 ;

rel:PROORISMOS POLI:10 ;

:HMERA "Tetarti" ;

:WRA "08:00:00" ;

rel:LEWFOREIO LEOFOREIO:10 ;

:TIMH_EISHTHRIOU 50.0 .

DROMOLOGIO:14

rel:AFETIRIA POLI:20 ;

rel:PROORISMOS POLI:10 ;

:HMERA "Pempti" ;

:WRA "08:00:00" ;

rel:LEWFOREIO LEOFOREIO:10 ;

:TIMH_EISHTHRIOU 50.0 .

DROMOLOGIO:15

rel:AFETIRIA POLI:20 ;

rel:PROORISMOS POLI:10 ;

:HMERA "Paraskeui" ;

:WRA "08:00:00" ;

rel:LEWFOREIO LEOFOREIO:10 ;

:TIMH_EISHTHRIOU 50.0 .

DROMOLOGIO:16

rel:AFETIRIA POLI:20 ;

rel:PROORISMOS POLI:10 ;

:HMERA "Sabbato" ;

:WRA "08:00:00" ;

rel:LEWFOREIO LEOFOREIO:10 ;

:TIMH_EISHTHRIOU 50.0 .

DROMOLOGIO:17

rel:AFETIRIA POLI:20 ;

rel:PROORISMOS POLI:10 ;

:HMERA "Kuriaki" ;

:WRA "08:00:00" ;

rel:LEWFOREIO LEOFOREIO:10 ;

:TIMH_EISHTHRIOU 50.0 .

DROMOLOGIO:21

rel:AFETIRIA POLI:30 ;

rel:PROORISMOS POLI:40 ;

:HMERA "Kuriaki" ;

:WRA "03:00:00" ;

rel:LEWFOREIO LEOFOREIO:20 ;

:TIMH_EISHTHRIOU 30.0 .

DROMOLOGIO:22

rel:AFETIRIA POLI:40 ;

rel:PROORISMOS POLI:30 ;

:HMERA "Deutera" ;

:WRA "08:00:00" ;

rel:LEWFOREIO LEOFOREIO:20 ;

:TIMH_EISHTHRIOU 30.0 .

DROMOLOGIO:31

rel:AFETIRIA POLI:10 ;

rel:PROORISMOS POLI:40 ;

:HMERA "Triti" ;

:WRA "08:00:00" ;

rel:LEWFOREIO LEOFOREIO:30 ;

:TIMH_EISHTHRIOU 10.0 .

DROMOLOGIO:32

rel:AFETIRIA POLI:40 ;

rel:PROORISMOS POLI:10 ;

:HMERA "Triti" ;

:WRA "12:00:00" ;

rel:LEWFOREIO LEOFOREIO:30 ;

:TIMH_EISHTHRIOU 10.0 .

DROMOLOGIO:33

rel:AFETIRIA POLI:10 ;

rel:PROORISMOS POLI:40 ;

:HMERA "Tetarti" ;

:WRA "08:00:00" ;

rel:LEWFOREIO LEOFOREIO:30 ;

:TIMH_EISHTHRIOU 10.0 .

DROMOLOGIO:34

rel:AFETIRIA POLI:40 ;

rel:PROORISMOS POLI:10 ;

:HMERA "Tetarti" ;

:WRA "12:00:00" ;

rel:LEWFOREIO LEOFOREIO:30 ;

:TIMH_EISHTHRIOU 10.0 .

LEOFOREIO:10

:ARITHMOS_KUKLOFORIAS "NHE1010" ;

rel:EDRA POLI:10 ;

:HMEROMHNIA_KATASKEUHS "2007-05-01" ;

rel:PLIROMA PLIROMA:1 ;

:ARI8MOS_XILIOMETRWN 10000 .

LEOFOREIO:20

:ARITHMOS_KUKLOFORIAS "NEE9990" ;

rel:EDRA POLI:10 ;

:HMEROMHNIA_KATASKEUHS "2006-06-01" ;

rel:PLIROMA PLIROMA:2 ;

:ARI8MOS_XILIOMETRWN 50000 .

LEOFOREIO:30

:ARITHMOS_KUKLOFORIAS "YOI7867" ;

rel:EDRA POLI:20 ;

:HMEROMHNIA_KATASKEUHS "2005-02-01" ;

rel:PLIROMA PLIROMA:3 ;

:ARI8MOS_XILIOMETRWN 100000 .

LEOFOREIO:40

:ARITHMOS_KUKLOFORIAS "PAY4532" ;

rel:EDRA POLI:30 ;

:HMEROMHNIA_KATASKEUHS "2004-09-01" ;

rel:PLIROMA PLIROMA:4 ;

:ARI8MOS_XILIOMETRWN 150000 .

ODIGOS:21

:ONOMA "Dimitris" ;

:EPWNUMO "Stamatiou" ;

:FULO "M" ;

:HLIKIA 40 ;

:DIEU8UNSH_KATOIKIAS "Kilkis 20" ;

:HMEROMHNIA_PROSLHPSHS "1995-05-25" ;

:TYPOS_DIPLOMATOS "Epaggelmatiko"^^xsd:String ;

:ARI8MOS_DROMOLOGIWN 123456 .

ODIGOS:22

:ONOMA "Katerina" ;

:EPWNUMO "Stauridou" ;

:FULO "F" ;

:HLIKIA 20 ;

:DIEU8UNSH_KATOIKIAS "Papafi 201" ;

:HMEROMHNIA_PROSLHPSHS "2001-01-14" ;

:TYPOS_DIPLOMATOS "Erasitexniko"^^xsd:String ;

:ARI8MOS_DROMOLOGIWN 3489762 .

ODIGOS:23

:ONOMA "Anna" ;

:EPWNUMO "Papasotiriou" ;

:FULO "F" ;

:HLIKIA 32 ;

:DIEU8UNSH_KATOIKIAS "Ermou 17" ;

:HMEROMHNIA_PROSLHPSHS "2003-11-01" ;

:TYPOS_DIPLOMATOS "Erasitexniko"^^xsd:String ;

:ARI8MOS_DROMOLOGIWN 4443233 .

PLIROMA:1

rel:ODIGOS ODIGOS:21 ;

rel:SYNODOS SUNODOS:14 .

PLIROMA:2

rel:ODIGOS ODIGOS:22 ;

rel:SYNODOS SUNODOS:11 .

PLIROMA:3

rel:ODIGOS ODIGOS:21 ;

rel:SYNODOS SUNODOS:12 .

PLIROMA:4

rel:ODIGOS ODIGOS:23 ;

rel:SYNODOS SUNODOS:13 .

POLI:10

:ONOMA "Thessaloniki" ;

:XWRA "Ellada" .

POLI:20

:ONOMA "Athina" ;

:XWRA "Ellada" .

POLI:30

:ONOMA "Florina" ;

:XWRA "Ellada" .

POLI:40

:ONOMA "Larisa" ;

:XWRA "Ellada" .

POLI:50

:ONOMA "Volos" ;

:XWRA "Ellada" .

POLI:60

:ONOMA "Drama" ;

:XWRA "Ellada" .

POLI:110

:ONOMA "Sofia" ;

:XWRA "Boulgaria" .

SUNODOS:11

:ONOMA "Kostas" ;

:EPWNUMO "Papadopoulos" ;

:FULO "M" ;

:HLIKIA 55 ;

:DIEU8UNSH_KATOIKIAS "Axaion 15" ;

:HMEROMHNIA_PROSLHPSHS "1975-05-25" ;

:VATHMOS_SUNODOU "A"^^xsd:String .

SUNODOS:12

:ONOMA "Giorgios" ;

:EPWNUMO "Giorgiadis" ;

:FULO "M" ;

:HLIKIA 53 ;

:DIEU8UNSH_KATOIKIAS "Serron 34" ;

:HMEROMHNIA_PROSLHPSHS "1976-04-11" ;

:VATHMOS_SUNODOU "A"^^xsd:String .

SUNODOS:13

:ONOMA "Sotiria" ;

:EPWNUMO "Bellou" ;

:FULO "F" ;

:HLIKIA 38 ;

:DIEU8UNSH_KATOIKIAS "Gravias 5" ;

:HMEROMHNIA_PROSLHPSHS "1999-06-02" ;

:VATHMOS_SUNODOU "B"^^xsd:String .

SUNODOS:14

:ONOMA "Sofia" ;

:EPWNUMO "Iatrou" ;

:FULO "F" ;

:HLIKIA 41 ;

:DIEU8UNSH_KATOIKIAS "Peramou 8" ;

:HMEROMHNIA_PROSLHPSHS "1997-06-02" ;

:VATHMOS_SUNODOU "B"^^xsd:String .

YPALLILOS:1

:ONOMA "Giannis" ;

:EPWNUMO "Parios" ;

:FULO "M" ;

:HLIKIA 35 ;

:DIEU8UNSH_KATOIKIAS "Tsimiski 1" ;

:HMEROMHNIA_PROSLHPSHS "2004-01-16" .

YPALLILOS:2

:ONOMA "Giorgos" ;

:EPWNUMO "Karampelas" ;

:FULO "M" ;

:HLIKIA 40 ;

:DIEU8UNSH_KATOIKIAS "Megalou Alexandr" ;

:HMEROMHNIA_PROSLHPSHS "2000-08-10" .

YPALLILOS:3

:ONOMA "Maria" ;

:EPWNUMO "Iouliou" ;

:FULO "F" ;

:HLIKIA 25 ;

:DIEU8UNSH_KATOIKIAS "Egnatias 10" ;

:HMEROMHNIA_PROSLHPSHS "2002-11-25" .

YPALLILOS:11

:ONOMA "Kostas" ;

:EPWNUMO "Papadopoulos" ;

:FULO "M" ;

:HLIKIA 55 ;

:DIEU8UNSH_KATOIKIAS "Axaion 15" ;

:HMEROMHNIA_PROSLHPSHS "1975-05-25" .

YPALLILOS:12

:ONOMA "Giorgios" ;

:EPWNUMO "Giorgiadis" ;

:FULO "M" ;

:HLIKIA 53 ;

:DIEU8UNSH_KATOIKIAS "Serron 34" ;

:HMEROMHNIA_PROSLHPSHS "1976-04-11" .

YPALLILOS:13

:ONOMA "Sotiria" ;

:EPWNUMO "Bellou" ;

:FULO "F" ;

:HLIKIA 38 ;

:DIEU8UNSH_KATOIKIAS "Gravias 5" ;

:HMEROMHNIA_PROSLHPSHS "1999-06-02" .

YPALLILOS:14

:ONOMA "Sofia" ;

:EPWNUMO "Iatrou" ;

:FULO "F" ;

:HLIKIA 41 ;

:DIEU8UNSH_KATOIKIAS "Peramou 8" ;

:HMEROMHNIA_PROSLHPSHS "1997-06-02" .

YPALLILOS:21

:ONOMA "Dimitris" ;

:EPWNUMO "Stamatiou" ;

:FULO "M" ;

:HLIKIA 40 ;

:DIEU8UNSH_KATOIKIAS "Kilkis 20" ;

:HMEROMHNIA_PROSLHPSHS "1995-05-25" .

YPALLILOS:22

:ONOMA "Katerina" ;

:EPWNUMO "Stauridou" ;

:FULO "F" ;

:HLIKIA 20 ;

:DIEU8UNSH_KATOIKIAS "Papafi 201" ;

:HMEROMHNIA_PROSLHPSHS "2001-01-14" .

```
YPALLILOS:23
        :ONOMA "Anna" ;
        :EPWNUMO "Papasotiriou" ;
        :FULO "F" ;
        :HLIKIA 32 ;
        :DIEU8UNSH_KATOIKIAS "Ermou 17" ;
        :HMEROMHNIA_PROSLHPSHS "2003-11-01" .


sc:DROMOLOGIO a rdfs:Class.

sc:AFETIRIA
         a rdfs:Property;
         rdfs:domain sc:DROMOLOGIO;
         rdfs:range sc:POLI.

sc:PROORISMOS
         a rdfs:Property;
         rdfs:domain sc:DROMOLOGIO;
         rdfs:range sc:POLI.

sc:LEWFOREIO
         a rdfs:Property;
         rdfs:domain sc:DROMOLOGIO;
         rdfs:range sc:LEOFOREIO.
```

```
sc:LEOFOREIO a rdfs:Class.

sc:EDRA
        a rdfs:Property;
        rdfs:domain sc:LEOFOREIO;
        rdfs:range sc:POLI.

sc:PLIROMA
        a rdfs:Property;
        rdfs:domain sc:LEOFOREIO;
        rdfs:range sc:PLIROMA.

sc:ODIGOS a rdfs:Class.

sc:PLIROMA a rdfs:Class.

sc:ODIGOS
        a rdfs:Property;
        rdfs:domain sc:PLIROMA;
        rdfs:range sc:ODIGOS.

sc:SYNODOS
        a rdfs:Property;
        rdfs:domain sc:PLIROMA;
        rdfs:range sc:SUNODOS.

sc:POLI a rdfs:Class.
```

```
sc:SUNODOS a rdfs:Class.

sc:YPALLILOS a rdfs:Class.

sc:ODIGOS rdfs:subClassOf sc:YPALLILOS.
sc:SUNODOS rdfs:subClassOf sc:YPALLILOS.
# Triggers
t:DROMOLOGIO1 t:Trigtime "After event";
        t:Trigevent "Insert";
        t:Valid "Valid".
t:LEOFOREIO1 t:Trigtime "After event";
        t:Trigevent "Insert";
        t:Valid "Valid".
```