ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Υπολογισμός κανόνων αλυσίδας πάνω σε γράφους

Ευθύμιος Θ. Θεοδωρακόπουλος
Μαριάννα Ρ. Ρεζκάλλα

Επιβλέποντες: **Ιωάννης Ιωαννίδης,** Καθηγητής
Εμμανουήλ Καρβούνης, Υποψήφιος Διδάκτωρ

ΑΘΗΝΑ

ΣΕΠΤΕΜΒΡΙΟΣ 2016

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Υπολογισμός κανόνων αλυσίδας πάνω σε γράφους

Ευθύμιος Θ. Θεοδωρακόπουλος
Α.Μ.: 1115201000028

Μαριάννα Ρ. Ρεζκάλλα
Α.Μ.: 1115201000154

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Ιωάννης Ιωαννίδης,** Καθηγητής
Εμμανουήλ Καρβούνης, Υποψήφιος Διδάκτωρ

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια, παρατηρείται η ανάπτυξη ενός μεγάλου αριθμού διαδικτυακών κοινωνικών δικτύων, πολλά από τα οποία έχουν προσελκύσει εκατοντάδες εκατομμυρίων χρήστες. Ενσωματωμένη στις βάσεις δεδομένων αυτών των κοινωνικών δικτύων βρίσκεται μία πληθώρα πληροφοριών, χρήσιμες για ένα μεγάλο εύρος εφαρμογών. Η ανάλυση των κοινωνικών δικτύων περικλείει θέματα, όπως υπολογισμό μετρικών πάνω σε γράφους. Πολλοί ερευνητές έχουν εκφράσει την ανάγκη για ένα καλύτερο υπολογιστικό μοντέλο ή γλώσσα επερωτήσεων για την επίτευξη του στόχου να επιτρέψουν τους χρήστες να εκφράσουν ερωτήματα πάνω στους προσωπικούς τους κοινωνικούς γράφους.

Με βάση τα παραπάνω, στα πλαίσια της παρούσας πτυχιακής εργασίας, αναπτύχθηκε ένα σύστημα, το οποίο επιλύει κανόνες σε μορφή αλυσίδας για την κάλυψη των μετρικών πάνω σε γράφους και παρέχει στο χρήστη ένα εύχρηστο περιβάλλον για την διατύπωση ερωτημάτων αλλά και την εξαγωγή αποτελεσμάτων.

Τέλος, η απόδοση υπήρξε κύριος παράγοντας στα πλαίσια ανάπτυξης του συστήματος και για αυτό το λόγο χρησιμοποιήθηκαν κατάλληλες δομές δεδομένων και αλγόριθμοι, με σκοπό ένα ικανοποιητικό αποτέλεσμα.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Επιστήμη των Υπολογιστών, Ανάλυση Γράφων, Θεωρία Γράφων

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: υπολογισμός κανόνων αλυσίδας, αλγόριθμοι γράφων, κοινωνικά δίκτυα, γράφοι ιδιοτήτων, ανάλυση δεδομένων

ABSTRACT

In recent years, we have witnessed the rise of a large number of online social networks, many of which have attracted hundreds of millions of users. Embedded in these databases of social networks is a wealth of information, useful for a wide range of applications. Social network analysis encompasses topics such as computation of general graph metrics. Many researchers expressed the need for a better computational model or query language to eventually achieve the goal of letting consumers express queries on their personal social graphs.

Based on the above, in the context of this thesis, we developed a system, which solves chain rules in order to cover computations on graphs. It also provides a user-friendly environment so as to express queries and get results.

Finally, the main factor of developing this system was the performance, which led us to use proper data structures and algorithms, in order to achieve a satisfying result.

SUBJECT AREA: Computer Science, Graph Analysis, Graph Theory

KEYWORDS: computation of chain rules, graph algoriyhms, social networks, labeled graphs, data analysis

Στις οικογένειες μας.

ΕΥΧΑΡΙΣΤΙΕΣ

Για τη διεκπεραίωση της παρούσας Πτυχιακής Εργασίας, θα θέλαμε να ευχαριστήσουμε τους επιβλέποντες, καθηγητή Ιωάννη Ιωαννίδη και υποψήφιο διδάκτωρ Εμμανουήλ Καρβούνη για τη συνεργασία και την πολύτιμη συμβολή τους στην ολοκλήρωση της. Τέλος θα θέλαμε να ευχαριστήσουμε τις οικογένειές μας για την διαχρονική υποστήριξη τους και τη συνεχή συμπαράσταση τους σε όλες τις βαθμίδες εκπαίδευσης μας.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΠΡΟΛΟΓΟΣ

Η παρούσα πτυχιακή εργασία αποτελεί μέρος των υποχρεώσεων για την λήψη πτυχίου στο Τμήμα Πληροφορικής και Τηλεπικοινωνιών του Εθνικού Καποδιστριακού Πανεπιστημίου Αθηνών. Η εργασία εκπονήθηκε κατά τη διάρκεια του ακαδημαΪκού έτους 2015-2016 υπό την επίβλεψη του Καθ, Ιωάννη Ιωαννίδη και Υποψήφιου Διδάκτωρ Εμμανουήλ Καρβούνη.

1. ΕΙΣΑΓΩΓΗ

Σε αυτό το κεφάλαιο γίνεται μια εισαγωγή στο θέμα της παρούσας πτυχιακής εργασίας. Αρχικά παρουσιάζεται το αντικείμενο της εργασίας, δηλαδή το τι αυτή πραγματεύεται. Στη συνέχεια αναφέρονται ο στόχος της, δηλαδή τι θα ήθελε να επιτύχει, και το κίνητρο το οποίο οδήγησε στην δημιουργία της. Τέλος επισημαίνεται ο τρόπος με τον οποίο διαρθρώνεται η εργασία.

1.1 Αντικείμενο εργασίας

Αντικείμενο της παρούσας πτυχιακής εργασίας είναι η σχεδίαση και η υλοποίηση ενός συστήματος-γλώσσας, το οποίο δέχεται ως είσοδο αρχεία που περιέχουν πληροφορία πάνω σε γράφους καθώς και μία σχέση μεταξύ αυτών, κάνει τους κατάλληλους υπολογισμούς και τέλος εξάγει τα αποτελέσματα. Συγκεκριμένα, η παραπάνω σχέση αποτελείται από κανόνες σε μορφή αλυσίδας (chain rules) [1], με τη δυνατότητα επέκτασης του όρου με ιδιότητες (property chain).

Το σύστημα αυτό είναι προσαρμοσμένο στην SQLite [2], μέσω του παρεχόμενου χαρακτηριστικού των εικονικών πινάκων (virtual tables) [3]. Χρησιμοποιώντας τους εικονικούς πίνακες δίνεται η δυνατότητα ανάπτυξης και χρήσης σύνθετων δομών δεδομένων, χωρίς όμως αυτές οι δομές να είναι εμφανείς στον χρήστη.

Ο χρήστης έχει τη δυνατότητα να κάνει κάποια ερωτήματα στο σύστημα έτσι ώστε να έχει τα επιθυμητά αποτελέσματα. Σύμφωνα με το παραπάνω σύστημα ο χρήστης θα μπορεί να κάνει σύνθετους υπολογισμούς με την ευχρηστία της SQLite και χωρίς την αναγκαία γνώση κάποιας περίπλοκης γλώσσας.

1.2 Στόχος εργασίας

Πρωταρχικός στόχος της εργασίας αυτής είναι η δημιουργία ενός εύχρηστου και φιλικού προς το χρήστη συστήματος για την επίλυση ερωτημάτων, χρησιμοποιώντας την ευελιξία που παρέχει η SQLite. Με αυτό τον τρόπο ο χρήστης δεν χρειάζεται εξειδίκευση σε κάποια γλώσσα προγραμματισμού χαμηλότερου επιπέδου, παρά μόνο στη γλώσσα που αναπτύχθηκε στα πλαίσια αυτής της πτυχιακής καθώς και τουλάχιστον βασικές γνώσεις SQL [4].

Το σύστημα παίρνει από την είσοδο που ορίζει ο χρήστης όλη την απαραίτητη πληροφορία που πρόκειται να επεξεργαστεί, σύμφωνα με τις προδιαγραφές του. Η παραπάνω είσοδος περιλαμβάνει τόσο τα αρχεία όσο και τους κανόνες πάνω σε αυτά. Οπότε από καθαρά τεχνική άποψη το εργαλείο που δημιουργήθηκε στα πλαίσια της παρούσας εργασίας αποσκοπεί στην παραγωγή αποτελεσμάτων, που πηγάζουν από το συνδυασμό των παραπάνω.

Όλα τα παραπάνω αποτελούν τη βάση έτσι ώστε να επιτευχθεί ο απώτερος στόχος, που είναι η ανάπτυξη εργαλείου για εύκολο και γρήγορο υπολογισμό σύνθετων ερωτημάτων, δίχως να εμπλέκεται ο χρήστης με τεχνικές λεπτομέρειες, και με την πλήρη προσήλωσή του να επικεντρώνεται στο περιεχόμενο, μέσα από μια εύκολη στη χρήση και στην εκμάθηση γλώσσα, χωρίς να επηρεάζεται από εξωγενής παράγοντες.

1.3 Κίνητρο εργασίας

Βασικό κίνητρο υπήρξε το πρόβλημα των συστάσεων (recommendations) επεξεργαζόμενοι τα δεδομένα σε μορφή γράφων. Συγκεκριμένα, το πρόβλημα της σύστασης ενός αντικειμένου σε ένα άτομο με βάση τους κοντινούς του ανθρώπους ήταν η πρώτη επαφή με τον χώρο. Πιο συγκεκριμένα, η μοντελοποίηση αυτού του προβλήματος εμφανίζεται στην επόμενη εικόνα.

```
RECOMMEND(X,Z,W):- TRUSTS(X,Y,W1),LIKES(Y,Z,W2),W=W1*W2.
```

Εικόνα 1: Μοντελοποίηση προβλήματος συστάσεων σε Datalog

Το επόμενο βήμα ήταν η γενίκευση αυτού του προβλήματος, έτσι ώστε το σύστημα να δέχεται οποιοδήποτε πρόβλημα αυτής της μορφής.

Σημαντικός παράγοντας αποτέλεσε επίσης το γεγονός ότι ένα τέτοιο εργαλείο θα έπρεπε να είναι φιλικό προς το χρήστη. Αυτό είχε σαν αποτέλεσμα την χρήση των εικονικών πινάκων της SQLite.

1.4 Διάρθρωση της εργασίας

Ολοκληρώνοντας το κεφάλαιο 1, το οποίο είναι μια εισαγωγή στην εργασία, αναφέρεται παρακάτω περιληπτικά το περιεχόμενο των επόμενων κεφαλαίων:

Στο κεφάλαιο 2 δίνεται το θεωρητικό υπόβαθρο, το οποίο σχετίζεται άμεσα με την ανάπτυξη του παρόντος εργαλείου. Συγκεκριμένα, περιγράφονται κάποιες βασικές έννοιες και ορισμοί, που αφορούν την θεωρία στην οποία βασίστηκε η σχεδίαση και η υλοποίηση του συστήματος.

Στο κεφάλαιο 3 περιγράφονται οι βασικές αρχές υλοποίησης. Ειδικότερα, αναφέρονται συνοπτικά οι προδιαγραφές και η αρχιτεκτονική του συστήματος. Στη συνέχεια παρουσιάζονται οι βασικές δομές δεδομένων που χρησιμοποιήθηκαν, καθώς και οι βασικοί αλγόριθμοι που υλοποιήθηκαν στα πλαίσια ανάπτυξης του συστήματος. Τέλος, περιγράφεται το συντακτικό της γλώσσας, σύμφωνα με το οποίο θα πρέπει ο χρήστης να δημιουργήσει τον εικονικό πίνακα.

Στο κεφάλαιο 4 παρουσιάζονται οι μετρήσεις που έγιναν και τα γραφήματα που εξάγονται από αυτές, ώστε να γίνει εμφανής η απόδοση του συστήματος.

Στο κεφάλαιο 5 παρουσιάζεται το εγχειρίδιο χρήσης, μέσα από το οποίο δίνονται σαφής οδηγίες για την εγκατάσταση του συστήματος, ώστε οποιοσδήποτε χρήστης να το χρησιμοποιήσει.

Στο κεφάλαιο 6, το οποίο είναι το τελευταίο, παρουσιάζονται τα συμπεράσματα που εξάγονται από την παρούσα πτυχιακή εργασία. Στη συνέχεια αναφέρονται κάποιες πιθανές μελλοντικές επεκτάσεις που μπορούν να εφαρμοστούν στο σύστημα και τέλος ακολουθεί ένας μικρός επίλογος.

2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΡΑΘΡΟ

Σε αυτό το κεφάλαιο δίνεται αρχικά ο ορισμός και η περιγραφή της έννοιας του κανόνα αλυσίδας, καθώς και της επέκτασης του που είναι ο κανόνας αλυσίδας με ιδιότητες.

Ακολουθεί μια σύντομη αναφορά στο τι είναι γράφος ιδιοτήτων (property graph) [5]. Στην συνέχεια παρουσιάζονται οι εμφωλευμένοι πίνακες με ουρά και δίνεται ο ορισμός τους. Μετά ακολουθεί η σύνδεση των γράφων ιδιοτήτων με τους εμφωλευμένους πίνακες (tail-nested tables) [6], καθώς και η ένωση τέτοιων πινάκων. Ύστερα, δίνεται η περιγραφή ενός αλγορίθμου διάσχισης και πράξεων πάνω σε γράφο, καθώς και αναλυτικό παράδειγμα με εικόνες.

Τέλος, αναφέρονται κάποια πράγματα σχετικά με τους εικονικούς πίνακες της SQLite.

2.1 Κανόνας αλυσίδας

Δεδομένου ενός συνόλου *P* δυαδικών κατηγορημάτων ονομάτων και ενός θετικού ακεραίου *n*, ένας κανόνας αλυσίδας ορίζεται να είναι ένα συνδετικό ερώτημα (conjunctive query) της μορφής:

$$q(X_1, X_{n+1}) : - a_1(X_1, X_2), a_2(X_2, X_3), \ldots, a_i(X_i, X_{i+1}), a_{i+1}(X_{i+1}, X_{i+2}), \ldots, a_{n-1}(X_{n-1}, X_n), a_n(X_n, X_{n+1}).$$

όπου $a_i \in P$, $i = 1, 2, \ldots, n$ και X_i , $i = 1, 2, \ldots, n + 1$ είναι διακριτές μεταβλητές.

Παρατηρείτε ότι το σώμα ενός ερωτήματος αλυσίδας περιέχει ως υποστόχους έναν αριθμό από δυαδικές οντότητες, οι οποίες εάν θεωρηθούν ως ένας γράφος, του οποίου ο κάθε κόμβος έχει αντιστοιχηθεί σε ένα στοιχείο από ένα σύνολο συμβόλων (labeled graph) [7], σχηματίζουν ένα απλό κατευθυνόμενο μονοπάτι, όπου ο αρχικός και τελικός κόμβος αυτού του μονοπατιού είναι οι παράμετροι της κεφαλής.

2.2 Κανόνας αλυσίδας με ιδιότητες

Η παραπάνω μορφή αλυσίδας που περιγράφηκε μπορεί να επεκταθεί εφόσον προστεθούν ιδιότητες για κάθε *Xi*, της μορφής *Yi1,Yi2,. . .,Yim*. Η προσθήκη ιδιοτήτων σε συνδυασμό με τον κανόνα αλυσίδας συνθέτουν τον όρο κανόνας αλυσίδας με ιδιότητες (property chain rule) και είναι της μορφής:

$$q(X_1, X_n, Y) :- a_1(X_1, Y_{11}, \ldots, Y_{1m}, X_2, Y_{21}, \ldots, Y_{2k}), a_2(X_2, Y_{31}, \ldots, Y_{3u}, X_3, Y_{41}, \ldots, Y_{4v}), \ldots, a_{n-1}(X_{n-1}, Y_{(nn-1)1}, \ldots, Y_{(nn-1)c}, X_n, Y_{nn1}, \ldots, Y_{nnb}),\ Y = Y_{11} \circ Y_{12} \circ \ldots \circ Y_{nnb}$$

όπου $\circ = +, -, *, /$

Παρατηρείτε ότι μπορούν να γίνουν πράξεις μεταξύ των διαφορετικών ιδιοτήτων κάθε κανόνα και να παραχθεί μια νέα ιδιότητα στο κύριο κατηγόρημα. Ακόμα μπορούν να γίνουν και απλές αναθέσεις ιδιοτήτων σε παραμέτρους του κύριου κατηγορήματος.

2.3 Γράφος Ιδιοτήτων

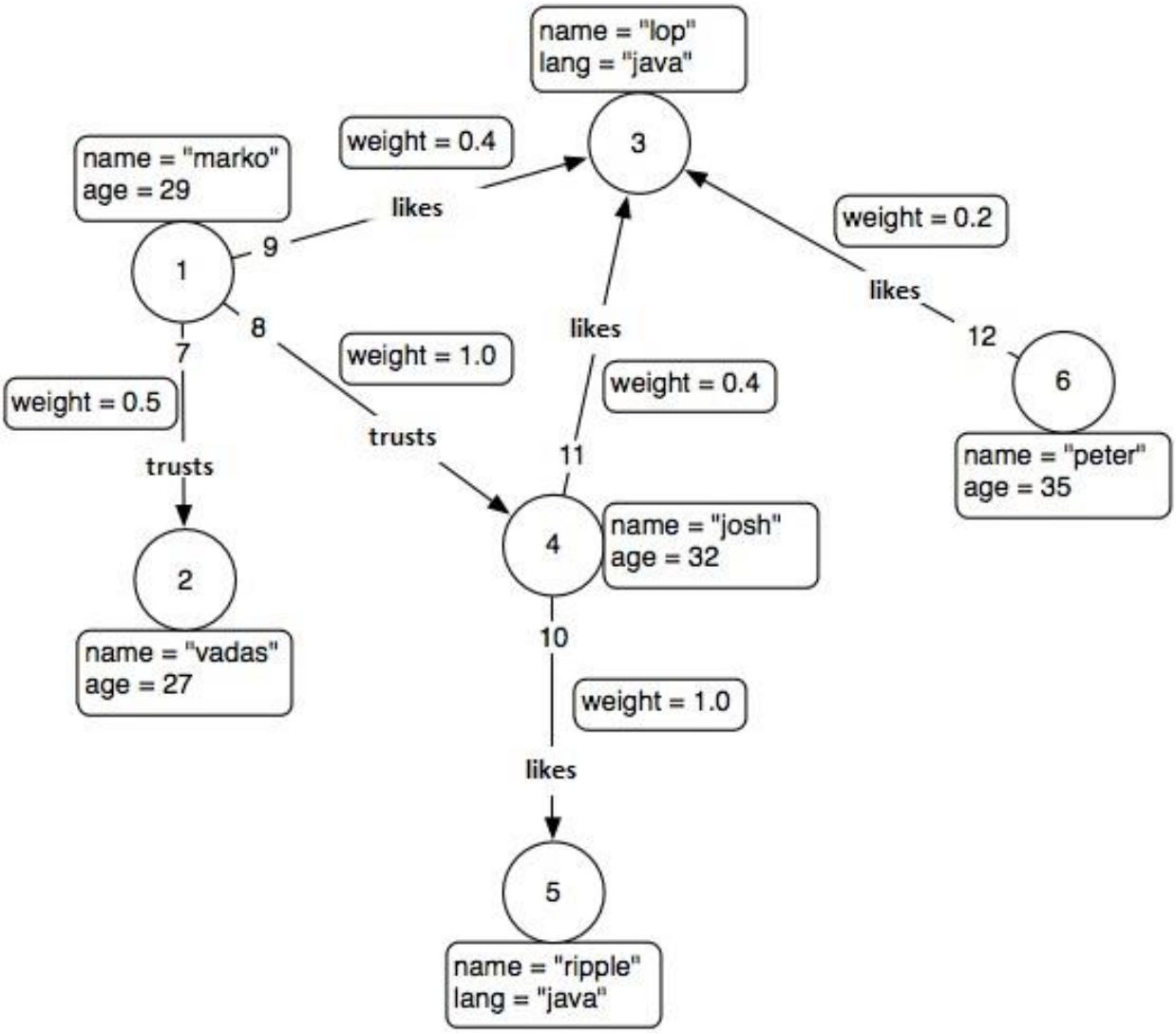
Με τον όρο γράφος ιδιοτήτων ορίζεται μια δομή δεδομένων τύπου γράφου, η οποία έχει ορισμένα χαρακτηριστικά. Αυτά τα χαρακτηριστικά περιγράφονται ως εξής.

Ο πρώτος όρος, *κλειδί/τιμή (key/value),* αναφέρεται στο γεγονός ότι τόσο οι κόμβοι (nodes) όσο και οι ακμές (edges) μπορούν να έχουν οποιοδήποτε αριθμό από ιδιότητες που συνδέονται με αυτά.

Ο δεύτερος όρος, *κατευθυνόμενος (directed),* αναφέρεται στο γεγονός ότι οι ακμές του γράφου έχουν μια κατευθυντικότητα, η οποία είναι ότι υπάρχει μια κεφαλή και μια ουρά σε κάθε ακμή.

Τέλος, ο τρίτος όρος, *πολυ-σχεσιακότητα (multi-relational),* αναφέρεται στο γεγονός ότι μπορούν να υπάρξουν πολλοί τύποι ακμών και αυτό έχει ως αποτέλεσμα να υπάρξουν πολλοί τύποι σχέσεων μεταξύ των ακμών.

Παράδειγμα τέτοιου γράφου φαίνεται στην παρακάτω εικόνα:



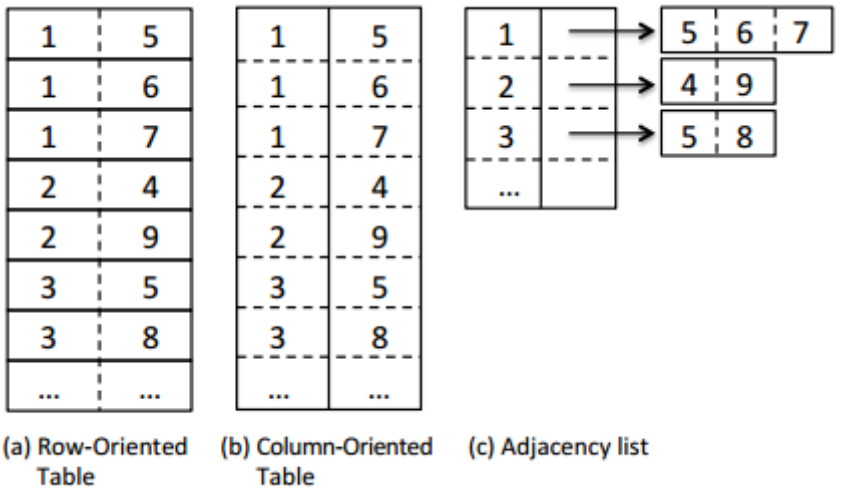
Εικόνα 2: Παράδειγμα γράφου ιδιοτήτων

Ένας γράφος ιδιοτήτων έχει τα εξής στοιχεία:

- ένα σύνολο κόμβων
  - κάθε κόμβος έχει ένα μοναδικό αναγνωριστικό.
  - κάθε κόμβος έχει ένα σύνολο εξερχόμενων ακμών.
  - κάθε κόμβος έχει ένα σύνολο εισερχόμενων ακμών.
  - κάθε κόμβος έχει μια συλλογή από ιδιότητες ορισμένες από ενα πίνακα κατακερματισμού από κλειδί σε τιμή.
- ένα σύνολο από ακμές
  - κάθε ακμή έχει ένα μοναδικό αναγνωριστικό.
  - κάθε ακμή έχει έναν εξερχόμενο κόμβο ουρά.
  - κάθε ακμή έχει έναν εισερχόμενο κόμβο κεφαλή.
  - κάθε ακμή έχει μια ετικέτα, η οποία υποδηλώνει τον τύπο της σχέσης μεταξύ των δύο κόμβων.
  - κάθε ακμή έχει μια συλλογή από ιδιότητες ιδιότητες ορισμένες από ενα πίνακα κατακερματισμού από κλειδί σε τιμή.
  -

2.4 Εμφωλευμένοι πίνακες με ουρά

Στις συμβατικές υλοποιήσεις της Datalog [8] ή σε σχεσιακά συστήματα βάσεων δεδομένων, τα δεδομένα αποθηκεύονται σε δυσδιάστατους πίνακες γραμμών και στηλών. Ένας πίνακας, ο οποίος είναι προσανατολισμένος με βάση τις στήλες (column-oriented table), αποθηκεύει τις τιμές στην ίδια στήλη συνεχόμενα, ενώ ένας πίνακας, ο οποίος είναι προσανατολισμένος με βάση τις γραμμές (row-oriented table), αποθηκεύει ολόκληρες εγγραφές (γραμμές) τη μια μετά την άλλη. Για την αποθήκευση πληροφοριών, όπως ακμές ενός γράφου, οι κόμβοι-πηγές των ακμών πρέπει να αποθηκευτούν επανειλημμένα, όπως φαίνεται και στην Εικόνα 2.



Εικόνα 3: Αναπαράσταση (a) Row-Oriented Table (b) Column-Oriented Table (c) Adjacency list

Σε προστακτικές γλώσσες προγραμματισμού οι γράφοι αναπαρίστανται συχνά ως μια λίστα γειτνίασης (adjacency list). Όπως φαίνεται στην Εικόνα 2 (c), μια λίστα γειτνίασης μπορεί να αποθηκεύσει συμπαγώς ακμές ενός γράφου ή οποιαδήποτε λίστα ιδιοτήτων, η οποία συνδέεται με κάποιο κόμβο. Αυτή η αναπαράσταση όχι μόνο εξοικονομεί χώρο, αλλά το πρόγραμμα γίνεται πιο αποδοτικό επειδή ένας και μόνο έλεγχος αρκεί για να συγκριθούν οι κόμβοι-πηγές όλων των ακμών στην ίδια λίστα γειτνίασης.

Σε αυτό το σημείο εισάγεται η έννοια των εμφωλευμένων πινάκων με ουρά, ως γενίκευση της λίστας γειτνίασης. Η τελευταία στήλη ενός τέτοιου πίνακα μπορεί να περιέχει δείκτες σε δυσδιάστατους πίνακες, των οποίων οι τελευταίες στήλες μπορούν να επεκταθούν σε άλλους εμφωλευμένους πίνακες με ουρά.

Ο εμφωλιασμός υποδεικνύεται από παρενθέσεις στη δήλωση του πίνακα. Για παράδειγμα, δηλώνεται μια σχέση *R* με τρεις εμφωλευμένους πίνακες, όπου *n1,n2,n3* είναι ο αριθμός των στηλών σε κάθε επίπεδο.

R(<type>$c_{1},\ldots,$<type>c_{n1-1},

(<type>$c_{n1,1},\ldots,$<type>$c_{n1,n2-1}$,

(<type>$c_{n1,n2,1},\ldots,$<type>$c_{n1,n2,n3}$)))

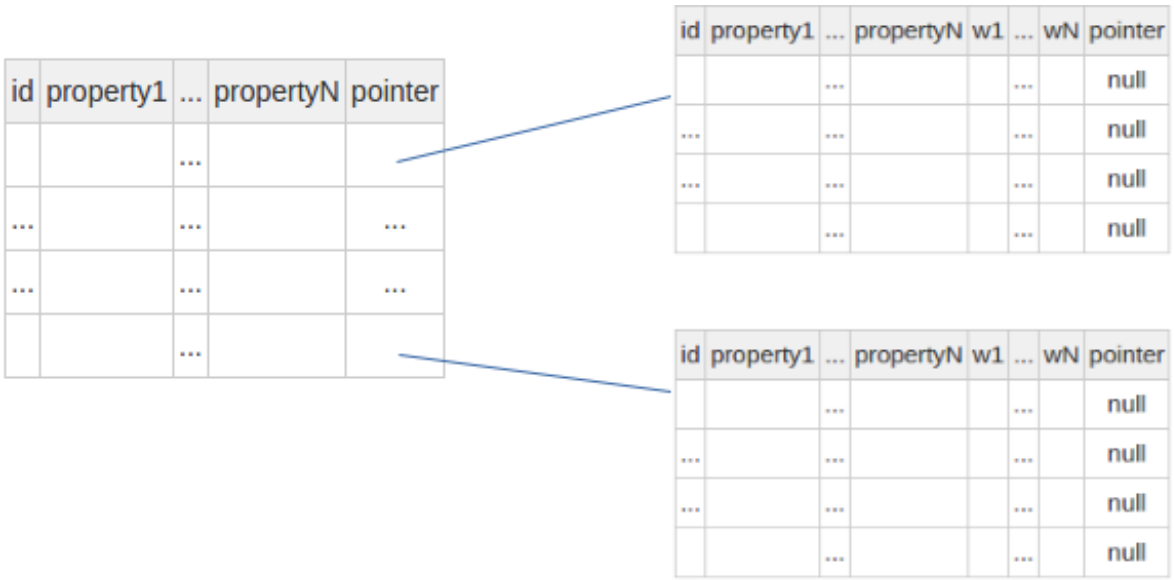
2.5 Σύνδεση γράφου ιδιοτήτων με εμφωλευμένους πίνακες με ουρά

Ο γράφος ιδιοτήτων μπορεί να αναπαρασταθεί αποδοτικά με τους εμφωλευμένους πίνακες με ουρά.

Πιο αναλυτικά, μια ακμή, η οποία εξέρχεται από έναν κόμβο και εισέρχεται σε έναν άλλο, εκφράζει μια σύνδεση-σχέση μεταξύ αυτών των δυο με κάποιο βάρος. Για παράδειγμα ο κόμβος Α εμπιστεύεται τον κόμβο Β με βάρος w. Ένας κόμβος μπορεί να έχει παραπάνω από μια σύνδεση-σχέση με έναν άλλο κόμβο, αλλά και την ίδια με διαφορετικούς κόμβους.

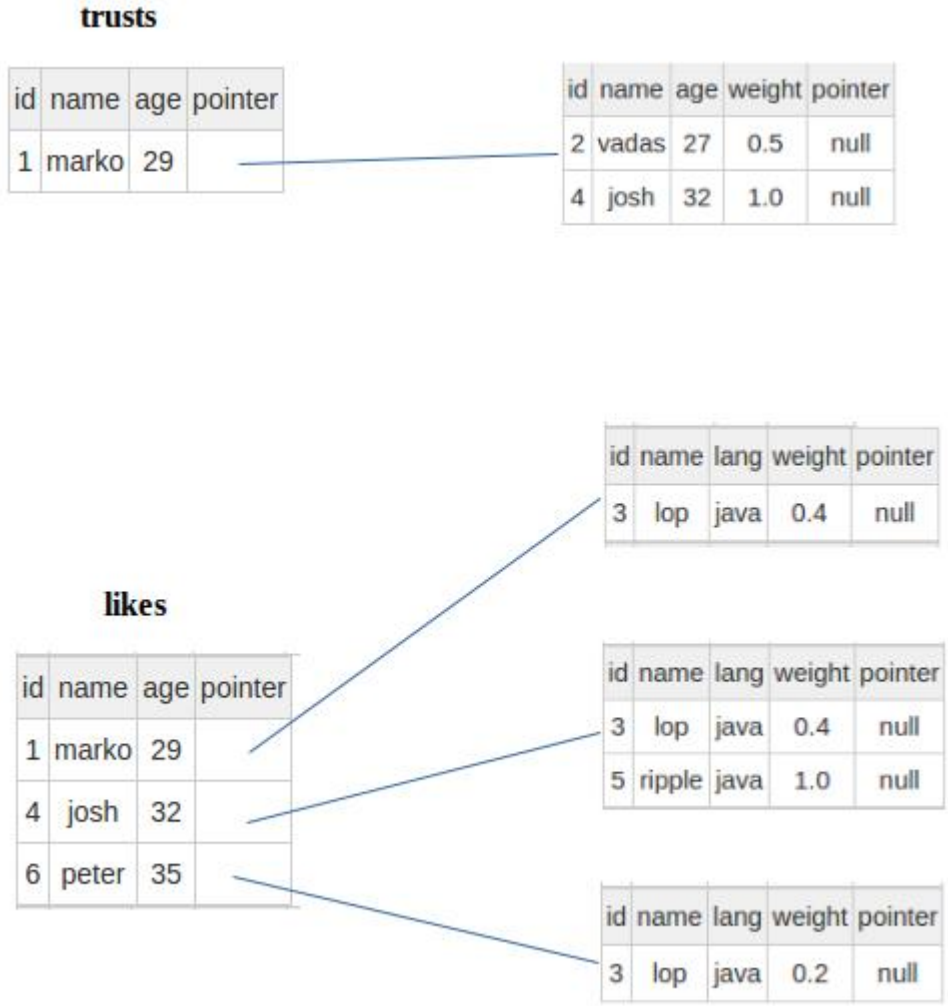
Αν ληφθούν υπόψιν οι παραπάνω αναφορές αλλά και οι ορισμοί, τότε για κάθε σύνδεση-σχέση αρχικοποιείται ένας εμφωλευμένος πίνακας με ουρά. Ο πίνακας αυτός, στο πρώτο επίπεδο έχει τους κόμβους εκκίνησης της συγκεκριμένης ακμής. Πιο συγκεκριμένα, στην πρώτη θέση κάθε εγγραφής είναι το αναγνωριστικό του κόμβου, στις επόμενες θέσεις βρίσκονται οι ιδιότητες-χαρακτηριστικά του, τα οποία έχουν σχέση ένα προς ένα με αυτόν και στην τελευταία θέση ένας δείκτης. Ο δείκτης αυτός δείχνει στο δεύτερο επίπεδο, δηλαδή στους κόμβους που συνδέονται με τον εκάστοτε κόμβο εκκίνησης. Το δεύτερο επίπεδο, περιέχει και αυτό στην πρώτη θέση κάθε εγγραφής το αναγνωριστικό του κόμβου, στις επόμενες θέσεις τις ιδιότητες-χαρακτηριστικά του, τα οποία έχουν σχέση ένα προς ένα με τον κόμβο και στις τελευταίες θέσεις τα βάρη της ακμής, σε περιπτώσεις που υπάρχουν περισσότερα από ένα.

Η γενική αυτή αναπαράσταση φαίνεται με σχήμα στην Εικόνα 3:



Εικόνα 4: Γενική αναπαράσταση γράφου ιδιοτήτων με εμφωλευμένους πίνακες με ουρά

Η αναπαράσταση του γράφου ιδιοτήτων της Εικόνας1 είναι η εξής:



Εικόνα 5: Παράδειγμα αναπαράστασης γράφου ιδιοτήτων με εμφωλευμένους πίνακες με ουρά

2.6 Ένωση εμφωλευμένων πινάκων (Join tail-nested tables)

Υπάρχουν δύο είδη πράξεων ένωσης (join operations), οι εμφωλευμένες ενώσεις βρόχου (nested loop joins) και οι ενώσεις βρόχου κατακερματισμού (hashed loop joins).

Στην πρώτη κατηγορία, για να γίνει επανάληψη σε μια στήλη ενός εμφωλευμένου πίνακα με ουρά, απλά εμφωλιάζονται οι επαναλήψεις των στηλών του γονικού πίνακα, από τους πιο εξωτερικούς, στους πιο εσωτερικούς πίνακες. Παρατηρείται ότι εάν η στήλη που πρόκειται να ενωθεί δεν βρίσκεται σε φύλλο του εμφωλευμένου πίνακα με ουρά, τότε κάθε στοιχείο, στο οποίο έχει γίνει επίσκεψη ίσως αντιστοιχεί σε παραπάνω από μία καταχωρήσεις. Σε αυτό έγκειται το πλεονέκτημα του σχήματος.

Στη δεύτερη κατηγορία, τιμές μιας στήλης κατακερματίζονται, έτσι ώστε να γίνεται απευθείας αναζήτηση των καταχωρήσεων που περιέχουν μια τιμή ενδιαφέροντος. Για να υποστηριχθεί αυτό για στήλες σε εμφωλευμένους πίνακες, ο κάθε ένας πίνακας περιλαμβάνει έναν δείκτη πίσω στο γονικό του πίνακα και στην εκάστοτε εγγραφή. Με αυτόν τον τρόπο, από μια καταχώρηση στη στήλη ενός εμφωλευμένου πίνακα, μπορεί εύκολα να εντοπιστεί η εγγραφή στο γονικό πίνακα, ο οποίος περιέχει την παραπάνω πληροφορία.

2.7 Διάσχιση και πράξεις πάνω σε γράφο

Όπως αναφέρθηκε παραπάνω, μια ακμή, η οποία εξέρχεται από έναν κόμβο και εισέρχεται σε έναν άλλο, εκφράζει μια σύνδεση-σχέση μεταξύ αυτών των δυο με κάποιο βάρος. Αυτή η περιγραφή είναι αρκετά γενική και μπορεί να επεκταθεί εάν μεταξύ του κόμβου εκκίνησης και του τελικού κόμβου παρεμβάλλονται ενδιάμεσοι κόμβοι, δημιουργώντας ένα γράφο, όπου οι ακμές του εκφράζουν την ίδια σύνδεση-σχέση.

Για να γίνει εξαγωγή της τελικής ακμής, πρέπει να γίνει διάσχιση του γράφου. Ένας από τους τρόπους ώστε να πραγματοποιηθεί αυτό είναι με την χρήση του αλγορίθμου αναζήτησης πρώτα κατά βάθος (BFS - Breadth-first search), καθώς και κάποιων αθροιστικών συναρτήσεων (aggregation functions) κατά τη διάρκεια του αλγορίθμου. Οι συναρτήσεις που χρησιμοποιούνται για να εξυπηρετήσουν τον παραπάνω σκοπό είναι, η συνάρτηση εξάπλωσης (expand aggregation function) και η συνάρτηση ένωσης μονοπατιών (concatenate aggregation function).

Η πρώτη αθροιστική συνάρτηση αναφέρεται στην περίπτωση, όπου ο αλγόριθμος αναζήτησης πρώτα κατά βάθος, εξερευνήσει έναν κόμβο και μέσω κάποιας ακμής του εξαπλωθεί σε κάποιον άλλον κόμβο. Ο δεύτερος κόμβος παίρνει τιμή με βάση την τιμή του κόμβου που εξερευνήθηκε και του βάρους της ακμής. Η παραπάνω τιμή υπολογίζεται σύμφωνα με τη μαθηματική πράξη που έχει οριστεί για τη συγκεκριμένη αθροιστική συνάρτηση.

Η δεύτερη αθροιστική συνάρτηση αναφέρεται στην περίπτωση, όπου ένας κόμβος έχει ήδη εξερευνηθεί από τον αλγόριθμο αλλά ένα διαφορετικό μονοπάτι ξαναφθάνει σε αυτόν. Γίνεται συγχώνευση των δύο μονοπατιών ανάλογα με την πράξη που έχει οριστεί για τη συνάρτηση αυτή. Ένα πρόβλημα που προκύπτει σε αυτό το σημείο είναι εάν ο κόμβος, στον οποίο φθάνει πάλι μονοπάτι πρέπει να μπει στην ουρά, ώστε να εξερευνηθεί ξανά. Η λύση σε αυτό δίνεται στο παρακάτω κομμάτι ψευδοκώδικα:

```
1  if( concat(wold,wnew != wold)
2  {
3      insertInQ( vertex );
4  }
```

Εικόνα 6: Ψευδοκώδικας για συγχώνευση ακμών και εισαγωγή κόμβου στην ουρά

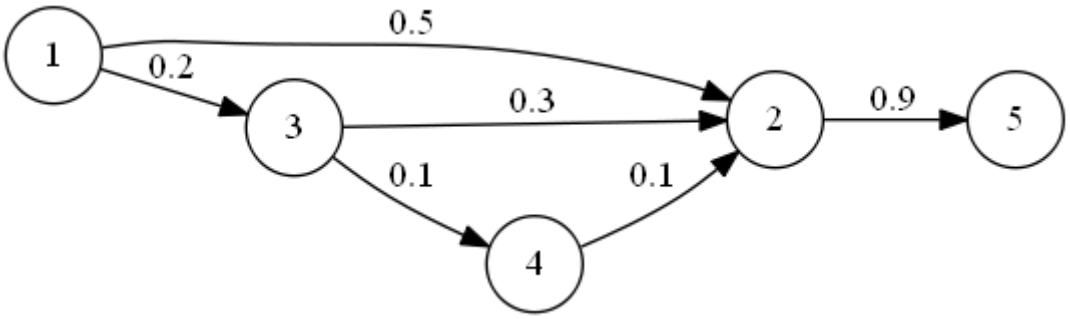
Στον παραπάνω ψευδοκώδικα, η μεταβλητή w_{old} αναφέρεται στην παλιά τιμή του κόμβου και η w_{new} στην τιμή που φέρνει το νέο μονοπάτι. Η συνάρτηση concat εκτελεί την πράξη που έχει οριστεί και η insertInQ τοποθετεί τον κόμβο στην ουρά Q, η οποία χρησιμοποιείται από τον αλγόριθμο

Για να γίνει πιο κατανοητή η παραπάνω προσέγγιση δίνεται το παρακάτω παράδειγμα:

Κόμβος εκκίνησης είναι ο κόμβος με αναγνωριστικό 1, τελικός κόμβος είναι ο κόμβος με αναγνωριστικό 5. Οι κόμβοι με αναγνωριστικό 2, 3, 4 είναι ενδιάμεσοι. Ως αθροιστική συνάρτηση για εξάπλωση είναι η πράξη της πρόσθεσης (+) και για ένωση, το ελάχιστο μονοπάτι (min).

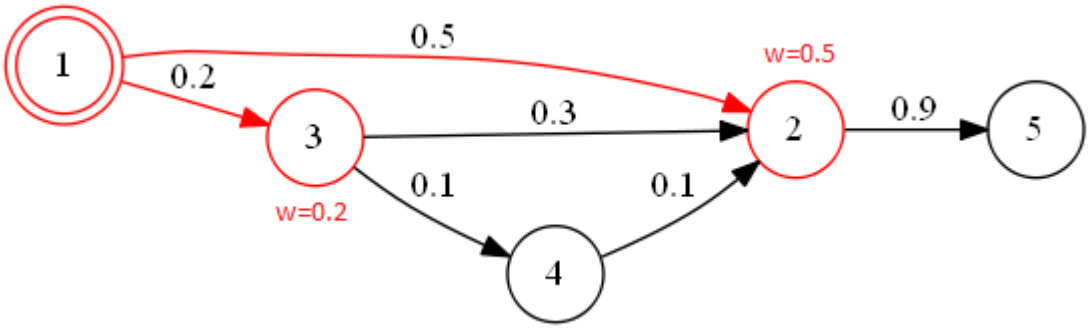
concat = min

aggregate = +



Εικόνα 7: Αρχική κατάσταση γράφου

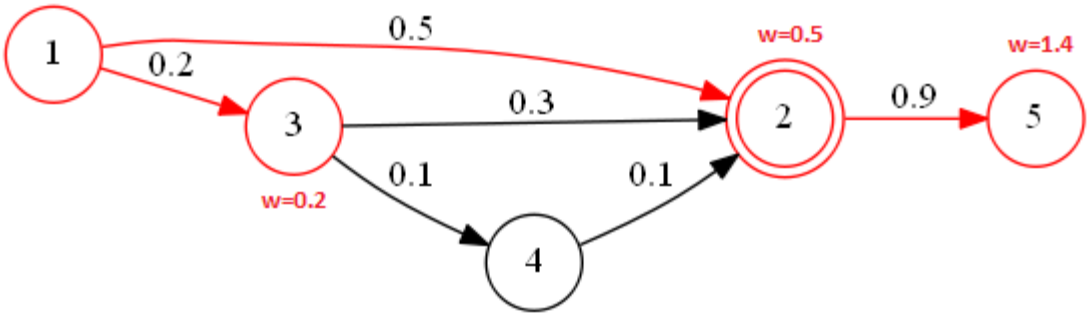
Στο πρώτο στάδιο του αλγορίθμου ανοίγουν οι γείτονες (κόμβοι 2, 3) του κόμβου 1, παίρνουν τιμές με βάση τα βάρη των ακμών και μπαίνουν στην ουρά Q.



Εικόνα 8: Κατάσταση γράφου μετά το άνοιγμα των γειτόνων του κόμβου 1

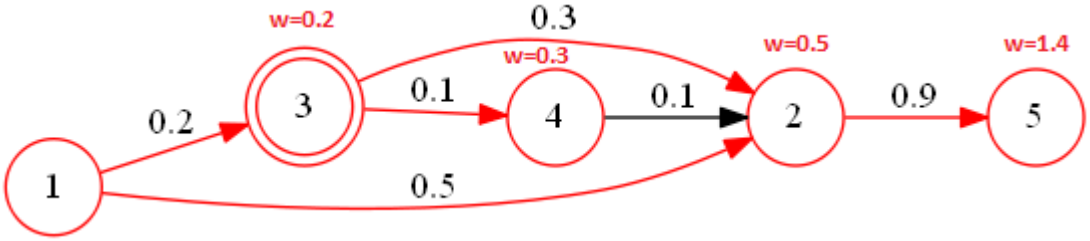
Στο δεύτερο στάδιο γίνεται εξαγωγή από την ουρά Q. Όταν δεν υπάρχει άλλος κόμβος τότε τελειώνει ο αλγόριθμος.

Αρχικά, γίνεται εξαγωγή ο κόμβος 2 και ανοίγει το γείτονα του (κόμβος 5). Αυτός παίρνει τιμή με βάση την τιμή του κόμβου 2 προσθέτοντας σε αυτή το βάρος της ακμής, όπως ορίζει η αθροιστική συνάρτηση εξάπλωσης και μπαίνει στην ουρά Q.



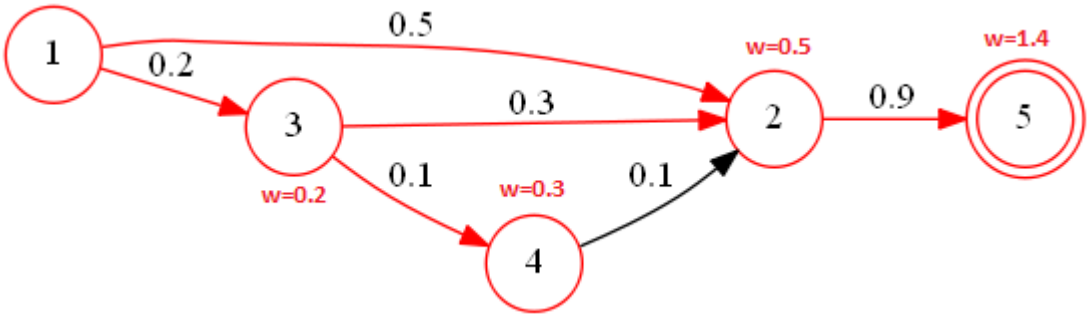
Εικόνα 9: Κατάσταση γράφου μετά το άνοιγμα του γείτονα του κόμβου 2

Στη συνέχεια, γίνεται εξαγωγή ο κόμβος 3 και ανοίγει τους γείτονες του (κόμβους 2, 4). Ο κόμβος 4 παίρνει τιμή με βάση την τιμή του κόμβου 3 προσθέτοντας σε αυτή το βάρος της ακμής. Ο κόμβος 2 έχει ήδη ανοιχτεί και πάρει τιμή. Σε αυτό το σημείο, χρησιμοποιείται ο ψευδοκώδικας της Εικόνας 5: min(0.5,0.5) = 0.5, οπότε ο κόμβος 2 δεν αλλάζει τιμή, ούτε μπαίνει στην ουρά Q.



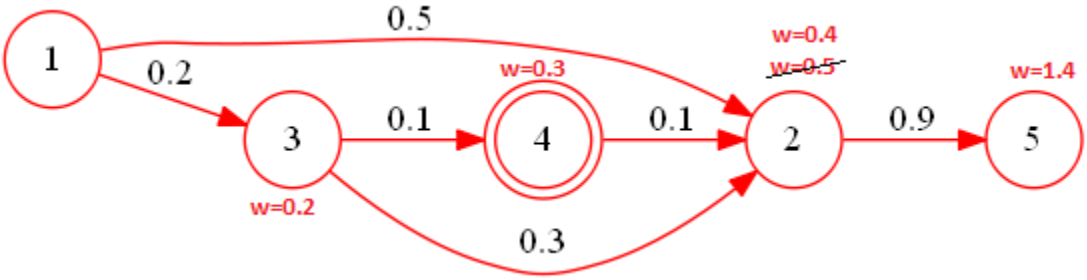
Εικόνα 10: Κατάσταση γράφου μετά το άνοιγμα των γειτόνων του κόμβου 3

Έπειτα, γίνεται εξαγωγή ο κόμβος 5, αλλά δεν έχει γείτονες.



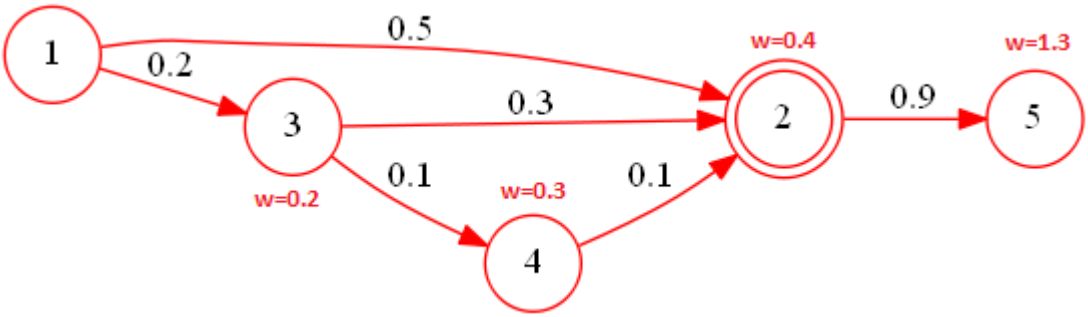
Εικόνα 11: Κατάσταση γράφου μετά το άνοιγμα του κόμβου 5

Μετά από αυτό, γίνεται εξαγωγή ο κόμβος 4 και ανοίγει το γείτονα του (κόμβος 2). Ο κόμβος 2 έχει ήδη ανοιχτεί και πάρει τιμή. Ξαναχρησιμοποιείται ο ψευδοκώδικας της Εικόνας 5: min(0.5,0.4) != 0.5, οπότε ο κόμβος 2 αλλάζει τιμή και μπαίνει στην ουρά Q.



Εικόνα 12: Κατάσταση γράφου μετά το άνοιγμα του γείτονα του κόμβου 4

Εν τέλει, γίνεται εξαγωγή ο κόμβος 2 και ανοίγει το γείτονα του (κόμβος 5). Αυτός παίρνει τιμή με βάση την νέα τιμή του κόμβου 2 προσθέτοντας σε αυτή το βάρος της ακμής και μπαίνει στην ουρά Q, από όπου θα γίνει εξαγωγή, δεν έχει γείτονες και τελειώνει ο αλγόριθμος.



Εικόνα 13: Κατάσταση γράφου μετά το άνοιγμα του γείτονα του κόμβου 2

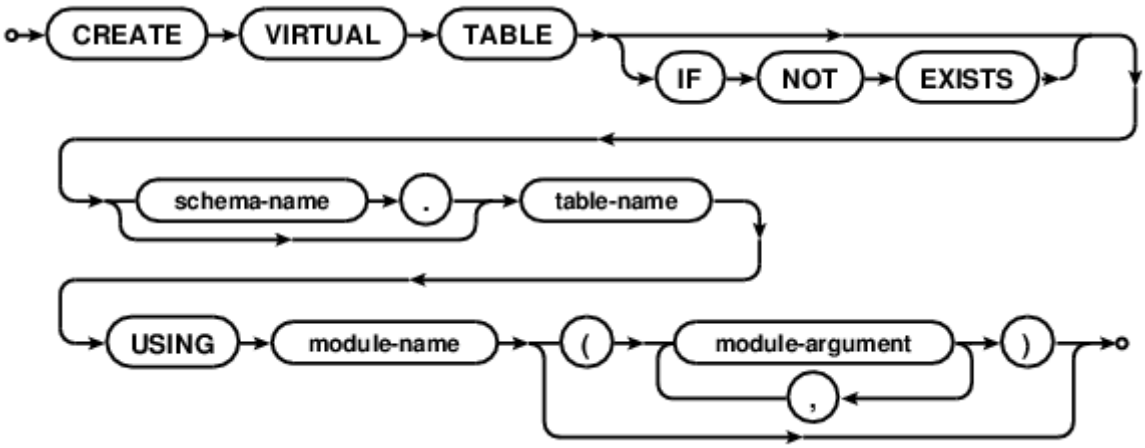
Οπότε η τελική σύνδεση-σχέση μεταξύ του κόμβου 1 και 5 είναι w=1.3 .

2.8 Εικονικοί πίνακες στην SQLite

Ένας εικονικός πίνακας είναι ένα αντικείμενο, το οποίο είναι καταχωρημένο με μια ανοιχτή σύνδεση σε SQLite βάση δεδομένων. Από την προοπτική μιας SQL κατάστασης, οι εικονικοί πίνακες μοιάζουν με οποιοδήποτε πίνακα ή όψη. Αλλά στο παρασκήνιο, ερωτήματα και ανανεώσεις σε έναν εικονικό πίνακα καλούν μεθόδους (callback methods) του αντικειμένου του εικονικού πίνακα, αντί να διαβάζουν και να γράφουν σε ένα αρχείο βάσης δεδομένων.

Ο μηχανισμός των εικονικών πινάκων επιτρέπει σε μια εφαρμογή να δημοσιεύσει διεπαφές, οι οποίες είναι προσβάσιμες από SQL καταστάσεις σαν να ήταν πίνακες. Μια SQL κατάσταση μπορεί να κάνει σχεδόν τα πάντα σε έναν εικονικό πίνακα, με αυτά που θα μπορούσε να κάνει σε έναν πραγματικό πίνακα.

Για να δημιουργηθεί ένας εικονικός πίνακας χρησιμοποιείται η CREATE VIRTUAL TABLE κατάσταση, η οποία φαίνεται στην Εικόνα 13.



Εικόνα 14: Create virtual table statement

3. ΒΑΣΙΚΕΣ ΑΡΧΕΣ ΥΛΟΠΟΙΗΣΗΣ

Σε αυτό το κεφάλαιο δίνονται αρχικά οι προδιαγραφές του συστήματος, καθώς και μια γενική εικόνα της αρχιτεκτονικής αυτού.

Στη συνέχεια, περιγράφεται ο τρόπος υλοποίησης του συστήματος. Αναλυτικότερα, γίνεται αναφορά στα εργαλεία πάνω στα οποία βασίζεται το σύστημα, στις κύριες δομές που χρησιμοποιεί και στους βασικούς αλγόριθμους που εκτελεί.

Τέλος, περιγράφεται αναλυτικά το συντακτικό της γλώσσας, δηλαδή ο τρόπος με τον οποίο καλείται ο χρήστης να δημιουργήσει έναν εικονικό πίνακα δίνοντας τις κατάλληλες παραμέτρους.

3.1 Προδιαγραφές του συστήματος

Όπως προαναφέρθηκε, ο τομέας των κοινωνικών δικτύων περιέχει μεγάλο όγκο πληροφορίας που διαρκώς αυξάνεται. Γίνονται πολλές προσπάθειες για την όλο και καλύτερη επεξεργασία αυτής της πληροφορίας με σκοπό να είναι διαχειρίσιμη. Η ανάγκη που στοχεύουν να καλύψουν οι προσπάθειες αυτές είναι η ανάπτυξη εύχρηστων εργαλείων για την ανάλυση της παραπάνω πληροφορίας.

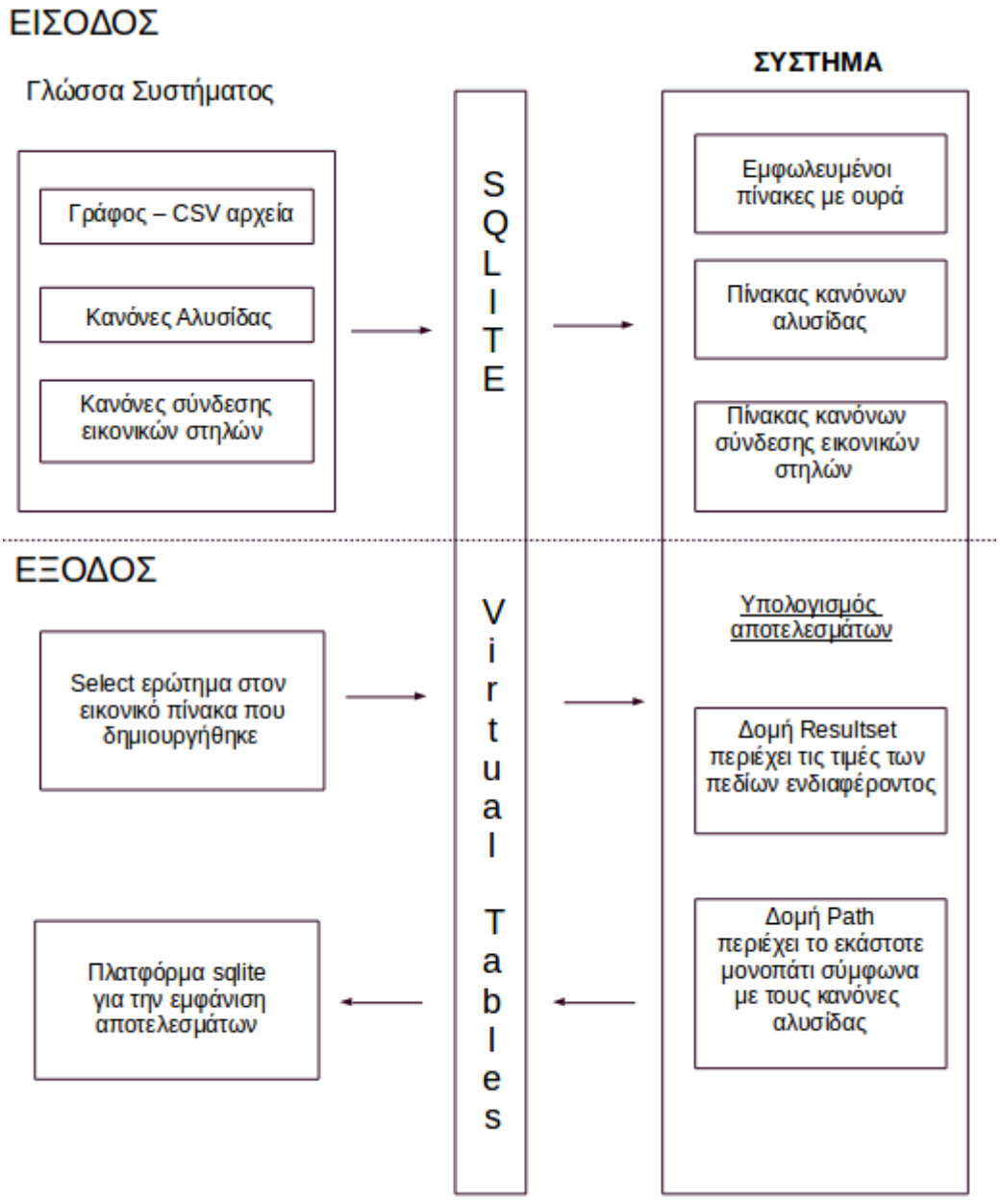
Προς αυτήν την κατεύθυνση και συνδυάζοντας ιδέες από υπάρχουσες τεχνολογίες και συστήματα, κατασκευάστηκε για την εργασία αυτή το παρόν σύστημα, έχοντας ως κίνητρο και βασική ιδέα τον τομέα των συστάσεων μεταξύ ατόμων σε κοινωνικά δίκτυα. Η ιδέα αυτή επεκτάθηκε ώστε το σύστημα να είναι γενικότερης μορφής και να αφήνει την ελευθερία στον χρήστη να δίνει τους δικούς του κανόνες σε μορφή αλυσίδας μαζί με την ανάλογη είσοδο δεδομένων, με σκοπό να δημιουργήσει τον δικό του γράφο ιδιοτήτων. Αυτό πραγματοποιείται γράφοντας στη γλώσσα που δημιουργήθηκε για τον παρόν εργαλείο και αλληλεπιδρώντας με την γλώσσα της SQL. Στη συνέχεια, ο χρήστης κάνοντας απλά ή σύνθετα ερωτήματα στην SQL, μπορεί να πάρει τα επιθυμητά αποτελέσματα.

Πιο αναλυτικά, το σύστημα αυτό δέχεται σαν είσοδο CSV (Column Separated Values) αρχεία που περιέχουν πληροφορία πάνω σε κοινωνικά δίκτυα, καθώς και τον κανόνα αλυσίδας που τα συνδέει. Εφαρμόζοντας τους αλγόριθμους που θα αναφερθούν παρακάτω, εξάγονται τα αποτελέσματα. Με την χρήση των εικονικών πινάκων της sqlite, ο χρήστης μπορεί να αξιοποιήσει το σύστημα αυτό και να διατυπώσει τα δικά του ερωτήματα.

Τελικός στόχος είναι το σύστημα να είναι όσο το δυνατό πιο εύχρηστο στο μέσο χρήστη, ώστε να μην χρειάζεται μεγάλο χρόνο εκμάθησης, να συνδυάζει καλή απόδοση για όσον το δυνατό μεγαλύτερα δεδομένα και να έχει δυνατότητες επέκτασης στο μέλλον, τόσο στο να μπορούν να προστεθούν νέες δυνατότητες, όσο και στο να βελτιωθούν υπάρχουσες.

3.2 Αρχιτεκτονική του συστήματος

Στην ενότητα αυτή δίνετε μια περιγραφή της αρχιτεκτονικής του συστήματος. Το σύστημα μπορεί να διαχωριστεί σε κάποιες βασικές λειτουργικές μονάδες όπως φαίνεται στο παρακάτω σχήμα:



Εικόνα 15: Σχήμα αρχιτεκτονικής του συστήματος

3.3 Υλοποίηση του συστήματος

Στην ενότητα αυτή περιγράφεται πιο αναλυτικά ο τρόπος που υλοποιήθηκε το σύστημα. Παρουσιάζονται αρχικά τα εργαλεία που χρησιμοποιήθηκαν και στη συνέχεια οι δομές δεδομένων, καθώς και οι βασικοί αλγόριθμοι. Με αυτόν τον τρόπο, μπορεί να φανεί πως οι παραπάνω θεωρητικές ιδέες και ορισμοί υλοποιήθηκαν ώστε το αποτέλεσμα να είναι αυτό που παρουσιάζεται στην Εικόνα 15.

3.3.1 Εργαλεία

Με την προϋπόθεση το σύστημα να είναι όσο πιο εύχρηστο και απλό για τον χρήστη, αναζητήθηκαν εργαλεία, τα οποία θα εξυπηρετούν αυτόν το σκοπό.

Το κύριο εργαλείο, το οποίο χρησιμοποιήθηκε είναι η SQLite. Αυτό συνέβη για δύο βασικούς λόγος. Ο πρώτος είναι διότι η SQLite είναι ένα σύστημα που υλοποιεί μια μηχανή βάσης δεδομένων SQL, η οποία είναι μια γλώσσα εκφραστική, αρκετά διαδεδομένη και σχετικά εύκολη στην εκμάθησή της. Ο δεύτερος λόγος είναι διότι περιέχει τους εικονικούς πίνακες που αναφέρθηκαν παραπάνω. Με βάση τα παραπάνω, η SQL επιλέχθηκε ως η γλώσσα αλληλεπίδρασης (είσοδος-έξοδος) με το σύστημα, μαζί με τη γλώσσα του συστήματος, η οποία θα περιγραφεί παρακάτω και αυτό πραγματοποιείται μέσω των εικονικών πινάκων. Με τους εικονικούς πίνακες ο χρήστης διαχειρίζεται το σύστημα σαν να ήταν σχεσιακοί πίνακες της βάσης, καθώς όλες οι λειτουργίες του έχουν γίνει αντιστοίχιση με τους πίνακες. Οπότε με ερωτήματα σε SQL, επιστρέφονται τα επιθυμητά αποτελέσματα.

Για την υλοποίηση των δομών δεδομένων, των λειτουργιών, αλλά και γενικά του συστήματος χρησιμοποιήθηκε η γλώσσα προγραμματισμού C, λόγω κυρίως της δυνατότητας διαχείρισης μνήμης και αυτομάτως της απόδοσης που προσφέρει. Για τη σύνδεση με τους εικονικούς πίνακες χρησιμοποιήθηκε η βιβλιοθήκη της C “sqlite3ext.h” [9], η οποία προσφέρει μια πολύ εύχρηστη διεπαφή της SQLite σε C.

Κατά την επιλογή, της γλώσσας προγραμματισμού, υπήρξε το δίλημμα να χρησιμοποιηθεί η C με την βιβλιοθήκη που αναφέρθηκε παραπάνω ή η Python με τη βιβλιοθήκη APSW [10], για την διεπαφή με την SQLite. Η Python είναι μια πολύ δημοφιλής διερμηνευμένη γλώσσα υψηλού επιπέδου, η οποία χρησιμοποιείται συνήθως για τη γρήγορη ανάπτυξη μικρών προγραμμάτων, σε περιπτώσεις που ο χρόνος του προγραμματιστή είναι πιο πολύτιμος από την ταχύτητα εκτέλεσης του προγράμματος. Η C είναι πιο γρήγορη, λόγω της διαχείρησης μνήμης αλλά και του σχεδιασμού της που περιλαμβάνει δομές που μεταφράζονται αποδοτικά σε τυπικές εντολές μηχανής (machine instructions) και εξ αιτίας αυτού χρησιμοποιείται συχνά σε εφαρμογές που παλιότερα γράφονταν σε συμβολική γλώσσα (assembly language). Αυτό ακριβώς το χαρακτηριστικό της, που έχει σαν συνέπεια και την αυξημένη ταχύτητα εκτέλεσης των εφαρμογών που γράφονται σε αυτή, είναι που την διαχωρίζει από άλλες γλώσσες προγραμματισμού. Σύμφωνα με μετρήσεις [11], ο χρόνος εκτέλεσης προγραμμάτων σε C είναι περίπου 2 με 10 φορές πιο γρήγορος από τον χρόνο εκτέλεσης σε Python.

3.3.2 Βασικές δομές δεδομένων

Η παρακάτω υποενότητα εστιάζεται στις βασικές δομές δεδομένων με βάση τις οποίες υλοποιήθηκε το σύστημα. Γίνεται αναφορά τόσο στη μορφή τους όσο και στο ρόλο που κατέχουν στη λειτουργία του συστήματος.

3.3.2.1 Tailnested

Η δομή Tailnested αντιπροσωπεύει έναν εμφωλευμένο πίνακα με ουρά και χρησιμοποιείται για την αναπαράσταση κανόνων μιας αλυσίδας ή σχέσεων μεταξύ αρχικών και τελικών κόμβων ενός γράφου στη μνήμη. Κάθε επίπεδο είναι στην ουσία ένας πίνακας γραμμικού κατακερματισμού (linear hashing) [12], με την επέκταση ότι περιέχει έναν δείκτη, έτσι ώστε να συνδέεται με το επόμενο επίπεδο.

3.3.2.2 Table_node

Η δομή Table_node χρησιμοποιείται για την δημιουργία ενός πίνακα tables, ο οποίος περιέχει τους εμφωλευμένους πίνακες με ουρά στην σειρά που βρίσκονται στην αλυσίδα. Στον πίνακα tables, υπάρχει συγκεντρωμένη η πληροφορία σχετικά με τους εμφωλευμένους πίνακες.

```
Table_node
    Tailnested *tailnested
```

Εικόνα 16: Δομή Table_node

3.3.2.3 Chain_node

Με την χρήση της παρακάτω δομής δημιουργείται ο πίνακας chain, που περιέχει τους επιμέρους κρίκους της αλυσίδας. Συγκεκριμένα, περιέχει δείκτες στον πίνακα tables, οι οποίοι αναφέρονται στους εμφωλευμένους πίνακες με ουρά όπου γίνεται η σύνδεση, καθώς και στις μεταβλητές τους.

```
Chain_node
    int table_index_1
    int variable_position_1
    int table_index_2
    int variable_position_2
```

Εικόνα 17: Δομή Chain_node

3.3.2.4 Path_node

Για τον υπολογισμό των αποτελεσμάτων κατασκευάζεται ένα μονοπάτι (path), που διασχίζει τους εμφωλευμένους πίνακες με ουρά σύμφωνα με την αλυσίδα. Ειδικότερα, περιέχει κάθε επιμέρους επίπεδο εμφωλευμένου πίνακα που συμμετέχει στην αλυσίδα, μέχρι και το επίπεδο ενδιαφέροντος.

```
Path_node
    int chain_index
    int table_index_of_chain
    int type
    void *pointer
    Path_node *next
```

Εικόνα 18: Δομή Path_node

Κάθε κόμβος του μονοπατιού περιέχει την εξής πληροφορία:

chain_index: δείκτης σε κάποιο κρίκο της αλυσίδας

table_index_of_chain: ο πίνακας της αλυσίδας στον οποίο αναφέρεται ο συγκεκριμένος κόμβος του μονοπατιού

type: ο τύπος της μεταβλητής pointer. Σε περίπτωση που η μεταβλητή type έχει την τιμή 1, τότε η μεταβλητή pointer δείχνει σε εμφωλευμένο πίνακα. Ενώ αν η μεταβλητή type έχει την τιμή 2, τότε η μεταβλητή pointer δείχνει σε εγγραφή κάποιου πίνακα.

pointer: αποτελεί δείκτη σε επιμέρους επίπεδο εμφωλευμένου πίνακα, ειδάλλως σε κάποια εγγραφή του πίνακα.

next: δείκτης σε επόμενο κόμβο του μονοπατιού.

3.3.2.5 Resultset_node

Η παρακάτω δομή χρησιμοποιείται για την δημιουργία του πίνακα resultset, ο οποίος περιέχει δείκτες σε στήλες εμφωλευμένων πινάκων καθώς και τις τιμές τους. Οι παραπάνω στήλες αποτελούν μέρος του αποτελέσματος και η δομή αυτή περιέχει συγκεντρωμένη την πληροφορία για την διευκόλυνση του υπολογισμού αλλά και της εμφάνισης του αποτελέσματος.

```
Resultset_node
    int table_index
    int variable_position
    int type
    void *value
```

Εικόνα 19: Δομή Resultset_node

Επιπρόσθετα, οι τύποι μεταβλητών που υποστηρίζονται είναι ακέραιος, συμβολοσειρά και δεκαδικός και προσδιορίζονται ανάλογα με την τιμή της μεταβλητής *type* 1,2,3 αντίστοιχα .

3.3.3 Βασικοί αλγόριθμοι

Η παρακάτω υποενότητα εστιάζεται στις βασικές δομές δεδομένων με βάση τις οποίες υλοποιήθηκε το σύστημα. Γίνεται αναφορά τόσο στη μορφή τους όσο και στο ρόλο που κατέχουν στη λειτουργία του συστήματος.

3.3.3.1 Bfs_graph

Στο υποκεφάλαιο 2.7 αναλύθηκε ο τρόπος διάσχισης και υπολογισμού πράξεων πάνω σε γράφο. Αυτός ο τρόπος χρησιμοποιήθηκε στο σύστημα ώστε να εξαχθούν σχέσεις μεταξύ αρχικών και τελικών κόμβων, με την ύπαρξη ενδιάμεσων κόμβων και ακμών που τους συνδέει. Παρακάτω παρατείθεται ο ψευδοκώδικας που περιγράφει την διαδικασία.

```
bfs_graph ()
    Αρχικοποίησε V με αρχικό κόμβο
    Αρχικοποίησε ουρά Q με αρχικό κόμβο
    Όσο Q δεν είναι άδεια
        Πάρε x από την κεφαλή της Q
        Για κάθε γείτονα y του x
            Υπολόγισε την νέα τιμή του y με expand
            Αν y υπάρχει στη V
                concat(παλιά τιμή του y, νέα τιμή του y)
                Ανανέωσε την τιμή του y στη V
                Αν η παλιά τιμή του y είναι διαφορετική από την τιμή μετά το concat
                    Βάλε το y στην Q
                Τέλος_Αν
            Αλλιώς
                Βάλε το y στο V
                Βάλε το y στην Q
            Τέλος_Αν
        Τέλος_Επανάληψης
    Τέλος_Επανάληψης
Τέλος_Διαδικασίας
```

Εικόνα 20: Ψευδοκώδικας bfs_graph

Στο σύστημα, ο αλγόριθμος έχει παραμετρικοποιηθεί, ώστε να τερματίζει σε ορισμένο βάθος. Αυτό πραγματοποιήθηκε διότι ανάλογα τις αθροιστικές συναρτήσεις που επιλέγονται και τον τύπο του γράφου, ο αλγόριθμος μπορεί να μην συγκλίνει ποτέ σε λύση ή να κάνει τους ίδιους κύκλους συνεχώς.

3.3.3.2 Compute_Resultset

Όπως αναφέρθηκε παραπάνω, ο πίνακας resultset περιέχει τις μεταβλητές των εμφωλευμένων πινάκων που συμμετέχουν στο τελικό αποτέλεσμα. Ο παρακάτω αλγόριθμος χρησιμοποιείται για την ανάθεση καθώς και την ανανέωση τιμών των μεταβλητών αυτών.

```
Compute_Resultset(Path_node *path, Resultset_node *resultset, int resultset_pointer)
    Όσο το path δείχνει στον ίδιο πίνακα με το resultset[resultset_pointer]
        Ανανέωσε την τιμή του resultset[resultset_pointer]
        Αύξησε το resultset_pointer
    Τέλος_Επανάληψης
Τέλος_Διαδικασίας
```

Εικόνα 21: Ψευδοκώδικας Compute_Resultset

Αναλυτικότερα, ο αλγόριθμος δέχεται σαν είσοδο ένα μονοπάτι, τον πίνακα resultset και έναν δείκτη στον πίνακα, ο οποίος προσδιορίζει την επόμενη μεταβλητή που θα πρέπει να ανανεωθεί. Σύμφωνα με την είσοδο και συγκεκριμένα μόνο με τον πρώτο κόμβο του μονοπατιού, ανανεώνονται όλες οι τιμές των μεταβλητών, για τις οποίες ισχύει η σχέση που ορίζεται στον ψευδοκώδικα.

3.3.3.3 Tailnested_dfs

Στη συνέχεια παρουσιάζεται ο αλγόριθμος για την εύρεση του επόμενου δυνατού μονοπατιού, σύμφωνα με την αλυσίδα που έχει οριστεί από τον χρήστη. Όπως είναι εμφανές και από τον ψευδοκώδικα, ο αλγόριθμος καλεί τον εαυτό του, που τον εντάσσει στην κατηγορία των αναδρομικών αλγόριθμων. Το γεγονός αυτό απλουστεύει τον αλγόριθμο, καθώς το ίδιο τμήμα του ψευδοκώδικα υπολογίζει τόσο ολόκληρο το μονοπάτι, όσο και τα επιμέρους τμήματά του .

```
tailnested_dfs (Path_node *path, Resultset_node *resultset, int resultset_pointer)
    Αν το path είναι κενό
        Απέτυχε
    Τέλος_Αν
    Αν path->pointer δείχνει σε πίνακα
        Βρες την επόμενη εγγραφή
        Αν δεν υπάρχει επόμενη εγγραφή
            tailnested_dfs(path->next, resultset, resultset_pointer)
        Τέλος_Αν
        compute_resultset(path, resultset, resultset_pointer)
    Αλλιώς_Αν path->pointer δείχνει σε εγγραφή
        Αρχή_Επανάληψης
            tailnested_dfs(path->next, resultset, resultset_pointer)
            Ψάξε για την επόμενη εγγραφή που ικανοποιεί την αλυσίδα
        Μέχρις_Ότου να βρεθεί εγγραφή ή να αποτύχει η tailnested_dfs
        compute_resultset(path, resultset, resultset_pointer)
    Τέλος_Αν
Τέλος_Διαδικασίας
```

Εικόνα 22: Ψευδοκώδικας tailnested_dfs

Αναλυτικότερα, ο αλγόριθμος παίρνει ως παραμέτρους ένα ήδη υπάρχων μονοπάτι, τον πίνακα resultset, που αναφέρθηκε στην υποενότητα 3.3.2.5, καθώς και έναν δείκτη, τον resultset_pointer, που προσδιορίζει ποια μεταβλητή του πίνακα θα ανανεωθεί. Αρχικά, ο αλγόριθμος ελέγχει αν το δεδομένο μονοπάτι είναι κενό, γεγονός που οδηγεί στην αποτυχία του, διότι δεν υπάρχει επόμενο δυνατό μονοπάτι, που να ικανοποιεί την αλυσίδα. Ειδάλλως, διακρίνονται δύο βασικές περιπτώσεις, σύμφωνα με τον δείκτη του εκάστοτε κόμβου του μονοπατιού. Δεδομένου ότι ο δείκτης δείχνει σε πίνακα, γίνεται μία αναζήτηση για την επόμενη εγγραφή του πίνακα. Σε περίπτωση αποτυχίας, καλείται ο ίδιος αλγόριθμος με τον επόμενο κόμβο για την επανεξέταση του υπομονοπατιού. Διαφορετικά, αν ο δείκτης του εκάστοτε κόμβου του μονοπατιού δείχνει σε εγγραφή, καλείται ο αλγόριθμος με το υπομονοπάτι μέχρις ότου να βρεθεί εγγραφή που να ικανοποιεί την αλυσίδα. Τόσο στην πρώτη, όσο και στην δεύτερη περίπτωση, καλείται ο αλγόριθμος compute_resultset, για την ανάθεση κατάλληλων τιμών στον πίνακα resultset. Μόλις ολοκληρωθεί η παραπάνω διαδικασία, υπάρχει διαθέσιμο το επόμενο μονοπάτι που οδηγεί στην επόμενη λύση.

3.3.3.4 Compute

Παραπάνω αναλύθηκε ο αλγόριθμος tailnested_dfs για τον υπαναπροσδιορισμό ενός υπάρχων μονοπατιού. Ακολούθως θα αναλυθεί ο αλγόριθμος για την δημιουργία του αρχικού μονοπατιού.

```
compute (Path_node *path, Chain_node *chain, Resultset_node *resultset, int resultset_pointer)
    Για chain_pointer από 0 μέχρι (μέγεθος του πίνακα chain)-1
        Αρχή_Επανάληψης
            Υπολόγισε τον επόμενο κόμβο του μονοπατιού και πρόσθεσέ τον στο path
            compute_resultset(path, resultset, resultset_pointer)
        Μέχρις_Ότου να φτάσει στο chain[chain_pointer].variable_position_1
        Αρχή_Επανάληψης
            Υπολόγισε τον επόμενο κόμβο του μονοπατιού και πρόσθεσέ τον στο path
            compute_resultset(path, resultset, resultset_pointer)
        Μέχρις_Ότου να φτάσει ένα επίπεδο πριν το chain[chain_pointer].variable_position_2
        Αν η τιμή του chain[chain_pointer].variable_position_1 δεν υπάρχει στο chain[chain_pointer].table_index_1
            tailnested_dfs(path, resultset, resultset_pointer)
        Αλλιώς
            Υπολόγισε τον επόμενο κόμβο του μονοπατιού και πρόσθεσέ τον στο path
        Τέλος_Αν
    Τέλος_Επανάληψης
Τέλος_Διαδικασίας
```

Εικόνα 23: Ψευδοκώδικας compute

Όπως παρατηρείται και από τον ψευδοκώδικα, η δημιουργία του μονοπατιού έχει άμεση σχέση με την αλυσίδα που έχει οριστεί από τον χρήστη. Αρχικά, γίνεται μία επανάληψη για την προσπέλαση κάθε κρίκου της αλυσίδας διαδοχικά. Για κάθε κρίκο της αλυσίδας, υπολογίζεται το μονοπάτι μέχρι την μεταβλητή του πρώτου πίνακα, καθώς και οι τιμές των αντίστοιχων μεταβλητών στον πίνακα *resultset*. Στη συνέχεια, υπολογίζεται το μονοπάτι που αναφέρεται στον δεύτερο πίνακα του εκάστοτε κρίκου της αλυσίδας και φτάνει ένα επίπεδο πριν την μεταβλητή αυτού. Διακρίνεται ειδική περίπτωση για τον κόμβο του μονοπατιού που προσδιορίζει την μεταβλητή του δεύτερου πίνακα. Το παραπάνω ισχύει, διότι θα πρέπει να εφαρμοστεί μία αναζήτηση στο επίπεδο του δεύτερου πίνακα που περιέχει την μεταβλητή, που συμμετέχει στην αλυσίδα, σύμφωνα με την τιμή της μεταβλητής του πρώτου πίνακα. Σε περίπτωση αποτυχίας της αναζήτησης, καλείται ο αλγόριθμος tailnested_dfs για τον επαναπροσδιορισμό του μονοπατιού που έχει δημιουργηθεί μέχρι εκείνο το σημείο. Μετά την ολοκλήρωση του αλγόριθμου compute έχει δημιουργηθεί το πρώτο μονοπάτι, που οδηγεί σε λύση, καθώς έχουν ανατεθεί και οι κατάλληλες τιμές στον πίνακα resultset, μέσω του αλγόριθμου *compute_resultset* που καλείται στα κατάλληλα σημεία.

3.4 Συντακτικό της γλώσσας του συστήματος

Στην ενότητα αυτή περιγράφεται αναλυτικά το συντακτικό της γλώσσας που αναπτύχθηκε για το παρόν σύστημα σε συνδυασμό με τους εικονικούς πίνακες της SQLite. Όπως αναφέρθηκε και παραπάνω, αυτό πραγματοποιείται με την *CREATE VIRTUAL TABLE* κατάσταση και κατάλληλα ορίσματα τα οποία αποτελούν τη γλώσσα και περιγράφονται στις παρακάτω υποενότητες.

3.4.1 Virtual tables columns

Το πρώτο όρισμα που παρουσιάζεται είναι αυτό το οποίο διευκρινίζει στο σύστημα τον αριθμό και τα ονόματα των στηλών του εικονικού πίνακα.

```
Number_of_columns|(vtab1,vtab2,...,vtabNumber_of_columns)
```

Εικόνα 24: Ορισμός virtual table columns

3.4.2 Connection between virtual table columns and tail-nested tables

Το δεύτερο όρισμα αντιστοιχεί κάθε στήλη του εικονικού πίνακα είτε με μια στήλη κάποιου εμφωλευμένου πίνακα με ουρά είτε με μια αλληλουχία πράξεων μεταξύ στηλών εμφωλευμένων πινάκων.

```
Number_of_distinct_parameters|vtab1 = tablename1.p1°...°tablenameN.pN
                             |...
                             |vtabNumber_of_columns = tablename1.p1
                                                     °...°
                                                     tablenameN.pN

    °=+,-,*,/
```

Εικόνα 25: Ορισμός connection between virtual table columns and tail-nested tables

Πιο αναλυτικά:

Number_of_distinct_parameters: ορίζει τον αριθμό των διακριτών στηλών εμφωλευμένων πινάκων με ουρά που εμφανίζονται παρακάτω.

vtab1…vtabNumber_of_columns: ορίζει την εκάστοτε στήλη του εικονικού πίνακα της ενότητας *3.4.1*.

tablename1.p1°...°tablenameN.pN: ορίζει την αλληλουχία πράξεων μεταξύ στηλών εμφωλευμένων πινάκων με ουρά. Εφόσον δεν υπάρχει πράξη, τότε γίνεται απλή ανάθεση.

3.4.3 Tail-nested structure

Στο τρίτο όρισμα ο χρήστης δίνει την πληροφορία για τον αριθμό αλλά και τα χαρακτηριστικά των εμφωλευμένων πινάκων με ουρά.

```
Number_of_parameters.Number_of_graphs | Tail-nested_from_graph OR
                                        Tail-nested_from_csv
                                      | ...
                                      | Tail-nested_from_graph OR
                                        Tail-nested_from_csv
```

Εικόνα 26: Ορισμός Tail-nested structure

Number_of_parameters: ορίζει τον αριθμό εμφωλευμένων πινάκων με ουρά που εμφανίζονται παρακάτω.

Number_of_graphs: ορίζει τον αριθμό των εμφωλευμένων πινάκων, οι οποίοι προέρχονται από γράφους.

3.4.3.1 Tail-nested_from_graph

Οι εμφωλευμένοι πίνακες με ουρά όταν προέρχονται από γράφο έχουν συγκεκριμένη μορφή, ώστε να αρχικοποιηθούν οι κατάλληλες δομές δεδομένων και στη συνέχεια να υπάρξει ως αποτέλεσμα ολοκληρωμένες σχέσεις μεταξύ αρχικών και τελικών κόμβων του γράφου.

```
( g,
  path_for_nodes.csv,
  path_for_edges.csv,
  tablename,
  number_of_aggregations*[
                          {parK.symbol_for_expand.symbom_for_concat}
                          &...&
                          {parK.symbol_for_expand.symbom_for_concat}
                          ],
  number_of_nesting*[!par1,par2,...,!parM,...,parK,...,parnumber_of_nesting]
)
```

Εικόνα 27: Ορισμός Tail-nested_from_graph

g: ορίζει ότι η συγκεκριμένη παράμετρος αναφέρεται σε γράφο.

path_for_nodes.csv: ορίζει το σχετικό ή απόλυτο μονοπάτι, στο οποίο βρίσκεται το αρχείο με την πληροφορία για του κόμβους του γράφου σε μορφή csv.

path_for_edges.csv: ορίζει το σχετικό ή απόλυτο μονοπάτι, στο οποίο βρίσκεται το αρχείο με την πληροφορία για τις ακμές του γράφου σε μορφή csv.

tablename: ορίζει το όνομα του εκάστοτε εμφωλευμένου πίνακα με ουρά, το οποίο θα δημιουργηθεί.

number_of_aggregations: ορίζει τον αριθμό των παραμέτρων-ιδιοτήτων που θα πραγματοποιηθούν αθροιστικές συναρτήσεις.

{parK.symbol_for_expand.symbom_for_concat}: ορίζει το όνομα της παραμέτρου, καθώς και το ποιες αθροιστικές συναρτήσεις θα εφαρμοστούν σε αυτή για επέκταση και συγχώνευση.

number_of_nesting: ορίζει τον αριθμό των εμφωλιασμών, δηλαδή πόσα επίπεδα θα έχει ο πίνακας.

[!par1,par2,...,!parM,...,parK,...,parnumber_of_nesting]: ορίζει τις παραμέτρους-στήλες του πίνακα και τα ονόματά τους. Στις παραμέτρους που υπάρχει το σύμβολο του θαυμαστικού πριν από αυτές, υποδηλώνεται αρχή επιπέδου εμφωλιασμού και η συγκεκριμένη παράμετρος είναι αναγνωριστικό αυτού του επιπέδου.

3.4.3.2 Tail-nested_from_csv

Όταν οι σχέσεις μεταξύ αρχικού και τελικού κόμβου δίνονται απευθείας σε μορφή αρχείου csv, χωρίς δηλαδή την ύπαρξη ενδιάμεσων κόμβων τότε η μορφή εισόδου είναι η εξής.

```
( c,
  path_file.csv,
  tablename,
  number_of_nesting*[!par1,par2,...,!parM,...,parK,...,parnumber_of_nesting]
)
```

Εικόνα 28: Ορισμός Tail-nested_from_csv

c: ορίζει ότι η συγκεκριμένη παράμετρος αναφέρεται στην περίπτωση που αναφερόμαστε.

path_file.csv: ορίζει το σχετικό ή απόλυτο μονοπάτι, στο οποίο βρίσκεται το αρχείο με την πληροφορία για τις σχέσεις μεταξύ δύο κόμβων σε μορφή csv.

tablename: ορίζει το όνομα του εκάστοτε εμφωλευμένου πίνακα με ουρά, το οποίο θα δημιουργηθεί.

number_of_nesting: ορίζει τον αριθμό των εμφωλιασμών, δηλαδή πόσα επίπεδα θα έχει ο πίνακας.

[!par1,par2,...,!parM,...,parK,...,parnumber_of_nesting]: ορίζει τις παραμέτρους-στήλες του πίνακα και τα ονόματά τους. Στις παραμέτρους που υπάρχει το σύμβολο του θαυμαστικού πριν από αυτές, υποδηλώνεται αρχή επιπέδου εμφωλιασμού και η συγκεκριμένη παράμετρος είναι αναγνωριστικό αυτού του επιπέδου.

3.4.4 Chain connections

Στο τελευταίο όρισμα ο χρήστης δίνει την πληροφορία για το ποιες παράμετροι-στήλες συνδέονται ανά δύο εμφωλευμένους πίνακες με ουρά, οι οποίοι αντιπροσωπεύουν και από μια σχέση μεταξύ κόμβων, ώστε να δημιουργηθεί ο κανόνας αλυσίδας που περιγράφηκε παραπάνω.

```
Number_of_connections | tablenameK.c = tablenameM.d
                      | ...
                      | tablenameK.c = tablenameM.d
```

Εικόνα 29: Ορισμός Chain connections

number_of_connections: ορίζει τον αριθμό των συνδέσεων.

tablenameK.c = tablenameM.d: ορίζει ποιες παράμετροι-στήλες συνδέονται ανά δύο εμφωλευμένους πίνακες με ουρά. Πρέπει να είναι με συγκεκριμένη σειρά, όπως η αλυσίδα που ορίζεται.

3.4.5 Σύνοψη συντακτικού γλώσσας συστήματος

Η σύνοψη των παραπάνω ορισμάτων σε συνδυασμό με την CREATE VIRTUAL TABLE φαίνεται στην εικόνα που ακολουθεί:

```
CREATE VIRTUAL TABLE x USING my_vtable(
                Virtual tables columns,
                Connection between virtual table columns and tail-nested tables,
                Tail-nested structure,
                Chain connections
                )
```

Εικόνα 30: Σύνοψη συντακτικού γλώσσας συστήματος

4. ΠΕΙΡΑΜΑΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

Σε αυτό το κεφάλαιο αναφέρονται κάποιες ενδεικτικές μετρήσεις για την αξιολόγηση του συστήματος που αναπτύχθηκε στα πλαίσια της πτυχιακής εργασίας.

4.1 Προδιαγραφές περιβάλλοντος δοκιμών

Όλες οι μετρήσεις διεξήχθησαν σε σύστημα με τα παρακάτω χαρακτηριστικά:

Λειτουργικό Σύστημα: Ubuntu 14.04 – 64 bit

Επεξεργαστής: Intel® Core™ i5-3570K (4 πυρήνες στα 3.4 GHz)

Μνήμη: 16GB

4.2 Συλλογή και επεξεργασία δεδομένων από γράφους

Αρχικά, χρησιμοποιήθηκαν γράφοι, οι οποίοι συλλέχθηκαν από την βιβλιοθήκη SNAP [13]. Στη συνέχεια υπέστησαν κατάλληλη επεξεργασία για να είναι σε μορφή συμβατή με το σύστημα. Στον παρακάτω πίνακα παρουσιάζονται οι γράφοι, καθώς και ο αριθμός των κόμβων και των ακμών τους.

Όνομα γράφου	Αριθμός κόμβων	Αριθμός ακμών
ca_GrQc	5.242	28.980
wiki_Vote	7.115	103.689
ca_CondMat	23.133	186.944
citHepTh	27.770	352.807
soc_Epinions1	75.879	508.837
Amazon302	262.111	1.234.190
web_NotreDame	352.729	1.497.134

Πίνακας 1: Γράφοι SNAP και χαρακτηριστικά

Αναλυτικότερα, η βιβλιοθήκη SNAP παρέχει ένα αρχείο ανά γράφο, το οποίο περιέχει πληροφορία για τις ακμές του. Μετά από επεξεργασία αυτού του αρχείου, εξήχθησαν τρία νέα αρχεία. Συγκεκριμένα, το πρώτο περιέχει τους κόμβους του γράφου, το δεύτερο αρχείο τις ακμές του γράφου καθώς και τυχαία βάρη για την κάθε μία, περιορισμένα στο πεδίο [-1,1] και το τρίτο αρχείο αποτελείται από τυχαίες ακμές, προερχόμενες από το δεύτερο με τυχαία βάρη. Ο αριθμός των ακμών αυτών ισούται με το 20% του αριθμού των ακμών του δεύτερου αρχείου.

Το ερώτημα που εκτελέστηκε για την διεξαγωγή των αποτελεσμάτων είναι παρόμοιο με αυτό που αναφέρεται στην Εικόνα 1 και είναι το εξής:

Recommend(X, Z, W) :- trusts(X, Y, W1), likes(Y, Z, W2), W=(W1+W2)/2.

Trusts:

- Γράφοι από snap.stanford.edu/data
- Concatenate : min
- Aggregate: +

Likes:

- Σχέσεις μεταξύ αρχικών και τελικών κόμβων
- 20% του αριθμού των ακμών του γράφου trusts

4.3 Αποτελέσματα μετρήσεων

Οι μετρήσεις πραγματοποιήθηκαν για τους γράφους, οι οποίοι περιγράφηκαν παραπάνω έχοντας σαν παράμετρο, στο κομμάτι της διάσχισης του γράφου με αθροιστικές συναρτήσεις, ένα ορισμένο βάθος για τον τερματισμό. Συγκεκριμένα, επιλέχθηκαν να παρουσιαστούν οι μετρήσεις με βάθος ένα και τρία.

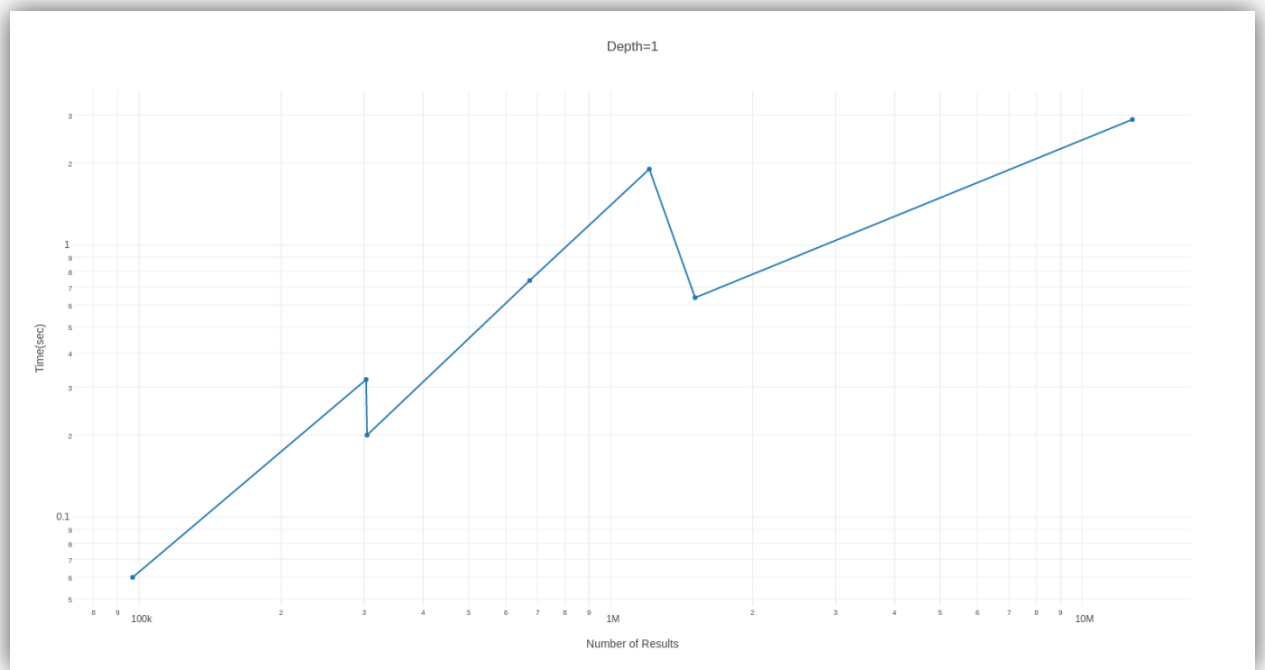
Στους παρακάτω πίνακες παρουσιάζονται οι γράφοι, ο αριθμός των αποτελεσμάτων που παρήγαγε το σύστημα καθώς και ο χρόνος εκτέλεσης σε δευτερόλεπτα ανά γράφο, ταξινομημένοι με βάση τον αριθμό αποτελεσμάτων, για βάθος ένα και τρία αντίστοιχα.

Μετά από κάθε πίνακα ακολουθεί το λογαριθμικό διάγραμμα, ώστε να υπάρξει πιο καθαρή εικόνα για την πορεία της καμπύλης.

Ενδιάμεσα από τους πίνακες και τα σχήματα παρεμβάλλονται παρατηρήσεις και σχόλια για την συμπεριφορά της απόδοσης του συστήματος.

Όνομα γράφου	Αριθμός αποτελεσμάτων	Χρόνος(s)
ca_GrQc	96.877	0,06
ca_CondMat	303.055	0,32
wiki_Vote	304.440	0,20
soc_Epinions1	673.698	0,74
Amazon302	1.208.190	1,90
citHepTh	1.510.786	0,64
web_NotreDame	12.783.472	2,89

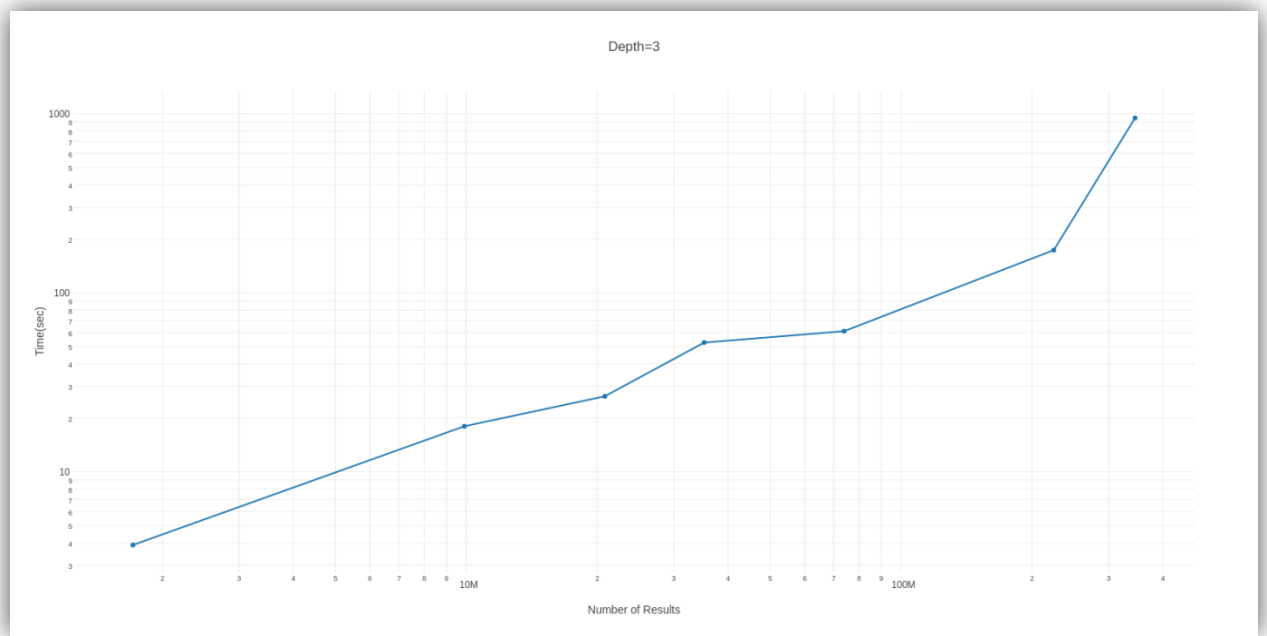
Πίνακας 2: Γράφοι SNAP, αριθμός αποτελεσμάτων, χρόνοι για βάθος 1



Σχήμα 1: Μετρήσεις γράφων SNAP για βάθος 1

Όνομα γράφου	Αριθμός αποτελεσμάτων	Χρόνος(s)
ca_GrQc	1.710.264	3,9
Amazon302	9.891.917	17,96
wiki_Vote	20.816.902	26,40
ca_CondMat	35.231.404	52,7
citHepTh	73.925.925	60,99
web_NotreDame	224.414.660	173,06
soc_Epinions1	345.178.789	946,77

Πίνακας 3: Γράφοι SNAP, αριθμός αποτελεσμάτων, χρόνοι για βάθος 3

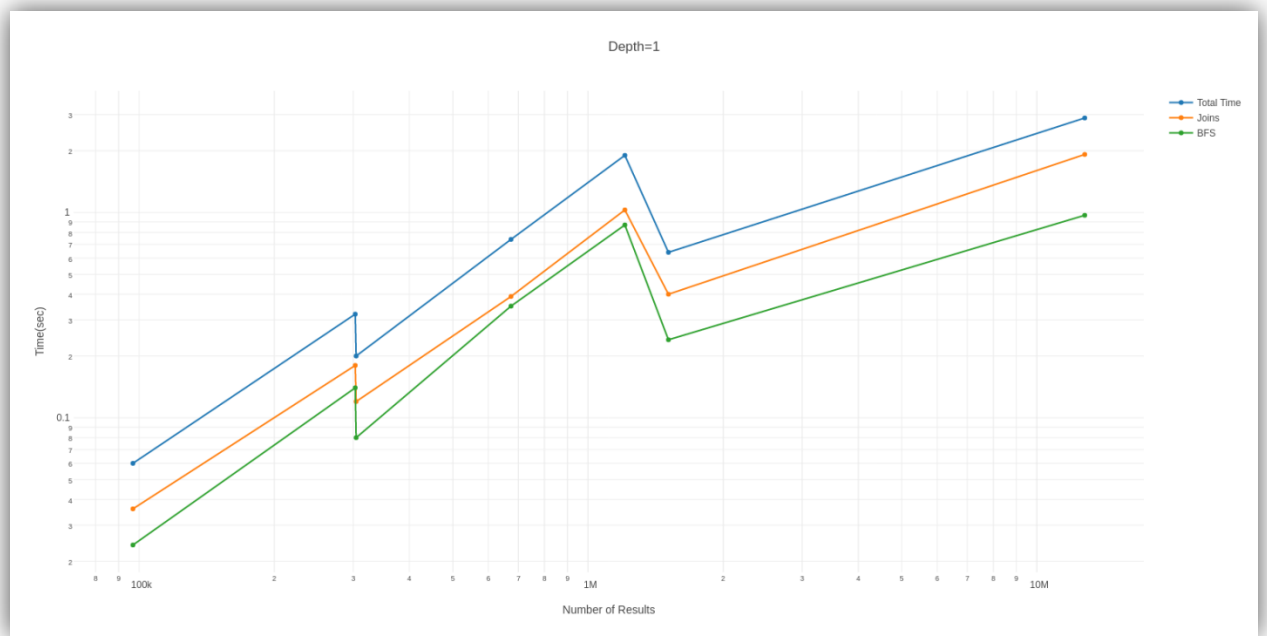


Σχήμα 2: Μετρήσεις γράφων SNAP για βάθος 3

Όπως παρατηρείται, η καμπύλη δεν ακολουθεί κάποια σταθερή πορεία, ώστε να μπορεί να χαρακτηριστεί με ασφάλεια στην εκτέλεση του συστήματος για βάθος ένα, ενώ για βάθος τρία, ξεκινάει γραμμικά και στην πορεία παίρνει εκθετική μορφή. Για αυτόν το λόγο, χρησιμοποιήθηκε η μέθοδος Profiling, για δυναμική ανάλυση του προγράμματος ώστε να γίνει μέτρηση του χρόνου των συναρτήσεων, οι οποίες είναι υπεύθυνες για την ένωση των εμφωλευμένων πινάκων και για την αναζήτηση κατά πλάτος με αθροιστικές συναρτήσεις. Πιο συγκεκριμένα, χρησιμοποιήθηκε το εργαλείο Callgrind για τον υπολογισμό των μετρήσεων και το εργαλείο KCachegrind [18] για την οπτικοποίηση των αποτελεσμάτων.

Όνομα γράφου	Αριθμός αποτελεσμάτων	Χρόνος ένωσης πινάκων(s)	Χρόνος BFS(s)	Τελικός Χρόνος(s)
ca_GrQc	96.877	0,036	0,024	0,06
ca_CondMat	303.055	0,18	0,14	0,32
wiki_Vote	304.440	0,12	0,08	0,20
soc_Epinions1	673.698	0,39	0,35	0,74
Amazon302	1.208.190	1,03	0,87	1,90
citHepTh	1.510.786	0,4	0,24	0,64
web_NotreDame	12.783.472	1,92	0,97	2,89

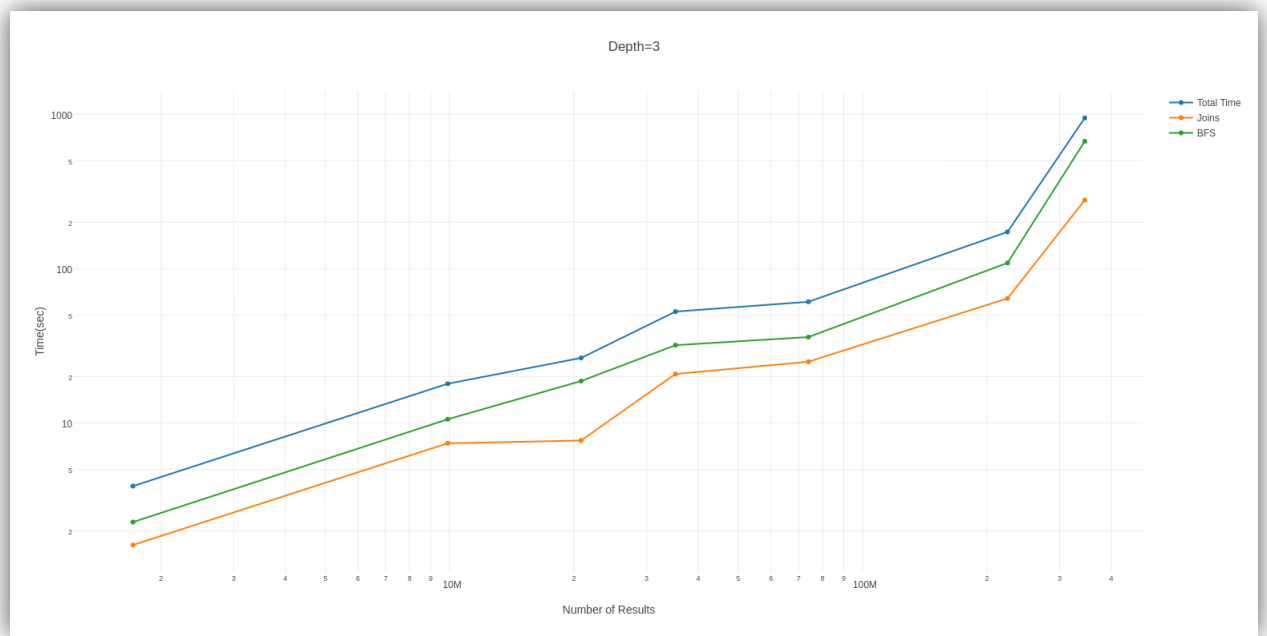
Πίνακας 4: Γράφοι SNAP, αριθμός αποτελεσμάτων, χρόνοι για ένωση πινάκων, bfs και τελικοί χρόνοι για βάθος 1



Σχήμα 3: Μετρήσεις γράφων SNAP (τελικοί χρόνοι, χρόνοι ένωσης πινάκων, χρόνοι bfs) για βάθος 1

Όνομα γράφου	Αριθμός αποτελεσμάτων	Χρόνος ένωσης πινάκων(s)	Χρόνος BFS(s)	Τελικός Χρόνος(s)
ca_GrQc	1.710.264	1,62	2,28	3,9
Amazon302	9.891.917	7,39	10,57	17,96
wiki_Vote	20.816.902	7,7	18,7	26,40
ca_CondMat	35.231.404	20,77	31,93	52,7
citHepTh	73.925.925	24,95	36,04	60,99
web_NotreDame	224.414.660	64,15	108,91	173,06
soc_Epinions1	345.178.789	278,19	668,58	946,77

Πίνακας 5: Γράφοι SNAP, αριθμός αποτελεσμάτων, χρόνοι για ένωση πινάκων, bfs και τελικοί χρόνοι για βάθος 3



Σχήμα 4: Μετρήσεις γράφων SNAP (τελικοί χρόνοι, χρόνοι ένωσης πινάκων, χρόνοι bfs) για βάθος 1

Από τα σχήματα 3 και 4, παρατηρείται αφενός ότι για βάθος ένα, η ένωση των πινάκων είναι πιο χρονοβόρα διαδικασία από ότι ο αλγόριθμος αναζήτησης κατά πλάτος με αθροιστικές συναρτήσεις, ενώ για βάθος τρία ισχύει το αντίθετο.

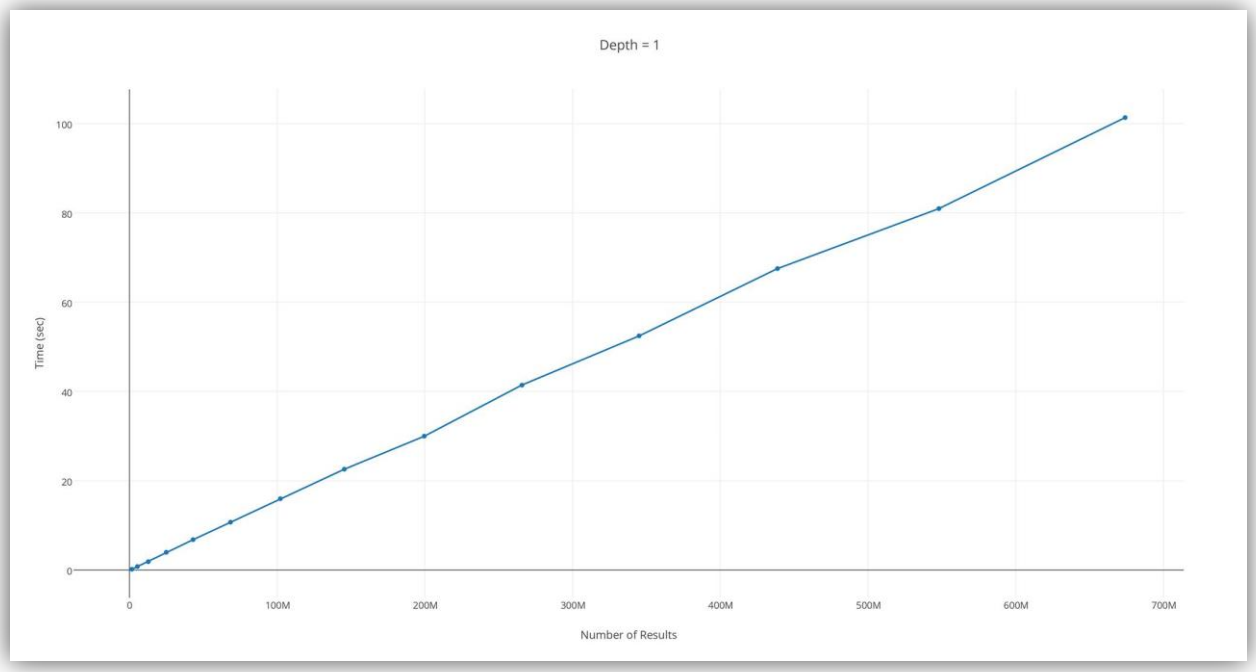
Ως γενικό συμπέρασμα συνάγεται ότι άλλοι παράγοντες πέρα από τον αριθμό των αποτελεσμάτων, όπως η πυκνότητα του γράφου, τα βάρη των ακμών, ο καταμερισμός των ακμών ανά κόμβο, αλλά και ο ο αριθμός των κόμβων που συνδέονται στους κανόνες αλυσίδας, παίζουν σημαντικό ρόλο στην απόδοση.

Σύμφωνα με τα παραπάνω, οι γράφοι από τη βιβλιοθήκη SNAP είναι ανομοιόμορφοι μεταξύ τους και η σύγκριση αυτών δεν εξάγει κάποιο ασφαλές συμπέρασμα. Αλλά από τους Πίνακες 2 και 3 μπορούν να φανούν οι χρόνοι για τον καθένα ξεχωριστά.

Στη συνέχεια, λόγω της παρατήρησης, ότι για να γίνει σύγκριση μεταξύ των χρόνων για την εκτέλεση του ερωτήματος σε γράφους, αυτοί πρέπει να είναι παρόμοιων χαρακτηριστικών, δημιουργήθηκαν πλήρεις γράφοι με σταθερά (0.5), αλλά και τυχαία βάρη. Οι πίνακες και τα διαγράμματα παρουσιάζονται παρακάτω:

Αριθμός κόμβων	Αριθμός ακμών	Αριθμός αποτελεσμάτων	Χρόνος(s)
200	39.800	1.584.040	0,17
300	89.700	5.364.060	0,78
400	159.600	12.736.080	1,87
500	249.500	24.900.100	3,97
600	359.400	43.056.120	6,80
700	489.300	68.404.140	10,72
800	639.200	102.144.160	15,98
900	809.100	145.476.180	22,59
1000	999.000	199.600.200	29,99
1100	1.208.900	265.716.220	41,42
1200	1.438.800	345.024.240	52,46
1300	1.688.700	438.724.260	67,53
1400	1.958.600	548.016.280	80,96
1500	2.248.500	674.100.300	101,35

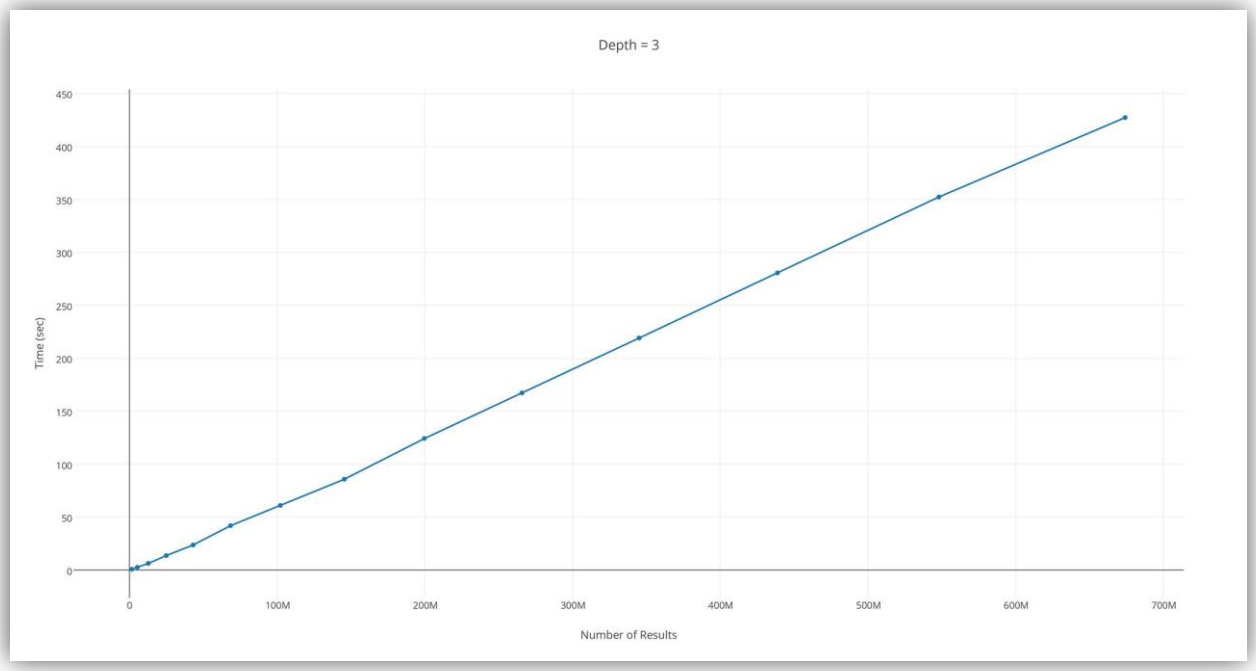
Πίνακας 6: Auto-generated graphs με σταθερά βάρη (0.5) για βάθος = 1



Σχήμα 5: Μετρήσεις auto-generated graphs με σταθερά βάρη (0.5) για βάθος 1

Αριθμός κόμβων	Αριθμός ακμών	Αριθμός αποτελεσμάτων	Χρόνος(s)
200	39.800	1.584.040	0,81
300	89.700	5.364.060	2,72
400	159.600	12.736.080	6,36
500	249.500	24.900.100	13,70
600	359.400	43.056.120	23,72
700	489.300	68.404.140	42,01
800	639.200	102.144.160	61,08
900	809.100	145.476.180	85,87
1000	999.000	199.600.200	124,31
1100	1.208.900	265.716.220	167,30
1200	1.438.800	345.024.240	219,20
1300	1.688.700	438.724.260	280,68
1400	1.958.600	548.016.280	352,29
1500	2.248.500	674.100.300	427,29

Πίνακας 7: Auto-generated graphs με σταθερά βάρη (0.5) για βάθος 3



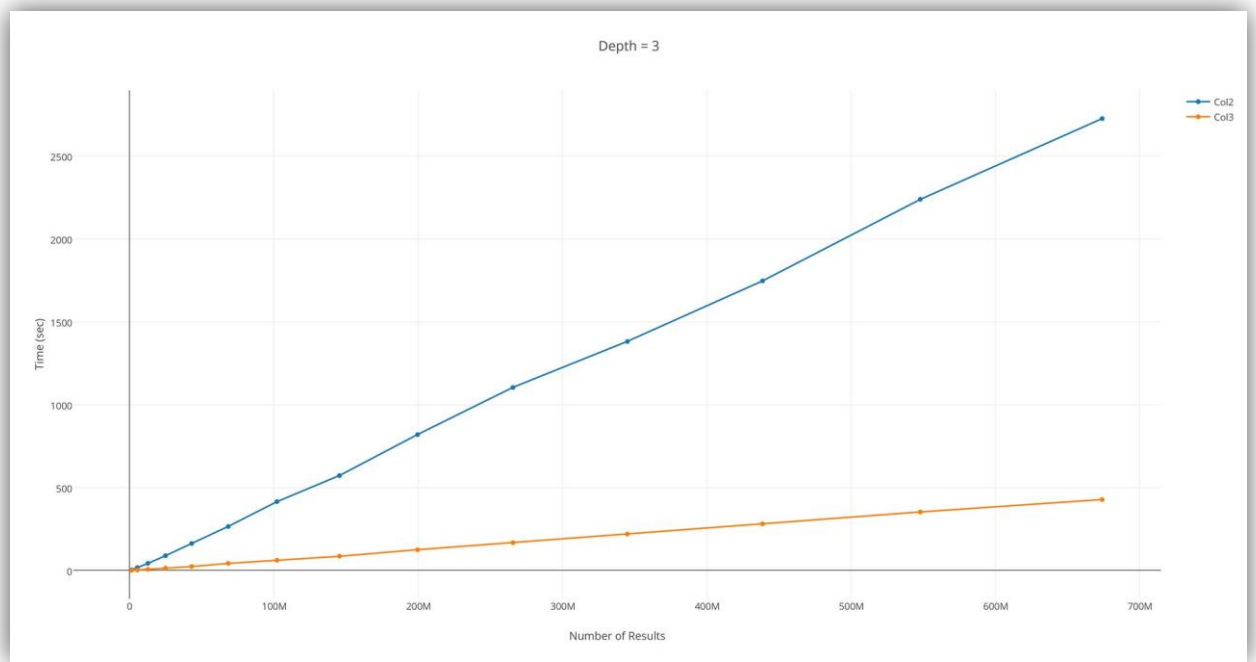
Σχήμα 6: Μετρήσεις auto-generated graphs με σταθερά βάρη (0.5) για βάθος 3

Σε αυτό το σημείο παρατηρείται ότι η καμπύλη στο διάγραμμα τείνει να είναι γραμμική, συμπέρασμα το οποίο εξάγεται με μεγαλύτερη ευκολία σε σχέση με τους γράφους SNAP, καθώς οι πλήρεις γράφοι που χρησιμοποιήθηκαν στις μετρήσεις είναι ομοιόμορφοι μεταξύ τους άρα και συγκρίσιμοι με βάση τον αριθμό των αποτελεσμάτων.

Τέλος, παρουσιάζεται μια σύγκριση ανάμεσα στους γράφους του Πίνακα 7 και του Πίνακα 8. Συγκεκριμένα, το χαρακτηριστικό που διαφέρει είναι τα βάρη των ακμών. Στην πρώτη μέτρηση τα βάρη είναι σταθερά, ενώ στην δεύτερη τα βάρη είναι τυχαία. Το παραπάνω αποσκοπεί στην καταγραφή της συμπεριφοράς του συστήματος σε μορφή διαγράμματος.

Αριθμός κόμβων	Αριθμός ακμών	Αριθμός αποτελεσμάτων	Χρόνος(s)
200	39.800	1.584.040	0,86
300	89.700	5.364.060	3,60
400	159.600	12.736.080	16,62
500	249.500	24.900.100	42,83
600	359.400	43.056.120	88,41
700	489.300	68.404.140	162,72
800	639.200	102.144.160	264,54
900	809.100	145.476.180	414,85
1000	999.000	199.600.200	572,09
1100	1.208.900	265.716.220	819,16
1200	1.438.800	345.024.240	1103,83
1300	1.688.700	438.724.260	1381,07
1400	1.958.600	548.016.280	1746,74
1500	2.248.500	674.100.300	2238,97

Πίνακας 8: Auto-generated graphs με τυχαία βάρη για βάθος 3



Σχήμα 7: Μετρήσεις auto-generated graphs με τυχαία βάρη για βάθος 3 και σύγκριση με auto-generated graphs με σταθερά βάρη (0.5) για βάθος 3

Παρατηρείται αύξηση στον χρόνο εκτέλεσης, ειδικά όσο ο αριθμός των αποτελεσμάτων μεγαλώνει. Το παραπάνω διάγραμμα, αλλά και η παρατήρηση των προηγούμενων έρχεται να επιβεβαιώσει την αρχική υπόθεση, ότι χαρακτηριστικά όπως η δομή του γράφου, τα βάρη των ακμών και η σύνδεση στους κανόνες αλυσίδας επηρεάζουν την απόδοση του συστήματος.

Οπότε συγκρίσεις μπορούν να γίνουν σε γράφους με παρόμοια χαρακτηριστικά, όπως οι πλήρεις γράφοι των Πινάκων 6, 7, όπου παρατηρείται από τα αντίστοιχα διαγράμματα ότι η καμπύλη τείνει να είναι γραμμική.

5. ΟΔΗΓΟΣ ΧΡΗΣΗΣ

Σε αυτό το κεφάλαιο δίνονται αρχικά οδηγίες για την εγκατάσταση προαπαιτούμενων εργαλείων, τα οποία απαιτούνται για το σύστημα. Στη συνέχεια, παρουσιάζονται οδηγίες για την εκτέλεση και τη χρήση του συστήματος.

5.1 Εγκατάσταση

Το σύστημα υλοποιήθηκε σε περιβάλλον Ubuntu 14.04 (64-bit) και με γλώσσα προγραμματισμού C.

Για τους παραπάνω λόγους, ο μεταγλωττιστής *gcc* [14] θα πρέπει να είναι εγκατεστημένος στο περιβάλλον. Αυτό πραγματοποιείται με τις ακόλουθες εντολές σε ένα τερματικό:

- *sudo add-apt-repository ppa:ubuntu-toolchain-r/test*
- *sudo apt-get update*
- *sudo apt-get install gcc-4.9*

Στη συνέχεια, επειδή το πρόγραμμα αλληλεπιδρά με τους εικονικούς πίνακες της SQLite πρέπει να είναι και αυτή εγκατεστημένη. Αντίστοιχα η εντολή είναι η ακόλουθη:

- sudo apt-get install sqlite3 libsqlite3-dev

5.2 Εκτέλεση και χρήση

Εφόσον εγκαταστάθηκαν με επιτυχία τα παραπάνω εργαλεία, το πρόγραμμα θα πρέπει να μεταγλωττιστεί. Στα αρχεία του προγράμματος περιέχεται αρχείο κατασκευής (*Makefile)* [15]. Ο χρήστης εφόσον ανοίξει ένα τερματικό και μεταβεί στον φάκελο που περιέχει τον κώδικα του συστήματος που αναπτύχθηκε, με την πληκτρολόγηση της εντολής *make* μεταγλωττίζει το πρόγραμμα, όπως φαίνεται και στην παρακάτω εικόνα:

```
                :~/Desktop/Ptuxiaki$ make
gcc -g -o SQLITE_MODEL.so SQLITE_MODEL.c chain_computation.c node.c parse_functions.c path_list.c property.c tail_nested.c hashmap_strings.c one_to_many_path_list.c one_to_many_path_join.c table.c Resultse
t.c evaluate.c parse_math_expr.c Vtable_columns.c graph.c graph_edge.c graph_node.c graph_table.c aggregation_array.c -lpthread -ldl -fPIC -shared;
tim@tim-Z68AP-D3:~/Desktop/Ptuxiaki$
```

Εικόνα 31: Εντολή make

Στη συνέχεια πληκτρολογώντας την εντολή *sqlite3* μεταβαίνει στο περιβάλλον της SQLite. Σε πρώτο στάδιο, φορτώνετε η επέκταση, η οποία είναι το σύστημα με την εντολή *.load ./SQLITE_MODEL* και ακριβώς μετά εισάγεται η εντολή *CREATE VIRTUAL_TABLE* με τις κατάλληλες παραμέτρους, όπως αναλύθηκε στο 3.4 υποκεφάλαιο.

Πιο συγκεκριμένα, στην εικόνα που ακολουθεί, φαίνεται μία παραδειγματική εκτέλεση των παραπάνω, με την *CREATE VIRTUAL_TABLE* να αντιπροσωπεύει ως ένα βαθμό το παράδειγμα της Εικόνας 1.

```
                :~/Desktop/Ptuxiaki$ sqlite3
SQLite version 3.8.2 2013-12-06 14:53:30
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .load ./SQLITE_MODEL
sqlite> CREATE VIRTUAL TABLE x USING my_vtable(7 | (x,y,yy,z,w,w1,w2),
6 | x= trusts.x | y= trusts.y | yy= likes.y | z= likes.z | w=((trusts.
w1) + likes.w2)/number.2| w1= trusts.w1 | w2= likes.w2,2.1|(g,./csv_ci
rcle/person.csv,./csv_circle/edge.csv,trusts,1*[{w1.+.min}],2*[!x,x1,x
2,x3,!y,y1,y2,y3,w1,w2])|(c,./csv_circle/likes.csv,likes,2*[!y,!z,w2])
,1| trusts.y=likes.y);
```

Εικόνα 32: Εντολές σε περιβάλλον SQLite

Τέλος, με τη χρήση απλών ή και σύνθετων ερωτημάτων SQL στο περιβάλλον της SQLite εξάγονται αποτελέσματα, όπως φαίνεται στο μικρό παράδειγμα της εικόνας που ακολουθεί:

```
sqlite> select x,z,w from x;

4|5|0.85
4|6|1.0
1|6|0.6
1|5|0.5
1|6|0.65
2|6|0.4
2|6|1.15
3|6|0.35
3|5|0.95
sqlite>
```

Εικόνα 33: Παράδειγμα select ερωτήματος στο σύστημα

6. ΣΥΜΠΕΡΑΣΜΑΤΑ

Στο τελευταίο κεφάλαιο αναφέρονται τα τελικά συμπεράσματα που προκύπτουν από την πτυχιακή εργασία. Αρχικά, παρουσιάζονται σχόλια πάνω στο σύστημα και στη συνέχεια κάποιες μελλοντικές επεκτάσεις που θα μπορούσαν να πραγματοποιηθούν με σκοπό τη βελτίωση του εργαλείου.

6.1 Σχόλια πάνω στο σύστημα

Το σύστημα που υλοποιήθηκε είχε ως αρχική προσδοκία να είναι όσο το δυνατό πιο εύχρηστο και φιλικό προς το χρήστη που το χρησιμοποιεί. Αυτή η προσδοκία εκπληρώθηκε σε σημαντικό βαθμό καθώς αποφεύχθηκαν περίπλοκα βοηθητικά προγράμματα ή δύσκολες παραμετρικοποιήσεις ως προς την εγκατάσταση ή την λειτουργία. Σε αυτήν την κατεύθυνση, η χρήση των εικονικών πινάκων της SQLite είχε καθοριστικό ρόλο, καθώς η μόνη προαπαιτούμενη γνώση που απαιτείται είναι η χρήση της γλώσσας SQL, σε συνδυασμό με τη γλώσσα του συστήματος, της οποίας το συντακτικό αναλύθηκε στο υποκεφάλαιο 3.4.

Επόμενος στόχος ήταν το σύστημα να δίνει την ελευθερία στο χρήστη να ορίσει οποιοδήποτε κανόνα αλυσίδας με ιδιότητες, είτε απλό είτε περίπλοκο, επιθυμεί. Η επίτευξη αυτού του στόχου έγινε σε πολύ μεγάλο βαθμό, καθώς τόσο η γλώσσα του συστήματος, όσο και οι δομές δεδομένων και οι αλγόριθμοι που χρησιμοποιήθηκαν συντέλεσαν σε αυτό. Ο μόνος περιορισμός για επιτυχή εκτέλεση του προγράμματος είναι η τήρηση των απλών κανόνων που έχουν οριστεί κατά μήκος αυτής της πτυχιακής.

Επιπλέον, βασική προϋπόθεση ήταν η εξαγωγή ορθών αποτελεσμάτων, ανάλογα τον ορισμό κανόνων αλυσίδας και των δεδομένων που δέχεται ως είσοδο το σύστημα. Κατά τη διάρκεια της υλοποίησης, αλλά και μετά το πέρας της, πραγματοποιήθηκαν πολλά πειράματα με διαφόρων ειδών δεδομένα, αλλά και σενάρια χρήσης, ώστε να διορθωθούν τυχόν ατέλειες και να παρουσιαστεί ένα άρτιο σύστημα.

Τέλος, υπήρξε η φιλοδοξία ενός αποδοτικού συστήματος, με όσο το δυνατόν καλύτερους χρόνους εξαγωγής αποτελεσμάτων για μεγάλα δεδομένα. Δεδομένου του παραπάνω, μελετήθηκαν διάφορες δημοσιεύσεις και τεχνικές, ώστε να βρεθούν ικανοποιητικοί αλγόριθμοι και δομές δεδομένων. Ύστερα από την ολοκλήρωση της υλοποίησης του εργαλείου, πραγματοποιήθηκαν πειραματικές μετρήσεις, όπου τα αποτελέσματα αυτών παρουσιάζονται στο κεφάλαιο 4. Σύμφωνα με τις μετρήσεις, η απόδοση είναι σε αρκετά ικανοποιητικό επίπεδο και εκπληρεί την παραπάνω φιλοδοξία.

6.2 Μελλοντικές επεκτάσεις

Σε αυτό το σημείο παρατίθενται κάποιες σκέψεις και ιδέες για μελλοντική επέκταση του συστήματος, το οποίο υλοποιήθηκε στα πλαίσια της πτυχιακής εργασίας.

Αρχικά, η διάδραση του χρήστη με το εργαλείο πραγματοποιείται μέσω τερματικού με τη βοήθεια της SQLite. Με αυτόν τον τρόπο πετυχαίνεται ευχρηστία, καθώς η μόνη προαπαιτούμενη γνώση είναι αυτής της γλώσσας SQL και του συστήματος, αλλά υπολείπεται εκφραστικότητας. Θα μπορούσε το περιβάλλον να γίνει ακόμα πιο φιλικό με μια UI εφαρμογή, στην οποία ο χρήστης θα δίνει τις παραμέτρους και το ερώτημα που επιθυμεί και θα παίρνει τα αντίστοιχα αποτελέσματα.

Στη συνέχεια, το συντακτικό της γλώσσας του συστήματος μπορεί να βελτιωθεί ως προς τη μορφή εισαγωγής των παραμέτρων, με σκοπό να είναι πιο εύκολο σε έναν νέο χρήστη να το χρησιμοποιήσει αλλά και να είναι πιο ευέλικτο σε μελλοντικές επεκτάσεις της γλώσσας.

Επιπρόσθετα, η διάσχιση και οι πράξεις πάνω σε γράφο γίνονται με βάση τον αλγόριθμο αναζήτησης κατά πλάτος με περιορισμό στο βάθος και σε συνδυασμό με αθροιστικές συναρτήσεις. Αυτός ο τρόπος θα ήταν πιο αποδοτικός για μεγάλα δεδομένα, εάν εισαγόταν η ιδέα του πολυνυματισμού. Ένας καλός αλγόριθμος ως προς αυτή την κατεύθυνση είναι ο Pregel [16].

Τέλος, μια επιπλέον επέκταση είναι η προσθήκη λογικής if-then-else πάνω στην αλυσίδα, όπως στη γλώσσα προγραμματισμού Prolog [17]. Αναλυτικότερα, η ύπαρξη περισσότερων του ενός σωμάτων κανόνων για ένα κύριο κατηγόρημα.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Chain rules	Κανόνες σε μορφή αλυσίδας
Property chain	Κανόνες σε μορφή αλυσίδας με ιδιότητες
Virtual tables	Εικονικοί πίνακες
Recommendations	Συστάσεις
Conjunctive query	Συνδετικό ερώτημα
Labeled graph	Γράφος ετικετών
Property graph	Γράφος ιδιοτήτων
Key/Value	Κλειδί/Τιμή
Directed	Κατευθυνόμενος
Multi-relational	Πολυ-σχεσιακότητα
Column-oriented table	Πίνακας προσανατολισμένος με βάση τις στήλες
Row-oriented table	Πίνακας προσανατολισμένος με βάση τις γραμμές
Adjacency list	Λίστα γειτνίασης
Tail-nested tables	Εμφωλευμένοι πίνακες με ουρά
Nodes	Κόμβοι
Edges	Ακμές
Join tail-nested tables	Ένωση εμφωλευμένων πινάκων
Join operations	Πράξεις ένωσης
Nested loop joins	Εμφωλευμένες ενώσεις βρόχου
Hashed loop joins	Ενώσεις βρόχου κατακερματισμού
Aggregation functions	Αθροιστικές συναρτήσεις
Expand aggregation function	Αθροιστική συνάρτηση εξάπλωσης
Concatenate aggregation function	Αθροιστική συνάρτηση ένωσης μονοπατιών
Callback methods	Μέθοδοι οπισθοδρόμησης
Assembly language	Συμβολική γλώσσα
Linear hashing	Γραμμικός κατακερματισμός
Path	Μονοπάτι
Makefile	Αρχείο κατασκευής

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

SQL	Structured Query Language
BFS	Breadth-first search
CSV	Column Separated Values
GCC	GNU Compiler Collection
UI	User Interface

ΠΑΡΑΡΤΗΜΑ

Παρακάτω παρουσιάζονται ενδεικτικά αρχεία εισόδου, τόσο για την αναπαράσταση γράφου, όσο και για τη δήλωση σχέσεων μεταξύ αρχικών και τελικών κόμβων, χωρίς την ύπαρξη ενδιάμεσων.

Αρχεία εισόδου για αναπαράσταση γράφου

```
1  id      | pr1       | pr2       | pr3
2  integer | string    | double    | integer
3  1       | Efthimis  | 0.3       | 11
4  2       | Marianna  | 0.22      | 22
5  3       | Nikos     | 7.8       | 33
6  4       | Kostas    | 1.2       | 44
7  5       | Maria     | 1         | 4
8  6       | Giannis   | 1.5       | 445
9
```

Εικόνα 34: Αρχείο εισόδου για την αναπαράσταση κόμβων γράφου

```
1  id1     | id2     | w1
2  integer | integer | double
3  1       | 2       | 0.6
4  1       | 3       | 0.5
5  2       | 4       | 0.3
6  3       | 4       | 0.2
7  4       | 1       | 0.7
```

Εικόνα 35: Αρχείο εισόδου για αναπαράσταση ακμών γράφου

Αρχεία εισόδου για δήλωση σχέσεων μεταξύ αρχικών και τελικών κόμβων

```
1  id1     | id2       | w2
2  integer | integer   | double
3  3       |6          | 0.8
4  4       |6          | 0.5
5  2       |5          | 0.4
6
```

Εικόνα 36: Αρχείο εισόδου για δήλωση σχέσεων μεταξύ αρχικών και τελικών κόμβων

ΑΝΑΦΟΡΕΣ

[1] J.D., Ullman, ed., *Theoretical Studies in Computer Science*, ACADEMIC Press,1992.

[2] J.A. Kreibich, *Using SQLite*, O'Reilly Media, 2010, p. 530.

[3] «The Virtual Table Mechanism Of SQLite,» [Ηλεκτρονικό]. Available: https://www.sqlite.org/vtab.html. [Προσπελάστηκε 28/9/16]

[4] C.J. Date and H. Darwen, *A Guide to SQL Standard*, Addison-Wesley Professional, 1996, p. 554.

[5] «Property Graph Model» [Ηλεκτρονικό]. Available: https://github.com/tinkerpop/blueprints/wiki/Property-Graph-Model [Προσπελάστηκε 29/9/16].

[6] J. Seo and S. Guo and M.S. Lam, "SociaLite: Datalog Extensions for Efficient Social Network Analysis", *Proc. 29h Int'l Conf. Data Eng. (ICDE 13)*, IEEE CS Press, 2013, pp. 49.

[7] «Graph labeling», [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Graph_labeling [Προσπελάστηκε 25/9/16].

[8] «Learn Datalog Today», [Ηλεκτρονικό]. Available: http://www.learndatalogtoday.org [Προσπελάστηκε 20/9/16].

[9] «Run-Time Loadable Extensions», [Ηλεκτρονικό]. Available: https://www.sqlite.org/loadext.html [Προσπελάστηκε 10/9/16].

[10] «Virtual Tables», [Ηλεκτρονικό]. Available: https://rogerbinns.github.io/apsw/vtable.html [Προσπελάστηκε 10/9/16].

[11] «Python 3 programs versus C gcc», [Ηλεκτρονικό]. Available: https://benchmarksgame.alioth.debian.org/u64q/compare.php?lang=python3&lang2=gcc [Προσπελάστηκε 10/9/16].

[12] W. Litwin, Linear Hashing: A New Tool for File and Table Addressing. In Proc. 6th International Conference on Very Large Databases (VLDB) , 1980, 212–223.

[13] J. Leskovec, «SNAP» [Ηλεκτρονικό]. Available: http://snap.stanford.edu/ [Προσπελάστηκε 29/9/16].

[14] «GCC, the GNU Compiler Collection», [Ηλεκτρονικό]. Available: https://gcc.gnu.org/ [Προσπελάστηκε 20/9/16].

[15] «GNU make», [Ηλεκτρονικό]. Available: https://www.gnu.org/software/make/manual/make.html [Προσπελάστηκε 20/9/16].

[16] Grzegorz Malewicz, Matthew H. Austern, Aart J. C. Bik,James C. Dehnert, Ilan Horn,Naty Leiser, and Grzegorz Czajkowski, "Pregel: A System for Large-Scale Graph Processing", *in Proc. ACM SIGMOD Intl. Conf. on Management of Data, 2010,* 135-146.

[17] «SWI Prolog», [Ηλεκτρονικό]. Available: http://www.swi-prolog.org/ [Προσπελάστηκε 25/9/16].

[18] «KCachegrind», [Ηλεκτρονικό]. Available: http://kcachegrind.sourceforge.net/html/Home.html [Προσπελάστηκε 25/9/16].