ΤΜΗΜΑ ΑΡΧΕΙΟΝΟΜΙΑΣ, ΒΙΒΛΙΟΘΗΚΟΝΟΜΙΑΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΗΣΗΣ

ΣΧΟΛΗ ΔΙΟΙΚΗΤΙΚΩΝ, ΟΙΚΟΝΟΜΙΚΩΝ ΚΑΙ ΚΟΙΝΩΝΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

DEPARTMENT OF ARCHIVAL, LIBRARY AND INFORMATION STUDIES

Πτυχιακή Εργασία

Αξιοποίηση βιβλιομετρικών δεδομένων από τη βάση αναφορών Scopus για την αξιολόγηση της ερευνητικής δραστηριότητας

Συγγραφέας

Γιώργος Γουργιωτόπουλος (ΑΜ: 14091)

Ιούλιος 2021

Επιβλέπων: Δημήτρης Κουής

Επιτροπή Εξέτασης

1. Ονοματεπώνυμο

DIMITRIOS KOUIS

Digitally signed by DIMITRIOS KOUIS
Date: 2021.07.21 10:09:49 +03'00'

2. Ονοματεπώνυμο

Alexandros Koulouris

Digitally signed by Alexandros Koulouris
Date: 2021.08.04 11:16:23 +03'00'

3. Ονοματεπώνυμο

Eftychia Vraimaki

Digitally signed by Eftychia Vraimaki
Date: 2021.07.21 14:45:06 +03'00'

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΠΤΥΧΙΑΚΗΣ/ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Ο κάτωθι υπογεγραμμένος Γουργιοτόπουλος Γεώργιος, με αριθμό μητρώου 14091 φοιτητής του Πανεπιστημίου Δυτικής Αττικής της Σχολής Διοικητικών, Οικονομικών και Κοινωνικών Επιστημών του Τμήματος Αρχειονομίας, Βιβλιοθηκονομίας και Συστημάτων Πληροφόρησης, δηλώνω υπεύθυνα ότι:

«Είμαι συγγραφέας αυτής της πτυχιακής/διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.

Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Γουργιοτόπουλος Γ.

Ο Δηλών

Ευχαριστίες – Αφιερώσεις

Η παρούσα πτυχιακή αφιερώνεται στο προσωπικό του τμήματος Αρχειονομίας, Βιβλιοθηκονομίας & Συστημάτων Πληροφόρησης, που με την εξαιρετική του δουλειά βοηθάει παιδιά (και όχι μόνο) να εξελιχθούν σε όλα τα επίπεδα και να ακολουθήσουν μια καλή σταδιοδρομία.

Ημερομηνία 01/07/2021

Συγγραφέας

Γουργιωτόπουλος Γιώργος

Περίληψη στα ελληνικά

Ο όγκος των διαθέσιμων πρωτογενών βιβλιομετρικών δεδομένων αυξάνεται συνεχώς λόγω της αποθήκευσης των ψηφιακών τεκμηρίων των επιστημονικών δημοσιευμάτων (άρθρα, ανακοινώσεις, μονογραφίες κ.λπ.) στα διάφορα πληροφοριακά συστήματα και αποθετήρια και τη μετέπειτα συγκεντρωτική ευρετηρίασή τους από τους καταλόγους αναφορών (citation indexes), όπως Scopus, Web of Science και Google Scholar.

Οι κατάλογοι αναφορών προσφέρουν μια σειρά από βιβλιομετρικούς δείκτες αλλά και τη δυνατότητα επεξεργασίας των πρωτογενών δεδομένων από τους ερευνητές για την δημιουργία νέων δεικτών ή την πραγματοποίηση στοχευμένων βιβλιομετρικών μελετών. Η πρόσβαση στα πρωτογενή βιβλιομετρικά δεδομένα είναι ελεύθερη ή συνδρομητική ανάλογα με την υπηρεσία καταλόγου αναφορών.

Στην παρούσα εργασία θα διεξαχθεί βιβλιογραφική ανασκόπηση σε ό, τι αφορά τις δυνατότητες των καταλόγων αναφορών στην παροχή βιβλιομετρικών δεδομένων.

Στη συνέχεια θα επεκταθεί περαιτέρω το πληροφοριακό εργαλείο που διαθέτει το Εργαστήριο Διαχείρισης της Πληροφορίας του τμήματος για την άντληση επιπλέον κατηγοριών δεδομένων από τον κατάλογο αναφορών Scopus με στόχο τη δημιουργία νέων βιβλιομετρικών δεικτών για την αξιολόγηση των ερευνητικών οντοτήτων.

Επιπλέον, θα επιχειρηθεί η συγκέντρωση επιπλέον βιβλιομετρικών δεδομένων με στόχο τον εμπλουτισμό των δυνατοτήτων του εργαλείου. Ως πεδίο εφαρμογής επιλέγονται τα τμήματα και το προσωπικό του Πανεπιστήμιου Δυτικής Αττικής. Η πτυχιακή εργασία θα ολοκληρωθεί με την αξιολόγηση και σχολιασμό των αποτελεσμάτων, την εξαγωγή συμπερασμάτων και προτάσεις για μελλοντική επέκταση.

Λέξεις Κλειδιά: Βάση Αναφορών, Scopus, Προγραμματιστική διεπαφή, Κανονικοποίηση βάσεων, Ανάκτηση πληροφοριών, Βιβλιομετρία, προγραμματισμός, Γλώσσα προγραμματισμού C#

Περίληψη στα αγγλικά

The volume of available bibliometric data is constantly increasing due to the high number of scientific publications (articles, announcements, monographs, etc.) that each year are produced and are deposited in the various information systems and repositories. The main source of bibliometric data is the citation indexes. such as Scopus, Web of Science and Google Scholar.

Citation indexes offer various bibliometric indicators as well as the ability for researchers to process raw data to create new metrics or perform bibliometric studies. Access to primary bibliographic data is provided through an API and is free or it is provided via subscription, depending on the provider.

In the present thesis, a bibliographic review will be conducted regarding the citation indexing capabilities in the provision of bibliometric data.

Then the bibliometric application that has been created in the Information Management Laboratory of the Department of Archives, Libraries and Information Studies (University of West Attica) will be further extended to be able to extract additional categories of data from the Scopus citation index.

In addition, an attempt will be made to collect additional bibliometric data in order to enrich the capabilities of the bibliometric tool. The departments and staff of the University of West Attica are selected as the field of application. The thesis will be completed with the evaluation and commentary of the results, the drawing of conclusions and proposals for future expansion.

Keywords: Citation index, Scopus, Application Programming Interface, Database normalization, Information retrieval, Bibliometrics, programming, C#

Περιεχόμενα

Κεφάλαιο 1. Εισαγωγή

1.1 Πλαίσιο, σκοπός και στόχοι της πτυχιακής εργασίας

Πρώτος στόχος της παρούσας εργασίας είναι η πραγματοποίηση βιβλιογραφικής έρευνας στο αντικείμενο της βιβλιομετρίας και των μεθόδων αξιολόγησης ερευνητικών οντοτήτων (προσώπων, έργων, ακαδημαϊκών μονάδων).

Επίσης, ως στόχος ορίζεται άντληση πρωτογενών δεδομένων που προσφέρει η βάση αναφορών Scopus, με έμφαση στη διερεύνηση των δυνατοτήτων που προσφέρει η προγραμματιστική διεπαφή (API) «Authors Retrieval» .

Ο δεύτερος στόχος περιλαμβάνει το σχεδιασμό και την υλοποίηση μιας εφαρμογής, η οποία θα έχει ως κύρια λειτουργία τη συλλογή δεδομένων με βάση τον ερευνητή ή ομάδες ερευνητών – όπως αυτές συγκροτούνται μέσα από ακαδημαϊκά τμήματα – τα οποία θα μας βοηθήσουν στο τομέα της μετέπειτα παραγωγής βιβλιομετρικών δεικτών (π.χ. δείκτες ακαδημαϊκής δραστηριότητας - απόδοσης). Η εφαρμογή υλοποιήθηκε στο περιβάλλον προγραμματισμού (IDE) Visual Studio της Microsoft χρησιμοποιώντας την αντικειμενοστρεφή γλώσσα προγραμματισμού γενικής χρήσης C#.

Σκοπός της εφαρμογής είναι υλοποίηση των αλλαγών που απαιτούνται για τη λειτουργία του νέου πλαισίου αξιολόγησης της ερευνητικής δραστηριότητας και τον υπολογισμό των νέων δεικτών και μετρικών.

Τέλος, η εφαρμογή που αναπτύχθηκε εφαρμόσθηκε για την άντληση των βιβλιομετρικών δεδομένων σε τμήματα του Πανεπιστημίου Δυτικής Αττικής και για την περαιτέρω ανάλυση τους, χρησιμοποιώντας στατιστικές μεθόδους, και εξάχθηκαν συμπεράσματα αναφορικά με την αξιολόγηση της ερευνητικής δραστηριότητας οντοτήτων.

1.2 Μεθοδολογία

Η μεθοδολογία που ακολουθήθηκε για την επίτευξη των στόχων της παρούσας πτυχιακής εργασίας περιγράφεται στις επόμενες παραγράφους.

Αρχικά, διεξήχθη η βιβλιογραφική έρευνα στο αντικείμενο της πτυχιακής εργασίας. Η παραπάνω διαδικασία πραγματοποιήθηκε με την αναζήτηση της απαραίτητης πληροφορίας σε μεγάλα ακαδημαϊκά αποθετήρια, όπως η Elsevier - Scopus. Επιπλέον, χρησιμοποιήθηκε το εργαλείο “Mendeley” ως εργαλείο συγκέντρωσης και οργάνωσης της βιβλιογραφικής έρευνας και των αναφορών της.

Έπειτα, πραγματοποιήθηκε σε θεωρητικό επίπεδο ο καθορισμός του νέου πλαισίου αξιολόγησης της ερευνητικής δραστηριότητας, με βάση τα δεδομένα που προσφέρει το API Authors της βάσης αναφορών Scopus. Πιο αναλυτικά, σε αυτό το κομμάτι έγινε ο καθορισμός των δεδομένων εισόδου, μαθηματικών τύπων και τεχνικών ανάλυσης δεδομένων που συγκεντρώθηκαν, για την υλοποίηση του πρακτικού περιεχομένου της έρευνας. Με άλλα λόγια, καθορίστηκαν οι απαιτήσεις λειτουργικότητας της εργασίας.

Στη συνέχεια, εξετάστηκαν οι αλλαγές που απαιτούνται για την αναβάθμιση του εργαλείου άντλησης βιβλιομετρικών δεδομένων που διαθέτει το εργαστήριο Διαχείρισης της Πληροφορίας (Προδιαγραφές ανάπτυξης) και ο σχεδιασμός και δημιουργία της νέας εφαρμογής συλλογής δεδομένων.

Τέλος, έγινε η χρήση του αναβαθμισμένου εργαλείου με πραγματικά δεδομένα από τμήματα του Πανεπιστημίου Δυτικής Αττικής με σκοπό την παραγωγή αποτελεσμάτων για τις μεθόδους που χρησιμοποιήθηκαν και παράθεση των συμπερασμάτων που προέκυψαν από αυτά.

1.3 Περιορισμοί

Ο μοναδικός περιορισμός μιας έρευνας, αντίστοιχης με την παρούσα, είναι η αναγκαία ύπαρξη συνδρομής στη βάση αναφορών Scopus, από την οποία συλλέγονται τα δεδομένα. Στην περίπτωση αυτής της εργασίας, χρησιμοποιήθηκε η δωρεάν συνδρομή που έχει προσφερθεί από τη Scopus προς το τμήμα Αρχειονομίας Βιβλιοθηκονομίας και Συστημάτων Πληροφόρησης του Πανεπιστημίου Δυτικής Αττικής.

1.4 Προγραμματιστικοί όροι

String : Προγραμματιστικό αντικείμενο. Αποτελεί μια ακολουθία αλφαριθμητικών χαρακτήρων ή συμβόλων ή αλλιώς κείμενο.

Class : Η κλάση είναι ένα σύνολο δηλώσεων που αφορούν στην περιγραφή μιας συγκεκριμένης κατηγορίας αντικειμένων. Απλοϊκά είναι το "καλούπι" με βάση από το οποίο περιγράφεται ένα αντικείμενο και οι ιδιότητες του. Μια κλάση μπορεί να έχει υπερ-κλάσεις (parent-class / superclass) ή υπο-κλάσεις (sub-class).

Object : Ένα αντικείμενο, είναι η προγραμματιστική περιγραφή μιας φυσικής ή νοητής οντότητας. Όπως προαναφέρθηκε τα αντικείμενα υπάγονται σε κλάσεις οι οποίες περιγράφουν τις ιδιότητές τους. Για να γίνει πιο εύκολα αντιληπτό αυτό, ας σκεφτούμε ένα αντικείμενο με όνομα "σκύλος ". Ο σκύλος θα μπορούσε να αποτελεί αντικείμενο της κλάσης "τετράποδα", που με τη σειρά της να ήταν υποκλάση της κλάσης "ζώα".

API : Η Διεπαφή Προγραμματισμού Εφαρμογών (Application Programming Interface) είναι η διεπαφή των προγραμματιστικών διαδικασιών που παρέχει ένα λειτουργικό σύστημα, βιβλιοθήκη ή εφαρμογή προκειμένου να επιτρέπει να γίνονται προς αυτά αιτήσεις από άλλα προγράμματα ή/και ανταλλαγή δεδομένων.

Http Request : Ένα (πιθανώς) παραμετροποιήσιμο προγραμματιστικό αίτημα που πραγματοποιείται μεταξύ της επικοινωνίας ενός client και ενός server.

Http Response : Η απάντηση που επιστρέφει ο server για ένα "http" αίτημα που δέχεται.

XML : Η XML αποτελεί μια γλώσσα σήμανσης, η οποία προσφέρει την εύκολη κατανόηση και περιγραφή αυθαίρετων δομών δεδομένων ενώ αυτόματα, μετατρέπεται και σε γλώσσα μηχανής.

XML Nodes : Η XML γλώσσα, χωρίζει το περιεχόμενό της σε γονικούς και παιδικούς κόμβους (nodes) με σκοπό την καλύτερη περιγραφή των δεδομένων που αντικατοπτρίζει.

(My)SQL Query : Ένα προγραμματιστικό ερώτημα που απευθύνεται στη βάση δεδομένων με σκοπό την άντληση δεδομένων ή την διεκπεραίωση κάποιας ενέργειας πάνω σε αυτή.

RDBMS : Ένα rdbms αποτελεί ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων. Ένα τέτοιο σύστημα προσφέρει τη δημιουργία και διαχείριση σχεσιακών βάσεων δεδομένων στις οποίες, οι πίνακες συνδέονται μεταξύ τους μέσω σχέσεων πληθικότητας σε κανονικοποιημένη μορφή.

1.5 Βιβλιομετρική βάση Scopus

Η βιβλιομετρική βάση Scopus δημιουργήθηκε το 2004 από την Eslevier και αποτελεί μια διεπιστημονική πηγή αναφορών (Norris & Oppenheim, 2007) . Πιο συγκεκριμένα, είναι μια από τις μεγαλύτερες «peer reviewed» βάσεις δεδομένων παγκοσμίως, η οποία καλύπτει περισσότερους από 24,000 ενεργούς τίτλους ακαδημαϊκών περιοδικών σε πολλαπλές θεματικές περιοχές υψηλού ερευνητικού ενδιαφέροντος όπως : βιοεπιστήμες, κοινωνικές επιστήμες, φυσικές επιστήμες και επιστήμες υγείας. Επιπλέον διαθέτει περισσότερους από 230,000 τίτλους βιβλίων και παραπάνω από 10,000,000 δημοσιεύσεις πρακτικών συνεδρίων. (Scopus, 2020)

Περιέχει τρία είδη ψηφιακών τεκμηρίων : σειρές βιβλίων, επιστημονικά περιοδικά και πρακτικά συνεδρίων. Όπως αναφέρουν οι Leslie και Chris (Adriaanse & Rensleigh, 2011) συγκριτικά με άλλες βάσεις δεδομένων ακαδημαϊκού - ερευνητικού περιεχομένου (όπως π.χ. Google Scholar ή Web of Science), η Scopus αποτελεί αυτή με τις λιγότερες “ασυνέπειες” αναφορικά με την εξακρίβωση και ποιότητα του περιεχομένου τους. Αναλυτικότερα, η κατάταξη του συνόλου των ετεροαναφορών μεταξύ των τριών πηγών που προαναφέρθηκαν “βρίσκει” το Google Scholar στην πρώτη θέση, ακολουθούμενο από το Web of Science και στην τελευταία θέση βρίσκεται η Scopus. Όμως, τα παραπάνω δεδομένα μολονότι ακριβή, μπορούν να οδηγήσουν σε παραπλανητικά συμπεράσματα. Αυτό συμβαίνει καθώς οι δύο προαναφερθείσες βάσεις δεδομένων, περιέχουν διπλότυπα ή και σε ορισμένες περιπτώσεις τριπλότυπα, με αποτέλεσμα στον συνολικό αριθμό αναφορών να συνυπολογίζονται τα ίδια τεκμήρια και επομένως να παράγουν ανακριβή δεδομένα. Επιπλέον, η Scopus παρέχει στους χρήστες της διαδικτυακά εργαλεία βιβλιομετρικής ανάλυσης των δημοσιεύσεων τους, υπολογίζοντας πληθώρα βιβλιομετρικών δεικτών όπως για παράδειγμα ο h-index αλλά και εργαλεία στατιστικής ανάλυσης όπως διαγράμματα συχνότητας δημοσιεύσεων σε συνάρτηση με το χρόνο κ.α. Στην παρούσα έρευνα, λοιπόν, χρησιμοποιήθηκε η βιβλιομετρική βάση δεδομένων Scopus για τους λόγους που έχουν ήδη αναφερθεί αλλά και λόγω του δημόσιου και τεκμηριωμένου REST (Representational State Transfer) API (Application Programming Interface), που προσφέρει στους χρήστες του, για τη συλλογή δεδομένων.

1.5.1 Βιβλιομετρικά δεδομένα ερευνητή μέσω Scopus

Στην περίπτωση των βιβλιομετρικών δεδομένων, η Scopus παρέχει μια σειρά από δείκτες και «analytics» για τους εγγεγραμμένους χρήστες και συνδρομητές της. Πιο συγκεκριμένα, χρησιμοποιώντας τις διαδικτυακές υπηρεσίες (web services) που προσφέρει μπορούμε να αναζητήσουμε ακαδημαϊκές - ερευνητικές δημοσιεύσεις με πληθώρα τρόπων όπως :

1. Η απλή αναζήτηση με βάση τον τίτλο ενός τεκμηρίου / δημοσίευσης.
2. Η απλή αναζήτηση με βάση το όνομα του ερευνητή / ακαδημαϊκού.
3. Η απλή αναζήτηση με βάση το ακαδημαϊκό ίδρυμα, στο οποίο έχει αναπτυχθεί ερευνητική εργασία.
4. Η σύνθετη αναζήτηση που μπορεί να βασίζεται σε πολλά διαφορετικά πεδία όπως (π.χ.) τίτλος, θεματική περιοχή, λέξεις κλειδιά και εκτελείται μέσα από ερωτήματα (queries), τα οποία συνδέονται μεταξύ τους με τελεστές “Boolean”. Στην περίπτωση της σύνθετης αναζήτησης, για τους αρχάριους χρήστες περιέχεται και εγχειρίδιο περιγραφής των παραπάνω ερωτημάτων, το οποίο περιέχει και παραδείγματα.

Αφού ο χρήστης καταφέρει να εντοπίσει το τεκμήριο(-α) ή τον συγγραφέα(-είς) (κ.α.) που επιθυμεί, του παρουσιάζονται μια πληθώρα δεδομένων.

Στην περίπτωση της αναζήτησης ενός **τεκμηρίου / δημοσίευσης** παρουσιάζονται τα παρακάτω βιβλιογραφικά, βιβλιομετρικά και στατιστικά δεδομένα :

1. Το όνομα της πηγής έκδοσης (π.χ. περιοδικού) και η βιβλιογραφική αναφορά.
2. Ο πλήρης τίτλος της δημοσίευσης, συνοδευόμενος από τον τύπο του τεκμηρίου (π.χ. «article»).
3. Το όνομα του συγγραφέα(-εων) και τα σχετιζόμενα με αυτούς ακαδημαϊκά ιδρύματα.
4. Η περίληψη της δημοσίευσης.
5. Οι λέξεις-κλειδιά που επέλεξε ο συγγραφέας(-είς) και οι λέξεις-κλειδιά που έχει αναγνωρίσει και καθιερώσει αυτόματα η βάση δεδομένων ανά θεματική κατηγορία.
6. Το σύνολο των αναφορών της δημοσίευσης (συμπεριλαμβανομένων των αυτο-αναφορών) καθώς και τα άρθρα, μέσα στα οποία έχουν αναφερθεί.
7. Ο βιβλιομετρικός δείκτης «Field Weighted Citation Impact» που περιγράφει τον αριθμό των ετεροαναφορών ενός άρθρου συγκριτικά με παρόμοια άρθρα της ίδιας θεματικής περιοχής.
8. Τα αναγνωριστικά της δημοσίευσης σε συνδυασμό με άλλες βιβλιογραφικές πληροφορίες όπως για παράδειγμα : το «ISSN» («International Standard Serial Number»), η αρχική γλώσσα δημοσίευσης και το «DOI» («Digital Object Identifier»).
9. Παρόμοιες ή αντίστοιχες ερευνητικές δημοσιεύσεις.
10. Τις αναφορές των δημοσιεύσεων που χρησιμοποιήθηκαν από τον συγγραφέα(-είς) για την περάτωση της έρευνας.

Στην περίπτωση της αναζήτησης ενός **ακαδημαϊκού / ερευνητή** παρουσιάζονται τα παρακάτω αποτελέσματα :

1. Το πλήρες όνομά του.
2. Το σχετιζόμενο με αυτόν ίδρυμα.
3. Το «author-id» ή «scopus-id» του , δηλαδή το βασικό αναγνωριστικό ενός συγγραφέα που παρέχεται από τη Scopus.
4. Διαφορετικές εκδοχές του ονόματός του, όπως αυτό έχει καταγραφεί σε δημοσιεύσεις του.
5. Ένα διάγραμμα όπου απεικονίζονται ο αριθμός των δημοσιεύσεων και το πλήθος των ετεροαναφορών τους σε συνάρτηση με το χρονικό διάστημα, στο οποίο καταγράφονται τα δεδομένα του από τη «Scopus».
6. Λίστες που περιέχουν :
    - Το συνολικό αριθμό των δημοσιεύσεών του και τις δημοσιεύσεις με τον τίτλο τους σε φθίνουσα χρονολογική ταξινόμηση.
    - Το συνολικό αριθμό των αναφορών του (μη συμπεριλαμβανομένων των αυτό-αναφορών) και τα άρθρα που τον έχουν αναφέρει με τον τίτλο τους σε φθίνουσα χρονολογική ταξινόμηση.
    - Το συνολικό αριθμό των συν-συγγραφέων με τους οποίους δημοσίευσε έρευνες και τα ονόματα τους ταξινομημένα με βάση τον αριθμό δημοσιεύσεων που είχε με τον καθένα από αυτούς. σε φθίνουσα σειρά.
    - Το ιστορικό του, στο οποίο ορίζονται οι ημερομηνίες έναρξης και λήξης (αν υπάρχει) της ερευνητικής του δραστηριότητας καθώς και ο συνολικός αριθμός των άρθρων που έχει χρησιμοποιήσει ως αναφορές ο ίδιος. Επιπλέον παρουσιάζονται τα περιοδικά στα οποία έχει δημοσιεύσει (ή οι εκδότες) και τα ιδρύματα στα οποία έχει εργαστεί στο παρελθόν ή με τα οποία σχετίζεται ως ερευνητής.

7. Ο βιβλιομετρικός δείκτης «h-index» του συγγραφέα που περιγράφει το N αριθμό δημοσιεύσεων με N και πάνω αναφορές (βλ. κεφ. 1) .
8. Ο συνολικός αριθμός των αναφορών του συμπεριλαμβανομένων και των αυτό-αναφορών.

Τέλος, στην περίπτωση αναζήτησης ενός ακαδημαϊκού ιδρύματος παρουσιάζονται τα παρακάτω αποτελέσματα :

1. Το επίσημο όνομα του ιδρύματος συνοδευόμενο από τη διεύθυνση (πόλη, χώρα) και τον ταχυδρομικό του κωδικό.
2. Άλλες εκδοχές του ονόματος του ιδρύματος όπως έχουν καταγραφεί σε δημοσιεύσεις της Scopus.
3. Το «affiliation_id», δηλαδή, το βασικό αναγνωριστικό που παρέχει η «Scopus» για ένα ακαδημαϊκό ίδρυμα / φορέα.
4. Λίστες που περιέχουν :
    - Τον αριθμό των δημοσιεύσεων του ιδρύματος ανά θεματική κατηγορία της «Scopus» ταξινομημένα σε φθίνουσα σειρά συνοδευόμενα από γράφημα που παρουσιάζει τα ποσοστά δημοσίευσης ανά θεματική κατηγορία.
    - Τα ονόματα των ιδρυμάτων με τα οποία έχει συνεργαστεί ο ακαδημαϊκός φορέας που έχει επιλεχθεί ταξινομημένα σε φθίνουσα σειρά με βάση τον συνολικό αριθμό των δημοσιεύσεων ανά φορέα.
    - Τα ονόματα των περιοδικών / εκδοτικών οίκων στα οποία έχει δημοσιεύσει ο φορέας ταξινομημένα σε φθίνουσα σειρά με βάση το συνολικό αριθμό των δημοσιεύσεων ανά περιοδικό / έκδοση.
5. 5. Ο συνολικός αριθμός των δημοσιεύσεων ενός ιδρύματος και ο συνολικός αριθμός των δημοσιεύσεων του τμήματος του ιδρύματος αυτού (σε περίπτωση που έχει γίνει αναζήτηση ενός συγκεκριμένου τμήματος που ανήκει σε έναν ακαδημαϊκό φορέα).
6. 6. Ο συνολικός αριθμός των συγγραφέων που ανήκουν στο ακαδημαϊκό ίδρυμα.

Συμπερασματικά, τα βιβλιομετρικά δεδομένα που παρέχει η «Scopus» είναι δομημένα με τέτοιο τρόπο, ώστε να διευκολύνεται η πολύ-επίπεδη αναζήτηση πληροφοριών, με αποτέλεσμα, συγκεντρωτικά, να αποτελούν μια πλήρη εικόνα ενός ακαδημαϊκού - ερευνητικού προφίλ.

1.5.2 Αναγνωριστικά συγγραφέων και Scopus ID

Ο ρόλος των μοναδικών αναγνωριστικών στη διαδικασία της δημοσίευσης εργασιών προσφέρει πληθώρα πλεονεκτημάτων για ολόκληρη την ακαδημαϊκή (- ερευνητική) κοινότητα και τους φορείς που σχετίζονται με αυτή. Επιτρέπει στους συγγραφείς να συνεργάζονται με μεγαλύτερη ευκολία για να παραχθούν νέες έρευνες σε κοινές θεματικές περιοχές, στα ακαδημαϊκά ιδρύματα να ελέγχουν και να συγκεντρώνουν την πρόοδο των ερευνητών τους με μεγαλύτερη ακρίβεια και ευκολία και στα περιοδικά (ακαδημαϊκά / επαγγελματικά) να απλοποιούνται οι διαδικασίες της δημοσίευσης και της αξιολόγησης των δημοσιεύσεων (Fenner, 2011). Αναφορικά με τους συγγραφείς, ο ρόλος της απόδοσης ενός μοναδικού αναγνωριστικού, γίνεται όλο και πιο σημαντικός (αν όχι αναγκαίος) με την πάροδο του χρόνου, λόγω της καθολικής ψηφιοποίησης της πληροφορίας σε συνδυασμό με τη ραγδαία τεχνολογική ανάπτυξη. Ένα μοναδικό αναγνωριστικό συγγραφέα («author unique identifier») στον ερευνητικό κλάδο είναι, ένας μοναδικός αριθμός, ο οποίος αντιστοιχεί σε έναν ερευνητή. Αναλυτικότερα, μέσω ενός «unique identifier» διευκολύνονται οι διαδικασίες της συγκομιδής και ανάλυσης των βιβλιογραφικών και κατ' επέκταση βιβλιομετρικών δεδομένων και η συγκέντρωση / απόθεση τους σε ένα ενιαίο προφίλ,

στο οποίο αντικατοπτρίζεται το σύνολο της δημοσιευμένης ακαδημαϊκής έρευνας ενός συγγραφέα. Παράλληλα, διασφαλίζει την καταγραφή της μελλοντικής παραγωγής δημοσιεύσεων στο ίδιο προφίλ προσδίδοντας ενημερότητα και καθολικότητα στην πληροφορία. Έχουν πραγματοποιηθεί διάφορες προσπάθειες για τη δημιουργία καθολικών αναγνωριστικών για τους συγγραφείς, όπως για παράδειγμα το «Researcher_ID» από τη βάση δεδομένων «Web of Science» ή το «author_id» ή «scopus_id» από τη Scopus. Το «Scopus_id» αποτελεί ένα αναγνωριστικό που παράγεται και χρησιμοποιείται από τη βάση δεδομένων της Scopus. Ορίζεται αυτόματα σε έναν συγγραφέα με τη δημιουργία μιας δημοσίευσης, μέσω ενός αλγορίθμου, ο οποίος εξετάζει το ίδρυμα, τη διεύθυνση, τη θεματική περιοχή, τον τίτλο, τις ετεροαναφορές και τα ονόματα των συν-συγγραφέων της, ώστε να αναγνωρίσει ομοιότητες που συγκεντρώνονται σε ένα κοινό "προφίλ" (Krämer, et al., 2017) . Όμως, η ακριβής διαδικασία και ο τρόπος με βάση τον οποίο εξετάζονται οι παραπάνω πληροφορίες, έτσι ώστε να διασταυρωθούν και να δημιουργήσουν - εν τέλη - ένα ενιαίο προφίλ, δεν έχουν δημοσιοποιηθεί. Το πρόβλημα που προκύπτει από την αυτόματη δημιουργία αναγνωριστικών (ειδικά στην περίπτωση της Scopus) είναι ο "κίνδυνος" της δημιουργίας πολλαπλών αναγνωριστικών που αφορούν το ίδιο άτομο. Αυτό μπορεί να συμβεί αν ο συγγραφέας δημοσιεύει μέσω ενός διαφορετικού ιδρύματος, σε μια νέα θεματική περιοχή, με νέους συν-συγγραφείς. Εφόσον δημιουργηθούν, αν ο ίδιος ο συγγραφέας δεν εντοπίσει το λάθος, έτσι ώστε να το υποβάλλει στους υπεύθυνους για τη διόρθωσή του, προκαλείται άμεσα αλλοίωση της καταγραφής της δημοσιευμένης ερευνητικής του δραστηριότητας και συνεπώς η διαδικασία αξιολόγησής του μέσω βιβλιομετρικών δεικτών καθίσταται ανακριβής. Επιπρόσθετα, πρόβλημα μπορεί να προκαλέσει σε ορισμένες περιπτώσεις, η συνωνυμία συγγραφέων, περιστατικό που λαμβάνει χώρο, σε μεγάλο ποσοστό, σε Ασιατικές χώρες. Παρά τα πιθανά προβλήματα, ο αλγόριθμος της Scopus είναι ακριβής σε υψηλό ποσοστό, όσον αφορά την κατηγοριοποίηση της πληροφορίας και την ένταξή της στα σωστά προφίλ, όπως αποδεικνύεται και από την έρευνα των Kawashima και Tomizawa (Kawashima & Tomizawa, 2015). Στη δημοσίευσή τους, έπειτα από σύγκριση μεταξύ των αποτελεσμάτων που προσφέρει η Scopus και η KAKEN (μεγαλύτερη ιαπωνική ακαδημαϊκή βάση δεδομένων) σχετικά με την ανάκληση και την ακρίβεια των αποτελεσμάτων, τα στατιστικά ευρήματα ήταν 98% και 99% αντιστοίχως. Σημαντικό είναι να αναφερθεί, βέβαια, είναι ότι το δείγμα στη συγκεκριμένη έρευνα, δεν αποτελεί απόδειξη της συνολικής ακρίβειας που προσφέρει η Scopus, εφόσον περιορίζεται σε εθνικό επίπεδο. Καθώς, λοιπόν, η δημιουργία διαφορετικών «scopus_ids» είναι αναπόφευκτη σε ορισμένες περιπτώσεις, το 2009 αναφέρθηκε για πρώτη φορά το ORCID («Open Researcher Contributor Identification Initiative») από την ευρύτερη ακαδημαϊκή κοινότητα, με σκοπό τη λύση του προβλήματος της λανθασμένης πιστοποίησης ακαδημαϊκών δημοσιεύσεων. Στις 5 Αυγούστου του 2010, στην πολιτεία «Delaware» των Η.Π.Α. δημιουργήθηκε ο ORCID, Inc. (Anon., 2010) ένας μη κερδοσκοπικός οργανισμός, αποτελούμενος από μέλη της ερευνητικής κοινότητας, ο οποίος "ανέλαβε" τη δημιουργία μιας λύσης του παραπάνω προβλήματος. Αναλυτικότερα, δημιουργήθηκε το αναγνωριστικό «ORCID», δηλαδή ένας μοναδικός αλφαριθμητικός κωδικός (χρησιμοποιεί όλους τους ASCII ή Unicode χαρακτήρες [0-9] και [Aa-Zz]) για τον κάθε χρήστη, με σκοπό την ένταξη όλων των διαφορετικών αναγνωριστικών που έχουν δημιουργηθεί για έναν ερευνητή, σε ένα μοναδικό ακαδημαϊκό προφίλ. Μετά από την ολοκλήρωση του παραπάνω συστήματος αυτοματοποιημένης απόδοσης μοναδικών "κωδικών" ORCID, το 2012, έπειτα από τη δημιουργία ενός συστήματος εγγραφής για τους ερευνητές, το εγχείρημα τέθηκε για πρώτη φορά σε λειτουργία. Σύντομα, γνωστά περιοδικά υψηλού κύρους όπως το Springer και το PLOS, ξεκίνησαν να χρησιμοποιούν το ORCID στη διαδικασία της υποβολής εργασιών προς δημοσίευση, με σκοπό την διευκόλυνση της προτυποποίησης της πληροφορίας. Το μοναδικό μειονέκτημα που μπορεί να αναγνωριστεί στο σύστημα του ORCID, είναι το γεγονός ότι δεν είναι πλήρως αυτοματοποιημένο, καθώς απαιτείται σε πρώτο επίπεδο, η δημιουργία ενός διαδικτυακού λογαριασμού από τον κάθε ερευνητή.

Συμπερασματικά, παρά το ότι έχουν προκύψει προβλήματα στην προσπάθεια της αυτοματοποιημένης απόδοσης μοναδικών αναγνωριστικών σε συγγραφείς, η ερευνητική κοινότητα δείχνει να πλησιάζει σε μια λύση, κυρίως μέσω του ORCID, είτε μέσω νέων μελλοντικών προτάσεων.

1.6 Προσφερόμενα API

Η Scopus , εκτός από τη δυνατότητα που προσφέρει στους εγγεγραμμένους χρήστες της για τη διαδικτυακή αναζήτηση δημοσιεύσεων και ερευνητικών οντοτήτων, προσφέρει επιπλέον μια σειρά από ανοιχτά «REST» («Representational State Transfer») «APIs» («Application Programming Interfaces»). Ένα «web API» αποτελεί τον διαδικτυακό μεσολαβητή ενός ηλεκτρονικού αιτήματος και της διεκπεραίωσης του, μεταξύ διαφορετικών εφαρμογών. Τα «REST APIs» χρησιμοποιούνται κυρίως από προγραμματιστές ή επιστήμονες δεδομένων («data scientists») για την περαιτέρω αξιοποίηση της πληροφορίας που προσφέρουν. Η πρόσβαση στα APIs γίνεται μέσω ενός «API key», δηλαδή ενός κωδικού ασφαλείας μεταξύ του API και του χρήστη, το οποίο διατίθεται δωρεάν σε οποιονδήποτε θέλει να το αποκτήσει, υπό την μοναδική προϋπόθεση ότι δε χρησιμοποιείται εκτός των πλαισίων της πολιτικής απορρήτου της «Elsevier». Για την απόκτηση ενός «API key», ο χρήστης πρέπει πρώτα να δημιουργήσει ένα λογαριασμό στην ιστοσελίδα της «Elsevier». Εάν ο χρήστης δεν διαθέτει κάποιο συνδρομητικό πακέτο της «Scopus», αφού λάβει το κλειδί έχει έναν περιορισμό ως προς τον αριθμό των "αιτημάτων" («http requests») που μπορεί να στείλει προς τη βάση δεδομένων (ο αριθμός ανέρχεται στα 10,000 αιτήματα). Επίσης, δίνεται πρόσβαση μόνο σε συγκεκριμένα δεδομένα και δεν μπορεί να αναζητήσει πληροφορίες με διαφορετικές μεθόδους από τη βασική μέθοδο αναζήτησης που προσφέρεται. Σε περίπτωση που κάποιος θέλει να έχει ευρύτερη πρόσβαση χωρίς περιορισμούς πρέπει να ανήκει ή να έχει πρόσβαση σε οργανισμούς, που διαθέτουν συνδρομητικά πακέτα της «Elsevie»r. Τα προσφερόμενα API χωρίζονται σε δύο γενικές κατηγορίες :

1. Η πρώτη αφορά την αναζήτηση πληροφοριών («Search»).
2. Η δεύτερη αφορά την ανάκτηση πληροφοριών («Retrieval»).

1.6.1 Αναζήτηση πληροφοριών – Search APIs

Η αναζήτηση γίνεται μέσω της προσθήκης του ονόματος ή του αναγνωριστικού («scopus id») της κάθε οντότητας, καθώς και του «API key», μέσω ενός «http request». Τα δεδομένα που ανακτώνται από τα προσφερόμενα API μπορούν να δομηθούν αυτόματα σε μορφή XML ή Json. Τα «Search APIs» της Scopus αποτελούνται από σύντομες, γενικές πληροφορίες σχετικά με τις οντότητες που περιγράφουν.

Στην περίπτωση της αναζήτησης συγγραφέων - ερευνητών η Scopus δίνει τη δυνατότητα στους προγραμματιστές να επικοινωνήσουν μέσω του «Author Search» («REST API»), με τη βάση δεδομένων τους και να αντλήσουν πληροφορίες για έναν συγγραφέα.

Αναλυτικότερα, μέσω ενός «http request» σε συνδυασμό με ένα συγκεκριμένο ερώτημα («query») διαμορφωμένο από τη Scopus, ο χρήστης μπορεί να αναζητήσει συγγραφείς με βάση το ονοματεπώνυμο τους, το ίδρυμα στο οποίο εργάζονται ή δημοσιεύουν ή και το «ORCID» τους. Το κομμάτι της συμπλήρωσης του ονόματος και του επιθέτου είναι υποχρεωτικό, ενώ τα υπόλοιπα πεδία δεν είναι.

Ένα «http request», αποτελεί στην ουσία ένα ερώτημα που αποστέλλεται από έναν «client» προς έναν «server» μέσω του διαδικτύου («internet»), στο οποίο και οι δύο πλευρές έχουν πρόσβαση. Ένας διαδικτυακός «client» αποτελεί λογισμικό, το οποίο βρίσκεται εγκατεστημένο σε συσκευές όπως ο προσωπικός υπολογιστής, το κινητό, η τηλεόραση (κ.α.), και η χρήση του περιορίζεται στο να στέλνει

δεδομένα με μορφή http requests («get, post, put, delete») και να "περιμένει" μια απάντηση («response») που στη συνέχεια αναλύει τοπικά. Ο διαδικτυακός «web server» (απομακρυσμένος υπολογιστής με πρόσβαση στο διαδίκτυο), διεκπεραιώνει όποιες διεργασίες του "ζητηθούν" και επιστρέφει στον «client» τα δεδομένα που θα διαχειριστεί. Ένας πιο εύκολος τρόπος να καταλάβουμε τι είναι η επικοινωνία μεταξύ ενός «server» και ενός «client», είναι να σκεφτούμε τη διαδικασία της παραγγελίας ενός καφέ από την καφετέρια. Ο πελάτης («client») παραγγέλνει («request») τον καφέ της επιλογής του σε έναν υπάλληλο. Ο υπάλληλος («server»), καταγράφει την παραγγελία εφόσον αυτή είναι στα πλαίσια της παραγωγής του καταστήματος, τη διεκπεραιώνει. Τέλος αφού ολοκληρώσει την παραγωγή του καφέ, τον παραδίδει στον πελάτη (response).

Author Search API :

HTTP REQUEST :

Το «http request» του author search API μπορεί να αποσταλεί είτε μέσω ενός τερματικού π.χ. («windows powershell, cmd»), είτε μέσω ενός browser. Εδώ παρουσιάζεται η δομή ενός «https» ερωτήματος μέσω του ενός τερματικού.

```
curl -X GET --header 'Accept: application/xml' 'https://api.elsevier.com/content/search/author?query=
```

Το παραπάνω ερώτημα, δέχεται ως παράμετρο ένα «query» (ερώτημα), το οποίο με τη σειρά του δέχεται περαιτέρω παραμετροποίηση:

Response Content Type application/xml

Parameters

Parameter	Value	Description	Parameter Type	Data Type
query	authlast(Kouis) and authfirst(Dimitrios) and affi	**Search query string**	query	string
apiKey	a67c47ae6eb8460606c4df18cc79793b	**Your API key**	query	string
httpAccept		Requested content type, overrides HTTP header value	query	string
insttoken		Specification for authorization, institution authtoken	query	string
access_token		Specification for active session, secured authtoken	query	string

Εικόνα 1. Περιγραφή http request από την ιστοσελίδα της Scopus για το Author Search API.

Στην παραπάνω εικόνα περιγράφονται τα πεδία καθώς και ο τύπος τους, τα οποία μπορούν να προστεθούν ως παράμετροι στο «http request». Στο συγκεκριμένο παράδειγμα το ερώτημα («query») δέχεται ως παραμέτρους το όνομα, το επίθετο, καθώς τον οργανισμό ενός ερευνητή, οι οποίες συνδέονται μεταξύ τους με λογικούς τελεστές «Boolean» («bool and»). Επιπλέον, προστίθεται σαν παράμετρος στο «request» και το «api-key», ώστε να πραγματοποιηθεί η πιστοποίηση του αιτήματος. Οι δύο αυτές παράμετροι είναι απαραίτητες για να λειτουργήσει το αίτημα του χρήστη, ενώ οι υπόλοιπες είναι προαιρετικές.

HTTP RESPONSE :

Η απάντηση έρχεται από τον «server», βασισμένη στη μορφή που επέλεξε ο χρήστης μέσω του «request» του. Στην εργασία χρησιμοποιήθηκε η δόμηση των δεδομένων της απάντησης σε μορφή XML. Περιέχει, επίσης, έναν κωδικό απάντησης («response code»),

καθώς και δύο πεδία στο «header» της:

την έκταση του περιεχομένου σε χαρακτήρες (συμπεριλαμβάνονται και τα κενά), καθώς και τον τύπο της δομής («content-type») σε συνδυασμό με την κωδικοποίησή της.

Εικόνα 2. Παράδειγμα http response, Author Search API

Ένα παράδειγμα XML απάντησης :


```
▼<entry>
  <link ref="self" href="https://api.elsevier.com/content/author/author_id/65076529
  <link ref="search" href="https://api.elsevier.com/content/search/author?query=au-
  <link ref="scopus-citedby" href="https://www.scopus.com/author/citedby.uri?partne
  <link ref="scopus-author" href="https://www.scopus.com/authid/detail.uri?partnerI
 ▼<prism:url>
    https://api.elsevier.com/content/author/author_id/6507652923
  </prism:url>
  <dc:identifier>AUTHOR_ID:6507652923</dc:identifier>
  <eid>9-s2.0-6507652923</eid>
  <orcid>0000-0002-5948-9766</orcid>
 ▼<preferred-name>
    <surname>Kouis</surname>
    <given-name>Dimitris</given-name>
    <initials>D.</initials>
  </preferred-name>
 ▼<name-variant>
    <surname>Kouis</surname>
    <given-name>Dimitrios</given-name>
    <initials>D.</initials>
  </name-variant>
 ▼<name-variant>
    <surname>Kouis</surname>
    <given-name>D.</given-name>
    <initials>D.</initials>
  </name-variant>
  <document-count>12</document-count>
  <subject-area abbrev="COMP" frequency="10">Computer Science (all)</subject-area>
  <subject-area abbrev="ENGI" frequency="6">Engineering (all)</subject-area>
  <subject-area abbrev="SOCI" frequency="5">Social Sciences (all)</subject-area>
```

Εικόνα 3 Παράδειγμα XML απάντησης

Η απάντηση χωρίζεται σε «entries» δηλαδή «XML nodes», το καθένα εκ των οποίων περιγράφει έναν συγγραφέα και τα πεδία πληροφοριών που του αντιστοιχούν. Σε αυτό το σημείο αξίζει να αναφερθεί, ότι η αναζήτηση για έναν συγγραφέα μπορεί υπό προϋποθέσεις να επιστρέψει περισσότερα από ένα αποτελέσματα («entries»). Οι λόγοι για τους οποίους ενδέχεται να προκύψει αυτό, είναι είτε επειδή υπάρχει συνωνυμία (ακόμα και μέσα στο ίδιο ίδρυμα), είτε επειδή ένας συγγραφέας έχει περισσότερα από ένα καταγεγραμμένα προφίλ στη «Scopus».

Ανάλυση πεδίων απάντησης :

Στο παράδειγμα της απάντησης το πεδίο «dc:identifier» αναφέρεται στο μοναδικό αναγνωριστικό του συγγραφέα («scopus author id»). Με τον ίδιο τρόπο παρατίθενται και τα αναγνωριστικά ORC-ID και EID, στα αντίστοιχα πεδία. Έπειτα, ακολουθεί το ονοματεπώνυμο του συγγραφέα, όπως είναι καταγεγραμμένο στη Scopus, καθώς και κάποιες παραλλαγές με τις οποία μπορεί να συναντηθεί. Στη συνέχεια, αναφέρονται οι θεματικές περιοχές στις οποίες έχει δημοσιεύσει αλλά και η συχνότητα με βάση την οποία δημοσιεύει στην καθεμία από αυτές ο συγγραφέας συγκριτικά με τις υπόλοιπες. Τέλος, περιγράφεται το ίδρυμα ή τα ιδρύματα στα οποία ανήκει ο συγγραφέας. Αναλυτικότερα, περιγράφονται το «url» (ιστότοπος) του ιδρύματος, το μοναδικό αναγνωριστικό που του έχει ορίσει η «scopus» («affiliation-id»), το όνομά του, η πόλη και η χώρα όπου βρίσκεται.

Affiliation Search API:

Αντίστοιχα, με το «author search API», το οποίο αναλύθηκε παραπάνω, η Scopus προσφέρει και την αναζήτηση ιδρυμάτων μέσω του API της.

HTTP REQUEST:

Το «http request» του «affiliation search api» συντάσσεται ως εξής :

```
curl -X GET --header 'Accept: text/xml' 'https://api.elsevier.com/content/search/affiliation?query=affil(UNIWA)&apiKey=7f59af901d2
```

Όπως και προηγουμένως η αναζήτηση γίνεται με βάση ένα ερώτημα - παράμετρο και το «api-key» ως απαραίτητα πεδία για τη διεκπεραίωση του αιτήματος. Οι υπόλοιπες παράμετροι, δε χρησιμοποιήθηκαν, όπως και στην προηγούμενη περίπτωση καθώς είναι προαιρετικές.

Response Content Type application/xml

Parameters

Parameter	Value	Description	Parameter Type	Data Type
query	affil(UNIWA)	**Affiliation search query string**	query	string
apiKey	7f59af901d2d86f78a1fd60c1bf9426a	**Your API key**	query	string
httpAccept		Requested content type, overrides HTTP header value	query	string
insttoken		Specification for authorization, institution authtoken	query	string
access_token		Specification for active session, secured authtoken	query	string

Εικόνα 4 Περιγραφή http request από την ιστοσελίδα της Scopus για το Affiliation Search API.

HTTP RESPONSE:

```
<opensearch:totalResults>468</opensearch:totalResults>
<opensearch:startIndex>0</opensearch:startIndex>
<opensearch:itemsPerPage>25</opensearch:itemsPerPage>
<opensearch:Query role="request" searchTerms="AFFIL(University of West Attica)" startPage="0"/>
```

Εικόνα 5 Περιγραφή xml απάντησης - Affiliation Search API

Στο πρώτο κομμάτι της απάντησης από ένα «affiliation search request» συναντούμε τον αριθμό των αποτελεσμάτων του ερωτήματος («total results»). Επίσης, παρατηρούμε των αριθμό των ιδρυμάτων (παρουσιάζονται ως «entries») που θέλουμε να εμφανίσουμε σε μια σελίδα, 25 στο παρακάτω παράδειγμα («items per page»). Επιπλέον, τα αποτελέσματα "επιστρέφονται" αριθμημένα από0 έως N, όπου N ισούται με τον συνολικό αριθμό των αποτελεσμάτων, μείον 1 (καθώς μετράμε από το 0), ανά 25 εγγραφές όπως αναφέρθηκε. Επομένως, στην περίπτωση που χρησιμοποιήσουμε το 0 ως δείκτη («start index»), θα πάρουμε πίσω τα «entries» με αρίθμηση από 0 έως 24. Για να προχωρήσουμε στα επόμενα 25, το μόνο που χρειάζεται είναι να προσθέσουμε στο «request» την παράμετρο «startIndex=25», ώστε να συνεχίσουμε από την 25η μέχρι τη 49η εγγραφή. Τέλος, στην παράμετρο AFFIL() μπορούμε να προσθέσουμε είτε την ονομασία του ιδρύματος που αναζητούμε (π.χ. «National Technical University of Athens»), είτε μια συντομογραφία (π.χ. «NTUA») του σε συνδυασμό με τη χώρα-περιοχή στην οποία βρίσκεται («Greece»).

1.6.2 Ανάκτηση πληροφοριών – Retrieval APIs

Το κομμάτι της ανάκτησης πληροφορίων από τη Scopus, περιλαμβάνει την ίδια διαδικασία, όπως και το κομμάτι της αναζήτησης, δηλαδή την αποστολή ενός «http request» (το οποίο δέχεται συγκεκριμένες παραμετροποιήσεις) και την παραλαβή μιας απάντησης με βάση αυτό, που περιέχει την πληροφορία που ζητήθηκε. Τα «retrieval APIs», δίνουν τη δυνατότητα στο χρήστη να έχει πρόσβαση σε μεγαλύτερο εύρος πληροφοριών σχετικά με τις οντότητες που περιγράφονται από αυτά συγκριτικά με τα αποτελέσματα της αναζήτησης πληροφοριών. Σε αυτό το σημείο, είναι σημαντικό να αναφερθεί, πως η πρόσβαση σε αυτές τις εφαρμογές, προσφέρεται αποκλειστικά σε συνδρομητές της Scopus. Οι διαδικτυακές εφαρμογές ανάκτησης πληροφοριών χωρίζονται σε 3 κατηγορίες :

1. «Affiliation Retrieval API».
2. «Author Retrieval API».
3. «Abstract Retrieval API».

Η κάθε μια από τις παραπάνω κατηγορίες περιγράφεται από τη Scopus σε μορφή JSON, XML και XML-RDF, ανάλογα με την επιλογή του χρήστη. Ας αναλύσουμε, λοιπόν, την πληροφορία που δεχόμαστε για την κάθε εφαρμογή ξεχωριστά.

Affiliation Retrieval API :

Το «affiliation retrieval api» της scopus, προσφέρει πληροφορίες σχετικά με ένα ακαδημαϊκό ίδρυμα ή σχολή ή τμήμα ή εργαστήριο. Σε αντίθεση με το αντίστοιχο «search api» περιέχει περισσότερες πληροφορίες σχετικά με την οντότητα που αναζητείται.

Ας αναλύσουμε την πληροφορία μέσα από ένα παράδειγμα.

HTTP REQUEST :

Request URL

```
https://api.elsevier.com/content/affiliation/affiliation_id/60110806?apiKey=7f59af901d2d86f78a1fd60c1bf9426a
```

ΠΑΡΑΜΕΤΡΟΙ ΕΡΩΤΗΜΑΤΟΣ :

Parameters

Parameter	Value	Description	Parameter Type	Data Type
affiliation_id	60110806	**affiliation_id value**	path	string
apiKey	7f59af901d2d86f78a1fd60c1bf9426a	**Your API key**	query	string
httpAccept	text/xml	Requested content type, overrides HTTP header value	query	string
insttoken		Specification for authorization, institution authtoken	query	string
access_token		Specification for active session, secured authtoken	query	string

Εικόνα 6 Περιγραφή http request, Affiliation Retrieval API

Αρχικά, το «http-request» της εφαρμογής απαιτεί ως απαραίτητη παράμετρο το «affiliation_id» ενός ακαδημαϊκού ιδρύματος ή σχολής κ.ο.κ. το οποίο αποτελεί ένα μοναδικό χαρακτηριστικό που ορίζεται από τη Scopus. Επιπλέον, όπως και σε όλες τις υπόλοιπες περιπτώσεις απαιτείται η προσθήκη του «api key» έτσι ώστε να ληφθεί η απάντηση («http response»). Τέλος, αφού οριστεί το αλφαριθμητικό κλειδί της scopus , ο χρήστης ορίζει τη μορφή των δεδομένων που θα «παραλάβει» στην απάντηση, μέσω της παραμέτρου «httpAccept», επιλέγοντας μια από τις επιλογές που αναφέρθηκαν παραπάνω (xml κ.α).

Αφού το ερώτημα αποσταλεί επιτυχώς με βάση τις παραπάνω παραμέτρους, δεχόμαστε την εξής απάντηση :

```
<affiliation-retrieval-response xmlns:ait="http://www.elsevier.com/xml/ani/ait" xmlns:ce="http://www.elsevier.com/xml/ani/common" xmlns:cto="http://www.elsevier.com/xml/cto/dtd"
 xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:prism="http://prismstandard.org/namespaces/basic/2.0/" xmlns:xocs="http://www.elsevier.com/xml/xocs/dtd"
 xmlns:xoe="http://www.elsevier.com/xml/xoe/dtd">
  <coredata>
    <prism:url>
      https://api.elsevier.com/content/affiliation/affiliation_id/60110806
    </prism:url>
    <dc:identifier>AFFILIATION_ID:60110806</dc:identifier>
    <eid>10-s2.0-60110806</eid>
    <author-count>1762</author-count>
    <document-count>3920</document-count>
    <link href="https://api.elsevier.com/content/affiliation/affiliation_id/60110806" rel="self"/>
    <link href="https://api.elsevier.com/content/search/scopus?query=af-id%2860110806%29" rel="search"/>
    <link href="https://www.scopus.com/affil/profile.uri?afid=60110806&partnerID=HzOxMe3b&origin=inward" rel="scopus-affiliation"/>
  </coredata>
  <affiliation-name>University of West Attica</affiliation-name>
  <name-variants>
    <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="1414" source="internal-ani">Technological Educational Institute Of Athens</name-variant>
    <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="721" source="internal-ani">Technological Educational Institution Of Athens</name-variant>
    <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="574" source="internal-ani">University Of West Attica</name-variant>
    <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="309" source="internal-ani">Technological Education Institute Of Piraeus</name-variant>
    <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="209" source="internal-ani">Technological Educational Institute Of Piraeus</name-variant>
    <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="190" source="internal-ani">
      Technological Educational Institution (tei) Of Athens
    </name-variant>
    <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="150" source="internal-ani">Technological Education Institute Of Athens</name-variant>
    <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="133" source="internal-ani">
      Technological Educational Institute (tei) Of Athens
    </name-variant>
    <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="119" source="internal-ani">Technol. Educ. Institution Of Athens</name-variant>
    <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="101" source="internal-ani">Tei Of Athens</name-variant>
  </name-variants>
  <address>250, Thivon & P. Ralli Street, Aigaleo,</address>
  <city>Athens</city>
  <country>Greece</country>
  <institution-profile affiliation-id="60110806">
    <status xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">update</status>
    <date-created xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" day="07" month="04" year="2018"/>
    <preferred-name xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" source="internal-ani">University of West Attica</preferred-name>
    <sort-name xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">West Attica, University of</sort-name>
    <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="1414" source="internal-ani">Technological Educational Institute Of Athens</name-variant>
    <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="721" source="internal-ani">Technological Educational Institution Of Athens</name-variant>
    <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="574" source="internal-ani">University Of West Attica</name-variant>
    <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="309" source="internal-ani">Technological Education Institute Of Piraeus</name-variant>
    <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="209" source="internal-ani">Technological Educational Institute Of Piraeus</name-variant>
    <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="190" source="internal-ani">
      Technological Educational Institution (tei) Of Athens
    </name-variant>
    <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="150" source="internal-ani">Technological Education Institute Of Athens</name-variant>
    <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="133" source="internal-ani">
      Technological Educational Institute (tei) Of Athens
    </name-variant>
    <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="119" source="internal-ani">Technol. Educ. Institution Of Athens</name-variant>
    <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="101" source="internal-ani">Tei Of Athens</name-variant>
    <address xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" country="grc">
      <address-part>250, Thivon & P. Ralli Street, Aigaleo,</address-part>
      <city>Athens</city>
      <state>Attica</state>
      <postal-code>12244</postal-code>
      <country>Greece</country>
    </address>
    <org-type xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">univ</org-type>
    <org-domain xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">aegeanconference.org</org-domain>
    <org-URL xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
      https://aegeanconference.org/content/university-west-attica-uniwa
    </org-URL>
  </institution-profile>
</affiliation-retrieval-response>
```


Εικόνα 7 Περιγραφή XML απάντησης - Affiliation Retrieval API.

Όπως βλέπουμε στην παραπάνω «απάντηση», πραγματοποιείται η περιγραφή του εκπαιδευτικού ιδρύματος «Πανεπιστήμιο Δυτικής Αττικής» («University of West Attica»), σύμφωνα με τα δεδομένα που έχουν συλλεχθεί από τη «Scopus» για αυτό. Όλη η πληροφορία είναι δομημένη με βάση τρεις γονικούς κόμβους : Τον “coredata”, τον “names-variants” και τον «institution profile». Οι τρείς αυτοί κόμβοι αποτελούν υπό-κόμβους του κύριου γονικού κόμβου «affiliation-retrieval-response». Ο πρώτος γονικός κόμβος («coredata»), περιέχει γενικές πληροφορίες για το ίδρυμα, όπως τα διάφορα αναγνωριστικά του («scopus-id, e-id, orcid») καθώς και το σύνολο τον ερευνητών που εργάζονται σε αυτό και τον συνολικό

αριθμό των δημοσιεύσεων τους. Ο δεύτερος γονικός κόμβος («name-variant»), περιέχει πληροφορίες σχετικά με τους διαφορετικούς τρόπους καταγραφής του ονόματος του ιδρύματος (τμήματος ή σχολής που αναζητήθηκε). Επιπρόσθετα, παρουσιάζεται και ο αριθμός των δημοσιεύσεων που έχουν καταγραφεί ανά παραλλαγή ονόματος, ως xml-χαρακτηριστικό («xml-attribute»). Τέλος, ο τρίτος γονικός κόμβος αναφέρεται σε πληροφορίες όπως η διεύθυνση, η χώρα, ο ταχυδρομικός κώδικας και ο ιστότοπος του ιδρύματος, ενώ υπάρχει και μια διπλότυπη εγγραφή του κόμβου «name-variant» που συναντήσαμε παραπάνω.

Abstract Retrieval API :

Το «abstract retrieval api» της «scopus», επιστρέφει πληροφορίες σχετικά με την περίληψη μιας δημοσίευσης. Η πλήρης περίληψη περιέχει πληθώρα μεταδεδομένων, τα οποία με τη σειρά τους, αντιστοιχούν σε προφίλ συγγραφέων και ιδρυμάτων.

HTTP REQUEST :

Request URL

```
https://api.elsevier.com/content/abstract/doi/10.1016/S0014-5793(01)03313-0?apiKey=7f59af901d2d86f78a1fd60c1bf9426a
```

Response Body

ΠΑΡΑΜΕΤΡΟΙ:

Parameters

Parameter	Value	Description	Parameter Type	Data Type
doi	10.1016/S0014-5793(01)03313-0	**doi value**	path	string
apiKey	7f59af901d2d86f78a1fd60c1bf9426a	**Your API key**	query	string
httpAccept		Requested content type, overrides HTTP header value	query	string
insttoken		Specification for authorization, institution authtoken	query	string
access_token		Specification for active session, secured authtoken	query	string

Εικόνα 8 Παράδειγμα http request - Abstract Retrieval API

Author Retrieval API :

Το συγκεκριμένο API, χρησιμοποιήθηκε για τη δημιουργία του εργαλείου άντλησης βιβλιομετρικών δεδομένων που θα αναλυθεί στη συνέχεια.

Κεφάλαιο 2. Μοντέλο Δεδομένων για την συλλογή των πληροφοριών

2.1 MySQL και τα εργαλεία της

Η «MySQL» δημιουργήθηκε απο μια Σουηδική εταιρεία, τη «MySQL AB», απο τους David Axmark, Allan Larsson και Michael «Monty» Widenius. Τα αρχικά «My» προέρχονται από το όνομα της κόρης του συνιδρυτή του συστήματος , Michael Widenius και το «SQL» προέρχεται από τη γνωστή γλώσσα προγραμματισμού Structured Query Language ή εν συντομία SQL (Oracle, 2020).

Το ακρωνύμιο «MySQL» αναφέρεται σε ένα «open-source» σύστημα οντοτήτων-συσχετίσεων για την αποθήκευση και διαχείριση δεδομένων. Επιπλέον, περιλαμβάνεται στο open-source «πακέτο» «LAMP» που αποτελεί το ακρωνύμιο των «Linux, Apache, MySQL και Perl / Php / Python». Η πρώτη «version» της γλώσσας «MySQL» κυκλοφόρησε στις 23 Μαΐου του 1995, όπου και χρησιμοποιήθηκε για προσωπική χρήση, βασισμένη και γραμμένη στη «low-level» γλώσσα προγραμματισμού «ISAM», την οποία οι ιδρυτές της «MySQL» θεώρησαν αργή και δύσχρηστη. Έτσι στη συνέχεια, ο κώδικας αναδιαρθρώθηκε με βάσεις τις γλώσσες C και C++. Το λογισμικό από την πρώτη εμπορική κυκλοφορία του, στη συνέχεια του 1995, απέκτησε μεγάλη απήχηση στον τομέα της πληροφορικής με αποτέλεσμα την ευρεία χρήση του σε ακαδημαϊκό αλλά και επαγγελματικό επίπεδο. Το 2008, η «MySQL AB» πουλήθηκε στη «Sun Microsystems» έναντι ενός δισεκατομμυρίου αμερικανικών δολαρίων. Το 2010, ωστόσο, η «Oracle» εξαγόρασε τη «Sun Microsystems». (Krill, 2010)

Τα βασικά εργαλεία που αντιπροσωπεύουν την εταιρεία είναι ο «MySQL Server» και η δομημένα γλώσσα ερωτημάτων «MySQL», όπως προαναφέρθηκε. Στην παρούσα εργασία, χρησιμοποιήθηκαν τα εργαλεία «MySQL Workbench» και «MySQL Server» (version 5.6). Το «MySQL Workbench» αποτελεί ένα πρόγραμμα που επιτρέπει στο χρήστη να δημιουργεί, τροποποιεί και να διαχειρίζεται βάσεις δεδομένων τύπου οντοτήτων - συσχετίσεων μέσα από ένα «GUI» («graphical user interface»), επιτυγχάνοντας ταυτόχρονα υψηλή λειτουργικότητα και ευκολία στη χρήση, ακόμα και από αρχάριους χρήστες. Παράλληλα, δίνει τη δυνατότητα για την αυτόματη παραγωγή μοντέλων δεδομένων σε μορφή εικόνας («reverse engineering diagram»), αλλά και επιτρέπει την άμεση επεξεργασία των δεδομένων μέσω της γλώσσας προγραμματισμού «MySQL». Ο «MySQL Server» είναι το λογισμικό που εγκαθίσταται τοπικά στον υπολογιστή το οποίο είναι υπεύθυνο για τη συνολική διαχείριση των πρωτοκόλλων επικοινωνίας μεταξύ των διαφόρων επιπέδων λογισμικού (π.χ. επικοινωνία εφαρμογής συλλογής δεδομένων - βάσης δεδομένων). Ο «MySQL Workbench», συνεπώς, δεν μπορεί να λειτουργήσει χωρίς την προ-εγκατάσταση του Server.

2.2 Παράδειγμα και ανάλυση του XML.

Στην παρούσα εργασία χρησιμοποιήθηκε το «Author Retrieval API» της «scopus». Όπως αναλύθηκε παραπάνω, η απάντηση που δεχόμαστε από την εφαρμογή της «scopus», αφορά αποκλειστικά τον συγγραφέα , σύμφωνα με το «scopus id» του οποίου, αποστείλαμε στο «request» μας. Σε αυτό το κομμάτι, θα αναλύσουμε τα πεδία της απάντησης των δεδομένων, η μορφή των οποίων ορίστηκε σε XML:

```
<author-retrieval-response xmlns:ait="http://www.elsevier.com/xml/ani/ait" xmlns:ce="http://www.elsevier.com/xml/ani/common" xmlns:cto="http://www.
xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:ns1="http://webservices.elsevier.com/schemas/search/fast/types/v4" xmlns:prism="http://prismstand
xmlns:xocs="http://www.elsevier.com/xml/xocs/dtd" xmlns:xoe="http://www.elsevier.com/xml/xoe/dtd" status="found">
 <coredata>
  <prism:url>
   http://api.elsevier.com/content/author/author_id/6507652923
  </prism:url>
  <dc:identifier>AUTHOR_ID:6507652923</dc:identifier>
  <eid>9-s2.0-6507652923</eid>
  <orcid>0000-0002-5948-9766</orcid>
  <document-count>13</document-count>
  <cited-by-count>27</cited-by-count>
  <citation-count>28</citation-count>
  <link href="https://www.scopus.com/authid/detail.uri?partnerID=HzOxMe3b&authorId=6507652923&origin=inward" rel="scopus-author"/>
  <link href="http://api.elsevier.com/content/author/author_id/6507652923" rel="self"/>
  <link href="http://api.elsevier.com/content/search/scopus?query=au-id%286507652923%29" rel="search"/>
 </coredata>
 <affiliation-current id="120863822" href="http://api.elsevier.com/content/affiliation/affiliation_id/120863822"/>
 <affiliation-history>
  <affiliation id="60002947" href="http://api.elsevier.com/content/affiliation/affiliation_id/60002947"/>
 </affiliation-history>
 <subject-areas>
  <subject-area abbrev="ENGI" code="2208">Electrical and Electronic Engineering</subject-area>
  <subject-area abbrev="SOCI" code="3321">Public Administration</subject-area>
  <subject-area abbrev="COMP" code="1700">Computer Science (all)</subject-area>
  <subject-area abbrev="COMP" code="1705">Computer Networks and Communications</subject-area>
  <subject-area abbrev="COMP" code="1710">Information Systems</subject-area>
  <subject-area abbrev="COMP" code="1702">Artificial Intelligence</subject-area>
  <subject-area abbrev="COMP" code="1707">Computer Vision and Pattern Recognition</subject-area>
  <subject-area abbrev="COMP" code="1712">Software</subject-area>
  <subject-area abbrev="MATH" code="2600">Mathematics (all)</subject-area>
  <subject-area abbrev="SOCI" code="3309">Library and Information Sciences</subject-area>
  <subject-area abbrev="COMP" code="1709">Human-Computer Interaction</subject-area>
  <subject-area abbrev="SOCI" code="3304">Education</subject-area>
  <subject-area abbrev="ENGI" code="2200">Engineering (all)</subject-area>
 </subject-areas>
```


Εικόνα 9 Ανάλυση πεδίων xml – Author Retrieval API

Όπως σε κάθε δενδροειδή δομή, έτσι και στην περιγραφική γλώσσα XML, έχουμε γονικούς και παιδικούς κόμβους οι οποίοι σε διαφορετικές αλληλουχίες περιγράφουν και συνδέουν την πληροφορία. Όπως διακρίνουμε στην παραπάνω εικόνα, η πληροφορία που αφορά έναν συγγραφέα, βρίσκεται κάτω από το πεδίο “author-retrieval-response”. Το συγκεκριμένο πεδίο αποτελεί και το γονικό κόμβο του συνόλου των δεδομένων μας. Σε αυτό τον κόμβο υπάγονται όλοι οι υπόλοιποι παιδικοί κόμβοι που θα αναλύσουμε στη συνέχεια. Ορισμένοι παιδικοί κόμβοι με τη σειρά τους αποτελούν γονικούς κόμβους που αποτελούνται από άλλους παιδικούς κόμβους κ.ο.κ. Στη συγκεκριμένη περίπτωση, η πληροφορία μας χωρίζεται σε πέντε βασικούς γονικούς κόμβους, οι οποίοι παράλληλα, αποτελούν τους παιδικούς κόμβους του πεδίου “author-retrieval-response”, όπως προαναφέρθηκε ενώ τέσσερις από αυτούς αναγράφονται στην παρακάτω εικόνα.


```
This XML file does not appear to have any style information associated with it. The document tree is shown below.

<author-retrieval-response xmlns:ait="http://www.elsevier.com/xml/ani/ait" xmlns:ce="http://www.elsevier.com/xml/ani/common" xmlns:cto="http://www.elsevie
xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:ns1="http://webservices.elsevier.com/schemas/search/fast/types/v4" xmlns:prism="http://prismstandard.org
xmlns:xocs="http://www.elsevier.com/xml/xocs/dtd" xmlns:xoe="http://www.elsevier.com/xml/xoe/dtd" status="found">
 <coredata>...</coredata>
  <affiliation-current id="120863822" href="http://api.elsevier.com/content/affiliation/affiliation_id/120863822"/>
 <affiliation-history>...</affiliation-history>
 <subject-areas>...</subject-areas>
 <author-profile>...</author-profile>
</author-retrieval-response>
```


Εικόνα 10 Κύριοι γονικοί κόμβοι απάντησης XML

Ο πρώτος από αυτούς ονομάζεται «coredata» και σε αυτόν, περιγράφονται :

- Το «url» της καταχωρημένης σελίδας που αφορά τον συγγραφέα μας στον ιστότοπο της «Scopus» από το πεδίο «prism-url».
- Το «scopus id» του συγγραφέα από το πεδίο «dc:identifier».
- Το «electronic id» του συγγραφέα από το πεδίο «eid»
- Το «orc_id» του συγγραφέα από το πεδίο «orcid»
- Το συνολικό αριθμό των δημοσιεύσεων του συγγραφέα από το πεδίο «documents-count».

- Το συνολικό αριθμό των αναφορών του συγγραφέα από το πεδίο «citations-count».
- Το συνολικό αριθμό των αναφορών του συγγραφέα εξαιρουμένων των αυτό-αναφορών.

Όπως γίνεται εύκολα αντιληπτό, ο παραπάνω κόμβος αφορά γενικές πληροφορίες, οι οποίες περιγράφουν τον συγγραφέα, όπως τα διάφορα αναγνωριστικά του και τα βασικά βιβλιομετρικά στοιχεία των δημοσιεύσεών του.

Ο επόμενος γονικός κόμβος που θα αναλυθεί ονομάζεται «affiliation-current». Σε αυτόν περιγράφεται το αναγνωριστικό καθώς και το όνομα του ιδρύματος στο οποίο είναι μέλος – εργάζεται ο ερευνητής την παρούσα περίοδο.

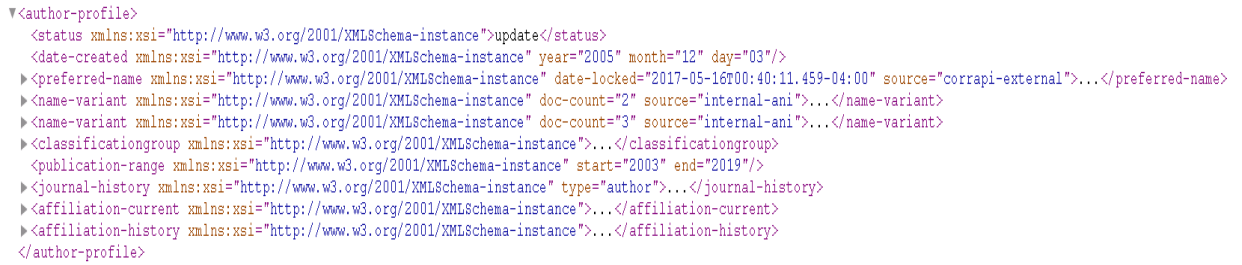
Ο επόμενος γονικός κόμβος, είναι ο «affiliation-history». Όπως υποδηλώνει η ονομασία του, σε αυτόν περιγράφονται τα αναγνωριστικά και οι ονομασίες των ιδρυμάτων, στα οποία έχει υπάρξει μέλος στο παρελθόν ο ερευνητής. Οι παραπάνω πληροφορίες, βρίσκονται μέσω στον παιδικό κόμβο του «affiliations-history», ο οποίος ονομάζεται affiliations. Επίσης σημαντικό είναι να αναφερθεί πως ο κόμβος «affiliations», περιέχει το «xml-χαρακτηριστικό»

(xml-attribute) «id», το οποίο σαν επιπρόσθετη πληροφορία περιγράφει το αναγνωριστικό του ιδρύματος, όπως προαναφέρθηκε.

Στη συνέχεια συναντούμε τον τέταρτο γονικό κόμβο της περιγραφής ενός συγγραφέα, με όνομα «subject areas» και αναφέρεται στις θεματικές περιοχές στις οποίες έχει δημοσιεύσει ο ερευνητής. Σε αυτόν τον κόμβο συναντούμε πολλούς παιδικούς κόμβους με το όνομα «subject-area», οι οποίο περιγράφονται περεταίρω από τα χαρακτηριστικά τους «abbrev» και «code». Με βάση τα παραπάνω η πληροφορία διαμορφώνεται ως :

Συντομογραφία θέματος («abbrev») | Κωδικός θέματος («code») | Θεματική κατηγορία («subject-area»).

Τέλος, θα αναλύσουμε την πληροφορία η οποία βρίσκεται στον τελευταίο γονικό κόμβο με ονομασία «author-profile», ο οποίος είναι και ο μεγαλύτερος σε έκταση, καθώς και περιγράφει το μεγαλύτερο κομμάτι των δεδομένων ενός ερευνητή. Περιέχει τα παρακάτω πεδία:



Εικόνα 11 Ανάλυση κυρίων πεδίων του γονικού κόμβου author profile

Ξεκινάμε με το πεδίο status, το οποίο αναφέρεται στο αν η πληροφορία που αναζητεί ο χρήστης, έχει ανανεωθεί από την πρώτη φορά που καταχωρήθηκε στη Scopus. Στη συνέχεια συναντούμε το πεδίο «date-created» , που αναφέρεται στην ημερομηνία δημιουργίας της πρώτης καταχώρησης που έγινε στη Scopus, για τον ερευνητή. Έπειτα, θα αναλύσουμε το πεδίο «preferred-name», το οποίο αναφέρεται στην προτιμώμενο τρόπο αναγραφής του ονοματεπωνύμου του ερευνητή, καθώς και το πεδίο «name variant», το οποίο αναφέρεται στους διαφορετικούς τρόπους με τους οποίους έχει καταχωρηθεί το ονοματεπώνυμο στις δημοσιεύσεις του.

```
<author-profile>
  <status xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">update</status>
  <date-created xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" year="2005" month="12" day="03"/>
  <preferred-name xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" date-locked="2017-05-16T00:40:11.459-04:00" source="corrapi-external">
    <initials>D.</initials>
    <indexed-name>Kouis D.</indexed-name>
    <surname>Kouis</surname>
    <given-name>Dimitris</given-name>
  </preferred-name>
  <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="2" source="internal-ani">
    <initials>D.</initials>
    <indexed-name>Kouis D.</indexed-name>
    <surname>Kouis</surname>
    <given-name>Dimitrios</given-name>
  </name-variant>
  <name-variant xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" doc-count="3" source="internal-ani">
    <initials>D.</initials>
    <indexed-name>Kouis D.</indexed-name>
    <surname>Kouis</surname>
    <given-name>D.</given-name>
  </name-variant>
```


Εικόνα 12 Παραλλαγές ονόματος συγγραφέα

Ο γονικός κόμβος «preferred-name», περιέχει τους υπό-κόμβους «initials» που περιγράφει τα αρχικά του συγγραφέα , «indexed-name» που περιγράφει το καταχωρημένο όνομα του συγγραφέα στη Scopus, «surname» που περιγράφει το καταχωρημένο επώνυμο του συγγραφέα στη Scopus και «given-name» που αναφέρεται και πάλι στο όνομα του συγγραφέα. Αντίστοιχα, στους κόμβους «name-variant», θα συναντήσουμε ακριβώς τα ίδια πεδία με παραλλαγές στο περιεχόμενο, καθώς αφορά διαφορετικούς τρόπους καταγραφής του ονοματεπωνύμου του ερευνητή, με μοναδική διαφορά, ότι ο κόμβος έχει το χαρακτηριστικό «doc-count», που αναφέρεται στον αριθμό των δημοσιεύσεων που έχει χρησιμοποιηθεί η περιγραφόμενη παραλλαγή.

Συνεχίζοντας, θα αναλύσουμε τον κόμβο «classificationgroup», ο οποίος αναφέρεται στις δημοσιεύσεις που έχει ο κάθε ερευνητής ανά θεματική περιοχή. Ο τρόπος με τον οποίο συνδέεται η θεματική περιοχή με το παρακάτω σχήμα δεδομένων, είναι μέσω των μοναδικών τετραψήφιων κωδικών («subject-code») που αντιστοιχούν σε θεματικές κατηγορίες.


```
<classificationgroup xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <classifications type="ASJC">
    <classification frequency="4">2208</classification>
    <classification frequency="1">3321</classification>
    <classification frequency="2">1700</classification>
    <classification frequency="3">1705</classification>
    <classification frequency="1">1710</classification>
    <classification frequency="1">1702</classification>
    <classification frequency="1">1707</classification>
    <classification frequency="1">1712</classification>
    <classification frequency="1">2600</classification>
    <classification frequency="4">3309</classification>
    <classification frequency="1">1709</classification>
    <classification frequency="1">3304</classification>
    <classification frequency="2">2200</classification>
  </classifications>
</classificationgroup>
<publication-range xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" start="2003" end="2019"/>
```


Εικόνα 13 Αριθμός δημοσιεύσεων συγγραφέα ανά κωδικό θέματος

Σημαντικό είναι να αναφερθεί πως (όπως και προηγουμένως) συναντάμε στον παιδικό κόμβο («classification») επιπρόσθετη πληροφορία μέσω ενός χαρακτηριστικού, του «frequency» σε αυτή την περίπτωση, που αντιστοιχεί στον αριθμό των δημοσιεύσεων.

Αντίστοιχα, θα αναλύσουμε τον επόμενο γονικό κόμβο με όνομα «journal-history». Ο κόμβος αναφέρεται στις δημοσιεύσεις που έχει ο συγγραφέας σε περιοδικά, καθώς και τον τύπο της δημοσίευσης.

```
<journal-history xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" type="author">
  <journal type="b">
    <sourcetitle>
      Recent Advances in Communications and Computer Science
    </sourcetitle>
    <sourcetitle-abbrev>Recent Adv. Commun. Comput. Sci.</sourcetitle-abbrev>
  </journal>
  <journal type="p">
    <sourcetitle>IEE Proceedings: Communications</sourcetitle>
    <sourcetitle-abbrev>IEE Proc Commun</sourcetitle-abbrev>
    <issn>13502425</issn>
  </journal>
  <journal type="j">
    <sourcetitle>
      International Journal of Wireless and Mobile Computing
    </sourcetitle>
    <sourcetitle-abbrev>Int. J. Wireless Mobile Comput.</sourcetitle-abbrev>
    <issn>17411084</issn>
  </journal>
  <journal type="j">
    <sourcetitle>Education and Information Technologies</sourcetitle>
    <sourcetitle-abbrev>Educ. Inf. Technol.</sourcetitle-abbrev>
    <issn>15737608</issn>
  </journal>
  <journal type="j">
    <sourcetitle>Bilgi Dunyasi</sourcetitle>
    <sourcetitle-abbrev>Bilgi Dunyasi</sourcetitle-abbrev>
    <issn>2148354X</issn>
  </journal>
  <journal type="j">
    <sourcetitle>
      International Journal on Artificial Intelligence Tools
    </sourcetitle>
    <sourcetitle-abbrev>Int. J. on Artif. Intell. Tools</sourcetitle-abbrev>
    <issn>17936349</issn>
  </journal>
  <journal type="p">
    <sourcetitle>
      GLOBECOM - IEEE Global Telecommunications Conference
    </sourcetitle>
    <sourcetitle-abbrev>GLOBECOM IEEE Global Telecommun. Conf.</sourcetitle-abbrev>
  </journal>
  <journal type="p">
    <sourcetitle>ACM International Conference Proceeding Series</sourcetitle>
    <sourcetitle-abbrev>ACM Int. Conf. Proc. Ser.</sourcetitle-abbrev>
  </journal>
  <journal type="k">
    <sourcetitle>Communications in Computer and Information Science</sourcetitle>
    <sourcetitle-abbrev>Commun. Comput. Info. Sci.</sourcetitle-abbrev>
    <issn>18650929</issn>
  </journal>
```


Εικόνα 14 Περιοδικά στα οποία έχει δημοσιεύσει ο συγγραφέας

Όπως παρατηρούμε ο κόμβος αποτελείται από παιδικούς κόμβους με ονομασία «journal», οι οποίοι με τη σειρά τους αποτελούν γονικούς κόμβους στα πεδία «sourcetitle», «sourcetitle-abbrev» και «issn». Κάθε εγγραφή «journal» έχει ως χαρακτηριστικό το πεδίο «type» , το οποίο περιγράφει τον τύπο της δημοσίευσης, ενώ τα υπόλοιπα πεδία περιγράφουν τον τίτλο, τη συντομογραφία του και το issn της δημοσίευσης.

Τέλος, θα αναφερθούμε στους κόμβους «affiliation-current» και «affiliation-history». Τους δύο αυτούς κόμβους τους συναντήσαμε και στην αρχή του XML (χωρίς να περιέχουν παιδικούς κόμβους), όπου αναφέρθηκε πως ο πρώτος σχετίζεται με το τρέχον ίδρυμα του ερευνητή ενώ ο δεύτερος με τα ιδρύματα στα οποία υπήρξε στο παρελθόν, όμως, στη συγκεκριμένη περίπτωση γίνεται αναφορά και στα τμήματα ή σχολές ή εργαστήρια ή και ιδρύματα στα οποία εργάστηκε ο ερευνητής στο παρελθόν. Με αυτό τον τρόπο, επομένως, μπορούμε να αποκτήσουμε μια πιο ολοκληρωμένη εικόνα για το ακαδημαϊκό προφίλ του συγγραφέα καθώς και το ιστορικό του.

Εικόνα 15 Επαγγελματικό ιστορικό στον ακαδημαϊκό χώρο για τον συγγραφέα

Πιο συγκεκριμένα, και οι δύο κόμβοι περιέχουν τον υπό-κόμβο «affiliation», ο οποίος έχει τα χαρακτηριστικά «id» και «parent», τα οποία αποτελούν τα αναγνωριστικά για το κάθε τμήμα ή σχολή. Μάλιστα το χαρακτηριστικό «parent», όπως μπορεί να γίνει εύκολα αντιληπτό, αναφέρεται στο ίδρυμα στο οποίο υπάγεται το εργαστήριο ή τμήμα ή σχολή που περιγράφεται. Επιπρόσθετα, ο υπό-κόμβος «affiliation» αποτελεί και γονικό κόμβο για τον υπό-κόμβο «ip-doc» στον οποίο περιέχεται όλη η πληροφορία που αναφέρθηκε παραπάνω. Πιο αναλυτικά , στον κόμβο «ip-doc» συναντούμε τα πεδία :

- «afdispname» που περιγράφει το όνομα με του ακαδημαϊκού τμήματος, ή σχολής, ή εργαστηρίου, ή ιδρύματος.
- «preferred-name» που περιγράφει τον προτιμώμενο τρόπο γραπτής αναφοράς του ονόματος του τμήματος κ.ο.κ.
- «parent-preferred-name» που περιγράφει το όνομα του ιδρύματος στο οποίο υπάγεται το τμήμα ή σχολή ή εργαστήριο ή ίδρυμα, το οποίο περιγράφεται.
- «sort-name» που περιγράφει το σύντομο όνομα του τμήματος κ.ο.κ.
- «address» που αποτελεί έναν νέο γονικό κόμβο για τα πεδία :
    - «address-part» που περιγράφει την πλήρη διεύθυνση της περιγραφόμενης οντότητας
    - «city» που περιγράφει την πόλη στην οποία βρίσκεται η περιγραφόμενη οντότητα.
    - «state» που περιγράφει την περιφέρεια στην οποία υπάγεται η περιγραφόμενη οντότητα.
    - «postal-code» που περιγράφει τον ταχυδρομικό κώδικα της περιγραφόμενης οντότητας.

- «country» που περιγράφει τη χώρα στην οποία βρίσκεται.

- «org-domain» που περιγράφει τον διαδικτυακό ιστότοπο που μπορεί να βρεθεί η περιγραφόμενη οντότητα.
- «org-URL» που περιγράφει την αποκλειστική ιστοσελίδα της περιγραφόμενης οντότητας.

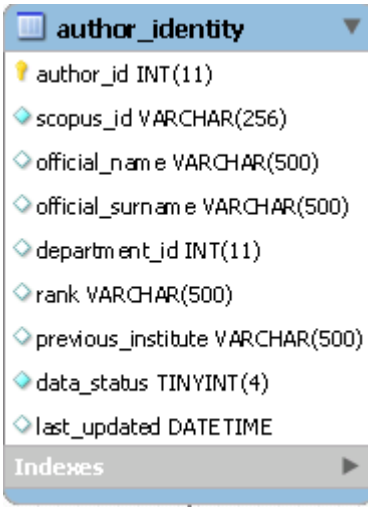
Σε αυτό το σημείο έχει περιγραφεί ολόκληρη η πληροφορία που αντλούμε από την εφαρμογή της Scopus. Ο τρόπος με τον οποίο έχει δομηθεί μπορεί να κάνει πολύ εύκολα αντιληπτή την περιγραφή της ακόμα και σε ένα φυσικό πρόσωπο, παρόλο που η μορφή περιγραφής των δεδομένων θα χρησιμοποιηθεί ως γλώσσα επικοινωνίας μεταξύ «μηχανών» στην εφαρμογή που δημιουργήθηκε.

2.3 Μοντέλο δεδομένων νέας εφαρμογής

Η Β.Δ. δημιουργήθηκε με σκοπό την άντληση και ταξινόμηση της πληροφορίας σε περιβάλλον οντοτήτων - συσχετίσεων, έτσι ώστε τα δεδομένα να αποθηκεύονται σε ένα περιβάλλον - δομημένο σχεσιακά - το οποίο επιτρέπει την εύκολη κατανόηση και επεξεργασία τους. Όλοι οι πίνακες σχετίζονται μεταξύ τους, μέσω των διαφορετικών αναγνωριστικών τους (ids), τα οποία είναι τα κύρια κλειδιά τους και συνδέονται μέσω σχέσεων πληθικότητας.

Το μοντέλο των δεδομένων, συστάθηκε με τέτοιο τρόπο, ώστε να περιγράφεται αναλυτικά και ολοκληρωμένα η οντότητα του ερευνητή μέσα από τα δεδομένα που έχουν συλλεχθεί από τη Scopus. Η πληροφορία που αφορά τον κάθε συγγραφέα καταγράφεται κάτω από ένα ενιαίο “προφίλ” , το οποίο περιέχει λεπτομέρειες για την παρούσα, καθώς και την πεπερασμένη ακαδημαϊκή του πορεία.

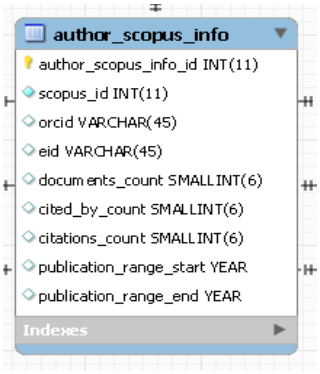
Η βασικότερη λειτουργία του, είναι η σύνδεση της πληροφορίας για τον κάθε συγγραφέα, η οποία σε αρκετές περιπτώσεις ενδέχεται να βρεθεί “διάσπαρτη”, καθιστώντας το έργο οποιουδήποτε την αναζητεί, αρκετά πιο περίπλοκο και χρονοβόρο.



Εικόνα 16 Πεδία πίνακα “Author Identity”

Ο πρώτος πίνακας με όνομα author_identity έχει ως σκοπό την περιγραφή της οντότητας του ερευνητή, όπως τον γνωρίζουμε απο τη Scopus. Το κύριο κλειδί του πίνακα είναι το αυτόματα αυξανόμενο πεδίο author_id, το οποίο στην ουσία αποτελεί αύξουσα αριθμητική ταξινόμηση από το 1 μέχρι το συνολικό αριθμό των εγγραφών του πίνακα. Επιπλέον περιλαμβάνει το πλήρες όνομα του κάθε ερευνητή (name, lastname) το τμήμα στο οποίο εργάζεται (department_id), την ημερομηνία (last_updated) κατά την οποία

δημιουργήθηκε η εγγραφή για τον καθένα καθώς και το scopus_id , που περιγράφει το ένα ή τα πολλαπλά διαφορετικά scopus_ids που ανήκουν σε έναν ερευνητή. Σε περίπτωση που υπάρχουν πολλαπλά scopus_ids στο πεδίο αναπαρίστανται διαχωρισμένα μέσω των χαρακτήρων “+OR+” (π.χ. 5123125+OR+25121521+OR+125361263 = 3 διαφορετικά scopus_ids). Περιέχει, επίσης, το πεδίο rank, το οποίο αναφέρεται στη βαθμίδα του ερευνητή στην ακαδημαϊκή κοινότητα, το οποίο αποτελεί το ξένο κλειδί που το συνδέει με τον πίνακα rank. Τέλος το πεδίο data_status μπορεί να πάρει τις τιμές 0 ή 1. Ανάλογα με το ποια τιμή έχει το παραπάνω πεδίο ο αλγόριθμος αποφασίζει αν θα ανανεώσει τα δεδομένα από την τελευταία φορά που τα συγκέντρωσε. Αυτή η λειτουργία θα διεκπεραιωθεί όταν στο πεδίο βρίσκεται ο αριθμός 1.



Εικόνα 17 πίνακα “author scopus info”

Στη συνέχεια, δημιουργήθηκε ο πίνακας author_scopus_info, ο οποίος συνδέει όλη την πληροφορία που λαμβάνεται από το scopus api μέσω του scopus_id, με το κάθε author_id (που ανήκει σε έναν ερευνητή). Το κύριο του κλειδί είναι το author_scopus_info_id (auto incremental identifier). Για κάθε διαφορετικό scopus_id, δημιουργείται μια νέα εγγραφή, η οποία συνδέεται μέσω του ξένου κλειδιού author_id με τον πίνακα author_identity. Ένας ερευνητής μπορεί να έχει πολλαπλά scopus ids, επομένως η σχέση πληθικότητας των δύο πινάκων είναι 1 / N.

Για παράδειγμα, έστω ότι ένας συγγραφέας έχει δύο διαφορετικά scopus_ids. Τα δεδομένα στον πίνακα author_identity θα παρουσιαστούν ως εξής :

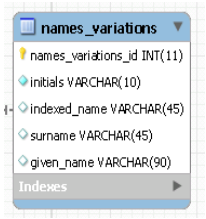
Πίνακας 1. AUTHOR IDENTITY TABLE - Παράδειγμα εγγραφής του πίνακα "author_identity"

author_id	official_name	official_surname	Scopus_id
1	Γιώργος	Γουργιωτόπουλος	12345+OR+67890

Στον πίνακα author_scopus_info τα διαφορετικά scopus_id θα διαχωριστούν και θα δημιουργήσουν 2 διαφορετικές εγγραφές με τον εξής τρόπο :

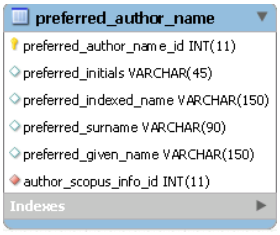
Πίνακας 2. AUTHOR SCOPUS INFO TABLE - Παράδειγμα εγγραφής του πίνακα "author_scopus_info"

author_scopus_info_id	scopus_id	author_Id
1	12345	1
2	67890	1



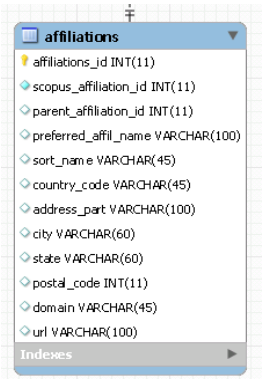
Εικόνα 18 Πεδία πίνακα "names variations"

Ο πίνακας names_variations, όπως υποδηλώνει το όνομα του περιέχει τις διαφορετικές παραλλαγές των ονομάτων τους, όπως έχουν καταγραφεί στις δημοσιεύσεις που βρίσκονται αποθηκευμένες στη scopus. Περιέχει το πεδίο initials, στο οποίο καταγράφονται τα "αρχικά" του κάθε ερευνητή, το πεδίο indexed_name, το οποίο είναι το όνομα του ερευνητή όπως είναι αποθηκευμένο στη Scopus, το πεδίο surname, που αναφέρεται στο επίθετό του και το πεδίο given name που περιέχει διάφορες παραλλαγές του ονόματός του (π.χ. Δημήτρης, Δημήτριος) .Συνδέεται μέσω του author_scopus_info_id ως ξένο κλειδί με τον πίνακα author_scopus_info. Ένας συγγραφέας μπορεί να έχει πολλές παραλλαγές ονόματος, επομένως η σχέση είναι πληθυκότητας των δύο πινάκων είναι 1 / N.



Εικόνα 19 Πεδία πίνακα "preferred_author_name"

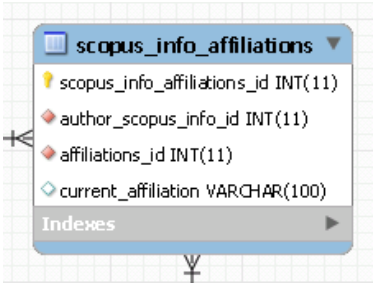
Ο πίνακας preferred_author_name έχει ως κύριο κλειδί το πεδίο preferred_author_name_id (auto incremental identifier). Ως πίνακας αναφέρεται στον τρόπο με τον οποίο προτιμά ο κάθε ερευνητής να εμφανίζεται το ονοματεπώνυμο του στις δημοσιεύσεις του. Περιέχει, τα ίδια πεδία με τον πίνακα names_variations και συνδέεται με τον πίνακα author_scopus_info μέσω του author_scopus_info_id. Η σχέση πληθικότητας είναι 1 / 1.



Εικόνα 20 Πεδία πίνακα "Affiliations"

Ο πίνακας affiliations, περιγράφει τα τμήματα-ιδρύματα για τα οποία έχει εργαστεί ο ερευνητής στο παρελθόν καθώς και αυτό στο οποίο εργάζεται τώρα (αν υπάρχει). Έχει ως κύριο κλειδί το πεδίο affiliations_id (auto incremental identifier). Το πεδίο preffered_affiliation_name περιέχει το όνομα με τον οποίο ένα ίδρυμα έχει "επιλέξει" να αναφέρεται στις δημοσιεύσεις του. Το πεδίο sort_name αποτελεί

τη συντομογραφία του οργανισμού. Το πεδίο parent_affiliation_id αναφέρεται στο ίδρυμα όπου ανήκει ένα τμήμα. Έπειτα, περιγράφονται ο κωδικός της χώρας στην οποία βρίσκεται ο οργανισμός (π.χ. GR), η διεύθυνση του, η πόλη στην οποία βρίσκεται, ο νομός και ο ταχυδρομικός κώδικάς του. Τέλος, στα πεδία domain και URL βρίσκονται οι υπερσύνδεσμοι των ιστοσελίδων του. Επειδή ένας συγγραφέας μπορεί να έχει εργαστεί σε πολλά ιδρύματα - τμήματα, αλλά και στα τμήματα εργάζονται πολλοί καθηγητές, θα συνδέσουμε τον πίνακα affilations με τον author_scopus_info μέσω ενός ενδιάμεσου πίνακα, αφού η σχέση πληθικότητάς τους είναι M/N.



Εικόνα 21 Πεδία πίνακα "scopus_info_affiliations"

Ο παραπάνω πίνακας, αποτελεί τον ενδιάμεσο πίνακα, ο οποίος συνδέει τους author_scopus_info και affiliations. Κύριο κλειδί του είναι το αυτοαυξανόμενο πεδίο scopus_info_affiliations_id. Συνδέεται με τους author_scopus_info και affiliations μέσω του author_scopus_info_id και του affiliations_id, αντιστοίχως. Επιπλέον, περιλαμβάνει το πεδίο current_affiliation, το οποίο δέχεται τις τιμές 0 και 1. Όπου υπάρχει συμπληρωμένος ο αριθμός 1 σημαίνει ότι ο ερευνητής, είναι καταγεγραμμένος στο αντίστοιχο ίδρυμα - οργανισμό την τρέχουσα περίοδο. Σκοπός αυτού του πίνακα είναι να συνδέσει για τον κάθε συγγραφέα, όλα τα καταγεγραμμένα ιδρύματα σε συνάρτηση με τα διαφορετικά scopus_id που μπορεί να έχει. Επομένως ένα παράδειγμα αναπαράστασης του πίνακα και του ρόλου του στη σύνδεση της πληροφορίας είναι :

Πίνακας 3. Author Identity - Παράδειγμα εγγραφής στον πίνακα "author_identity"

author_id	official_name	official_surname	Scopus_id
1	Γιώργος	Γουργιωτόπουλος	12345+OR+67890

Πίνακας 4. ***Author Scopus Info*** *- Παράδειγμα εγγραφής στον πίνακα "author_scopus_info"*

author_scopus_info_id	scopus_id	author_Id
1	12345	1
2	67890	1

Πίνακας 5. Affiliations - Παράδειγμα εγγραφής στον πίνακα "affiliations"

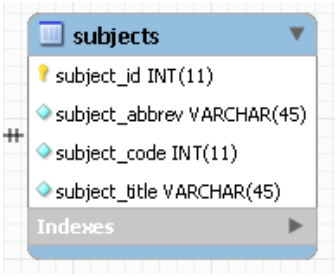
affiliation_id	Preferred_affiliation_name
1	Department of Engineering
2	Department of Librarianship

Πίνακας 6. Scopus info affiliations - Παράδειγμα εγγραφής στον πίνακα "scopus_info_affiliations" (Το πεδίο "author_id" έχει προστεθεί για επεξηγηματικούς σκοπούς.)

scopus_info_affiliations_id	author_scopus_info_id	affiliation_id	current_affiliation	author_id
1	1	1	0	1

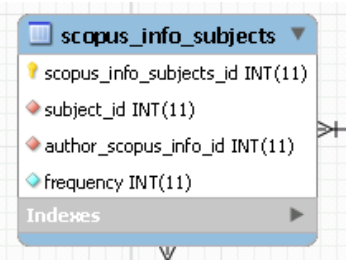
2	2	2	1	1

Η εγγραφή με «affiliation_id» «1» στον πίνακα affiliations, έχει ως «preferred_affiliation_name» : «Department of Librarianship». Η εγγραφή με «author_scopus_info_id» 1 αντιστοιχεί στον ερευνητή με name και surname : “Γιώργος Γουργιωτόπουλος” και «author_identity_id» : ¨1” . Επιπλέον, στον ίδιο συγγραφέα, αντιστοιχεί και το «author_scopus_info_id» : “2” (επειδή έχει δύο διαφορετικά «scopus_ids»), το οποίο συνδέεται με το «affiliation_id» : “2”, το οποίο αντιστοιχεί στο affiliation με «preferred_affiliation_name» : «Department of Engineering». Τέλος, στην περίπτωση του «Department of Engineering» το πεδίο «current_affiliation» έχει την τιμή 1, επομένως, ο ερευνητής εμφανίζεται να εργάζεται στο παραπάνω τμήμα, με βάση τα δεδομένα που συλλέχθηκαν.



Εικόνα 22 Πεδία πίνακα "subjects"

Στον πίνακα subjects, περιγράφονται οι θεματικές περιοχές στις οποίες έχει δημοσιεύσει ένας συγγραφέας. Κύριο κλειδί του είναι το αυτό-αυξανόμενο πεδίο subject_id. Το πεδίο subject_abbrev, αναφέρεται στη συντομογραφία ενός θέματος, το πεδίο subject_code στον τετραψήφιο κωδικό του, και το πεδίο subject_title στο πλήρες όνομα του, όπως αυτά έχουν καταγραφεί και ταξινομηθεί στη Scopus. Αντίστοιχα, με τον πίνακα affiliations η σχέση πληθικότητας του subject με έναν author είναι M / N. Αυτό συμβαίνει καθώς ένας συγγραφέας μπορεί να έχει δημοσιεύσει σε πολλές διαφορετικές θεματικές κατηγορίες, αλλά σε πολλές θεματικές κατηγορίες ενδέχεται να έχουν δημοσιεύσει και πολλοί ερευνητές. Επομένως, σχηματίστηκε ένας ακόμα ενδιάμεσος πίνακας, για να διευκολύνει την πρόσβαση σε αυτή την πληροφορία.



Εικόνα 23 Πεδία πίνακα "scopus info subjects"

Στον παραπάνω πίνακα, όπως συνέβη και με τον πίνακα scopus_info_affiliations συνδέουμε τους πίνακες author_scopus_info και subjects και κατά συνέπεια τον πίνακα author_identity. Κύριο κλειδί του είναι το αυτο-αυξανόμενο πεδίο scopus_info_subjects_id. Επιπλέον, περιέχει και το πεδίο frequency, το οποίο περιγράφει τη συχνότητα, με βάση την οποία ένας συγγραφέας δημοσιεύει σε μια θεματική περιοχή, όπως έχει καταγραφεί από τη Scopus. Ακολουθεί ένα παράδειγμα αναπαράστασης της σύνδεσης της πληροφορίας :

Πίνακας 7. Author Identity Table - Παράδειγμα εγγραφής πίνακα "author_identity"

author_id	official_name	official_surname	Scopus_id
1	Γιώργος	Γουργιωτόπουλος	12345+OR+67890

Πίνακας 8. Author Scopus Info Table - Παράδειγμα εγγραφής πίνακα "author_scopus_info"

author_scopus_info_id	scopus_id	author_Id
1	12345	1
2	67890	1

Πίνακας 9. Subjects Table- Παράδειγμα εγγραφής πίνακα "subjects"

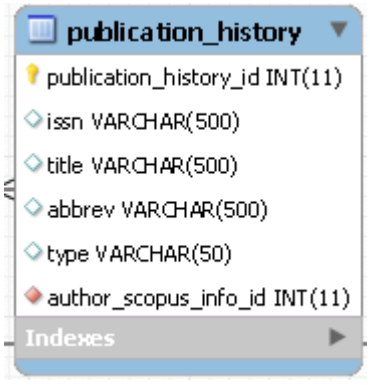
subject _id	subject _ abbrev	subject _ code	subject _ title
1	COMP	1100	Computer Science
2	COMP	1200	Mathematics

Πίνακας 10. Scopus_info_subjects Table - Παράδειγμα εγγραφής πίνακα "scopus_info_subjects"

scopus_info_ subjects_id	subject _ id	author_scopus_info_id	frequency
1	1	1	1
2	2	2	3

Με βάση το παραπάνω παράδειγμα μπορούμε να συμπεράνουμε με ασφάλεια, ότι ο συγγραφέας δημοσιεύει με μεγαλύτερη συχνότητα στη θεματική κατηγορία των μαθηματικών, παρά αυτή της πληροφορικής. Συνδέοντας, λοιπόν, τη συνολική πληροφορία μέσω της κανονικοποίησης διευκολύνουμε την πρόσβαση στα σωστά δεδομένα και επομένως αποκτούμε ολοκληρωμένη αντίληψη ως προς την πληροφορία που εξετάζουμε ενώ μειώνεται δραστικά η πιθανότητα για λανθασμένη ερμηνεία της.

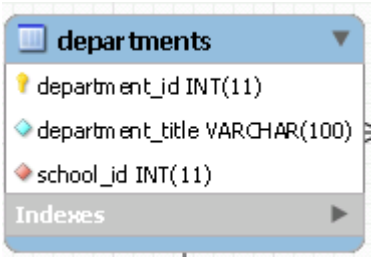
Χωρίζοντας τα δεδομένα σχέσεων M / N μέσω ενδιάμεσων πινάκων, θα αποθηκευτούν θα αποθηκευτούν όλα τα μοναδικά πεδία για να αποφευχθεί η δημιουργία διπλοτύπων και θα τους δοθεί ένα id. Με αυτή την τεχνική μειώνεται κατά μεγάλο βαθμός το πλήθος των εγγραφών ενός πίνακα, ενώ η πληροφορία παραμένει ανεπηρέαστη.



Εικόνα 24 Πεδία πίνακα "publication_history"

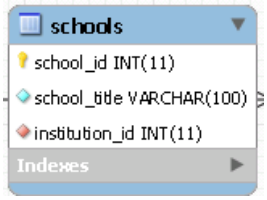
Σκοπός του πίνακα publication_history είναι να περιγράψει το ιστορικό δημοσιεύσεων που έχει ο κάθε ερευνητής. Περιέχει το αυτόματα αυξανόμενο πεδίο publication_history_id, το πεδίο issn, το οποίο περιγράφει το issn της δημοσίευσης,

το πεδίο title, που περιγράφει τον τίτλο της δημοσίευσης, το πεδίο abbrev που περιγράφει τη συντομογραφία του τίτλου, το πεδίο type, το οποίο αναφέρεται στον τύπο της δημοσίευσης (δημοσίευση σε περιοδικό ή σε πρακτικά συνεδρίου), καθώς και το πεδίο author_scopus_info_id, το οποίο ως ξένο κλειδί του πίνακα έχει ως σκοπό τη σύνδεση της πληροφορίας με στόχο την εγκυρότερη πρόσβαση σε αυτή.



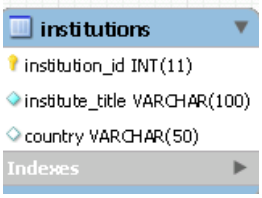
Εικόνα 25 Πεδία πίνακα "departments"

Στον πίνακα «departments», περιγράφονται τα τμήματα στα οποία εργάζονται οι ερευνητές που έχουν καταγραφεί στον πίνακα «author_identity». Περιέχει το πεδίο «department-id» (αυτό-αυξανόμενο αναγνωριστικό), το πεδίο «department-title», το οποίο περιέχει τον τίτλο του περιγραφόμενου τμήματος καθώς και το πεδίο «school-id», το οποίο αναφέρεται στην σχολή στην οποία ανήκει η κάθε εγγραφή ενός τμήματος.



Εικόνα 26 Πεδία πίνακα "schools"

Στον πίνακα «schools», περιγράφονται οι σχολές στις οποίες ανήκουν τα τμήματα που έχουν καταγραφεί στον πίνακα «departments». Περιέχει τα πεδία «school_id» (αυτό-αυξανόμενο αναγνωριστικό) , το πεδίο «school_title» που περιέχει το όνομα της σχολής αλλά και το πεδίο «institution_id», το οποίο αναφέρεται στο ίδρυμα στο οποίο ανήκει η κάθε εγγραφή μιας σχολής.



Εικόνα 27 Πεδία πίνακα "institutions"

Στον πίνακα «institutions», περιγράφονται τα ιδρύματα στα οποία ανήκουν οι σχολές που έχουν καταγραφεί στον πίνακα «schools». Περιέχει τα πεδία «institution_id» (αυτό-αυξανόμενο αναγνωριστικό), «institute_title» το οποίο αναφέρεται στο όνομα του ιδρύματος και το πεδίο «country», το οποίο απευθύνεται στη χώρα στην οποία βρίσκεται.

Σε αυτό το σημείο αξίζει να αναφερθεί, πως οι τρείς τελευταίοι πίνακες συστάθηκαν και αποτελούν πίνακες θεμελιώδους σημασίας, για την εκτέλεση του πειράματος εφαρμογής του βιβλιομετρικού

εργαλείου. Με τη χρήση των τριών αυτών πινάκων, η πληροφορία που έχει συλλεχθεί για τον κάθε συγγραφέα, μπορεί να αναλυθεί σε τμηματικό , σχολικό ή και ιδρυματικό επίπεδο, δίνοντας τη δυνατότητα της παραγωγής χρήσιμων βιβλιομετρικών και γενικών στατιστικών στοιχείων.

Κεφάλαιο 3. Εφαρμογή άντλησης δεδομένων από Scopus

3.1 Περιβάλλον ανάπτυξης

Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής είναι η C# καθώς και το πρώτο εργαλείο άντλησης βιβλιομετρικών δεδομένων είναι γραμμένο στην ίδια γλώσσα. Επομένως, με αυτόν τον τρόπο εξασφαλίζονται η ομοιομορφία και η ευκολία στην περεταίρω επέκτασή τους.

Η «C#» (προφέρεται Σι Σαρπ) είναι μια αντικειμενοστρεφής γλώσσα προγραμματισμού γενικής χρήσης, η οποία δημιουργήθηκε από τη Microsoft και πιο συγκεκριμένα, σχεδιάστηκε από τον Anders Hejlsberg και την ομάδα του κατά τη διάρκεια του έργου «.NET initiative», και χρονολογείται περίπου στο 2000. Το όνομα «C sharp» εμπνεύστηκε από μια μουσική σημειογραφία. Εξαιτίας διαφόρων τεχνικών περιορισμών προβολής και επειδή το σύμβολο «sharp ♯» δεν υφίσταται στα απλά πληκτρολόγια, χρησιμοποιήθηκε το αποτέλεσμα του συνδυασμού «shift + 3 (#)» για την αναπαράσταση της γλώσσας. Η συγκεκριμένη κατάληξη έχει χρησιμοποιηθεί από αρκετές ακόμα γλώσσες της .NET όπως η «A#» η «F#» και η «Eiffel#». Η «C#» παρότι έχει κατηγορηθεί πολλές φορές ότι αποτελεί μια αντιγραφή της «Java», είναι βασισμένη στη «C++» .

Το περιβάλλον ανάπτυξης που χρησιμοποιήθηκε ήταν το «Visual Studio Community Version», δηλαδή ένα δωρεάν «IDE» («integrated development environment») της Microsoft, το οποίο επιτρέπει την εύκολη και γρήγορη δημιουργία προγραμμάτων μέσω γραφικού περιβάλλοντος σε πληθώρα γλωσσών προγραμματισμού της «.NET», συμπεριλαμβανομένης και της «C#». Επιπλέον, διαθέτει μια πληθώρα διαφορετικών βιβλιοθηκών κώδικα, έτοιμες για να εξυπηρετήσουν την κάθε ανάγκη του χρήστη.

Στην περίπτωση του προγράμματος που αναπτύχθηκε για το εργαστήριο της πληροφορίας του τμήματος βιβλιοθηκονομίας και συστημάτων πληροφόρησης, χρησιμοποιήθηκε μια εφαρμογή χωρίς γραφικό περιβάλλον («console application»), η οποία μπορεί να «τρέξει» μέσω του «Microsoft visual studio», είτε μέσω τερματικού.

Ρόλος της είναι η δημιουργία διαδικτυακής σύνδεσης μεταξύ του τοπικού υπολογιστή και της «Scopus», και η άντληση δεδομένων, τα οποία αποθηκεύονται και επεξεργάζονται, εξίσου σε τοπικό επίπεδο.

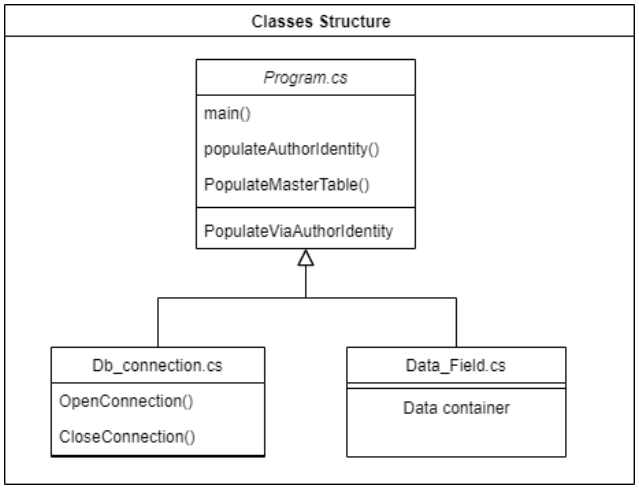
3.2 Δομή – Κλάσεις - Μεταβλητές

Για τη δημιουργία της επικοινωνίας με τη «MySql» βάση δεδομένων, χρησιμοποιήθηκε η βιβλιοθήκη κώδικα «MySql.Data.MySqlClient». Η πρώτη κλάση του «framework» αφορά αποκλειστικά τη σύνδεση του προγράμματος, με τη βάση δεδομένων με την οποία αλληλεπιδρά. Ονομάστηκε «db_connection» και περιέχει τέσσερις δημόσιες («public») μεθόδους, δύο δημόσιες μεταβλητές και ένα αντικείμενο. Οι μεταβλητές είναι τα δύο διαφορετικά «connection strings» («string» = ακολουθία χαρακτήρων ή αλλιώς κείμενο) για τη βάση «citations» («citations_connectionString») και για τη βάση «bibliometrics» («bibliometrics_connectionString») καθώς και το αντικείμενο «connection», μέσω του οποίου δημιουργούμε τη σύνδεση. Οι μέθοδοι που περιέχονται είναι η «OpenCitationsConnection» ,και η «CloseCitationsConnection», οι οποίες αφορούν τη σύνδεση με τη «MySQL» βάση δεδομένων «citations», καθώς και η «OpenBibliometricsConnection» και η «CloseBibliometricsConnection», οι οποίες αφορούν τη σύνδεση με τη «MySQL» βάση δεδομένων «bibliometrics». Δημιουργήθηκαν, ώστε οι δύο διαφορετικές συνδέσεις να συγχωνευτούν σε μια κλάση και κατά συνέπεια ο κώδικας να είναι πιο ευανάγνωστος. Επιπλέον, καλούνται σε διάφορα σημεία του κώδικα και βοηθούν στην καλύτερη

διαχείριση του. Η «OpenCitationsConnection()» δημιουργεί μια εκδοχή του αντικειμένου «MySqlConnection» το οποίο “περιμένει” ως παράμετρο ένα «connection string» και στην θέση του χρησιμοποιούμε το «connectionString» της βάσης δεδομένων «citations» («citations_connectionString»). Με την ίδια λογική έχει δομηθεί και η «OpenBibliometricsConnection()» χρησιμοποιώντας σε αυτή την περίπτωση το «connectionString» της βάσης δεδομένων «bibliometrics» («bibliometrics_connectionString»).

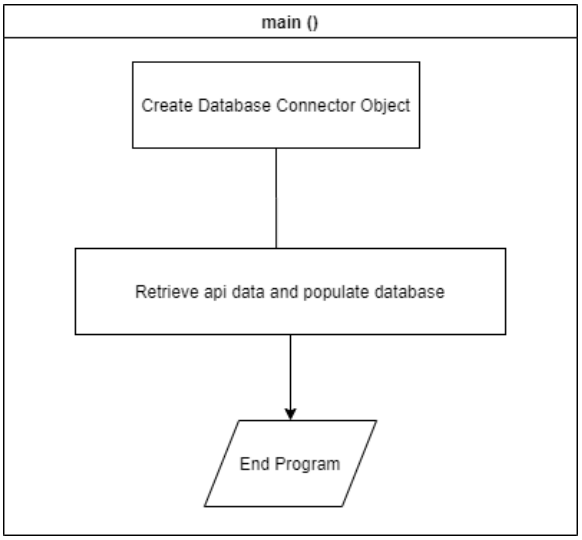
Η δεύτερη κλάση που δημιουργήθηκε είναι η κλάση «Data_field» και όπως υποδηλώνει το όνομα της είναι η κλάση στην οποία αποθηκεύονται τα δεδομένα τα οποία συλλέγει το πρόγραμμα μέσω του REST API. Αποτελεί ένα αντικείμενο το οποίο έχει ως «properties» μεταβλητές τύπου «string» οι οποίες έχουν ονομαστεί ακριβώς όπως και τα πεδία στα οποία αντιστοιχούν στη Β.Δ. με σκοπό να διευκολύνεται η διαδικασία της μεταφοράς τους.

3.2.1 Κώδικας και διαγράμματα



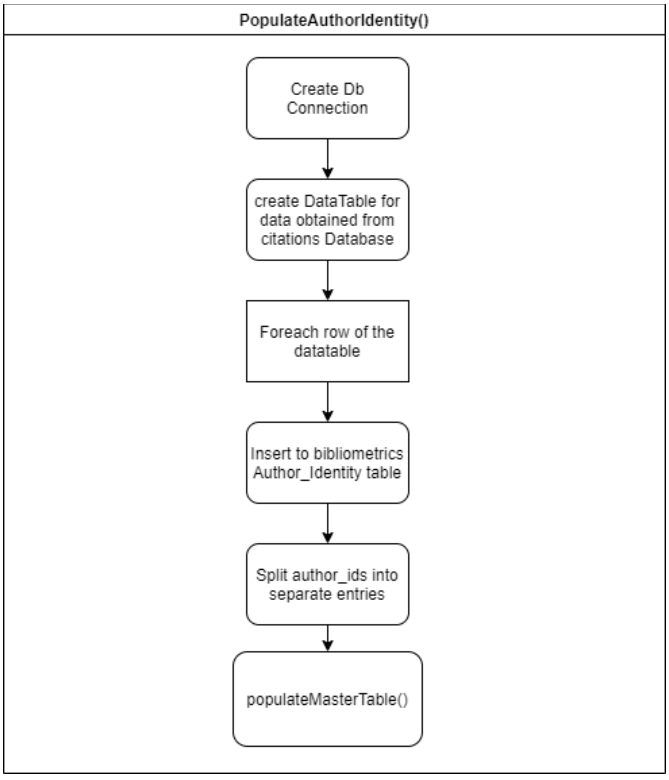
Εικόνα 28 Κλάσεις και μέθοδοι κώδικα του προγραμματιστικού εργαλείου

Όπως παρατηρούμε στο παραπάνω διάγραμμα, για τη δόμηση του προγράμματος δημιουργήθηκαν τρείς βασικές κλάσεις. Η πρώτη είναι υπεύθυνη για τη δημιουργία συνδέσεων με τη βάση δεδομένων, η δεύτερη για την αποθήκευση των δεδομένων και ή τρίτη για την εκτέλεση του προγράμματος. Αναλυτικότερα, η κλάση με όνομα “Db_connection”, περιέχει βασικές μεθόδους για την εκτέλεση των διαδικασιών σύνδεσης και αποσύνδεσης με τη βάση δεδομένων. Η κλάση με όνομα “Data_Field”, περιγράφει τη δομή των δεδομένων που αντλούνται από τη Scopus για τον κάθε συγγραφέα. Περιέχει, δηλαδή όλες τις πιθανές πληροφορίες που θα συλλεχθούν σε μεταβλητές κειμένου (string), σε ένα αντικείμενο. Τέλος, αυτές οι δύο κλάσεις αλληλεπιδρούν μεταξύ τους με την τελευταία κλάση “Program.cs” ώστε να επιτευχθεί η εκτέλεση του προγράμματος. Πιο συγκεκριμένα, η μέθοδος “main” της τελευταίας κλάσης, είναι η μέθοδος που εκτελείται πρώτη (με την έναρξη του προγράμματος) και αποτελεί τον «καθοδηγητή» του προγράμματος. Σε αυτήν αρχικοποιούνται οι κλάσεις – αντικείμενα που αναλύθηκαν παραπάνω. Μια απλή περιγραφή των λειτουργιών που εκτελούνται στη “main” περιγράφονται στο παρακάτω διάγραμμα:



Εικόνα 29 Διάγραμμα ροής προγράμματος

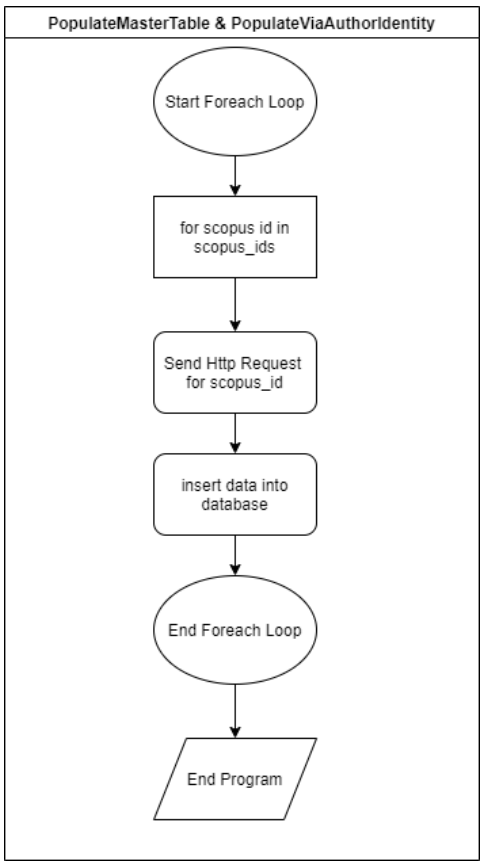
Επιπλέον, η κλάση “Program.cs” εκτός από τη “main” μέθοδο περιέχει και τρείς ακόμα μεθόδους (όπως παρατηρήσαμε παραπάνω) :



Εικόνα 30 Διάγραμμα ροής μεθόδου PopulateAuthorIdentity()

Η πρώτη εκ των τριών, είναι η “PopulateAuthorIdentity” η οποία έχει ως σκοπό τη μεταφορά εγγραφών από ένα προ-υπάρχον μοντέλο βάσης δεδομένων (με όνομα citations), στη βάση δεδομένων που χρησιμοποιήθηκε για την παρούσα εφαρμογή (με όνομα bibliometrics) . Πιο συγκεκριμένα, η μέθοδος δημιουργεί συνδέσεις με τα δύο διαφορετικά σχήματα βάσεων δεδομένων. Έπειτα με βάση ενός SQL ερωτήματος αποθηκεύει στη μνήμη τυχαίας προσπέλασης όλους του συγγραφείς με τα ονοματεπώνυμά τους, τα διαφορετικά τους αναγνωριστικά και άλλες επιπρόσθετες πληροφορίες. Η κάθε εγγραφή (απάντηση του ερωτήματος),αποθηκεύεται με τη σειρά της σε ένα αντικείμενο που αντικατοπτρίζει μια

γραμμή ενός πίνακα, χωρισμένη σε κελιά. Επομένως, δημιουργείται ένα εικονικό αντικείμενο (virtual object) – αντίγραφο του πίνακα από τον οποίο αντλήθηκαν δεδομένα και στη συνέχεια χρησιμοποιήθηκε η επαναληπτική δομή “for” έτσι ώστε να επεξεργασθεί ή κάθε γραμμή αυτού του πίνακα. Για κάθε γραμμή, λοιπόν, εκτελέσθηκαν κάποιες τροποποιήσεις έτσι ώστε τα δεδομένα να μορφοποιηθούν με βάση της απαιτήσεις του νέου μοντέλου δεδομένων και στη συνέχεια, εισήχθησαν στη νέα βάση (γραμμή-γραμμή). Το πιο σημαντικό κομμάτι που αξίζει να αναφερθεί εδώ είναι ότι αντλήθηκαν τα διαφορετικά scopus-id, τα οποία αντιστοιχούν στον ίδιο συγγραφέα και χωρίστηκαν σε διαφορετικές εγγραφές, οι οποίες όμως αντιστοιχούν στην ίδια οντότητα. Αφού χωρίστηκαν, αποθηκεύτηκαν σε μια λίστα, για κάθε μέρος της οποίας, εκτελείται η μέθοδος “populateMasterTable”. Εδώ αξίζει να επισημανθεί το πλήθος τις λίστας μπορεί να αποτελέσει από 1-N, ανάλογα με τον αριθμό των χωρισμένων αναγνωριστικών.



Εικόνα 31 Διάγραμμα ροής μεθόδου PopulateMasterTable

Η μέθοδος populate master table, έχει ως βασικό σκοπό τη δημιουργία ξεχωριστών εγγραφών για τον κάθε συγγραφέα, σε ένα νέο πίνακα του σχήματος δεδομένων, στον οποίο το κάθε διαφορετικό αναγνωριστικό συνδέεται με έναν ερευνητή με σχέση πλυθηκότητας ένα προς ένα (one-to-one). Αυτό σημαίνει (όπως έχει αναλυθεί και σε προηγούμενο παράρτημα) πως στην περίπτωση που υπάρχουν περισσότερα από ένα αναγνωριστικά δημιουργείται μια ξεχωριστή εγγραφή για το καθένα από αυτά.

Σε αυτό το σημείο, σημαντικό είναι να αναφερθεί ένα “bug” που παρατηρήθηκε στη χρήση της γλώσσας προγραμματισμού C#, αναφορικά με το κομμάτι της εισαγωγής δεδομένων στη βάση δεδομένων. Πιο συγκεκριμένα, παρατηρήθηκε ότι στο σημείο της εισαγωγής των δεδομένων στη βάση ορισμένα κομμάτια του κώδικα δεν εκτελούνταν σειριακά αλλά ασύγχρονα, προκαλώντας προβλήματα όπως η προσθήκη δεδομένων στη βάση που αφορούσαν οντότητες (συγγραφείς), οι οποίοι δεν είχαν

αποθηκευτεί ακόμα στους αντίστοιχους πίνακες. Γι’ αυτό το λόγο, η μέθοδος που αρχικά σχεδιάστηκε ως υπεύθυνη για τη συλλογή και αποθήκευση των δεδομένων, χωρίστηκε σε δύο παρόμοιες μεθόδους, ώστε να βεβαιωθεί ότι πρώτα εισάγεται ο κάθε συγγραφέας στον αντίστοιχο πίνακα και έπειτα οι επιπρόσθετες πληροφορίες για αυτόν στους υπόλοιπους.

Συνεπώς, η μέθοδος «populateMasterTable()», λειτουργεί μέσα σε έναν επαναληπτικό βρόγχο «for» για κάθε διαφορετικό αναγνωριστικό που έχει αποθηκευτεί στη λίστα (ενδέχεται να είναι και μόνο ένα) και στη συνέχεια δημιουργεί ένα «GET http request», το οποίο «ζητάει» δεδομένα από το «web api» της Scopus. Σε περίπτωση που το διαδικτυακό αυτό αίτημα λάβει ως απάντηση τον κωδικό 200 (response code 200) και η απάντηση δεν είναι κενή προχωρούμε στο κομμάτι της εισαγωγής των δεδομένων στη βάση, διαφορετικά συνεχίζουμε με το επόμενο αναγνωριστικό της λίστας. Στην περίπτωση της εισαγωγής των δεδομένων στη Β.Δ. χρησιμοποιήθηκε η βιβλιοθήκη της C# «xml». Η απάντηση που δεχόμαστε από τη διαδικτυακό αίτημα αποθηκεύεται σε μορφή κειμένου και στη συνέχεια δημιουργείται ένα εικονικό xml έγγραφο (xml document), στη μνήμη τυχαίας προσπέλασης, στο οποίο «φορτώνουμε» («load») την απάντηση. Ο λόγος που επιλέχθηκε αυτή η βιβλιοθήκη είναι η ευκολία που προσφέρει στην πρόσβαση του (εμφωλευμένου) κειμένου γονικών – παιδικών κόμβων (parent-child nodes). Αφού δημιουργηθεί το XML έγγραφο και φορτωθεί η απάντηση, το έγγραφο χωρίζεται με βάση τους βασικούς γονικούς του κόμβους και ξεκινάει η διαδικασία εισαγωγής της πληροφορίας στο αντίστοιχο αντικείμενο που έχει δημιουργηθεί (Datafield.cs) και έπειτα στους αντίστοιχους πίνακες της βάσης δεδομένων. Αναφορικά με την εισαγωγή των δεδομένων στη βάση, χρησιμοποιήθηκαν δύο τεχνικές, ανάλογα με τον πίνακα στον οποίο γίνεται η αποθήκευση των δεδομένων . Η πρώτη αφορά δεδομένα που συλλέγονται στη συνολική τους μορφή και επομένως χρησιμοποιούνται απλά «sql insert» ερωτήματα. Η δεύτερη αφορά δεδομένα τα οποία για δομικούς λόγους του σχήματος (σχέσεις M-N / π.χ. πίνακας subjects) αποθηκεύονται μόνο μια φορά στη βάση και αν ξανασυναντηθούν αγνοούνται. Αφού η παραπάνω διαδικασία ολοκληρωθεί με επιτυχία, τα δεδομένα έχουν εισαχθεί στη Β.Δ. και προχωρούμε στο κομμάτι της επεξεργασίας – ανάλυσης των δεδομένων.

Κεφάλαιο 4. Εφαρμογή χρήσης του βιβλιομετρικού εργαλείου

4.1 Σενάριο εφαρμογής

Το εργαλείο δομήθηκε με σκοπό την ομαλή - αυτοματοποιημένη συλλογή βιβλιομετρικών δεδομένων της «Scopus», που αφορούν μεμονωμένες ακαδημαϊκές οντότητες – συγγραφείς . Όμως, οι πληροφορίες που έχουν συλλεχθεί στο σύνολο τους αφορούν ολόκληρα τμήματα, σχολές ή και εκπαιδευτικά ιδρύματα. Σε παλαιότερο εργαλείο του εργαστηρίου πληροφορίας του τμήματος Αρχειονομίας, Βιβλιοθηκονομίας και Συστημάτων Πληροφόρησης, έχουν συλλεχθεί δεδομένα για το ακαδημαϊκό προσωπικό των σχολών και τμημάτων του Πανεπιστημίου Δυτικής Αττικής καθώς και άλλων πανεπιστημίων. Οι καταγεγραμμένες αυτές πληροφορίες αποτέλεσαν τη βάση για τη δημιουργία του σεναρίου εφαρμογής του συγκεκριμένου εργαλείου.

Αρχικά μελετήθηκε το ήδη υπάρχον εργαλείο και χρησιμοποιήθηκε για την άντληση δεδομένων και την εξοικείωση με τη χρήση ενός API, σε προγραμματιστικό επίπεδο. Το παραπάνω, σχεδιάστηκε με βάση το “Scopus search API”, ένα από τα προσφερόμενα APIs της Scopus, όπως αναλύθηκε προηγουμένως και τα δεδομένα του αποθηκεύτηκαν σε μια MySQL βάση δεδομένων (σχεσιακή βάση δεδομένων). Στη συνέχεια, μελετήθηκαν τα υπόλοιπα API που προσφέρει η Scopus και επιλέχθηκε το “Author retrieval API” για την άντληση νέων βιβλιομετρικών δεδομένων που δύναται να χρησιμοποιηθούν για την αξιολόγηση της ερευνητικής δραστηριότητας ακαδημαϊκών οντοτήτων. Συνεπώς, σχεδιάστηκε το νέο εργαλείο, με βάση τα πρότυπα του ήδη υπάρχοντος. Αρχικά, δημιουργήθηκε ένα νέο σχεσιακό μοντέλο δεδομένων, το οποίο αντικατοπτρίζει τις πληροφορίες που αντλήθηκαν σε μια νέα “MySQL database”. Στη συνέχεια, συστάθηκε ο κώδικας μέσω του οποίου πραγματοποιήθηκε η άντληση και αποθήκευση των δεδομένων στο μοντέλο, αφού πρώτα τα αποτελέσματα ελέγχθηκαν για την εγκυρότητά τους σε σχέση με αυτά που εμφανίζονται σε μια αντίστοιχη αναζήτηση μέσω της ιστοσελίδας της Scopus.

Αναλυτικότερα, πραγματοποιήθηκε αποθήκευση βιβλιομετρικών δεδομένων αναφορικά με όλους τους ακαδημαϊκούς – συγγραφείς του πανεπιστημίου δυτικής αττικής, διαχωρισμένους ανά τμήμα και σχολή. Όπως παρατηρήθηκε παραπάνω στο κομμάτι της ανάλυσης του XML «Author Scopus Retrieval API» αλλά και του μοντέλου δεδομένων που παράχθηκε, συλλέχθηκαν πληροφορίες όπως ο αριθμός των δημοσιεύσεων και οι διαφορετικές θεματικές κατηγορίες στις οποίες έχει δημοσιεύσει ο κάθε ερευνητής, τα περιοδικά και τα συνέδρια στα οποία έχουν συμμετάσχει, το επαγγελματικό ιστορικό τους στον ακαδημαϊκό τομέα κ.α.

Στη συνέχεια τα παραπάνω δεδομένα αναλύθηκαν μέσω SQL ερωτημάτων, σε επίπεδο βάσης δεδομένων, για την εξαγωγή συμπερασμάτων αναφορικά με την ερευνητική δραστηριότητα των συγγραφέων αλλά και άλλων χρήσιμων στατιστικών δεδομένων. Οι περισσότερες κατηγορίες ερωτημάτων ανάλυσης δεδομένων αφορούν όλα τα τμήματα του «ΠΑΔΑ» σε συλλογικό επίπεδο, με διαφορά σε αυτές των θεματικών περιοχών και του ιστορικού δημοσιεύσεων που χρησιμοποιήθηκαν ως παράδειγμα, τα τμήματα Αρχειονομίας Βιβλιοθηκονομίας και Συστημάτων Πληροφόρησης, Ηλεκτρολόγων Μηχανικών και Μηχανικών Πληροφορικής καθώς και το Τμήμα Βιοϊατρικών Επιστημών.

Τέλος, τα αποτελέσματα που προέκυψαν οπτικοποιήθηκαν με τη μορφή διαγραμμάτων, ώστε να εξαχθούν με μεγαλύτερη ευκολία συμπεράσματα.

4.1.1 Κατηγορίες ερωτημάτων SQL

Αφού το προγραμματιστικό εργαλείο της άντλησης δεδομένων από το API της Scopus, ολοκλήρωσε τη συλλογή των βιβλιομετρικών δεδομένων με επιτυχία για το ακαδημαϊκό προσωπικό του κάθε τμήματος του Πανεπιστημίου Δυτικής Αττικής, σχεδιάστηκε πληθώρα ερωτημάτων SQL, τα οποία δημιουργήθηκαν με τέτοιο τρόπο ώστε τα αποτελέσματα να μπορούν να χωριστούν σε ιδρυματικό, σχολικό, τμηματικό ή προσωπικό επίπεδο.

Στα παρακάτω παραδείγματα αναλύθηκαν κάποια SQL ερωτήματα, τα οποία αφορούν το τμήμα Βιβλιοθηκονομίας Αρχειονομίας και Συστημάτων πληροφόρησης ή το Πανεπιστήμιο Δυτικής Αττικής αναφορικά με :

1. Τις διαφορετικές θεματικές κατηγορίες στις οποίες έχουν δημοσιεύσει τμήματα του Πανεπιστημίου Δυτικής Αττικής.
2. Τον αριθμό των δημοσιεύσεων τμημάτων του «ΠΑΔΑ» σε επιστημονικά περιοδικά.
3. Τον συνολικό αριθμό των επιστημονικών τους δημοσιεύσεων σε συνδυασμό με άλλα χρήσιμα βιβλιομετρικά δεδομένα όπως π.χ. ετεροαναφορές.
4. Το επαγγελματικό ιστορικό των συγγραφέων στον ακαδημαϊκό χώρο.

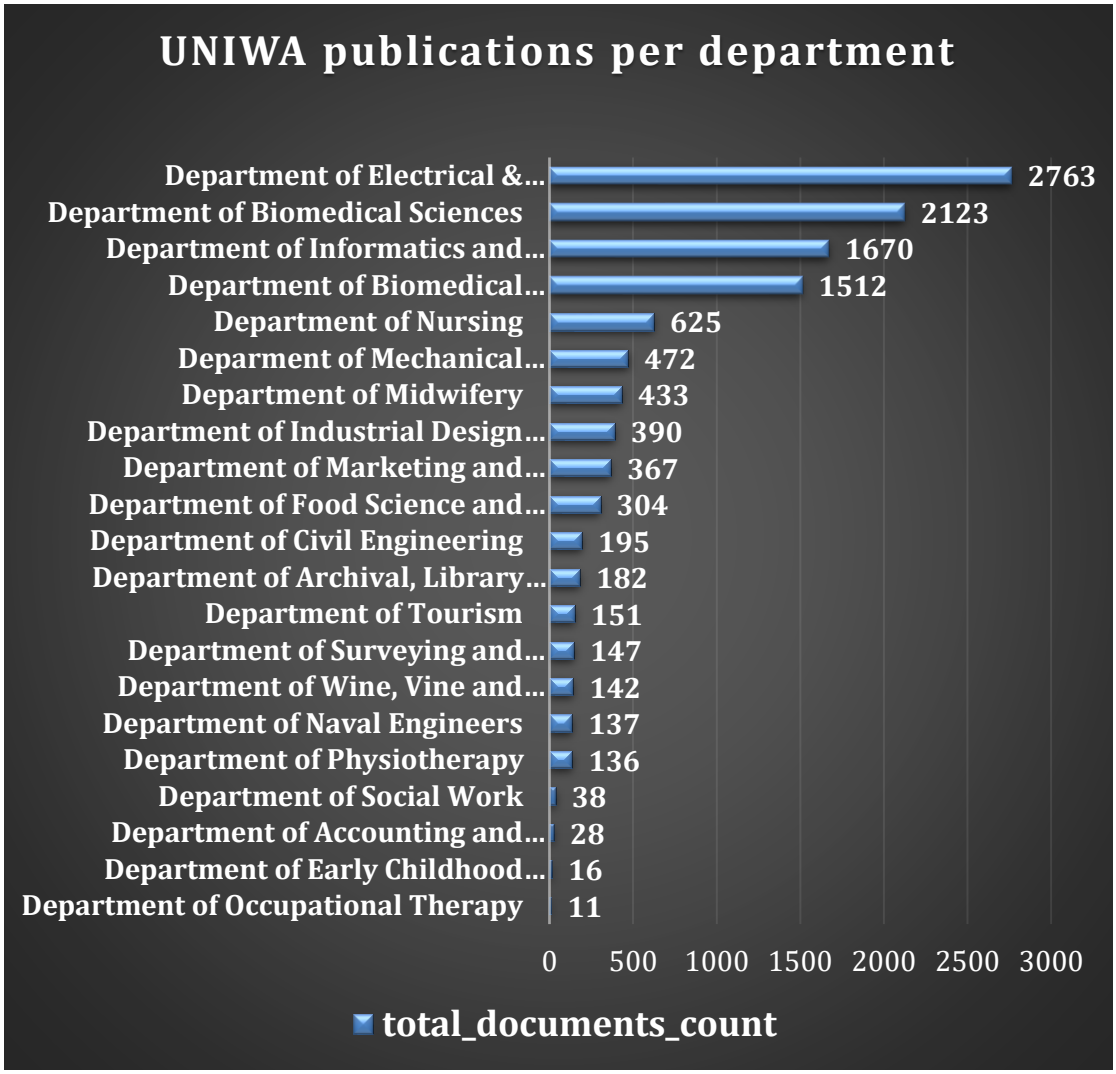
4.2 Ερωτήματα SQL και Αποτελέσματα

Το πρώτο ερώτημα αφορά το Πανεπιστήμιο Δυτικής στο σύνολό του (όπως έχει καταγραφεί στο εργαλείο άντλησης δεδομένων) και το κάθε τμήμα του ξεχωριστά καθώς περιέχει το συνολικό αριθμό δημοσιεύσεων, αλλα και τον αριθμό των αυτοαναφορών και των ετεροαναφορών τους.

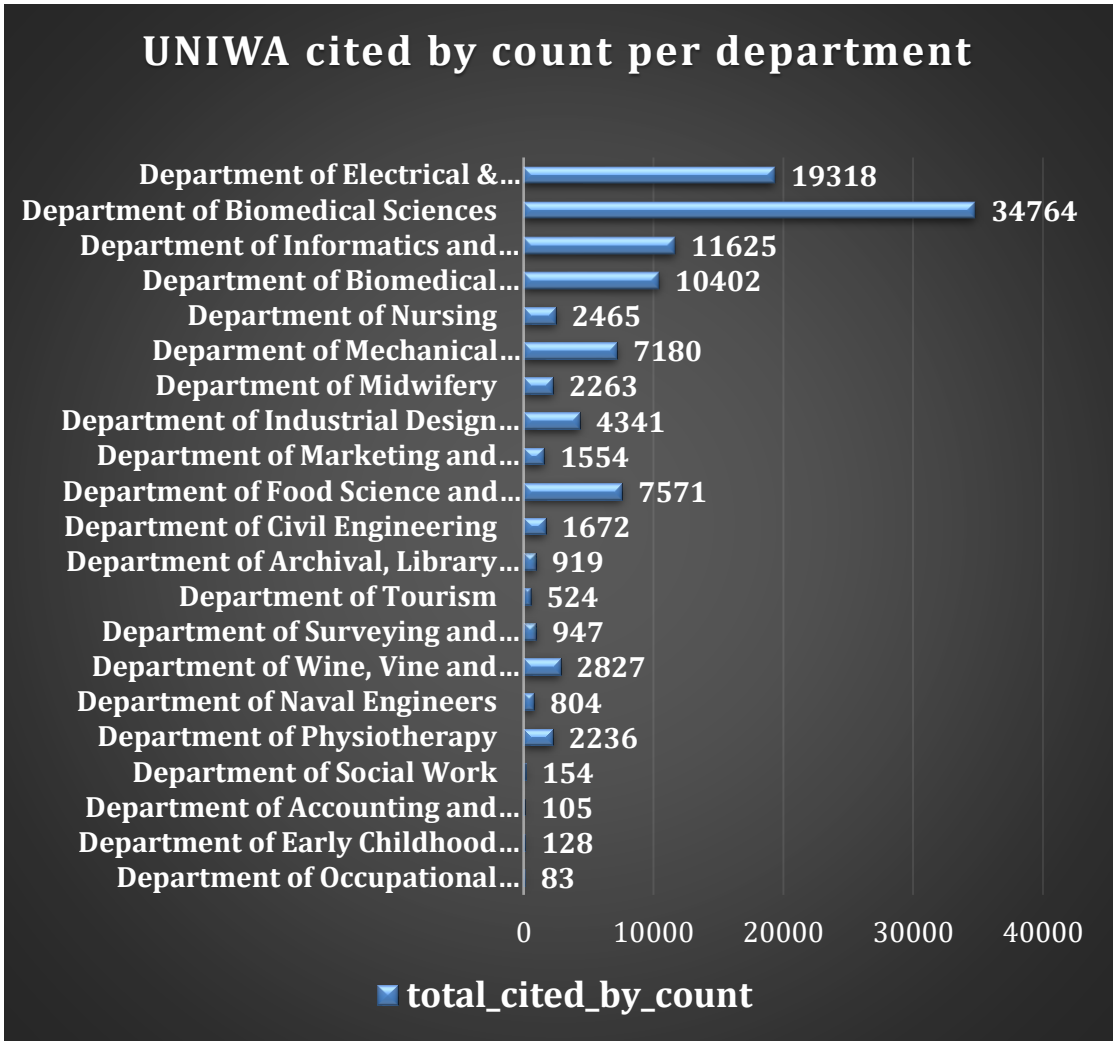
Το MySQL ερώτημα είναι :

```
select departments.department_title,SUM(author_scopus_info.documents_count) as total_documents_count,
SUM(author_scopus_info.cited_by_count) as total_cited_by_count,
SUM(author_scopus_info.citations_count) as total_citations_count,
join author_scopus_info on author_identity.author_id = author_scopus_info.author_id
join departments on departments.department_id = author_identity.department_id
group by departments.department_id
```

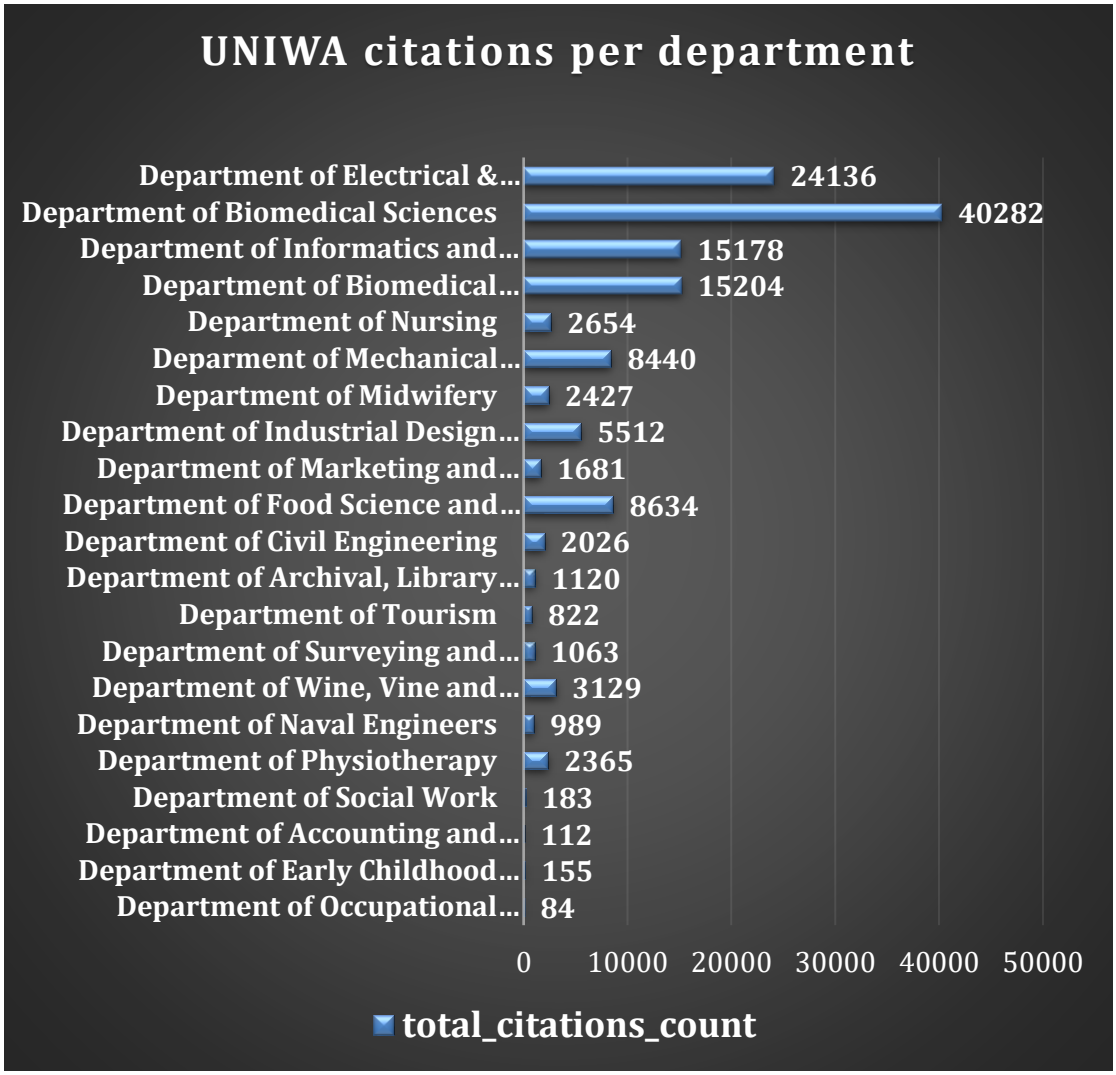
Σε αυτό το ερώτημα χρησιμοποιήθηκε η μέθοδος SUM, ώστε να αθροιστούν τα διαφορετικά πεδία (αυτοαναφορών και ετεροαναφορών) σε συνδυασμό με τον τμηματικό διαχωρισμό τους, μέσω της μεθόδου group by.



Εικόνα 32 Τμήματα Πανεπιστημίου Δυτικής Αττικής : Συνολικός αριθμός δημοσιεύσεων



Εικόνα 33 Τμήματα Πανεπιστημίου Δυτικής Αττικής : Συνολικός αριθμός αναφορών σε άλλες έρευνες



Εικόνα 34 Τμήματα Πανεπιστημίου Δυτικής Αττικής : Συνολικός αριθμός ετεροαναφορών.

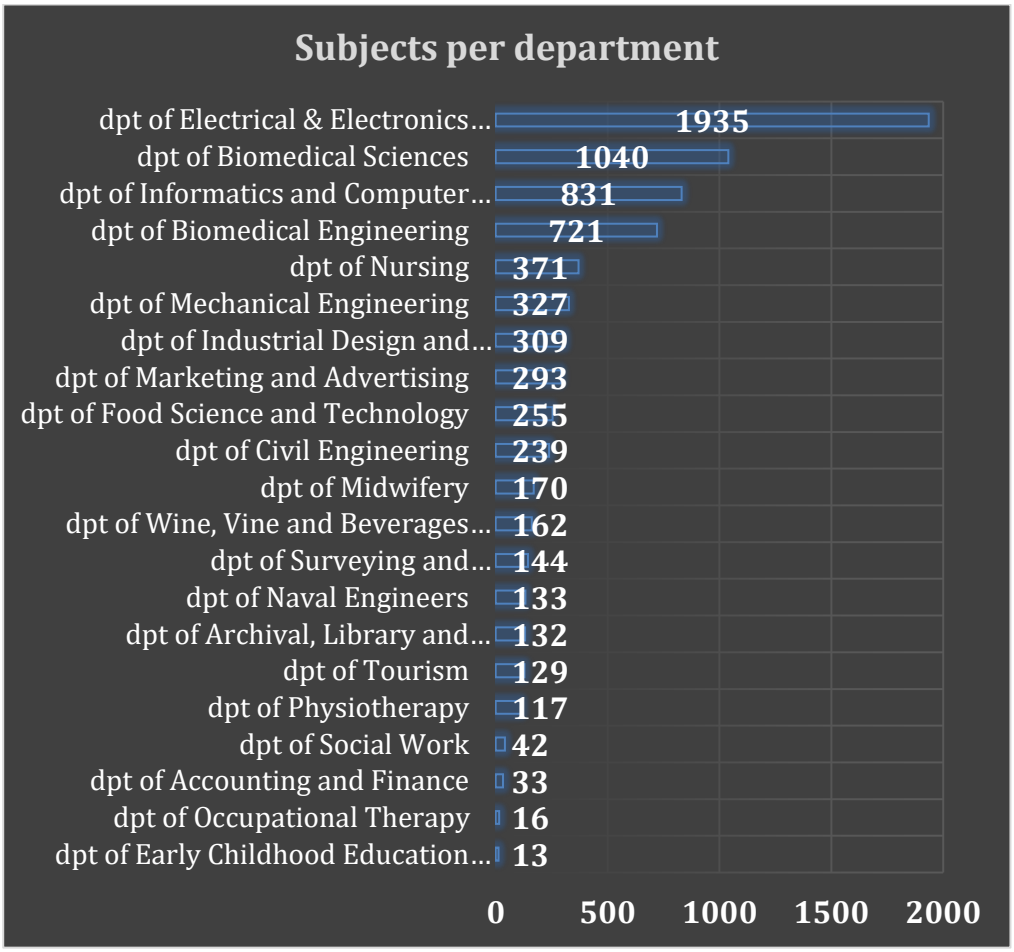
Όπως αναγνωρίζεται με βάση τα παραπάνω διαγράμματα, το τμήμα των Ηλεκτρολόγων & Ηλεκτρολόγων Μηχανικών κατέχει το μεγαλύτερο αριθμό δημοσιεύσεων, ενώ το τμήμα Βιοϊατρικών επιστημών έχει τις περισσότερες ετεροαναφορές, αλλά και χρησιμοποιεί περισσότερες βιβλιογραφικές αναζητήσεις στις έρευνές του, συγκριτικά με τα υπόλοιπα τμήματα, με βάση την εικόνα 7. Για τη συνέχεια της έρευνας, επιλέχθηκαν τα παραπάνω δύο τμήματα συνδυαστικά με το τμήμα Βιβλιοθηκονομίας Αρχειονομίας και Συστημάτων Πληροφόρησης ώστε να εξαχθούν συμπεράσματα αναφορικά με τον συνολικό αριθμό των δημοσιεύσεων τους σε πληθώρα επιστημονικών περιοδικών και συνεδρίων που έχουν δημοσιεύσει καθώς και στον αριθμό των διαφορετικών scopus profiles που έχουν χρησιμοποιηθεί για να δημοσιεύσουν σε διάφορες θεματικές κατηγορίες.

Το επόμενο ερώτημα αφορά τις διαφορετικές θεματικές κατηγορίες στις οποίες έχουν καταγραφεί δημοσιεύσεις από το αποθετήριο της Scopus για κάποια απο τα τμήματα του Πανεπιστημίου Δυτικής Αττικής.

Ο κώδικας του ερωτήματος είναι ο παρακάτω :

```
select departments.department_title, COUNT(subjects.subject_id) AS subjects_per_department from subjects
join scopus_info_subjects on scopus_info_subjects.subject_id = subjects.subject_id
join author_scopus_info on author_scopus_info.author_scopus_info_id = scopus_info_subjects.author_scopus_info_id
join author_identity on author_identity.author_id = author_scopus_info.author_id
join departments on departments.department_id = author_identity.department_id
group by departments.department_id;
```

Το παραπάνω ερώτημα δομήθηκε με βάση το μοντέλο δεδομένων που αναλύθηκε σε προηγούμενο παράρτημα της παρούσας εργασίας. Πιο συγκεκριμένα, χρησιμοποιείται η μέθοδος count της MySql, η οποία όπως υποδηλώνει καταγραφεί τον αριθμό του κάθε πεδίου “subject_id” (δηλαδή της κάθε θεματικής κατηγορίας) που υπάρχει στην ένωση μεταξύ των πινάκων subjects, scopus_info_subjects, author_scopus_info και author_identity, χωρισμένα ανα τμήμα μέσω της μεθόδου της MySQL “group by”. Εδώ αξίζει να αναφερθεί πως στην περίπτωση που δεν είχε χρησιμοποιηθεί η μέθοδος group by, το αποτέλεσμα του παραπάνω ερωτήματος θα αποτελούσε απλώς τον συνολικό αριθμό των θεματικών κατηγοριών που έχουν καταγραφεί στο εργαλείο άντληση πληροφοριών απο τον κατάλογο της Scopus.



Όπως παρατηρούμε από τα παραπάνω διαγράμματα, το τμήμα Ηλεκτρολόγων Μηχανικών κατέχει το μεγαλύτερο εύρος σε διαφορετικές θεματικές κατηγορίες με ποσοστό 26% του συνόλου, ενώ ακολουθεί το τμήμα Βιοϊατρικών Επιστημών με ποσοστό 14%.

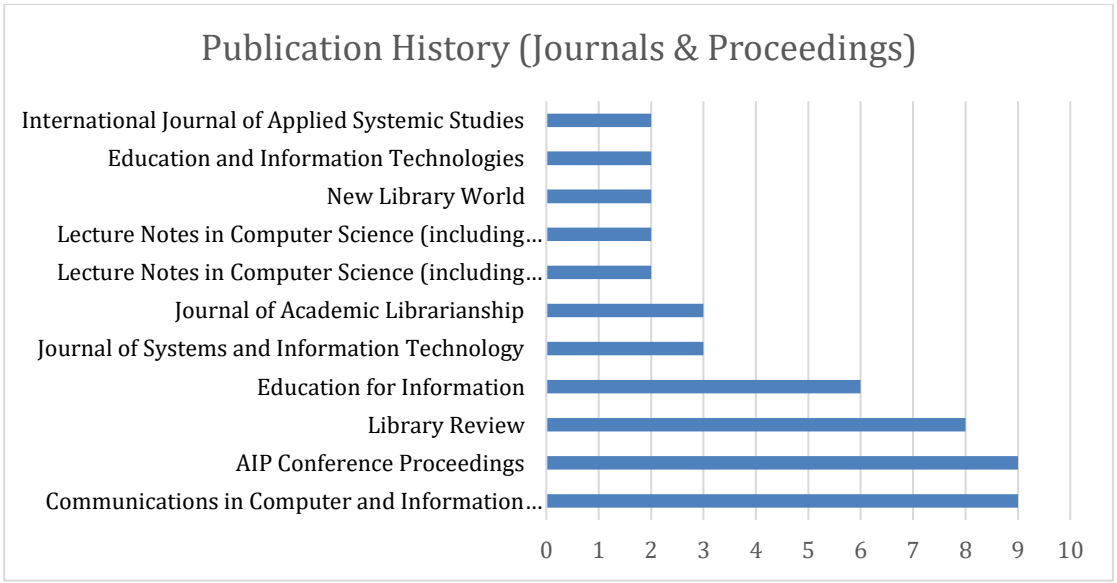
Το δεύτερο ερώτημα που συντάχθηκε αφορά των αριθμό δημοσιεύσεων των παραπάνω τριών τμημάτων ανά επιστημονικό περιοδικό.

Το MySQL ερώτημα που χρησιμοποιήθηκε είναι το παρακάτω :

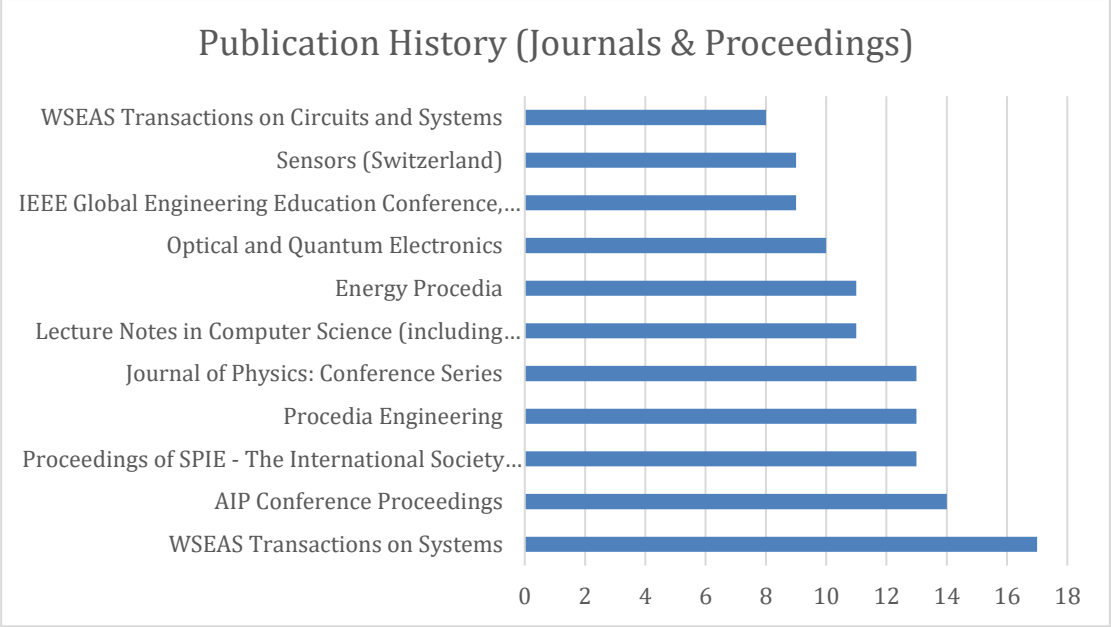
```
SELECT issn, COUNT(*) publications_number, publication_history.title, departments.department_title FROM publication_history
join author_scopus_info on author_scopus_info.author_scopus_info_id = publication_history.author_scopus_info_id
join author_identity on author_scopus_info.author_id = author_identity.author_id
join departments on departments.department_id = author_identity.department_id
where publication_history.issn is not null and publication_history.issn <> '' and departments.department_id = 1
GROUP BY issn
order by publications_number desc;
```

Παρόμοια με το προηγούμενο παράδειγμα ερωτήματος, συνδέθηκαν οι πίνακες author_scopus_info, author_identity, department και Publication_history μέσω της μεθόδου join. Βασικό ρόλο παίζει η μέθοδος group by μέσω της οποίας τα όμοια μεταξύ τους «issn» που υπάρχουν στον πίνακα “publication history” συγκεντρώνονται σε ένα πεδίο επιτρέποντας στη μέθοδο count να «μετρήσει» τις φορές που βρίσκει το κάθε πανομοιότυπο «issn» και να τις αποτυπώσει σαν άθροισμα για τον κάθε ξεχωριστό κωδικό (κάθε κωδικός αντιστοιχεί σε έναν τίτλο επιστημονικού περιοδικού ή συνεδρίου). Σε αυτό το σημείο σημειώνεται πως χρησιμοποιήθηκαν όσες επιστημονικές δημοσιεύσεις είχαν καταγεγραμμένο το «issn» τους στον κατάλογο της Scopus και όχι αυτές που δεν είχαν το πεδίο συμπληρωμένο, με αποτέλεσμα κάποιες δημοσιεύσεις να μην έχουν συμπεριληφθεί. Τέλος, τα αποτελέσματα παρατέθηκαν σε αύξουσα παράταξη με βάση των αριθμό των δημοσιεύσεων.

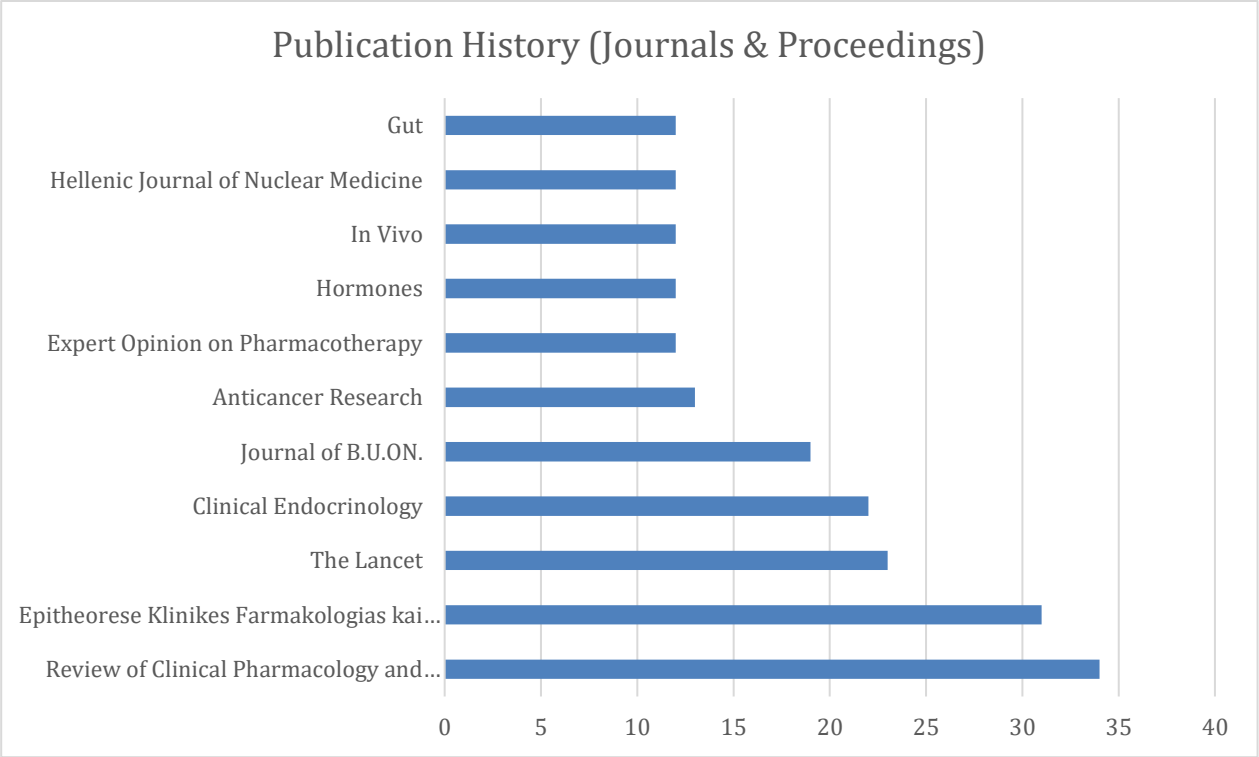
Ακολουθεί η οπτικοποιήση του παραπάνω ερωτήματος για το κάθε τμήμα :



Εικόνα 35 Δημοσιεύσεις περιοδικών / συνεδρίων τμήμα Βιβλιοθηκονομίας Αρχειονομίας & Συστημάτων πληροφόρησης



Εικόνα 36 Δημοσιεύσεις περιοδικών / συνεδρίων τμήμα Ηλεκτρολόγων & Ηλεκτρολόγων Μηχανικών



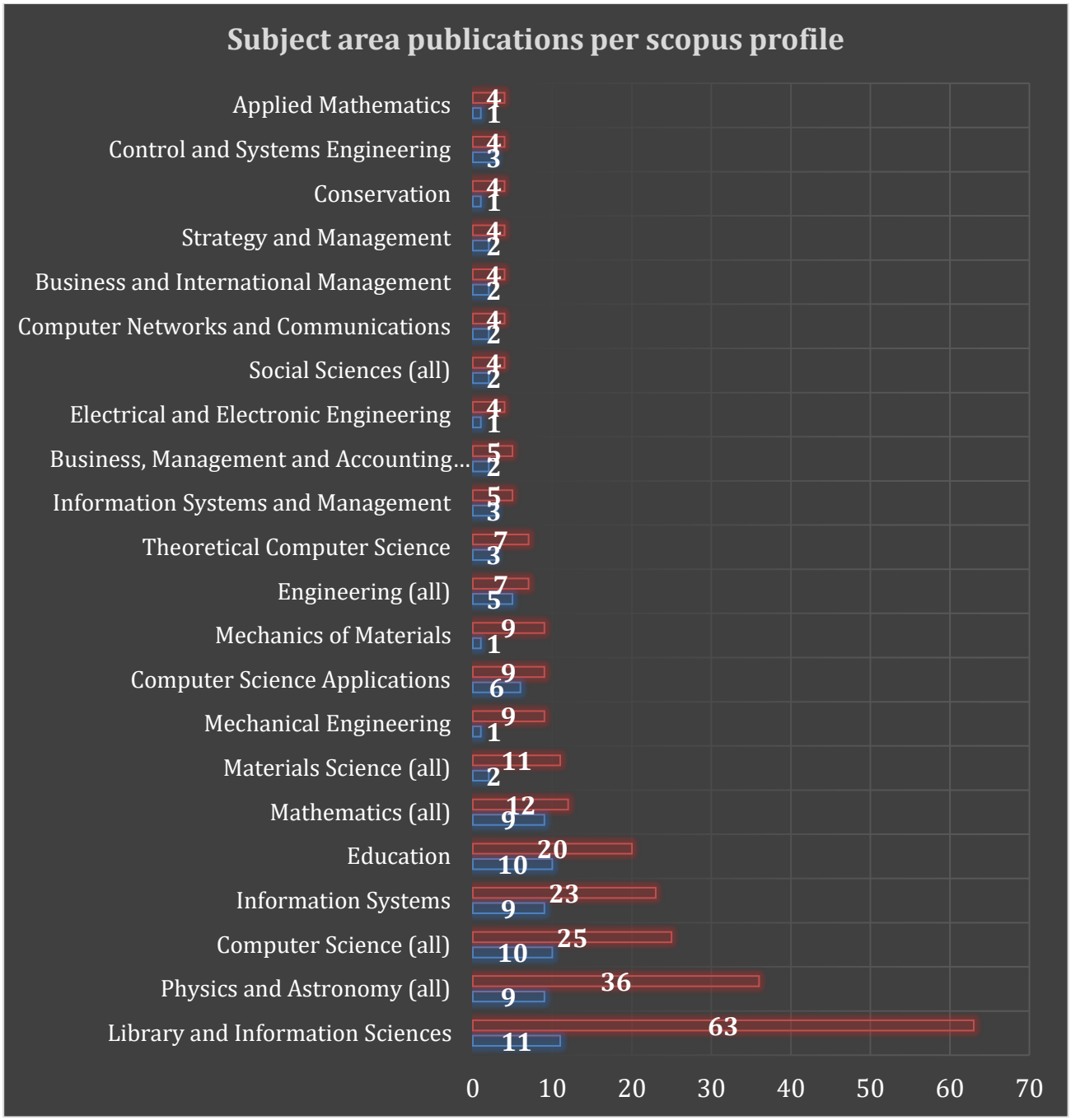
Εικόνα 8 Δημοσιεύσεις περιοδικών / συνεδρίων τμήμα Βιοϊατρικών Επιστημών

Σε αυτό το σημείο αξίζει να αναφερθεί πως τα παραπάνω διαγράμματα περιέχουν μόνο τα 11 περιοδικά / συνέδρια στα οποία το κάθε τμήμα έχει τις περισσότερες δημοσιεύσεις. Όπως παρατηρείται το τμήμα Βιοϊατρικών Επιστημών έχει τις περισσότερες δημοσιεύσεις σε μεμονωμένα περιοδικά (34 σε 1 περιοδικό) ακολουθούμενο από το τμήμα Ηλεκτρολόγων και Ηλεκτρολόγων Μηχανικών (17 σε 1 περιοδικό) και το τμήμα Βιβλιοθηκονομίας Αρχειονομίας και Συστημάτων Πληροφόρησης (9 σε 1 περιοδικό).

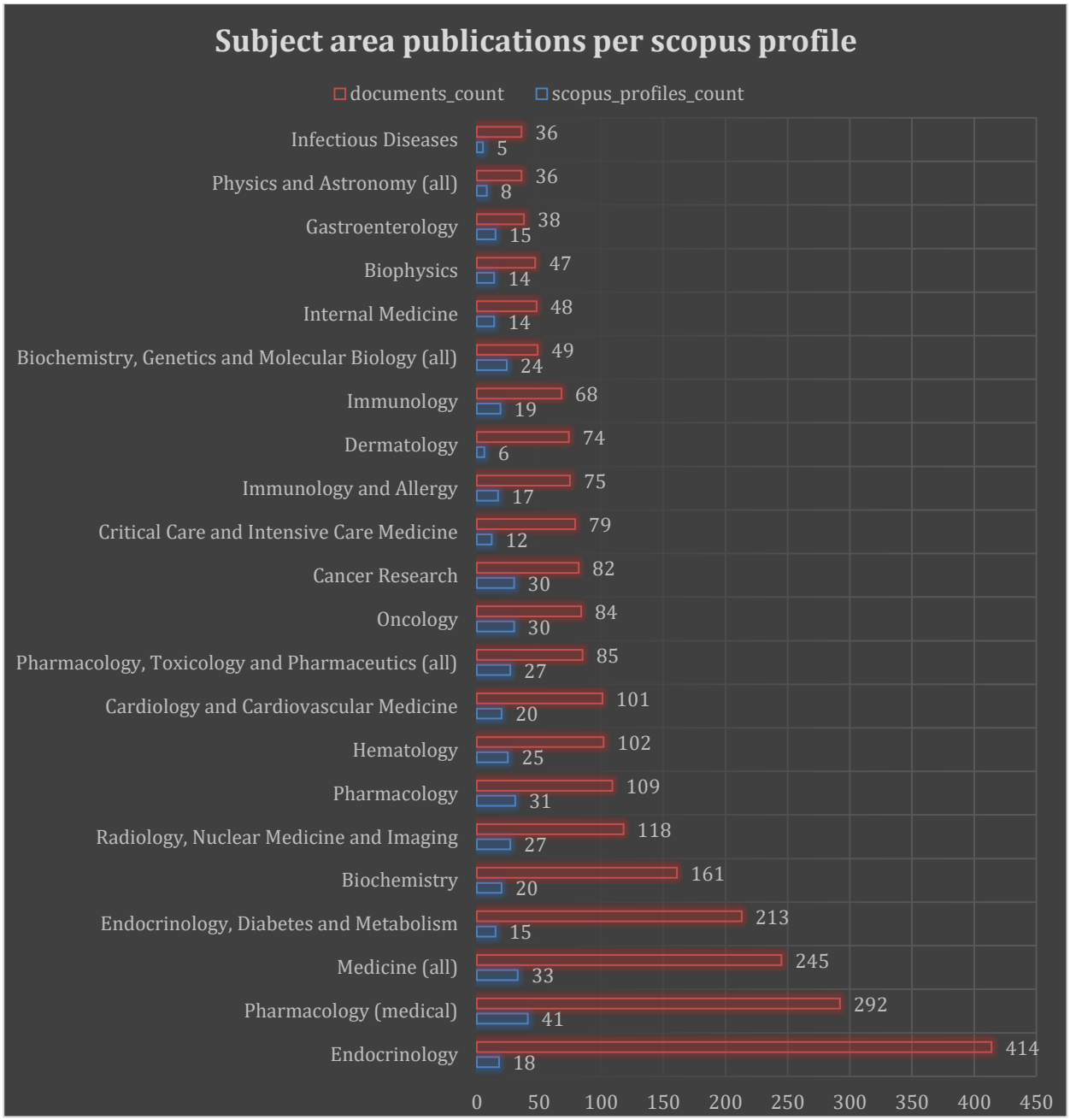
Το επόμενο MySQL ερώτημα δημιουργήθηκε με σκοπό την ανάδειξη των θεματικής κατηγοριών στις οποίες έχουν δημοσιεύσει διαφορετικά scopus προφίλ για τα τρία αυτά τμήματα.

```
select departments.department_title,subjects.subject_title, subjects.subject_code,count(subject_code) as subject_count, sum(scopus_info_subjects.frequency) as frequency_per_subject from subjects
join scopus_info_subjects on subjects.subject_id = scopus_info_subjects.subject_id
join author_scopus_info on scopus_info_subjects.author_scopus_info_id = author_scopus_info.author_scopus_info_id
join author_identity on author_scopus_info.author_id = author_identity.author_id
join departments on departments.department_id = author_identity.department_id
where departments.department_id = 1
group by subjects.subject_code, departments.department_id
order by frequency_per_subject desc;
```

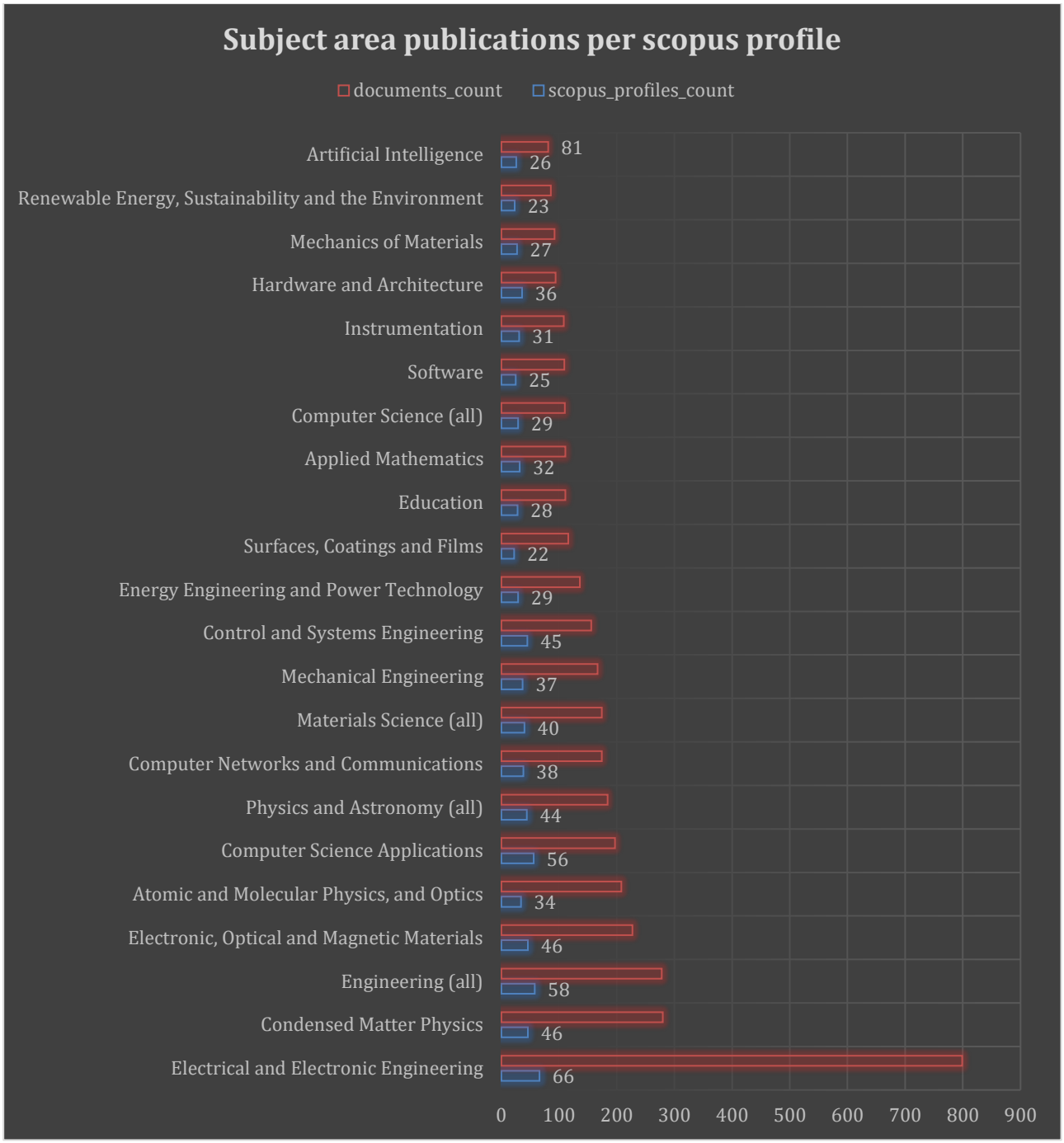
Στο παραπάνω ερώτημα η μέθοδος count αφορά τον κωδικό της κάθε θεματικής ενότητας. Συνδέοντας τους πίνακες scopus_info_subjects, author_scopus_info, author_identity και departments με αυτόν των subjects, σε συνδυασμό με τη μέθοδο group by, τα αποτελέσματα χωρίστηκαν ανά θεματική κατηγορία με τη μέθοδο count να προσδίδει και πάλι το άθροισμα των πανομοιότυπων θεματικών περιοχών. Παρατίθενται τα αποτελέσματα για τα τρία παραπάνω τμήματα :



Εικόνα 37 Δημοσιεύσεις ανά θεματική κατηγορία για διαφορετικούς λογαριασμούς scopus, Τμήμα Βιβλιοθηκονομίας Αρχειονομίας & Συστημάτων Πληροφόρησης



Εικόνα 38 Δημοσιεύσεις ανά θεματική κατηγορία για διαφορετικούς λογαριασμούς scopus, Τμήμα Βιοϊατρικών Επιστημών



Εικόνα 39Δημοσιεύσεις ανά θεματική κατηγορία για διαφορετικούς λογαριασμούς scopus, Τμήμα Ηλεκτρολόγων & Ηλεκτρολόγων Μηχανικών

Όπως παρατηρείται με βάση τα παραπάνω διαγράμματα, το τμήμα Ηλεκτρολόγων & Ηλεκτρολόγων Μηχανικών έχει τον μεγαλύτερο αριθμό δημοσιεύσεων σε μια θεματική κατηγορία από 66 διαφορετικούς λογαριασμούς scopus (Electrical and Electronic Engineering, 800 δημοσιεύσεις) , ενώ ακολουθούν το τμήμα Βιοϊατρικών επιστημών (Endocrinology, 414 δημοσιεύσεις από 18 διαφορετικούς λογαριασμούς scopus) και το τμήμα Βιβλιοθηκονομίας και Συστημάτων Πληροφόρησης (Library and Information Sciences, 63 δημοσιεύσεις από 11 διαφορετικούς λογαριασμούς scopus). Σε αυτό το σημείο αξίζει να αναφερθεί πως επιλέχθηκαν οι είκοσι-δύο θεματικές κατηγορίες με τις περισσότερες δημοσιεύσεις για το κάθε τμήμα και παρατέθηκαν σε αύξουσα σειρά με βάση τον αριθμό των δημοσιεύσεων. Επιπλέον σημειώνεται ότι μια δημοσίευση μπορεί να υπάγεται σε παραπάνω από μία

θεματικές κατηγορίες, επομένως το συνολικό τους άθροισμα, δεν αναλογεί απαραίτητα στον συνολικό αριθμό των δημοσιεύσεων του κάθε τμήματος, αντίστοιχα. Όπως παρατηρείται και με βάση τα προηγούμενα αποτελέσματα τα τμήματα με τις περισσότερες δημοσιεύσεις

Τέλος, επισημαίνεται πως το εργαλείο μπορεί να χρησιμοποιηθεί και για την ανάκτηση γενικών πληροφοριών για τους ερευνητές όπως το επαγγελματικό τους ιστορικό, οι διαφορετικοί τρόποι καταγραφής του ονοματεπωνύμου τους στις δημοσιεύσεις τους κ.α.
Η τελευταία κατηγορία ερωτήματος SQL αφορά το επαγγελματικό ιστορικό για ορισμένους ερευνητές του τμήματος Βιβλιοθηκονομίας και συστημάτων πληροφόρησης.

```
select author_scopus_info.author_id,official_name,official_surname,scopus_affiliation_id,parent_affiliation_id,ip_doc,preferred_affil_name,
affiliations.city, affiliations.postal_code, affiliations.country, affiliations.domain,affiliations.url,current_affiliation,
author_identity.previous_institute from author_scopus_info
inner join author_identity on author_identity.author_id = author_scopus_info.author_id
inner join scopus_info_affiliations on author_scopus_info.author_scopus_info_id = scopus_info_affiliations.author_scopus_info_id
inner join affiliations on scopus_info_affiliations.affiliations_id = affiliations.affiliations_id;
```

George A.	Giannakopoulos	Department of Library Science and Information Systems	Athens	1
George A.	Giannakopoulos	Technological Educational Institute of Athens	Athens	0
George A.	Giannakopoulos	Department of Library Science and Information Systems	Athens	0
George A.	Giannakopoulos	Department of Informatics	Athens	0

Εικόνα 40 Ακαδημαϊκό Ιστορικό : Γιαννακόπουλος, Γ.

Το πεδίο με την τιμή «1» αντιστοιχεί στον ακαδημαϊκό οργανισμό στον οποίο εργάζεται ο ερευνητής, την παρούσα χρονική περίοδο.

Dimitris	Kouis	Department of Archival	Athens	1
Dimitris	Kouis	Hellenic Academic Libraries Link	Athens	0
Dimitris	Kouis	Department of Library Science and Information Systems	Athens	0
Dimitris	Kouis	Department of Library Science and Information Systems	Athens	0
Dimitris	Kouis	Department of Electrical and Computer Engineering	Athens	0
Dimitris	Kouis	National Technical University of Athens	Athens	0
Dimitris	Kouis	Computer Networks Laboratory	Athens	0
Dimitris	Kouis	Department of Electrical Engineering	Athens	0

Εικόνα 41Ακαδημαϊκό Ιστορικό : Κουής, Δ.

Damianos	Sakkas	University of Peloponnese	Tripolis	1
Damianos	Sakkas	Department of Informatics and Telecommunications	Tripolis	0
Damianos	Sakkas	Department of Computer Science and Technology	Tripolis	0
Damianos	Sakkas	Department of Informatics and Telecommunications	Piraeus	0
Damianos	Sakkas	Department of Informatics and Telecommunications	Tripolis	0
Damianos	Sakkas	Department of Computer Science and Technology	Tripolis	0
Damianos	Sakkas	Department of Computer Science	Piraeus	0
Damianos	Sakkas	Department of Computer Science and Technology	Tripolis	0
Damianos	Sakkas	Department of Computer Science and Technology	Tripolis	0

Εικόνα 42 Ακαδημαϊκό Ιστορικό : Σακκάς, Δ.

Συμπεράσματα

Με βάση τα παραπάνω ερωτήματα είναι ασφαλές να εξαχθεί ως συμπέρασμα, πως τα τμήματα με τις περισσότερες δημοσιεύσεις συνήθως είναι και αυτά με τις περισσότερες αυτοαναφορές και ετεροαναφορές. Η παραπάνω, όμως διαπίστωση, δε μπορεί να αποτελέσει, αυτούσια, κριτήριο αξιολόγησης της συνολικής ερευνητικής δραστηριότητας ενός τμήματος ή σχολής, καθώς υπάρχει πληθώρα διαφορετικών παραγόντων οι οποίοι δε συμπεριλαμβάνονται, όπως π.χ. ο “impact factor” των περιοδικών στα οποία έχει δημοσιεύσει και άλλοι βιβλιομετρικοί δείκτες, οι οποίοι στοχεύουν στην ανάδειξη του ποιοτικού περιεχομένου των δημοσιεύσεων.

Επιπλέον, με βάση τα παραπάνω διαγράμματα αναδεικνύονται οι θεματικές περιοχές στις οποίες ένα τμήμα επιλέγει να δημοσιεύσει πιο συχνά, γεγονός που υποδεικνύει την ερευνητική «κατεύθυνση» αυτού. Τύποι δεδομένων σαν και αυτά μπορούν να χρησιμοποιηθούν για να βοηθήσουν νέους ή μελλοντικούς φοιτητές να κατανοήσουν και να έρθουν σε μια πρώτη επαφή, με τα ερευνητικά πεδία στα οποία ίσως εργαστούν μελλοντικά, ή και μεταπτυχιακούς / διδακτορικούς ερευνητές ώστε να γνωρίζουν ποιες είναι οι βασικότερες θεματικές ενότητες του εκάστοτε τμήματος και να διευκολύνονται στην επιλογή ενός ακαδημαϊκού τμήματος έναντι άλλων, με σκοπό τη διεκπεραίωση της ερευνητική τους δραστηριότητας.

Αντιστοίχως, τα περιοδικά τα οποία δέχονται περισσότερες δημοσιεύσεις από το κάθε τμήμα, μπορούν να γνωστοποιούνται στους μελλοντικούς φοιτητές ή διδάκτορες, ώστε να γνωρίζουν που έχουν περισσότερες πιθανότητες να στοχεύσουν για μια επικείμενη επιστημονική δημοσίευση. Επιπρόσθετα, τα περιοδικά αυτά μπορούν να αναζητηθούν και να αναλυθούν από τους ερευνητές, ώστε να έχουν πρόσβαση σε προηγούμενες έρευνες συμφοιτητών τους ή συνεργατών τους, οι οποίες να επιδέχονται ενδεχόμενες προεκτάσεις ή απλά για να διευρύνουν τις γνώσεις τους στα ερευνητικά πεδία της επιλογής τους. Τέλος, τα περιοδικά στα οποία έχουν δημοσιεύσει περισσότερο τμήματα μπορούν να αναδείξουν και το επίπεδο των ερευνών οι οποίες πραγματοποιούνται εντός του τμήματος, χωρίς όμως αυτό το γεγονός να αποτελεί μοναδικό παράγοντα για την αξιολόγηση της ποιότητάς τους.

Συμπερασματικά, ένα εργαλείο συλλογής βιβλιομετρικών δεδομένων όπως αυτό που συστάθηκε στην παρούσα εργασία μπορεί να προσφέρει πληθώρα διαφορετικών δυνατοτήτων, οι οποίες μπορούν να χρησιμοποιηθούν από όλες τις ακαδημαϊκές οντότητες ή και εξωτερικούς συνεργάτες αυτών. Δεδομένα τέτοιου τύπου μπορούν να δώσουν μια πιο αναλυτική και ολοκληρωμένη εικόνα για την ερευνητική δραστηριότητα που διεξάγουν τα τμήματα και οι σχολές / ιδρύματα όπου ανήκουν, είτε προς τους ίδιους τους φοιτητές του τμήματος, είτε προς τους ακαδημαϊκούς που εργάζονται σε αυτά. Παράλληλα, προσδίδουν φερεγγυότητα και εγκυρότητα για τους ίδιους τους οργανισμούς, ενώ μπορούν να

χρησιμοποιηθούν για να προσελκύσουν νέους ερευνητές και ακαδημαϊκό προσωπικό. Επιπρόσθετα, τα δεδομένα αυτά προσφέρουν τη δυνατότητα παραγωγής βιβλιομετρικών δεικτών (όπως π.χ. h-index, n-index κ.λ.π.) έπειτα από περεταίρω ανάλυση, διευκολύνοντας οργανισμούς και πρόσωπα να αναδείξουν το επίπεδο των ερευνών τους καθώς και διατηρώντας μια σταθερή καταγραφή της πορείας τους.

6.1 Αξιοποίηση / Μελλοντικές Επεκτάσεις της έρευνας

Η παρούσα έρευνα, υλοποιήθηκε με σκοπό να αναδείξει τις δυνατότητες που μπορούν να προσφέρουν εργαλεία συλλογής βιβλιομετρικών δεδομένων για τους ακαδημαϊκούς φορείς. Αλλά ένα τέτοιο εργαλείο, κατά την προσωπική μου άποψη, είναι «μισό» χωρίς την περεταίρω επέκτασή του. Αναλυτικότερα, θα μπορούσε να δημιουργηθεί ένα νέο «API» μέσω του οποίου θα προσφέρονται κάθε τύπου πληροφορίες που έχουν συλλεχθεί για όλα επίπεδα : ερευνητικό, τμηματικό, σχολικό και ιδρυματικό. Επιπλέον, όπως προαναφέρθηκε θα μπορούσαν να προστεθούν στην παραπάνω διεπαφή προγραμματισμού εφαρμογών (API) και οι βιβλιομετρικοί δείκτες που μπορούν να προκύψουν έπειτα από περαιτέρω ανάλυση των δεδομένων. Επιπρόσθετα, θα μπορούσε να δημιουργηθούν νέα, αντίστοιχα εργαλεία, τα οποία αντλούν δεδομένα από άλλους παρόχους βιβλιομετρικών δεδομένων όπως π.χ. Google Scholar ή Web of Science, με σκοπό την σύγκριση των αποτελεσμάτων σε αντίστοιχα μοντέλα δεδομένων, με σκοπό την επαλήθευση της εγκυρότητάς τους και την περισσότερο ολοκληρωμένη καταγραφή τους. Τέλος, με βάση της παραπάνω διεπαφές προγραμματισμού εφαρμογών θα μπορούσε να στηριχθεί η δημιουργία ενός ελληνικού ακαδημαϊκού "portal" με πληροφορίες για όλα τα ακαδημαϊκά ιδρύματα της χώρας και της πορείας τους, συνδυαστικά με την οπτικοποίηση των δεδομένων που αφορούν το καθένα από αυτά και άλλες γενικές πληροφορίες για αυτά.

Αναφορές

Adriaanse, L. S. & Rensleigh, C., 2011. Comparing Web of Science, Scopus and Google Scholar from an Environmental Sciences perspective. *South African Journal of Libraries and Information Science*, Ιανουάριος, pp. 169-178.

Anon., 2010. Delaware: s.n.

Kawashima, H. & Tomizawa, H., 2015. Accuracy evaluation of Scopus Author ID based on the largest funding database in Japan. *Scientometrics*, Ιούνιος, Issue 103, pp. 1061-1071.

Krämer, T., Momeni, F. & Mayr, P., 2017. Coverage of Author Identifiers in Web of Science and Scopus. Μάρτιος.

Norris, M. & Oppenheim, C., 2007. Comparing alternatives to the Web of Science for coverage of the social sciences' literature. *Journal of Informetrics*, 1(2), pp. 161-169.

Oracle, χ.χ. *History of MySQL,* s.l.: s.n.

Scopus, 2020. *Scopus Roadmap: What's coming up in 2020 & 2021?.* s.l.:s.n.

Παράρτημα Ι - Κώδικας

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;
using System.Xml;
using System.IO;
using System.Diagnostics;
using System.Net;
using MySql.Data.MySqlClient;

namespace Bibliometrics_app
{



    class Program
    {



        static void Main(string[] args)
        {
            Db_connection cit = new Db_connection();
            Populate_authorIdentity(cit);
            Console.ReadKey();
        }


        /// <summary>
        /// Populates the datatable bibliometrics.author_identity through datatable
citations.author
        /// </summary>
        /// <param name="connection"></param>
        /// <param name="cit"></param>
        ///

        public static void Populate_authorIdentity(Db_connection cit)
        {
            cit = new Db_connection(); // create a ciatations_connections class object in
order to connect to citations database.
            cit.con = new MySqlConnection(cit.connectionString);
            cit.OpenConection();  // open the citations database connection
            DataTable citations_author = new DataTable();
            MySqlDataAdapter adapter = new MySqlDataAdapter("select name,
surname,scopus_id,department,rank,previous_institute from citations2.author where
citations2.author.department = 1 OR 2 OR 3;", cit.con); // select the names and
scopus_ids of authors that have a scopus_id in citations.author table
          // @_ OLD QUERY _@ select name,surname, scopus_id from citations2.author where
scopus_id and name is not null and school = 10;
            adapter.Fill(citations_author); // store the citations.author authors to the
C# DataTable citations_author which contains 2 columns : name  and scopus_id.
            cit.CloseConnection(); // close the citations database connection
            Db_connection bib = new Db_connection(); // creta a db connection object
            bib.con = new MySqlConnection(bib.bib_connectionString); // pass the
connection string into the connection
            bib.OpenBibliometricsConnection(); // open bibliometrics database connection
```

```
            author_identity = new DataTable(); // create our C# datatable to convert the
data into the desired format from the filled (citations_author) datatable.
            author_identity.Columns.Add("official_name", typeof(string)); // crete
collumns : official_name, official_surname , scopus_id into the C# author_identity
Datatable.
            author_identity.Columns.Add("official_surname", typeof(string));
            author_identity.Columns.Add("scopus_id", typeof(string));
            author_identity.Columns.Add("department", typeof(string));
            author_identity.Columns.Add("rank", typeof(string));
            author_identity.Columns.Add("previous_institute", typeof(string));

            string name = "";
            string surname = "";
            string scopus_id = "";
            string department = "";
            string rank = "";
            string previous_institute = "";

            for (int i = 0; i < citations_author.Rows.Count; i++) // foreach row of the
citations_author (stored) C# datatable split the first column of citations_author
datatable (name) into 2 : first_name and last_name.
            {
                name = citations_author.Rows[i][0].ToString();
                surname = citations_author.Rows[i][1].ToString();
                scopus_id = citations_author.Rows[i][2].ToString();
                department = citations_author.Rows[i][3].ToString();
                rank = citations_author.Rows[i][4].ToString();
                previous_institute = citations_author.Rows[i][5].ToString();

                 MySqlCommand cmd1 = new MySqlCommand("INSERT IGNORE INTO
bibliometrics.author_identity (scopus_id,
official_name,official_surname,department_id,rank,previous_institute,data_status,last_upd
ated) "
                                                                    + $"VALUES
('{scopus_id}','{name}','{surname}','{department}','{rank}','{previous_institute}',0,now(
));", bib.con);
                // Pass the last inserted datarows of author_identity DataTable object to
our Database Table (author_identity) into a mysql command.


                 cmd1.ExecuteNonQuery(); // Execute the mysql command.

                MySqlCommand getmaxid = new MySqlCommand("SELECT MAX(author_id) from
bibliometrics.author_identity;", bib.con); // Create a MySql command to the get max id
from our database table.
                max_author_identity_id = Convert.ToInt32(getmaxid.ExecuteScalar());  //
Execute the MySqlCommand and convert it to an integer and store into a public variable
that will be used through the program.

                  if (scopus_id.Contains(") OR AU-ID(")) // check if we have one or more
scopus_ids
                   {
                   scopus_ids = scopus_id.Split(new[] { ") OR AU-ID(" },
StringSplitOptions.None); // split the last inserted (scopus_id) row and pass it into a
public string array variable  as a key to iterate via it, in order to collect all the
necesairy data.
                        populateMasterTable(); // call and execute the customly created
populate Master Table method that populates our database 's  master table.
                   }
                  else  // if we have one
                   {
```

```
                        scopus_ids = new string[] { scopus_id.ToString() }; // the
public string[] contains a single value since it doesn't split into anything.
                        populateMasterTable(); // call and execute the customly created
populate Master Table method that populates our database 's  master table.
                    }

            }
            bib.CloseBibliometricsConnection();
           // connection.Close(); // close the bibliometrics database connection.
        }

        /// <summary>
        /// Foreach scopus_id send an api request: fetch , store and insert the data in
bibliometrics.author_scopus_info table.
        /// </summary>
        public static void populateMasterTable()
        {
            foreach (string s_id in scopus_ids) // if scopus_ids > 1 we iterate through
the array foreach scopus_id it contains.Otherwise it only runs once.
            {
                Debug.WriteLine($"----- Scopus ID in populateMasterTable:{s_id} -----
\n");
                Console.WriteLine($"Starting information retrieval for scopus id : {s_id
} ");
                Db_connection bib_con = new Db_connection();
                bib_con.con = new MySqlConnection(bib_con.bib_connectionString);
                bib_con.OpenBibliometricsConnection();
                //connection.Open(); // open a connection to the bibliometrics database.
                HttpWebRequest req =
(HttpWebRequest)WebRequest.Create(string.Format("https://api.elsevier.com/content/author/
author_id/{0}?apiKey={1}", s_id, apiKey)); // create a webrequest with our string from
the public string array and the API's key.
                req.Method = "GET";
                req.ContentType = "application/x-www-form-urlencoded";
                req.Timeout = 480000;
                HttpWebResponse res = (HttpWebResponse)req.GetResponse(); // get the
response.
                Stream resst = res.GetResponseStream(); // get the stream' s response.
                StreamReader sr = new StreamReader(resst); // create a new streamreader
object and pass in the response.
                reply = sr.ReadToEnd(); // read the response to the end and store into a
string.
                resst.Flush();
                resst.Close();
                res.Close();

                if (reply != null) // if the string isn't empty
                {
                    XmlDocument document = new XmlDocument(); // create a new xml
document
                    document.LoadXml(reply); // and load the string into it.
                    XmlNodeList nodes = document.GetElementsByTagName("author-retrieval-
response"); // create a nodelist.
                    Data_field df = new Data_field(); // create a new datafield class
object named df.

                    foreach (XmlNode node in nodes) // foreach node in the author-
retrieval-response nodelist (named nodes)
                    {
                        XmlElement element = (XmlElement)node; // convert each node into
an xml element
```

```
                        XmlNodeList childnodes = element.ChildNodes; // get the element's
child nodes and create another nodelist from them named called childnodes
                        // the child nodes are : coredata, affiliation-current,
affiliation-history, subject-areas and author-profile.

                        foreach (XmlNode child in childnodes)  // iterate through the
child nodes
                        {
                            XmlElement ele = (XmlElement)child;  // convert each child to
an element.

                            if (ele.Name == "coredata")
                            {
                                Debug.WriteLine($"----- START COREDATA ----- ");
                                XmlNodeList CoredataChilds = ele.ChildNodes; // get the
coredata's element child nodes.

                                foreach (XmlElement e in CoredataChilds) // foreach
element into the the coredata's child nodes
                                {
                                    if (e.Name == "dc:identifier") // if the element's
name is dc:identifier
                                    {
                                        string replaced = e.InnerText; // create a string
that contains the value of the xmlelement (dc:identifier)
                                        df.scopus_id = replaced.Replace("AUTHOR_ID:",
""); // remove text "AUTHOR_ID:" from the stored string value and pass it into the df
class equivalent string.
                                        df.scopus_id.Trim(); // Trim the df class string
for whitespaces.
                                    }

                                    if (e.Name == "eid") df.eid = e.InnerText; // get the
inner text of element and and store it into the df class equivalent string
                                    if (e.Name == "orcid") df.orcid = e.InnerText;
                                    if (e.Name == "document-count") df.document_count =
e.InnerText;
                                    if (e.Name == "cited-by-count") df.citedby_count =
e.InnerText;
                                    if (e.Name == "citation-count") df.citation_count =
e.InnerText;

                                }

                            }

                            if (ele.Name == "author-profile")
                            {
                                Debug.WriteLine($"----- START AUTHOR PROFILE ----- ");
                                XmlNodeList evenmorechilds = ele.ChildNodes;

                                foreach (XmlNode authNode in evenmorechilds) //
evenmorechilds == <status>, <date-created>, <preferred-name> <name-variant>,
                                {
// <classificationgroup>, <publication-range>, <journal-history>, <affiliation-current>,

// <affiliation-history>
                                    XmlElement author_profile_element =
(XmlElement)authNode;
                                    if (author_profile_element.Name == "publication-
range")
                                    {
```

```
                                        Debug.WriteLine($"----- START PUB RANGE  ----- "
);
                                        df.pub_start =
author_profile_element.Attributes["start"].Value.ToString(); // get attributes and store
them into variables.
                                        df.pub_end =
author_profile_element.Attributes["end"].Value.ToString();

                                        MySqlCommand cmd = new MySqlCommand("INSERT INTO
bibliometrics.author_scopus_info
(scopus_id,orcid,eid,documents_count,cited_by_count,citations_count,publication_range_sta
rt,publication_range_end, author_id)"
                                + $"
VALUES({s_id},'{df.orcid}','{df.eid}',{df.document_count},{df.citedby_count},{df.citation
_count},{df.pub_start},{df.pub_end}, {max_author_identity_id});", bib_con.con);
                                        cmd.ExecuteNonQuery(); // populate the
bibliometrics.author_scopus_info table for the last author_id inserted into the
author_identity table.

                                        MySqlCommand command =
bib_con.con.CreateCommand();
                                        string sqlCommandString = "SELECT
MAX(author_scopus_info_id) from bibliometrics.author_scopus_info;";
                                        command.CommandText = sqlCommandString;
                                        max_author_scopus_info_id =
Convert.ToInt32(command.ExecuteScalar()); // get the last inserted id from the (just
populated) author_scopus_info table and convert it to a global variable of type int.

                                    }
                                }
                            }
                        }
                    }
                    bib_con.CloseConnection(); // close the connection to the
bibliometrics database
                    PopulateViaAuthorIdentity(df); // call the populate via author
identity method foreach scopus_id  (hover over PopulateViaAuthorIdentity() for more
info).
                    Console.WriteLine("-----------------------------------------------
");
                    Console.WriteLine($"Retrieved all information for scopus id : {s_id}
");
                    Console.WriteLine();
                }
            }

        }
        /// <summary>
        ///  Populates the rest of the bibliometrics database tables.
        /// </summary>
        public static void PopulateViaAuthorIdentity(Data_field df)
        {
                Db_connection bibliometrics = new Db_connection();
                bibliometrics.con = new
MySqlConnection(bibliometrics.bib_connectionString);
                bibliometrics.OpenBibliometricsConnection();

                if (reply != null)
                {
                    XmlDocument document = new XmlDocument(); // create a new xml
document
```

```
                    document.LoadXml(reply); // load it with  our global streamreader
reply string.
                    XmlNodeList nodes = document.GetElementsByTagName("author-retrieval-
response"); // get the parent node.
                    //Data_field df = new Data_field();

                    foreach (XmlNode node in nodes)
                    {

                        XmlElement element = (XmlElement)node;
                        XmlNodeList childnodes = element.ChildNodes; // create the
childnodes list

                        foreach (XmlNode child in childnodes)
                        {
                            XmlElement ele = (XmlElement)child; // coredata, affiliation-
current, affiliation-history, subject-areas,author-profile || 5 ChildNodes apo to
<author-retrieval-response> parent node
                                                                //
Console.WriteLine(ele.Name + "\n ");  (childnodes)
                            if (ele.Name == "author-profile")
                            {

                                XmlNodeList evenmorechilds = ele.ChildNodes; // create
more childs to iterate through.
                                foreach (XmlNode authNode in evenmorechilds) //
evenmorechilds == <status>, <date-created>, <preferred-name> <name-variant>,
                                {
// <classificationgroup>, <publication-range>, <journal-history>, <affiliation-current>,

// <affiliation-history>
                                    XmlElement author_profile_element =
(XmlElement)authNode;

                                 if (author_profile_element.Name == "preferred-name") //
if the element's name is preferred-name
                                    {
                                    Debug.WriteLine("-----START NAME VARIATIONS -----
\n");
                                    foreach (XmlElement pref_name_ele in
author_profile_element.ChildNodes) // get it's child nodes and store them into datafield
class variables
                                        {
                                            if (pref_name_ele.Name == "initials")
df.preferred_initials = pref_name_ele.InnerText;
                                            if (pref_name_ele.Name == "indexed-name")
df.preferred_indexed_name = pref_name_ele.InnerText;
                                            if (pref_name_ele.Name == "surname")
df.preferred_surname = pref_name_ele.InnerText;
                                            if (pref_name_ele.Name == "given-name")
df.preferred_given_name = pref_name_ele.InnerText;
                                        }
                                    MySqlCommand command = new MySqlCommand
                                        ("INSERT INTO bibliometrics.preferred_author_name
(preferred_initials, preferred_indexed_name, preferred_surname, preferred_given_name,
author_scopus_info_id) "

+$"VALUES('{df.preferred_initials}',"
                                                                            +$"
'{df.preferred_indexed_name}',"
                                                                            +$"
'{df.preferred_surname}',"
```

```
                                                                    +$"
'{df.preferred_given_name}',"
                                                                    +$"
{max_author_scopus_info_id});", bibliometrics.con);
                                    command.ExecuteNonQuery();
                                    // create the query to insert the variables into our
bibliometrics database table and execute it
                                }

                                    if (author_profile_element.Name == "name-variant") //
check for name variations
                                    {
                                    if (author_profile_element.HasAttribute("doc-count"))

                                    {
                                        df.nv_doc_count =
author_profile_element.Attributes["doc-count"].Value.ToString(); // get the documents
count foreach name variation and store it into a class variable
                                    }
                                    else { df.nv_doc_count = "0"; }

                                        XmlNodeList variant_nodes =
author_profile_element.ChildNodes;

                                        foreach (XmlNode namevar in variant_nodes)
                                        {

                                            XmlElement variation = (XmlElement)namevar;

                                            if (variation.Name == "initials")
df.nv_initials = variation.InnerText;  // get the elements' names and store them into
variables.
                                            if (variation.Name == "indexed-name")
df.nv_indexed_name = variation.InnerText;
                                            if (variation.Name == "surname")
df.nv_surname = variation.InnerText;
                                            if (variation.Name == "given-name")
df.nv_given_name = variation.InnerText;

                                        }
                                        MySqlCommand cmd1 = new MySqlCommand("INSERT INTO
bibliometrics.names_variations (initials, indexed_name, surname, given_name, doc_count,
author_scopus_info_id) "

+$"VALUES('{df.nv_initials}', "
                                                                        +$"
'{df.nv_indexed_name}', "
                                                                        +$"
'{df.nv_surname}', "
                                                                        +$"
'{df.nv_given_name}', "
                                                                        +$"
{df.nv_doc_count}, "
                                                                        +$"
{max_author_scopus_info_id});", bibliometrics.con);
                                        cmd1.ExecuteNonQuery(); // create and execute the
query for names_variations table.

                                    }

                                    if (author_profile_element.Name ==
"classificationgroup")
```

```
                                    {
                                        Debug.WriteLine("----- START CLASSIFICATIONS ----
- \n");
                                        if (author_profile_element.FirstChild.Name ==
"classifications")
                                        {

                                            XmlNodeList classifications =
author_profile_element.FirstChild.ChildNodes; // create the childnodes off of the first
child of "classificationgroup" node.

                                            foreach (XmlElement classification in
classifications)
                                            {

                                                string[] clsfctns = new [] {
classification.Attributes["frequency"].Value.ToString(), classification.InnerText }; //
string array to store the frequency and the subject code.
                                                DataTable dt = new DataTable();
                                                MySqlDataAdapter da = new
MySqlDataAdapter($"SELECT subject_id, subject_code from subjects WHERE {
classification.InnerText } = subject_code", bibliometrics.con);
                                                da.Fill(dt); // check if the subject code
exists into the datatable (dt)
                                                if (dt.Rows.Count > 0) // the rows count
always returns > 0 because the subjects table always fills first and the subject_code
fields are the identical.
                                                {
                                                    foreach(DataRow dr in dt.Rows)

                                                        {
                                                            subject_id =
Convert.ToInt32(dr["subject_id"].ToString()); // get the subject id of the row
                                                        }
                                                MySqlCommand command = new
MySqlCommand("insert into bibliometrics.scopus_info_subjects (subject_id, frequency,
author_scopus_info_id) " +
                                                $" VALUES ({subject_id} ,
{classification.Attributes["frequency"].Value.ToString()},
{max_author_scopus_info_id});", bibliometrics.con);
                                                command.ExecuteNonQuery(); // insert into
scopus_info_subjects table the frequency foreach subject area that an author has
published on.
                                            }

                                        }

                                    }

                                }



                                if (author_profile_element.Name == "journal-history")
                                {
                                    Debug.WriteLine("-----START PUBLICATION HISTORY--
-----\n");
                                    XmlNodeList journal_history =
author_profile_element.ChildNodes;

                                    foreach (XmlElement journal in journal_history)
                                    {
```

```
                                        if (journal.Name == "journal")
                                        {
                                            if ((journal.Attributes["type"].Value !=
null) || (journal.Attributes["type"].Value != ""))
                                            {
                                            df.journal_type =
journal.Attributes["type"].Value;
                                            df.journal_type =
MySqlHelper.EscapeString(df.journal_type);
                                        }
                                        XmlNodeList journal_info =
journal.ChildNodes;


                                            foreach (XmlElement j_info in
journal_info)
                                            {
                                            if (j_info.Name == "sourcetitle" &&
j_info.Name != "")
                                            {
                                                df.journal_sourcetitle =
j_info.InnerText;
                                                df.journal_sourcetitle =
MySqlHelper.EscapeString(df.journal_sourcetitle);
                                            }

                                            if (j_info.Name == "sourcetitle-abbrev"
&& j_info.Name != "")
                                            {
                                                df.journal_sourcetitle_abbrev =
j_info.InnerText
                                                df.journal_sourcetitle_abbrev =
MySqlHelper.EscapeString(df.journal_sourcetitle_abbrev);
                                            }
                                                if ((j_info.Name == "issn") &&
(j_info.InnerText != "")) // if the element has a value
                                                {
                                                    df.journal_issn =
j_info.InnerText; // store the value into a class string
                                                }
                                                else
                                                {
                                                    df.journal_issn = ""; // else
store it as null.
                                                }
                                            }
                                        }

                                        MySqlCommand cmd1 = new MySqlCommand
                                        ("INSERT INTO
bibliometrics.publication_history (issn, title, abbrev, type, author_scopus_info_id) "
+$"VALUES('{df.journal_issn}',"
                                                                +$"
'{df.journal_sourcetitle}',"
                                                                +$"
'{df.journal_sourcetitle_abbrev}',"
                                                                +$"
'{df.journal_type}',"
                                                                +$"
{max_author_scopus_info_id});", bibliometrics.con);
```

```
                                        cmd1.ExecuteNonQuery(); // foreach <journal>
element insert the data which correspond to the latest inserted author (max_author_id)
into publication_history table.
                                    }

                                }

                                if (author_profile_element.Name == "affiliation-
current")
                                {
                                    Debug.WriteLine("----- START AFFILIATION CURRENT
----- \n");
                                    XmlNodeList aff =
author_profile_element.ChildNodes;
                                    foreach (XmlElement el in aff)
                                    {
                                        if (el.Name == "affiliation")
                                        {
                                            if (el.HasAttribute("affiliation-id"))
current_scopus_affiliation_id =  el.Attributes["affiliation-id"].Value.ToString(); //
check if attributes exist to avoid exceptions.
                                            if
(el.HasAttribute("parent"))df.aff_parent_id = el.Attributes["parent"].Value.ToString();

                                            XmlNodeList affil = el.ChildNodes;
                                            foreach (XmlElement ee in affil)
                                            {
                                                if (ee.Name == "ip-doc")
                                                {
                                                    if
(ee.HasAttribute("id"))df.aff_ip_doc_id = ee.Attributes["id"].Value.ToString();
                                                }

                                                XmlNodeList affiliation =
ee.ChildNodes; // get affilation childnodes.

                                                foreach (XmlElement elem in
affiliation)
                                                {
                                                    if (elem.Name == "preferred-name"
&& elem.Name != "" )
                                                    {
                                                        df.aff_pref_name =
(elem.InnerText); // store the elements' text into class variables.
                                                        df.aff_pref_name =
MySqlHelper.EscapeString(df.aff_pref_name);
                                                    }
                                                    if (elem.Name == "sort-name" &&
elem.Name != "")
                                                    {
                                                        df.aff_sort_name =
elem.InnerText;
                                                        df.aff_sort_name =
MySqlHelper.EscapeString(df.aff_sort_name);
                                                    }
                                                    if (elem.Name == "address")  //
check for the <address> node.
                                                    {
                                                        XmlNodeList addressnodes =
elem.ChildNodes; // create a new nodelist with the childnodes of <address> node.
```

```
                                                    foreach (XmlNode ad in
addressnodes) // foreach element in the nodelist get the inner text and store into class
variables.
                                                    {
                                                    if (ad.Name == "address-part"
&& ad.Name != "")
                                                    {
                                                        df.aff_adress_part =
ad.InnerText;
                                                        df.aff_adress_part =
MySqlHelper.EscapeString(df.aff_adress_part);
                                                    }
                                                    if (ad.Name == "city" &&
ad.Name != "")
                                                    {
                                                        df.aff_city =
ad.InnerText;
                                                        df.aff_city =
MySqlHelper.EscapeString(df.aff_city);
                                                    }
                                                    if (ad.Name == "state" &&
ad.Name != "")
                                                    {
                                                        df.aff_state =
ad.InnerText;
                                                        df.aff_state =
MySqlHelper.EscapeString(df.aff_state);
                                                    }
                                                    if (ad.Name == "postal-code")
df.aff_postal_code = ad.InnerText;
                                                    if (ad.Name == "country")
df.aff_country = ad.InnerText;

                                                    }
                                                }

                                                if (elem.Name == "org-domain")
df.aff_domain = elem.InnerText; // store more strings.
                                                if (elem.Name == "org-URL")
df.aff_url = elem.InnerText;
                                            }

                                        }


                                    }
                                }
                                    DataTable affiliations = new DataTable(); //
create the affiliations datatable.
                                    MySqlDataAdapter affilcheck = new
MySqlDataAdapter($"select affiliations_id,scopus_affiliation_id from
bibliometrics.affiliations where scopus_affiliation_id =
{current_scopus_affiliation_id}", bibliometrics.con);
                                    affilcheck.Fill(affiliations); // check if
the scopus_affiliation_id already exists in the datatable.
                                    if (affiliations.Rows.Count == 0) // if it
doesn't.
                                    {
                                        int aff_max_id; // create the
affiliations_max_id int.
                                        MySqlCommand commandafill = new
MySqlCommand(
```

```
                                                "insert into
bibliometrics.affiliations (scopus_affiliation_id, parent_affiliation_id, ip_doc,
preferred_affil_name, sort_name, country, address_part, " +
                                                  $"city, state, postal_code,
domain, url) "
                                                +
$"VALUES('{current_scopus_affiliation_id}','"
                                                  +$"
'{df.aff_parent_id}','"
                                                  +$"
'{df.aff_ip_doc_id}', "
                                                  +$"
'{df.aff_pref_name}','"
                                                  +$"
'{df.aff_sort_name}','"
                                                  +$"      '{df.aff_country}','"
                                                  +$"
'{df.aff_adress_part}','"
                                                  +$"      '{df.aff_city}','"
                                                  +$"      '{df.aff_state}','"
                                                  +$"
'{df.aff_postal_code}','"
                                                  +$"      '{df.aff_domain}','"
                                                  +$"      '{df.aff_url}');",
bibliometrics.con);

                                            commandafill.ExecuteNonQuery(); // insert
all the new values into the affiliations table.
                                            MySqlCommand get_aff_max_id = new
MySqlCommand("SELECT MAX(affiliations_id) from bibliometrics.affiliations;",
bibliometrics.con);   // get the last id(max_id) inserted and store into the aff_max_id
integer variable.
                                            aff_max_id =
Convert.ToInt32(get_aff_max_id.ExecuteScalar());
                                            //MySqlCommand auth_sc_affil_command1 =
new MySqlCommand($"insert into bibliometrics.scopus_info_affiliations
(author_scopus_info_id,affiliations_id,current_affiliation)
VALUES({max_author_scopus_info_id},{aff_max_id},'1');", bibliometrics.con);

//auth_sc_affil_command1.ExecuteNonQuery(); // insert into scopus_info_affiliations table
the foreign key from affiliations (aff_max_id) [connects each affiliation_id with each
author] and adds value '1' since it's the author's current affiliation.
                                        }


                                    }

                                    if (author_profile_element.Name == "affiliation-
history") // same logic as the one applied onto the affiliation_current node.
                                    {
// find the nodes, create childnodes , get the elements' innertext , store it into class
variables to use on our insert query.
                                        Debug.WriteLine("----- START AFFILIATION HISTORY
----- \n");
                                        XmlNodeList affhis =
author_profile_element.ChildNodes;
                                        foreach (XmlElement e1 in affhis)
                                        {
                                            if (e1.Name == "affiliation")
                                            {
                                                if (e1.HasAttribute("affiliation-id"))
df.aff_scopus_affiliation_id = e1.Attributes["affiliation-id"].Value.ToString();
```

```
                                                if (e1.HasAttribute("parent"))
df.aff_parent_id = e1.Attributes["parent"].Value.ToString();

                                                XmlNodeList affil = e1.ChildNodes;
                                                foreach (XmlElement ee2 in affil)
                                                {
                                                    if (ee2.Name == "ip-doc")
                                                    {
                                                        if (ee2.HasAttribute("id"))
df.aff_ip_doc_id = ee2.Attributes["id"].Value.ToString();
                                                    }

                                                    XmlNodeList affiliation =
ee2.ChildNodes;

                                                foreach (XmlElement elem3 in affiliation)
                                                {

                                                    if (elem3.Name == "preferred-name" &&
elem3.Name != "")
                                                    {
                                                        df.aff_pref_name =
(elem3.InnerText); // store the elements' text into class variables.
                                                        df.aff_pref_name =
MySqlHelper.EscapeString(df.aff_pref_name);
                                                    }
                                                    if (elem3.Name == "sort-name" &&
elem3.Name != "")
                                                    {
                                                        df.aff_sort_name =
elem3.InnerText;
                                                        df.aff_sort_name =
MySqlHelper.EscapeString(df.aff_sort_name);
                                                    }
                                                    if (elem3.Name == "address")
                                                    {
                                                        XmlNodeList addressnodes =
elem3.ChildNodes;
                                                        foreach (XmlNode ad in
addressnodes)
                                                        {
                                                            if (ad.Name == "address-part"
&& ad.Name != "")
                                                            {
                                                                df.aff_adress_part =
ad.InnerText;
                                                                df.aff_adress_part =
MySqlHelper.EscapeString(df.aff_adress_part);
                                                            }
                                                            if (ad.Name == "city" &&
ad.Name != "")
                                                            {
                                                                df.aff_city =
ad.InnerText;
                                                                df.aff_city =
MySqlHelper.EscapeString(df.aff_city);
                                                            }
                                                            if (ad.Name == "state" &&
ad.Name != "")
                                                            {
                                                                df.aff_state =
ad.InnerText;
```

```
                                                    df.aff_state =
MySqlHelper.EscapeString(df.aff_state);
                                                }
                                                if (ad.Name == "postal-code")
df.aff_postal_code = ad.InnerText;
                                                if (ad.Name == "country")
df.aff_country = ad.InnerText;
                                            }
                                        }

                                        if (elem3.Name == "org-domain")
df.aff_domain = elem3.InnerText;
                                        if (elem3.Name == "org-URL")
df.aff_url = elem3.InnerText;
                                    }

                                    }

                                DataTable affiliations = new DataTable(); //
create the affiliations datatable
                                MySqlDataAdapter affilcheck = new
MySqlDataAdapter($"select affiliations_id,scopus_affiliation_id from
bibliometrics.affiliations where scopus_affiliation_id = {df.aff_scopus_affiliation_id}",
bibliometrics.con);
                                affilcheck.Fill(affiliations); // check if
the scopus_affiliation_id exists in the table.
                                if (affiliations.Rows.Count == 0)  // if it
doesn't
                                {
                                    int aff_max_id;
                                    MySqlCommand commandafill = new
MySqlCommand
                                        ("insert into
bibliometrics.affiliations (scopus_affiliation_id, parent_affiliation_id, ip_doc,
preferred_affil_name, sort_name, country, address_part, "
                                        +$"city, state, postal_code, domain,
url) "
+$"VALUES('{df.aff_scopus_affiliation_id}',"
                                        +$"       '{df.aff_parent_id}', "
                                        +$"       '{df.aff_ip_doc_id}',"
                                        +$"       '{df.aff_pref_name}',"
                                        +$"       '{df.aff_sort_name}',"
                                        +$"       '{df.aff_country}',"
                                        +$"       '{df.aff_adress_part}',"
                                        +$"       '{df.aff_city}',"
                                        +$"       '{df.aff_state}',"
                                        +$"       '{df.aff_postal_code}',"
                                        + $"       '{df.aff_domain}',"
                                        + $"       '{df.aff_url}');",
bibliometrics.con);

                                    commandafill.ExecuteNonQuery(); // insert
it into affiliations with the rest of the stored strings.

                                    MySqlCommand get_aff_max_id = new
MySqlCommand("SELECT MAX(affiliations_id) from bibliometrics.affiliations;",
bibliometrics.con);
                                    aff_max_id =
Convert.ToInt32(get_aff_max_id.ExecuteScalar()); // get the id of the row we just
inserted
```

```
                                                if (df.aff_scopus_affiliation_id ==
current_scopus_affiliation_id) // check if the scopus_id of the affiliation is the same
one with the one we stored as a global variable
                                                {
// from the affiliations_current nodelist.
                                                    MySqlCommand auth_sc_affil_command1 =
new MySqlCommand($"insert into bibliometrics.scopus_info_affiliations
(author_scopus_info_id,affiliations_id, current_affiliation)
VALUES({max_author_scopus_info_id},{aff_max_id},'1');", bibliometrics.con);

auth_sc_affil_command1.ExecuteNonQuery(); // if they are the same just insert the value
'1' on current affiliation column along with the rest of the values.
                                                }
                                                else
                                                {
                                                    MySqlCommand auth_sc_affil_command1 =
new MySqlCommand($"insert into bibliometrics.scopus_info_affiliations
(author_scopus_info_id,affiliations_id, current_affiliation)
VALUES({max_author_scopus_info_id},{aff_max_id},'0');", bibliometrics.con);

auth_sc_affil_command1.ExecuteNonQuery(); // if they are not insert the value '0'.
                                                }
                                            }
                                            else
                                            {

                                                foreach (DataRow dr in affiliations.Rows)
// if the scopus_affiliation_id exists
                                                {
                                                    affiliations_id =
Convert.ToInt32(dr["affiliations_id"].ToString()); // get the row's id
                                                }
                                                if (df.aff_scopus_affiliation_id ==
current_scopus_affiliation_id) // same logic applies as before(lines 537-545) to
eventually populate the scopus_info_affiliation table having skipped the insertion
                                                {
// into the affiliations table since each affiliation has to be unique.
                                                    MySqlCommand auth_sc_affil_command2 =
new MySqlCommand($"insert into bibliometrics.scopus_info_affiliations
(author_scopus_info_id,affiliations_id, current_affiliation)
VALUES({max_author_scopus_info_id},{affiliations_id},'1');", bibliometrics.con);

auth_sc_affil_command2.ExecuteNonQuery();
                                                }
                                                else
                                                {
                                                    MySqlCommand auth_sc_affil_command3 =
new MySqlCommand($"insert into bibliometrics.scopus_info_affiliations
(author_scopus_info_id,affiliations_id, current_affiliation)
VALUES({max_author_scopus_info_id},{affiliations_id},'0');", bibliometrics.con);

auth_sc_affil_command3.ExecuteNonQuery();
                                                }
                                            }
                                        }
                                        }

                                    }
                                    }
                                }
```

```
                            if (ele.Name == "subject-areas") // POPULATES BEFORE ANY
OTHER TABLE IN THIS METHOD (FIRST)
                            {
                                Debug.WriteLine("----- START SUBJECT AREAS ----- \n");
                                XmlNodeList subjects =
document.GetElementsByTagName("subject-areas");  // get the parent node.
                                foreach (XmlNode subject_area in subjects)
                                {
                                    XmlElement sub_element = (XmlElement)subject_area;
                                    XmlNodeList Subjectschilds = sub_element.ChildNodes;
// create it's child nodes.
                                    foreach (XmlNode childElement in Subjectschilds)
                                    {
                                        XmlElement elem = (XmlElement)childElement; //
foreach subject element
                                        string[] subjectsarr = new string[] {
elem.Attributes["abbrev"].Value.ToString(), elem.Attributes["code"].Value.ToString(),
elem.InnerText }; // create a string array of 3 values foreach subject element
                                        DataTable checksubjects = new DataTable();     //
create checksubjects datatable
                                        MySqlDataAdapter adapter = new
MySqlDataAdapter($"SELECT subject_id,subject_code FROM bibliometrics.subjects where
subject_code = { elem.Attributes["code"].Value.ToString()}", bibliometrics.con);
                                        adapter.Fill(checksubjects);  // check if the
subject code already exists into subject table.
                                       if (checksubjects.Rows.Count == 0) // if it
doesn't insert it into the table along with the rest of the stored values.
                                          {
                                                MySqlCommand cmd = new
MySqlCommand("INSERT INTO bibliometrics.subjects (subject_abbrev, subject_code ,
subject_title) " +

$"VALUES
('{elem.Attributes["abbrev"].Value.ToString()}','{elem.Attributes["code"].Value.ToString(
)}', '{elem.InnerText}');", bibliometrics.con);
                                                cmd.ExecuteNonQuery();
                                          } // otherwise do nothing.
                                    }

                                }


                            }
                        }
                    }
                }
            bibliometrics.CloseBibliometricsConnection();
        }

        // GLOBAL VARIABLES.
        public static string apiKey = "a67c47ae6eb8460606c4df18cc79793b";
        public static string[] scopus_ids;
        public static DataTable author_identity;
        public static string reply;
        public static int subject_id;
        public static int affiliations_id;
        public static string current_scopus_affiliation_id;
        public static long max_author_scopus_info_id;
        public static long max_author_identity_id;
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;
using System.Xml;
using System.IO;
using System.Diagnostics;
using System.Net;
using MySql.Data.MySqlClient;

namespace Bibliometrics_app
{
    public class Db_connection
    {
        public string connectionString =
"Server=localhost;Port=3306;Database=citations2;Uid=root;Pwd=cris$14#@;charset=greek"; //
connection string for the citations database.
        public MySqlConnection con; // connection variable

        /// <summary>
        /// opens the connection to citations database
        /// </summary>
        public void OpenConection()
        {
            con = new MySqlConnection(connectionString);
            con.Open();
        }

        /// <summary>
        /// closes the connection to citations database
        /// </summary>
        public void CloseConnection()
        {
            con.Close();
        }

        public string bib_connectionString =
"Server=localhost;Port=3306;Database=bibliometrics;Uid=root;Pwd=cris$14#@;charset=greek";
// connection string for the bibliometrics database.

        /// <summary>
        /// opens the connection to bibliometrics database
```

```
        /// </summary>
        public void OpenBibliometricsConnection()
        {
            con = new MySqlConnection(bib_connectionString);
            con.Open();
        }

        /// <summary>
        /// closes the connection to bibliometrics database
        /// </summary>
        public void CloseBibliometricsConnection()
        {
            con.Close();
        }

    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data;
using System.Data.SqlClient;
using System.Xml;
using System.IO;
using System.Diagnostics;
using System.Net;
using MySql.Data.MySqlClient;

namespace Bibliometrics_app
{
    public class Data_field
    {
        public string eid, scopus_id, orcid, document_count, citedby_count,
citation_count, pub_start, pub_end; // author_scopus_info
        public string journal_type, journal_sourcetitle, journal_sourcetitle_abbrev,
journal_issn; // publication_history
        public string nv_doc_count, nv_initials, nv_indexed_name, nv_surname,
nv_given_name; // names_variations
        public string preferred_initials, preferred_indexed_name, preferred_surname,
preferred_given_name; // preferred_author_name
        public string aff_scopus_affiliation_id, aff_parent_id, aff_ip_doc_id,
aff_pref_name, aff_sort_name, aff_country, aff_adress_part, aff_city, aff_state,
aff_postal_code, aff_domain, aff_url; // affiliation current - affiliation history

    }


}
```