

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ



ΣΗΜΑΣΙΟΛΟΓΙΚΟΣ ΙΣΤΟΣ

ΚΩΝΣΤΑ ΕΡΩΦΙΛΗ

ΜΑΝΔΑΝΑ ΕΛΕΝΗ

Περιεχόμενα

Μέρος πρώτο : RDFS	1
Μέρος Δεύτερο : -OWL	11

Μέρος πρώτο : RDFS

Η Εργασία αφορά τον σχεδιασμό μίας οντολογίας που περιγράφει ένα πεδίο της επιλογής με τη χρήση της RDF/S. Το πεδίο που επιλέξαμε να μοντελοποιήσουμε είναι το μέσο κοινωνικής δικτύωσης **Facebook**.

1. Βασική Ιδέα

Αποφασίσαμε να συμπεριλάβουμε κάποια από τα βασικά πράγματα που μπορεί να βρει κανείς και να κάνει στο Facebook. Έχουμε προσεγγίσει ικανοποιητικά την οντότητα «χρήστης» δίνοντας του τη δυνατότητα να έχει φίλους, να αλληλοεπιδρά με αναρτημένο περιεχόμενο, να συμμετέχει σε ομάδες και ακόμα να πουλάει προϊόντα στο Facebook Marketplace. Παράλληλα δώσαμε βαρύτητα και στους οργανωτικούς ρόλους που μπορεί να έχει ένας χρήστης σε σελίδες και ομάδες. Τέλος προσπαθήσαμε να μοντελοποιήσουμε και την ορατότητα των ομάδων και των αναρτήσεων.

2. Κλάσεις

Οι κλάσεις που δημιουργήσαμε αναλύονται στο παρακάτω κομμάτι. Ξεκινάμε από αυτές που αποτελούν τον κορμό της υλοποίησης.

i. Κλάση Entity

Η κλάση αυτή αναπαριστά οντότητες όπως τις αντιλαμβανόμαστε στον πραγματικό κόσμο, δηλαδή ανθρώπους και οργανισμούς. Για αυτό και έχει ως υποκλάσεις τη *User*, η οποία αναπαριστά χρήστες και *Page*, η οποία αναπαριστά σελίδες. Κάνουμε την παραδοχή ότι οι σελίδες

αναπαριστούν μία οντότητα (πραγματικού κόσμου) διότι έχει πολλά κοινά χαρακτηριστικά με έναν χρήστη όσον αφορά την αλληλεπίδραση με την πλατφόρμα. Για παράδειγμα μία σελίδα μπορεί επίσης να κάνει likes σε posts ή άλλες σελίδες (ενώ μία ομάδα πχ δεν μπορεί). Ένας χρήστης έχει όμως διαφορετικές ιδιότητες από μία σελίδα, για παράδειγμα έχει στοιχεία όπως ονοματεπώνυμο, ημερομηνία γέννησης, λίστα φίλων κ.α. ενώ μία σελίδα έχει όνομα, έχει τον δημιουργό της και τους διάφορους διαχειριστές, ημερομηνία που δημιουργήθηκε Κ.Ο.Κ.

ii. *Κλάση Publication*

Η κλάση αυτή αναπαριστά τις αναρτήσεις που γίνονται στο Facebook. Θεωρούμε πως κάθε πράγμα που μπορεί να δημοσιοποιήσει ο χρήστης ανήκει στην μεγαλύτερη κλάση της ανάρτησης. Άμεσες υποκλάσεις της Publication είναι Event, Comment, Post και Review. Η τελευταίες τρεις υπό-κλάσεις αφορούν αναρτήσεις με σώμα κειμένου που κάνει ένα αντικείμενο της κλάσης Entity, όπως σχόλια, ανάρτηση και αξιολογήσεις. Η υπό-κλάση Event αφορά τις εκδηλώσεις που μπορεί να κάνει κανείς στο Facebook. Τέλος υπό-κλάση της Publication, είναι και η Media που αφορά αναρτήσεις που περιλαμβάνουν εικόνες ή βίντεο. Ο λόγος που διαχωρίζονται είναι για να κρατήσουμε θεωρητικά μέτα-δεδομένα που αφορούν πολυμέσα αλλά για λόγους απλότητας στα πλαίσια της εργασίας κρατήσαμε απλά το όνομα του αρχείου. Όλες οι αναρτήσεις όμως έχουν ένα σώμα κειμένου, δημιουργό, ορατότητα και άλλες ιδιότητες.

iii. *Κλάση Group*

Η κλάση αυτή προφανώς αναπαριστά τις ομάδες του Facebook. Οι ομάδες μπορούν να είναι είτε ιδιωτικές είτε δημόσιες, προφανώς μπορεί να έχει μέλη και φυσικά να έχουν διάφορους χρήστες οι οποίοι έχουν διάφορους ρόλους όπως Admin.

iv. *Κλάση Visibility*

Η κλάση visibility αφορά την ορατότητα που έχουν κάποιες οντότητες του Facebook. Για παράδειγμα οι ομάδες μπορούν να είναι είτε ιδιωτικές είτε δημόσιες. Παράλληλα οι δημοσιεύσεις μπορούν να είναι είτε δημόσιες, ιδιωτικές είτε να εμφανίζονται μόνο σε φίλους του χρήστη που κάνει τη κάθε ανάρτηση. Προφανώς δεν υπάρχουν ιδιωτικά προφίλ με την ίδια έννοια ή σελίδες.

v. *Κλάση Product*

Η κλάση αυτή αναπαριστά τα αντικείμενα που μπορεί να πουλήσει κανείς στο Marketplace. Ένα προϊόν μπορεί να ανήκει σε κάποια κατηγορία που ουσιαστικά το περιγράφει. Τα προϊόντα όμως δεν πωλούνται από χρήστες, ένας χρήστης γίνεται πωλητής και μετά το πουλάει ως πωλητής, έχει δηλαδή έμμεση σχέση με ένα προϊόν. Η υλοποίηση αυτή έγινε με αυτόν τον τρόπο διότι θέλαμε να δώσουμε την

δυνατότητα να γίνεται αξιολόγηση του πωλητή αλλά προφανώς δεν μπορεί να γίνει αξιολόγηση ενός χρήστη.

vi. *Κλάση Seller*

Η κλάση αυτή απλά μετατρέπει υπάρχοντες χρήστες σε πωλητές.

vii. *Κλάση Category*

Η κλάση `Category` έχει μερικές ενδεικτικές κατηγορίες για τα πλαίσια της εργασίας που χρησιμοποιούνται για να δοθούν στα προϊόντα του Marketplace.

viii. *Κλάση Tag*

Η κλάση αναπαριστά τα tags που μπορεί να κάνει ένας χρήστη σε σχόλια, media και posts. Δεχόμαστε λόγω των περιορισμών που έχουμε στη γλώσσα ότι μπορούν πολλοί χρήστες να κάνουν το ίδιο tag που δεν ισχύει προφανώς στο πραγματικό Facebook.

ix. *Κλάση Marketplace*

Η κλάση Marketplace αναπαριστά το Marketplace που υπάρχει στο Facebook το οποίο περιέχει προϊόντα.

3. Ιδιότητες

Οι ιδιότητες που δημιουργήσαμε αναλύονται στο παρακάτω κομμάτι. Αντιμετωπίσαμε κάποιους περιορισμούς από τη γλώσσα και κάποιες από αυτές που θα έπρεπε να είναι αμφίδρομες δεν είναι. Επίσης πολλές ιδιότητες για να μπορέσουν να υλοποιηθούν σωστά έσπασαν σε δύο πχ ένας χρήστης κάνει ένα σχόλιο, ένα σχόλιο έχει δικό του σώμα, το ίδιο σχόλιο περιέχει ένα tag κτλ. Τελευταία παρατήρηση, αξιοποιήσαμε πλήρως τις υπό-ιδιότητες διότι είχαμε θέμα όταν βάζαμε στο domain 2 διαφορετικές κλάσεις, και φυσικά αυτός ο περιορισμός έπαιξε καταλυτικό ρόλο στην δημιουργία των (υπό)κλάσεων της υλοποίησης μας.

i. *Ιδιότητα κλάσης Entity*

Η κλάση Entity έχει ιδιότητες που παίρνουν τιμές και από απλούς τύπους αλλά και από άλλα αντικείμενα. Οι πρώτες, που παίρνει απευθείας η κλάση Entity, είναι αυτές που αφορούν τις εικόνες του προφίλ (εξωφύλλου και προφίλ) και είναι οι `Entity_Cover` και η `PFP`. Η ιδιότητα `Entity_Cover` είναι υπό-ιδιότητα της κλάσης `Cover` που αφορά τα εξώφυλλα που μπορούν να βάλουν οι οντότητες: `Entity`, `Group` και `Event`. Αφορούν επίσης την τοποθεσία στην οποία έχει δηλώσει ότι είναι η βάση μίας οντότητας: `Entity_Location`, που είναι υπό-κλάση της `Location` γιατί τοποθεσία πέρα από τα Entities μπορούν να έχουν και όλες οι αναρτήσεις. Όσον αφορά τις κοινές (και για τη `User` και `Page`) που παίρνουν τιμές από άλλα αντικείμενα είναι η λίστα με τις οντότητες που ακολουθούν, η `Follow`. Ουσιαστικά σε κάθε χρήστη και σελίδα αντιστοιχεί μία λίστα όλων των χρηστών και σελίδων

που ακολουθεί. Τέλος έχουμε τις ιδιότητες που συνδέουν ένα Entity με μία σελίδα, ως Admin/ Moderator/ Advertiser/ Editor/ Analyst οι οποίες είναι αντίστοιχα Page_Admin/Advertiser/Analyst/Editor/Moderator. Προφανώς όμως υπάρχουν και πολλές διαφορετικές ιδιότητες μεταξύ των χρηστών και των σελίδων. Ιδιότητες με τιμές απλούς τύπους : για τους χρήστες υπάρχουν η ιδιότητα User_Name και όλες οι υποκλάσεις τις η οποίες αφορούν το First/Last_Name του χρήστη και την προφορά του κάθε ονόματος First/Last_Name_Pronunciation. Έχουμε δώσει την δυνατότητα στον χρήστη να δηλώνει και το pronoun του ως string γιατί θα ήταν πλεονασμός να δηλώσουμε όλα τα υπαρκτά pronouns. Επίσης ο χρήστης μπορεί να συνδεθεί με γλώσσες που γνωρίζει τις οποίες δίνει ως string μέσω της Language ιδιότητας. Τέλος για τον χρήστη υπάρχει και η ιδιότητα που τον συνδέει με την ημερομηνία γέννησης του η Birth_Date. Στις ιδιότητες του χρήστη όπου η τιμή είναι αντικείμενα έχουμε τη λίστα των φίλων, που είναι ίδια η λογική με την λίστα των οντοτήτων που ακολουθεί μία οντότητα, Friend. Τέλος η σύνδεση του χρήστη με μία σελίδα ως δημιουργό της σελίδας, User_Created_Page. Ιδιότητες με τιμές απλούς τύπους : για τους σελίδες είναι το όνομα της σελίδας και η ημερομηνία που δημιουργήθηκε η σελίδα, Page_Name και Page_Creation_Date, το page creation date είναι υπό-ιδιότητα του Creation_Date που αφορά γενικότερα ημερομηνίες δημιουργίας και έχει ως υπό-ιδιότητες ημερομηνία δημιουργίας για ομάδες, περιεχόμενο και προϊόντα. Τέλος, στις ιδιότητες σελίδας όπου η τιμή είναι άλλα αντικείμενα έχουμε την ιδιότητα Page_Likes μίας σελίδας που αποτελεί μία λίστα με όλους τους χρήστες που δήλωσαν ότι τους αρέσει η σελίδα.

ii. *Ιδιότητες κλάσης Publication*

Η κλάση Publication έχει ιδιότητες που παίρνουν τιμές και από απλούς τύπους αλλά και από άλλα αντικείμενα. Οι ιδιότητες που παίρνουν απλούς τύπους και αφορούν γενικά την κλάση Publication , δηλαδή ιδιότητες που βρίσκονται σε κάθε υπό-κλάση της είναι οι : Content_Created_Date η οποία παίρνει ως τιμή την ημερομηνία που αναρτήθηκε το συγκεκριμένο Publication. Η ιδιότητα Publication_Body παίρνει μια τιμή string η οποία αποτελεί την περιγραφή για το συγκεκριμένο Publication. Οι ιδιότητες που παίρνουν ως τιμή αντικείμενα άλλης κλάσης και είναι κοινά στην κλάση Publication είναι οι εξής: Η Content_Visibility η οποία παίρνει σαν τιμή ένα αντικείμενο της κλάσης visibility δηλαδή των υπό-κλάσεων της Public & Private ώστε να δηλώσουμε ένα αντικείμενο της κλάσης Publication_ είναι ορατό για όλους (public) ή όχι (private). Οι Publication_Created_by Publication_Likes παίρνουν ως τιμή αντικείμενα από την κλάση Entity και τις υποκλάσεις (User

και Page) και δηλώνουν ποιος δημιούργησε αυτό το Publication και ποιοι έχουν κάνει like σε αυτό αντίστοιχα. Για κάθε υποκλάση της Publication υπάρχουν όμως και ξεχωριστές ιδιότητες. Αρχικά, για την κλάση Comment έχουμε την ιδιότητα Comment_By όπου δηλώνουμε ποιος έχει κάνει το συγκεκριμένο σχόλιο. Ένα σχόλιο μπορεί να πραγματοποιηθεί από έναν User ή ένα Page. Η ιδιότητα Comment_Contains_Tag παίρνει σαν τιμή ένα αντικείμενο της κλάσης Tag και δηλώνει αν το σχόλιο αυτό έχει αυτό το tag. Η Reply ιδιότητα παίρνει σαν τιμή επίσης ένα αντικείμενο της κλάσης Comment και αναπαριστά αν ένα σχόλιο είναι απάντηση σε κάποιο άλλο. Για την κλάση Event έχουμε τις εξής ιδιότητες: Η Event_Cover_Art η οποία παίρνει ως τιμή ένα αντικείμενο της κλάσης Image και αναπαριστά το εξώφυλλο που έχει ένα Event (σαν αφίσα). Οι ιδιότητες Event_Date και Event_Name παίρνουν ως τιμές απλούς τύπους ημερομηνία και string και δηλώνουν την ημερομηνία που είναι προγραμματισμένο το Event να πραγματοποιηθεί και το όνομα του αντίστοιχα. Η κλάση Post έχει μία μόνο επιπλέον ιδιότητα την Post_Contains_Tag η οποία παίρνει σαν τιμή ένα αντικείμενο της κλάσης Tag και δηλώνει αν το Post αυτό έχει αυτό το tag. Η κλάση Review έχει τις ιδιότητες Reviews_Page που παίρνει σαν τιμή ένα αντικείμενο της κλάσης Page και δηλώνει ποια σελίδα κάνει Review και την Reviews_Seller η οποία παίρνει σαν τιμή ένα αντικείμενο της κλάσης Seller και κάνει ουσιαστικά Review έναν πωλητή. Οι Reviews_Page και Reviews_Seller είναι υποκλάσεις της Reviews καθώς και οι δυο έχουν κοινή λειτουργία αλλά αναφέρονται σε διαφορετικές κλάσεις. Τέλος η κλάση Media έχει 2 ακόμα ιδιότητες την File_Name η οποία παίρνει σαν τιμή ένα string ως το όνομα του αρχείου και την Media_Contains_Tag η οποία παίρνει σαν τιμή ένα αντικείμενο της κλάσης Tag και δηλώνει αν το βίντεο ή η φωτογραφία αυτό έχει αυτό το tag. Η κλάση αυτή έχει δυο υποκλάσεις την Image και Video οι οποίες δεν έχουν κάποια επιπλέον ιδιότητα από αυτές της Media. Τέλος παρατηρούμε ότι οι Comment_Contains_Tag, Post_Contains_Tag και Media_Contains_Tag έχουν παρόμοια λειτουργία και γι' αυτόν τον λόγο δημιουργούμε μια υπερ-κλάση Contains_Tag και τις βάζουμε ως υποκλάσεις σε αυτήν.

iii. *Ιδιότητες κλάσης Group*

Η κλάση Group έχει ιδιότητες που παίρνουν τιμές και από απλούς τύπους αλλά και από άλλα αντικείμενα. Οι ιδιότητες με τιμές απλούς τύπους είναι οι Group_Creation_Date και Group_Name παίρνουν σαν τιμές date και string αντίστοιχα και αναπαριστούν την ημερομηνία δημιουργίας του group και το όνομα του. Οι υπόλοιπες ιδιότητες παίρνουν τιμές από άλλα αντικείμενα. Πιο συγκεκριμένα, η

`Group_Admin` αναπαριστά ποιος είναι ο admin ενός γκρουπ και παίρνει τιμή από αντικείμενα της κλάσης `Entity`. Η ιδιότητα `Group_Background_Cover` παίρνει τιμές από την κλάση `Image` και οι ιδιότητες `Group_Created_by` και `Group_Moderator` παίρνουν ως τιμές αντικείμενα από την κλάση `Entity` και αναπαριστούν ποιος δημιούργησε την κλάση και ποιος είναι ο moderator της. Η ιδιότητα `Group_Visibility` δείχνει αν το γκρουπ αυτό είναι private ή public και μπορεί να είναι ορατό από χρήστες και σελίδες στο Facebook και γι' αυτό παίρνει τιμές από την κλάση `Visibility`. Τέλος η ιδιότητα `Group_Member` αναπαριστά ποιοι χρήστες και σελίδες είναι μέλη αυτού του γκρούπ και παίρνει τιμές από την κλάση `Entity`.

iv. *Ιδιότητες κλάσης* `Product`

Η κλάση `Product` έχει για ιδιότητες που παίρνουν απλές τιμές τη τιμή του προϊόντος `Price`, η ημερομηνία που αναρτήθηκε η αγγελία του `Product_Creation_Date`, η περιγραφή του προϊόντος `Product_Description` και ο τίτλος της αγγελίας, δηλαδή το όνομα του προϊόντος `Product_Name`. Από αυτές η πρώτη είναι αριθμός, η δεύτερη είναι ημερομηνία και οι υπόλοιπες είναι strings. Για ιδιότητες που παίρνουν τιμές από άλλα αντικείμενα έχει τις `Marketplace_Product` που ενώνει ένα προϊόν με το Marketplace, το `Product_Category` που υποδηλώνει τη κατηγορία στην οποία ανήκει ένα προϊόν, στο ειδικότερο της είδος ή και όχι και τις εικόνες που έχει ανεβάσει ένας πωλητής για το προϊόν που έχει ανεβάσει, `Product_Images`.

v. *Ιδιότητες κλάσης* `Seller`

Όπως έχει αναφερθεί και πιο πάνω η κλάση αυτή απλά μετατρέπει υπάρχοντες χρήστες σε πωλητές. Οπότε η μοναδική ιδιότητα που έχει είναι η `isEntity` η οποία παίρνει τιμές από την κλάση `Entity`, δηλαδή από άλλα αντικείμενα.

vi. *Ιδιότητα κλάσης* `Tag`

Η κλάση αυτή έχει δύο ιδιότητες που παίρνουν σαν τιμές αντικείμενα από άλλες κλάσεις. Αυτές είναι οι `Tag_By_Entity` και `Tagging`. Η πρώτη δείχνει από ποιον έγινε το Tag και η δεύτερη ποιον κάνει Tag. Και οι δυο έχουν σαν τιμή ένα αντικείμενο της κλάσης `Entity`.

4. Queries

Τα 3 Queries που δημιουργήσαμε ώστε να τρέχουν πάνω στην οντολογία είναι τα εξής:

vii. *Query No.1*

Το query αυτό βρίσκει το group που έχει τα περισσότερα μέλη και εμφανίζει το όνομα του μαζί με τα ονόματα των μελών του. Το συγκεκριμένο query είναι εμφωλευμένο. Ξεκινάει βρίσκοντας σε ποια group ανήκει ο κάθε χρήστης. Στην συνέχεια βρίσκει το group που έχει τους περισσότερους χρήστες ως μέλη του και στην συνέχεια με την Filter() εμφανίζει μόνο τα ονόματα των χρηστών που ανήκουν στο group με τα περισσότερα μέλη.

```
PREFIX rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:     <http://www.w3.org/2000/01/rdf-schema#>
PREFIX Facebook: <http://example.org/Facebook#>

SELECT ?name ?group_name
WHERE {
  ?x Facebook:Group_Name ?group_name.
  ?x Facebook:Members ?members.
  ?members Facebook:First_Name ?name
  {
    SELECT ?Gn (COUNT (?members) AS ?number_of_members)
    WHERE
    {
      ?x Facebook:Group_Name ?Gn.
      ?x Facebook:Members ?members.
    }
    GROUP BY ?Gn
    ORDER BY DESC ( ?number_of_members )
    LIMIT 1
  }
  FILTER (?group_name = ?Gn)
}
ORDER BY ?group_name
```

Τα αποτελέσματα σύμφωνα με τα instances μας ήταν τα παρακάτω:

name	group_name
 Erofil	 forest hikers
 Kostas	 forest hikers
 Konstantina	 forest hikers
 Giorgos	 forest hikers
 Dimitris	 forest hikers
 Maria	 forest hikers

viii. *Query No.2*

Το παρακάτω query αναζητά το δυνητικό κέρδος άνω των διακοσίων ευρώ των πωλητών από την πώληση των προϊόντων τεχνολογίας. Βρίσκει ουσιαστικά τα προϊόντα που η κατηγορία τους είναι η υπό-κλάση Electronics και αθροίζει την τιμή τους. Γίνεται γκρουπάρισμα με βάση τις οντότητες και εμφανίζονται μόνο αυτές που έχουν τελικό ποσό μεγαλύτερο από 200. Για να εμφανίζονται όλα τα ποσά με τον ίδιο τρόπο απλά το πολλαπλασιάσαμε το 1.0 και η μορφή είναι σε όλα πχ. 160.0.

```
PREFIX rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:     <http://www.w3.org/2000/01/rdf-schema#>
PREFIX fn:       <http://www.w3.org/2005/xpath-functions#>
PREFIX xsd:      <http://www.w3.org/2001/XMLSchema#>
PREFIX Facebook: <http://example.org/Facebook#>

SELECT ?entity (SUM (?amount*1.0) AS ?total_amount)
WHERE {
    ?type rdfs:subClassOf Facebook:Electronics.
    ?category rdf:type ?type.
    ?product Facebook:Product_Category ?category.
    ?product Facebook:Price ?amount.
    ?product Facebook:Interchange ?seller.
    ?seller Facebook:isEntity ?entity.
    ?seller rdf:type Facebook:Seller
}
GROUP BY ?entity
HAVING (?total_amount > 200)
```

Τα αποτελέσματα σύμφωνα με τα instances μας ήταν τα παρακάτω

[entity]	total_amount
Facebook:User_7	1500.0















ix. *Query No.3*

Το τρίτο query ξεκινάει βρίσκοντας τους admin της κάθε σελίδας και σε πόσες σελίδες έχουν τον ρόλο του admin μετά ενώνει το όνομα και το επίθετο τους και τα εμφανίζει με φθίνουσα σειρά, από τον χρήστη που είναι admin στις περισσότερες σελίδες.

```
PREFIX rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:     <http://www.w3.org/2000/01/rdf-schema#>
PREFIX fn:       <http://www.w3.org/2005/xpath-functions#>
PREFIX Facebook: <http://example.org/Facebook#>

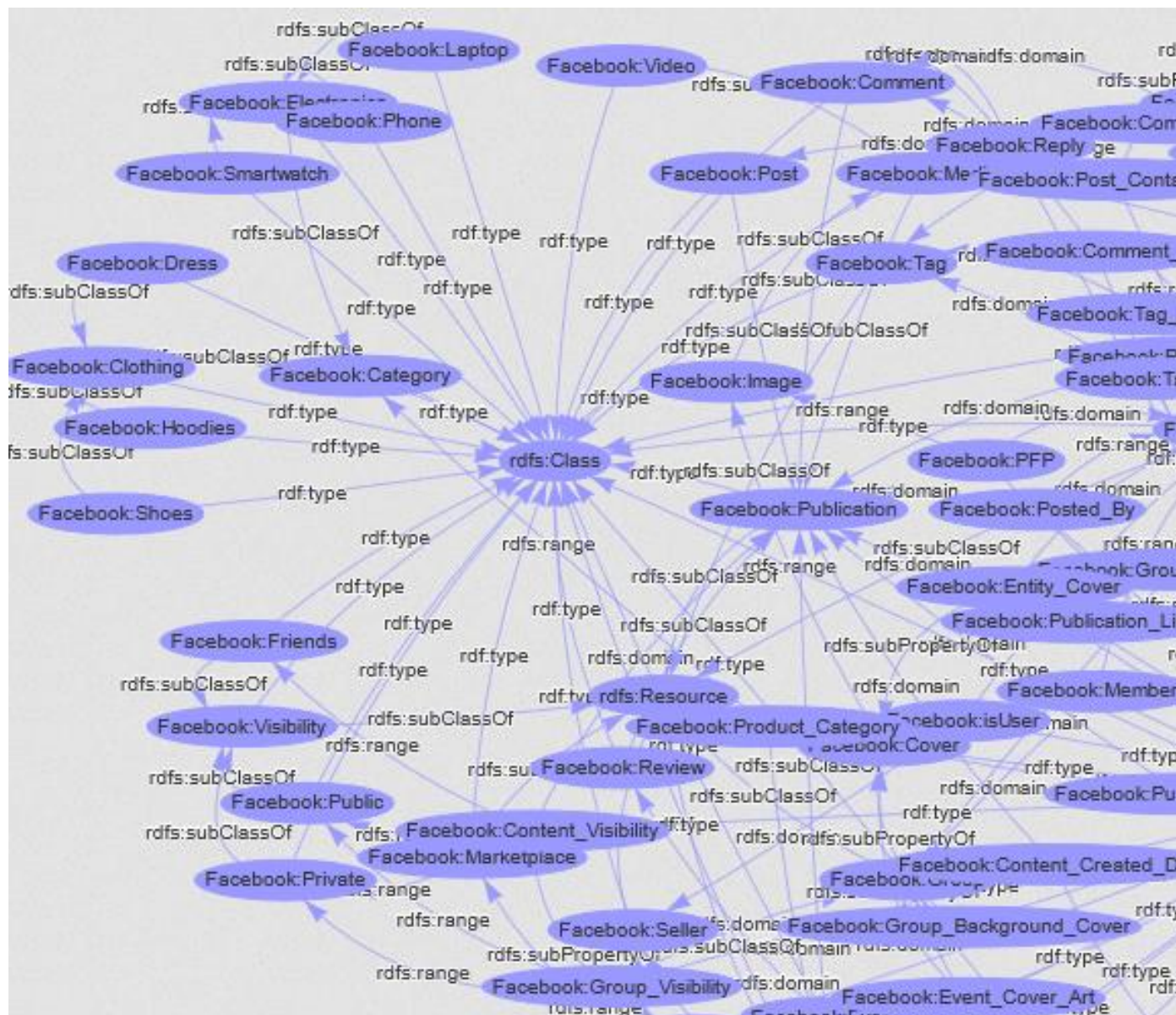
SELECT DISTINCT(fn:concat (?first, " ", ?last) AS ?full_name) (COUNT (?page) AS ?count)
WHERE
{
    ?user Facebook:Last_Name ?last.
    ?user Facebook:First_Name ?first.
    ?page Facebook:Page_Admin ?user.
    ?page rdf:type Facebook:Page
}
GROUP BY ?first ?last
ORDER BY DESC (?count)
```

Τα αποτελέσματα σύμφωνα με τα instances μας ήταν τα παρακάτω:

full_name	count
 Dimitris Papakostantinou	 3
 Giorgos Blake	 2
 Eleni Mnd	 1
 Erofilis Konsta	 1
 Ilias Konstas	 1
 Kostas Giotas	 1
 Nikos Mandanas	 1

5. Γραφική Αναπαράσταση

Για γραφική αναπαράσταση επιλέξαμε ένα μικρό κομμάτι του γραφήματος που δημιουργήσαμε με τη βοήθεια του [RDF Playground](#).



Μέρος Δεύτερο : OWL

Στο δεύτερο μέρος της εργασίας η επέκταση του προαναφερθέντος μοντέλου με τη σημασιολογία και τις δομές που μας παρέχει η OWL.

1. Κλάσεις

Στην ενότητα αυτή θα αναλύσουμε τις νέες κλάσεις που προσθέσαμε στη μοντελοποίηση μας με τη χρήση των restrictions, intersection, union και disjoint της OWL

i. *Union*

Η κλάσεις που αφορούν την ένωση άλλων κλάσεων είναι η κλάση `Entity` η οποία είναι η ένωση των κλάσεων `User` και `Page`, διότι όπως αναλύσαμε και παραπάνω μία Οντότητα-`Entity` είναι οι χρήστες και οι σελίδες λόγω των πολλών κοινών χαρακτηριστικών τους και της κοινής τους συμπεριφοράς στην μοντελοποίηση. Η κλάση `Media` που αφορά την ένωση των κλάσεων `Image` και `Video` για τους αντίστοιχους λόγους.

ii. *Disjoint*

Προφανώς πολλές κλάσεις έχουν αντικείμενα τα οποία είναι διαφορετικά μεταξύ τους. Αυτές είναι οι `Image` και `Video`, γιατί μία εικόνα δεν μπορεί να και βίντεο. Οι `User` και `Page` είναι επίσης διάφορες, οι `Clothing` και `Electronics`, οι `Dress Hoodies & Shoes` και τέλος οι `Laptop Phone & Smartwatch`. Επιπλέον έχουμε τις `Group` και `Entity`, τις `Public` και `Private`, και όλες τις υποκλάσεις της κλάσης `Publication`, πιο συγκεκριμένα: `Review`, `Media`, `Post`, `Comment` και `Event`. Έχουμε και τις `Marketplace Entity` και `Group` και `Category Marketplace`

iii. *Intersection of*

Οι κλάσεις που αποτελούνται από την τομή άλλων κλάσεων είναι η κλάση `Page_Managers` και είναι οι τομή όσων χρηστών/σελίδων είναι `Admin` αλλά και `Moderator` σε μία σελίδα-όχι απαραίτητα στην ίδια. Οφείλουμε να αναφέρουμε ότι στην πραγματική εφαρμογή δεν γνωρίζουμε την ύπαρξη κάποια τέτοιας ιδιότητας αλλά την προσθέσαμε για την ικανοποίηση των προϋποθέσεων την εργασίας. Στην συνέχεια έχουμε την κλάση `User_Seller` που παίρνει τιμές από τις κλάσεις `User` και `Seller` και αποτελείται από τα αντικείμενα που ανήκουν και στις 2 αυτές κλάσεις. Έχουμε επίσης

τις κλάσεις `Not_Tagged_Media` και `Unpurchased_product` που δημιουργούνται με τον ίδιο τρόπο: Παίρνουμε το συμπλήρωμα της κλάσης `Tagged_Media` και `Purchased_Product` αντίστοιχα και βρίσκουμε την τομή τους με τις κλάσεις `Media` και `Product` αντίστοιχα ώστε να επιστραφούν όλα τα αντικείμενα τύπου `Media` που δεν έχουν κάποιο `Tag` και όλα τα αντικείμενα τύπου `Product` που δεν έχουν αγορασθεί. Με τον ίδιο τρόπο (δηλαδή με το συμπλήρωμα) δημιουργούμε και τις κλάσεις `Not_Tagged_Comment` και `Not_Tagged_Post`.

iv. *Restriction*

Οι κλάσεις που δημιουργήθηκαν από περιορισμούς διαχωρίζονται σε δύο κατηγορίες: τις *equivalent* και τις *subclass*.

Πρώτα θα αναφερθούμε στις *equivalent* που είναι κλάσεις με ίδια ακριβώς χαρακτηριστικά με τη υπέρ-κλάση τους. Αυτές είναι οι κλάσεις `Private/Public_Post`, `Private/Public_Event`, `Private/Public_Image`, `Private/Public_Video` και `Private/Public_Group`. Όλες οι κλάσεις βασίζονται στην ίδια βασική ιδέα: την έκφραση των υπερ-κλάσεων τους με βάση την ορατότητα που έχουν. Στην δική μας μοντελοποίηση έχουμε δύο ειδών ορατότητα την `Public` και `Private` και σε αυτές βασίζονται οι παραπάνω κλάσεις. Πχ. ένα `Post` με `Visibility: Public` θα ανήκει και στην κλάση `Public_Post`.

Στα *restrictions* τύπου *subclass* έχουμε τις κλάσεις `Tagged_Post`, διότι περιέχει το `Tag` που δεν περιέχει υποχρεωτικά οποιοδήποτε `Post`. Έχουμε επίσης την κλάση `Purchased_Product` που δεν περιέχει υποχρεωτικά όλα τα αντικείμενα της κλάσης `Product` αλλά μπορεί να περιέχει ένα σύνολο από αυτά, και παίρνει αντικείμενα τα οποία έχει αγοράσει ένας `User`. Το *restriction* μπαίνει πάνω στην ιδιότητα `Buys` που έχει ως `domain` και `range` έναν `User` & `Product` αντίστοιχα. Στην συνέχεια έχουμε την κλάση `Tagged_Media` και `Tagged_Comment` που μοιάζουν με την `Tagged_Post` που εξηγήσαμε πιο πάνω. Έχουν την ίδια λογική απλά χρησιμοποιούμε διαφορετικό `object property` για να κάνουμε το *restriction*. Στην συνέχεια έχουμε 3 κλάσεις την `Gold_Rank`, `Silver_Rank` και `Bronze_Rank` που είναι υποκλάσεις της `Entity_Rank` και έχουν την εξής λειτουργία: Για την `Gold_Rank` έχουμε τον περιορισμό ότι αντικείμενα της είναι όσα στιγμιότυπα της κλάσης `Entity` έχουν δημιουργήσει ένα αντικείμενο της κλάσης `Publication` το λιγότερο 10 φορές. Με απλά λόγια όποια σελίδα ή χρήστης έχει κάνει πάνω από 10 `post` για παράδειγμα. Η `Silver_Rank` & `Bronze_Rank` λειτουργούν με τον ίδιο τρόπο απλά αλλάζει το *restriction* σε 7 και 5 αντίστοιχα το λιγότερο

αντικείμενα της κλάσης *Publication*. Γνωρίζουμε ότι κάτι τέτοιο δεν υπάρχει στο Facebook αλλά για λόγους της εργασίας αποφασίσαμε ότι ταιριάζει στην οντολογία μας. Στην συνέχεια έχουμε την κλάση *Suspicious_User* που περιλαμβάνει τους χρήστες οι οποίοι έχουν ακριβώς 0 φίλους, οπότε το *Restriction* μπαίνει στην ιδιότητα *Friend* και έχει συγκεκριμένη τιμή ίσον 0. Μετά έχουμε την κλάση *Popular_Group* όπου σε αυτήν ανήκουν όλα τα αντικείμενα τύπου *Group* που έχουν μέλη τουλάχιστον 20. Επειδή η οντολογία μας προσομοιώνει το Facebook σε ένα παράδειγμα που θα αντιπροσώπευε το κανονικό Facebook το νούμερο αυτό θα ήταν πολύ μεγαλύτερο, αλλά τώρα για να έχει νόημα στην οντολογία μας χρησιμοποιείται το 20. Τέλος η τελευταία κλάση που δημιουργήσαμε με περιορισμό είναι η *Inactive_Page* όπου περιέχει όλες τις σελίδες που έχουν κάνει το πολύ 1 post και θεωρούμε ότι δεν είναι ενεργές.

2. Ιδιότητες

Στην ενότητα αυτή θα αναλύσουμε τις νέες ιδιότητες που προστέθηκαν και τις

v. *Symmetric*

Συμμετρικές είναι οι ιδιότητες που αναπαριστούν μία αμφίδρομη σχέση μεταξύ *subject* και *object*. Στην μοντελοποίηση μας τέτοιες αμφίδρομες σχέσεις αναπαριστούν οι ιδιότητες *Friend*, *Group_Admin*, *Group_Moderator*, *Page_Admin*, *Page_Moderator*, *Page_Analyst* και *Page_Advertiser*.

vi. *Inverse of*

Όταν μια ιδιότητα είναι *inverse of* κάποιας άλλης, αυτό σημαίνει ότι αντιστρέφονται οι θέσεις του *subject* και *predicate* στις ιδιότητες αυτές. Στην οντολογία μας έχουμε τις εξής αντίστροφες ιδιότητες: *buys-is_bought* στις οποίες, στην πρώτη *domain* είναι ένας χρήστης και *range* είναι ένα αντικείμενο της κλάσης *Product*, ενώ στην δεύτερη είναι αντίστροφα. Οι ιδιότητες *isSeller-isEntity* είναι επίσης αντίστροφες καθώς η πρώτη έχει *domain* ένα αντικείμενο της κλάσης *Seller* και *range* ένα αντικείμενο της κλάσης *Entity* και η δεύτερη το αντίστροφο και δηλώνει ότι ένας πωλητής είναι ένας χρήστης. Συνδέει δηλαδή τους πωλητές με τους χρήστες. Στην συνέχεια έχουμε τις αντίστροφες ιδιότητες *sells-interchange* όπου η πρώτη έχει *domain* ένα αντικείμενο της κλάσης *Seller* και *range* ένα αντικείμενο της κλάσης *Product* και η δεύτερη το αντίστροφο και δηλώνει ποια προϊόντα πουλάει

ένας seller και η αντίστροφη ιδιότητα δηλώνει ποια προϊόντα πωλούνται από έναν seller. Οι ιδιότητες `comments-comment_by` είναι αντίστροφες, με την πρώτη να έχει domain ένα αντικείμενο της κλάσης `Entity` και range ένα αντικείμενο της κλάσης `Comment` και δηλώνει μια οντότητα που κάνει ένα comment, ενώ η δεύτερη το αντίστροφο domain/range και δηλώνει από ποια οντότητα έγινε το comment. Τέλος, έχουμε τις ιδιότητες `entity_is_member`, `members` όπου η πρώτη έχει domain ένα αντικείμενο της κλάσης `Entity` και range ένα αντικείμενο της κλάσης `Group` και η δεύτερη το αντίστροφο και δηλώνει τα αντικείμενα που είναι μέλη ενός Group.

vii. *Transitive*

Στις μεταβατικές ιδιότητες έχουμε ότι αν $?a : p ?b$ και $?b : p : c$ τότε $?a : p : c$. Οπότε στις ιδιότητες `Page_Is_Older`, `User_Is_Older`, `Group_Has_More_members`, `User_Has_More_Friends` και `Product_is_Cheaper_Than` αν έχουμε για παράδειγμα $:page1 :Page_Is_Older :page2$ & $:page2 :Page_Is_Older :page3$ τότε λόγω του ότι η ιδιότητα είναι μεταβατική θα βγει ότι $:page1 :Page_Is_older :page3$. Και με αντίστοιχο τρόπο λειτουργούν και οι υπόλοιπες 4 Transitive ιδιότητες. Οι μεταβατικές ιδιότητες που υλοποιήσαμε για να έχουν ακριβή αποτελέσματα αλλά και οι μεταβατικές σχέσεις να επιστρέφουν το σωστό αποτέλεσμα θα πρέπει να χρησιμοποιηθεί σε συνδιασμό με ένα Query ώστε να ελέγχουμε και τις τιμές των data properities, δηλαδή να συγκρίνουμε την ημερομηνία δημιουργίας των 2 αντικειμένων ή ότιδηποτε άλλο είναι απαραίτητο. Οι `Page_Is_Older`, `User_Is_Older` ελέγχουν την ημερομηνία δημιουργίας του χρήστη ή της σελίδας. Η `Group_Has_More_members` και `User_Has_More_Friends` ελέγχει το πλήθος το μελών ή των φίλων που έχει ένα Group ή ένας χρήστης αντίστοιχα.

viii. *Functional Property*

Εδώ, βάλαμε στο data property `Price` να είναι functional, γιατί ένα προϊόν έχει 1 μόνο τιμή, αλλά μπορεί πολλά προϊόντα να έχουν την ίδια τιμή. Ένα ακόμα Functional property που βάλαμε είναι στο object property `Marketplace_Product`, και δηλώνει ότι ένα αντικείμενο τύπου `Product` μπορεί να ανήκει μόνο σε ένα Marketplace αλλά ένα Marketplace μπορεί να περιλαμβάνει πολλά αντικείμενα τύπου `Product`.

ix. *Inverse Functional Property*

Το `objectproperty Comment` είναι *inverse functional property*, γιατί έχει ως `domain: Entity` και `range: Comment` και δηλώνει ότι ένα αντικείμενο τύπου `Entity` μπορεί να έχει κάνει πολλά `comments`, ενώ ένα `comment` μπορεί να γίνει μόνο από ένα αντικείμενο της κλάσης `Entity`.

3. Ασυνέπειες

Στο σημείο αυτό θα αναλύσουμε κάποιες από τις ασυνέπειες που αντιμετωπίσαμε κατά της υλοποίηση και χρειάστηκε να αλλάξουμε την αρχική μας ιδέα ή να διορθώσουμε τυχόν λάθη που κάναμε και τεχνητές ασυνέπειες που μας ζητήθηκαν στα πλαίσια της εργασίας.

x. *Ασυνέπειες που αντιμετωπίσαμε κατά την υλοποίηση*


Κατά την υλοποίηση αντιμετωπίσαμε κάποιες ασυνέπειες στην προσπάθεια μας να κάνουμε τις γλώσσες που έχει δηλώσει ο χρήστης στο προφίλ του ότι γνωρίζει ως *Functional Property* κάτι τέτοιο όμως δεν θα μπορούσε να γίνει διότι ο χρήστης μπορεί να γνωρίζει παραπάνω από μία γλώσσες. Επίσης είχαμε προσπαθήσει να δηλώσουμε την κλάση `Publication` ως την ένωση των υπό-κλάσεων της `Post`, `Video`, `Image` και `Event` κάτι το οποίο είναι λάθος.

xi. *Τεχνητές ασυνέπειες*

Ασυνέπειες που δημιουργήσαμε εμείς για το πλαίσιο της εργασίας είναι οι παρακάτω:

1. `Tagged_Post` / `Not_Tagged_Post`






Αν πάμε να ορίσουμε ως *equivalent* την κλάση `Tagged_Post` με την **not** `Not_Tagged_Post` δηλαδή το συμπλήρωμα της `Not_Tagged_Post` θα πάρουμε ασυνέπεια καθώς η εντολή **not** `Not_Tagged_Post` θα φέρει πίσω όλα όσα δεν ανήκουν στην κλάση `Not_Tagged_Post` ακόμα και αυτά που δεν είναι `Post`. Για παράδειγμα θα επιστρέψει ένα αντικείμενο τύπου `Image` και θα πάει να το βάλει στην κλάση `Tagged_Post` όμως εμείς έχουμε ήδη πει ότι αυτή παίρνει αντικείμενα τύπου `Post` και επιπλέον η `Post` είναι *disjoint* με την `Image`.

 Inconsistent ontology explanation

☒ Show regular justifications ☒ All justifications
☐ Show laconic justifications ☐ Limit justifications to

Explanation 1 ☐ Display laconic explanation

Explanation for: `owl:Thing SubClassOf owl:Nothing`

1)	DisjointClasses: <code>Comment</code> , <code>Event</code> , <code>Image</code> , <code>Post</code> , <code>Review</code> , <code>Video</code>	In ALL other justifications	
2)	Tagged_Post EquivalentTo not (<code>Not_Tagged_Post</code>)	In ALL other justifications	
3)	<code>Image_17 Type</code> <code>Image</code>	In 3 other justifications	
4)	<code>Not_Tagged_Post SubClassOf</code> <code>Post</code>	In 122 other justifications	
5)	<code>Tagged_Post SubClassOf</code> <code>Post</code>	In 123 other justifications	

2. Tagged_Post_with_Image

Αν δημιουργήσουμε μια κλάση που να περιέχει όλα τα Post που έχουν κάποιο Tag και ένα αντικείμενο της κλάσης Image, δηλαδή αν κάνω την τομή των κλάσεων Tagged_Post, Image αυτό δημιουργεί ασυνέπεια καθώς οι κλάσεις Post (όπου είναι υπερκλάση της Tagged_Post) και η Image είναι disjoint:

Explanation for Tagged_post_with_Image EquivalentTo owl:Nothing

☒ Show regular justifications ☒ All justifications
☐ Show laconic justifications ☐ Limit justifications to 2

Explanation 1 ☐ Display laconic explanation

Explanation for: Tagged_post_with_Image EquivalentTo owl:Nothing

1)	Tagged_post_with_Image EquivalentTo Image and Tagged_Post	In ALL other justifications	?
2)	Tagged_Post SubClassOf Post	In NO other justifications	?
3)	DisjointClasses: Comment, Event, Image, Post, Review, Video	In ALL other justifications	?

3. Image individual

Αν πάμε να προσθέσουμε σε ένα instance τύπου Image ότι είναι και instance τύπου Video, τότε αυτό θα βγάλει ασυνέπεια καθώς έχουμε ήδη ορίσει ότι οι κλάσεις Image και Video είναι Disjoint μεταξύ τους. Την ίδια ασυνέπεια θα βγάλει αντίστοιχα και σε άλλες αντίστοιχες περιπτώσεις

Inconsistent ontology explanation

☒ Show regular justifications ☒ All justifications
☐ Show laconic justifications ☐ Limit justifications to 2

Explanation 1 ☐ Display laconic explanation

Explanation for: owl:Thing SubClassOf owl:Nothing

1)	Image_3 Type Image	In NO other justifications	?
2)	DisjointClasses: Comment, Event, Image, Post, Review, Video	In ALL other justifications	?
3)	Image_3 Type Video	In ALL other justifications	?

Inferred

Explanation 2 ☐ Display laconic explanation

4. Συνεπής Οντολογία

Η οντολογία μας δεν παρουσιάζει καμία ασυνέπεια.

```
INFO 16:23:07 ----- Running Reasoner -----
INFO 16:23:07 Pre-computing inferences:
INFO 16:23:07   - class hierarchy
INFO 16:23:07   - object property hierarchy
INFO 16:23:07   - data property hierarchy
INFO 16:23:07   - class assertions
INFO 16:23:07   - object property assertions
INFO 16:23:07   - same individuals
INFO 16:23:08 Ontologies processed in 1109 ms by Hermit
INFO 16:23:08
```

Show log file Preferences Time stamp Clear log

Reasoner active ☒ Show Inferences

OK