

A Synthetic Benchmark for Evaluating RDF Streaming Data Systems

by

SKLIRI IRENE
(MTH283)

Bachelor's degree in Informatics Engineering, Hellenic Mediterranean University, 2022

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING SCHOOL OF
ENGINEERING

HELLENIC MEDITERRANEAN UNIVERSITY

2024

Approved by:

Dr. Papadakis Nikolaos

Ένα συνθετικό σημείο αναφοράς (Benchmark) για την αξιολόγηση συστημάτων δεδομένων ροής RDF

από
ΣΚΛΗΡΗ ΕΙΡΗΝΗ
(ΜΤΗ283)

Πτυχίο Μηχανικών Πληροφορικής, Ελληνικό Μεσογειακό Πανεπιστήμιο, 2022

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

υποβάλλεται σε μερική εκπλήρωση των απαιτήσεων για την απόκτηση του πτυχίου

ΜΕΤΑΠΤΥΧΙΑΚΟ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΕΛΛΗΝΙΚΟ ΜΕΣΟΓΕΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

2024

Εγκεκριμένο από:

Δρ. Παπαδάκης Νικόλαος

Acknowledgements

Η παρούσα διπλωματική εργασία πραγματοποιήθηκε στο Ηράκλειο Κρήτης, στο Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Ελληνικού Μεσογειακού Πανεπιστημίου το έτος 2024. Η συγγραφή της εργασίας έλαβε χώρα στην Αθήνα κατά το ακαδημαϊκό έτος 2023/2024.

Η πιο σημαντική παρουσία στη ζωή μας είναι η οικογένειά μας. Γι' αυτό, θα ήθελα να εκφράσω την βαθιά μου ευγνωμοσύνη προς τους γονείς μου, Μαρία και Γιώργο, των οποίων η αμέριστη υποστήριξή τους, μου επέτρεψε να φτάσω ως εδώ και να βαδίσω προς τον «επίλογο» των σπουδών μου. Τους ευχαριστώ ειλικρινά για την ψυχολογική και οικονομική υποστήριξη καθ' όλη τη διάρκεια αυτών των ετών της πορείας μου. Η πίστη τους στις δυνατότητές μου και η συνεχής ενθάρρυνσή τους αποτέλεσαν το θεμέλιο της ακαδημαϊκής και προσωπικής μου ανάπτυξης.

Ένα μεγάλο ευχαριστώ στους ανεκτίμητους φίλους μου, που αποτέλεσαν πηγή μεγάλης υποστήριξης τόσο για τις σπουδές όσο και για την καριέρα μου. Οι συμβουλές τους και η στήριξή τους ήταν και είναι ανεκτίμητες.

Τέλος, θα ήθελα να εκφράσω τις ευχαριστίες μου στους επιβλέποντες καθηγητές της διπλωματικής μου εργασίας, Καθηγητή Νικόλαο Παπαδάκη και Καθηγητή Χαρίδημο Κονδυλάκη, των οποίων η καθοδήγησή τους, μου επέτρεψε να ολοκληρώσω αυτό το έργο.

Αθήνα, Ιούνιος 2024

Abstract

RDF Stream Processing (RSP) has emerged as a critical research area, driven by the growing demand for real-time analysis and management of continuous RDF data streams. Despite recent progress, the field still faces significant challenges, particularly regarding standardization, system efficiency, and evaluation. This study examines the current state of RSP systems, with a focus on evaluating available benchmarks. We analyze these benchmarks based on key performance criteria, such as concurrent join execution, full-text search capabilities, and duplicate removal, and identify significant shortcomings in their ability to holistically assess system performance.

To address these shortcomings, we implemented a Java-based benchmark using RSP4J, which utilizes the RSP-QL language for query creation. This benchmark introduces additional evaluation criteria to overcome the limitations of existing approaches and provides a more reliable measure of system performance across various scenarios. This master's thesis aims to advance the field of RDF Stream Processing by contributing to standardization efforts and by developing more efficient and comprehensive evaluation tools.

Περίληψη

Η Επεξεργασία Ροών RDF (RDF Stream Processing - RSP) έχει αναδειχθεί σε έναν κρίσιμο τομέα έρευνας, υποκινούμενο από την αυξανόμενη ζήτηση για ανάλυση και διαχείριση συνεχών ροών δεδομένων RDF σε πραγματικό χρόνο. Παρά την πρόοδό του, ο τομέας εξακολουθεί να αντιμετωπίζει σημαντικές προκλήσεις, ιδίως όσον αφορά την τυποποίηση, την αποδοτικότητα των συστημάτων και την αξιολόγηση τους. Η παρούσα μελέτη εξετάζει την τρέχουσα κατάσταση των συστημάτων RSP, με έμφαση στην αξιολόγηση των διαθέσιμων benchmarks. Αναλύουμε αυτά τα benchmarks με βάση καίρια κριτήρια απόδοσης, όπως η ταυτόχρονη εκτέλεση join, οι δυνατότητες αναζήτησης πλήρους κειμένου και η αφαίρεση διπλότυπων, εντοπίζοντας σημαντικές ελλείψεις στην ικανότητά τους να αξιολογήσουν ολιστικά την απόδοση των συστημάτων. Για την αντιμετώπιση αυτών των αδυναμιών, υλοποιήθηκε ένα benchmark σε Java, βασισμένο στο RSP4J, το οποίο χρησιμοποιεί την γλώσσα RSP-QL για τη δημιουργία επερωτήσεων. Το συγκεκριμένο benchmark εισάγει επιπλέον κριτήρια αξιολόγησης, με στόχο την υπέρβαση των περιορισμών των υπαρχόντων προσεγγίσεων και προσφέρει ένα πιο αξιόπιστο μέτρο της απόδοσης των συστημάτων σε διάφορα σενάρια. Η εργασία αυτή στοχεύει στην προώθηση του πεδίου της επεξεργασίας ροών RDF, συμβάλλοντας τόσο στις προσπάθειες τυποποίησης όσο και στην ανάπτυξη πιο αποδοτικών και ολοκληρωμένων εργαλείων αξιολόγησης.

Πίνακας Περιεχομένων

Acknowledgements	3
Abstract	4
Περίληψη	5
Κεφάλαιο 1: Introduction	7
1.1 Evolution of the Web	7
1.2 Semantic web	8
Κεφάλαιο 2: RDF	11
2.1 RDF Streaming	12
2.2 RDF streaming languages	13
2.3 RDF Stream Processing Systems	14
Κεφάλαιο 3: State of the art	16
3.1 Benchmarks	17
3.2 RDF Stream Processing Benchmarks	18
3.3 Choke points and their meaning	20
3.4 Comparison of existing benchmarks based on choke points	23
Κεφάλαιο 4: Development of an RDF Streaming benchmark	32
4.1 Hardware and Software	32
4.2 Programming Languages and Libraries	32
4.2.1 Java	33
4.2.2 RspQL	34
4.2.3 Rps4j	35
4.3 Approach & Implementation	35
4.3.1 Explanation of the Benchmark Queries Based on the Choke Points	41
Κεφάλαιο 5: Conclusions	52
5.1 Conclusion	52
5.2 Future Work	53
References	55

Κεφάλαιο 1: Introduction

1.1 Evolution of the Web

Το World Wide Web αποτελεί πλέον ένα από τα καινοτόμα τεχνολογικά επιτεύγματα της σύγχρονης εποχής. Συγκεκριμένα, είναι ένα ανοιχτό σύστημα συνδεδεμένων πληροφοριών το οποίο δίνει την δυνατότητα στους χρήστες να έχουν δωρεάν πρόσβαση, αναζήτηση και ανταλλαγή πληροφοριών σε όλο το κόσμο. Ο ιστός δημιουργεί hyperTexts, δηλαδή την διασύνδεση πολλών μη ιεραρχημένων πληροφοριών τα οποία μπορούν να πάρουν διάφορες μορφές (εικόνα, βίντεο) πέρα του γραπτού λόγου. Αναπτύχθηκε την χρονολογία του 1989 από τον Tim Berners-Lee και Robert Cailliau οι οποίοι εργάζονταν στο CERN της Ελβετίας.

Η ιδέα της δημιουργίας του W3 ξεκίνησε από την ανάγκη κοινοποίησης και διαχείρισης πληροφοριών μεταξύ των ερευνών στο CERN. Η πρώτη υλοποίηση ξεκίνησε το 1990. Αναπτύχθηκαν τα βασικά στοιχεία τα οποία ήταν το πρωτόκολλο μεταφοράς Υπερκειμένου (HTTP), η γλώσσα σήμανσης υπερκειμένου (HTML) καθώς και το πρώτο πρόγραμμα περιήγησης, το οποίο λειτουργούσε και ως επεξεργαστής. Λόγω της ραγδαίας ανάπτυξης του ιστού, τα έγγραφα πλέον “μπορούσαν” να επικοινωνήσουν το ένα με το άλλο επιτρέποντας στους χρήστες να έχουν πρόσβαση σε οποιοδήποτε έγγραφο χωρίς καμία δυσκολία. Αυτή η επικοινωνία εξυπηρετήθηκε χάρει στα URL. Τα URL, παρέχουν μία μοναδική διεύθυνση για κάθε έγγραφο στον ιστό. Στις αρχές του 1990, αναπτύχθηκε το User Interface, με άλλα λόγια, τα προγράμματα περιήγησης μετατράπηκαν πιο φιλικά προς τον χρήστη για την καλύτερη περιήγηση και διευκόλυνση τους. Όσον αφορά την τεχνική εξέλιξη του ιστού, δημιουργήθηκαν περαιτέρω πρωτόκολλα για την βελτιστοποίηση της λειτουργικότητας. Επιπλέον στην ανάπτυξη της επόμενης γενιάς του W3 πρωτοκόλλου, ο Carl Barker εισήγαγε ελέγχους πρόσβασης με κωδικούς ασφαλείας για την διασφάλιση ευαίσθητων και προσωπικών πληροφοριών των χρηστών. Συνοψίζοντας, ο Παγκόσμιος Ιστός ξεκίνησε ως ένα έργο για τη διευκόλυνση της κοινοποίησης πληροφοριών μεταξύ των επιστημόνων στο CERN. Πλέον όμως, ο Ιστός έχει εξελιχθεί σε ένα πανταχού παρόν παγκόσμιο σύστημα πληροφόρησης, επαναστατώντας τον τρόπο με τον οποίο ο χρήστης έχει πρόσβαση και αλληλεπιδρά με πληροφορίες.

Η ραγδαία εξέλιξη και ανάπτυξη του Παγκόσμιου Ιστού έχουν οδηγήσει στη δημιουργία του Semantic Web, δηλαδή μιας επέκτασης που στοχεύει στην κατανόηση και στην

διαχείριση των δεδομένων από μηχανές, μέσω της χρήσης του RDF (Resource Description Framework).

Παρακάτω, θα γίνει ανάλυση του Σημασιολογικού Ιστού καθώς και το RDF, δίνοντας έμφαση στις δυνατότητες και τις προκλήσεις που παρουσιάζουν για την αξιοποίηση και επεξεργασία τεράστιων όγκων δεδομένων. [1] [2]

1.2 Semantic web

The Semantic Web, or in other words Web 3.0, θεωρείται η επόμενη σημαντική εξέλιξη και επέκταση στη διασύνδεση και αναπαράσταση των πληροφοριών. Ο όρος εισήχθη από τον Tim Berners-Lee, ο οποίος οραματίστηκε έναν Ιστό όπου οι υπολογιστές θα μπορούν να επεξεργάζονται δεδομένα με έναν πιο “ανθρώπινο τρόπο”, δηλαδή να κατανοούν το νόημα των δεδομένων και να τα επεξεργάζονται αυτόνομα και αποτελεσματικά. Η πρώτη έκδοση του Σημασιολογικού Ιστού (Web 1.0) ήταν η αρχή της ψηφιακής εποχής τη δεκαετία του 1990. Ο Ιστός αποτελούνταν κυρίως από στατικά έγγραφα (HTML), στα οποία επιτρεπόταν μόνο η ανάγνωση και δεν υπήρχε καμία αλληλεπίδραση με αυτά, δηλαδή τροποποίηση περιεχομένου. Η δεύτερη έκδοση του Σημασιολογικού Ιστού (Web 2.0) ξεκίνησε το 2004 λόγω της ανόδου των κοινωνικών μέσων επικοινωνίας όπου μέσω αυτού, αλληλεπιδρούσαν οι χρήστες. Ο Ιστός πλέον έδωσε την δυνατότητα στην ανθρωπότητα, να μπορεί να γράψει και να τροποποιήσει αρχεία. Αυτό πραγματοποιήθηκε χάρη στην ανάπτυξη των HTML5, CSS και JavaScript. Η τρίτη έκδοση του Σημασιολογικού Ιστού (Web 3.0) δεν έχει υλοποιηθεί ακόμα ολοκληρωτικά. Θεωρείται όμως από πολλούς ως το επόμενο στάδιο εξέλιξης. Κάποια στοιχεία τείνουν προς την εκτέλεση της υλοποίησης.

Ο ιστός, δίνει την δυνατότητα επικοινωνίας και κατανόησης των δεδομένων από έναν υπολογιστή σε έναν άλλον, με σκοπό να μπορούν να εκτελούν όλο και πιο εξελιγμένες λειτουργίες. Εκτός από τον εμπλουτισμό και ενίσχυση των δεδομένων με σημασιολογικά μεταδεδομένα, ο απώτερος σκοπός του Σημασιολογικού Ιστού είναι η ανάπτυξη ευφυούς επικοινωνίας με τις μηχανές που να μοιάζει με την ανθρώπινη επικοινωνία. Πλέον οι αλγόριθμοι πλοηγούνται σε έναν μεγάλο όγκο περιεχομένου το οποίο εξελίσσεται ολοένα και πιο αδόμητο και περίπλοκο (βίντεο, εικόνες, ήχος, αισθητήρες). Τα δεδομένα, τα οποία είναι εμπλουτισμένα με σημασιολογικά και δομημένα μεταδεδομένα, επιτρέπουν στα

συστήματα υπολογιστών να διαχειρίζονται και να βρίσκουν τις πληροφορίες για τον άνθρωπο με πολύ μεγαλύτερη ακρίβεια.

Ο Σημασιολογικός Ιστός έχει εφαρμοστεί σε πολλούς τομείς όπως στις βάσεις δεδομένων και τις κινητές επικοινωνίες. Οι γλώσσες RDF και OWL χρησιμοποιούνται ευρέως για την ανταλλαγή σημασιολογικών δεδομένων. Η ανάπτυξή του, περιλαμβάνει τη δημιουργία μιας ποικίλλας σημασιολογικής υποδομής όπου περιέχει βάσεις γνώσης, οντολογίες, πράκτορες, συλλογισμούς και εξελεγμένες δυνατότητες σημασιολογικής αναζήτησης. Για να επικοινωνούν αποτελεσματικά οι πράκτορες στον Σημασιολογικό Ιστό, πρέπει να κατανοούν και να ερμηνεύουν με ακρίβεια τη σημασία των πληροφοριών. Αυτό προϋποθέτει κοινά αποδεκτούς όρους, που μπορεί να δημιουργήσει δυσκολίες σε ένα ποικιλόμορφο, ανοιχτό περιβάλλον. Με την ταχεία εξέλιξή του, ο Σημασιολογικός Ιστός οραματίζεται να είναι πολυεπίπεδος, συνθέσιμος και επεκτάσιμος. Οι “έξυπνοι πράκτορες” θα αποτελούν σημαντικό ρόλο, εκτελώντας εργασίες μέσω της επικοινωνίας, την κατανόηση και τον συνδυασμό πληροφοριών από διάφορους πόρους.

Ένα ακόμη προτέρημά του, είναι η οργάνωση της σύνδεσης των δεδομένων με τέτοιο τρόπο ώστε να είναι ευκολότερη και πιο βέλτιστη η πρόσβαση και η χρήση τους από τις μηχανές. Αυτό μπορεί να επιτευχθεί με τη χρήση προτύπων όπως το RDF (Resource Description Framework) και το OWL (Web Ontology Language), τα οποία αναπαριστούν τη γνώση και την δημιουργία σχέσεων μεταξύ των δεδομένων αυτών.

Με την ενσωμάτωση αυτών των τεχνολογιών, αναπτύσσονται προηγμένες εφαρμογές. Μερικές από αυτές είναι οι έξυπνες μηχανές αναζήτησης ή οι προσωπικοί βοηθοί. Αυτοί έχουν την δυνατότητα να καταλάβουν το περιεχόμενο μιας αναζήτησης, ή να μπορούν να επεξεργαστούν φυσική γλώσσα καθώς και να εκτελούν σύνθετες ενέργειες.

Η ανάπτυξη του Σημασιολογικού Ιστού έχει επίσης επηρεάσει σημαντικά την εξέλιξη της μηχανικής μάθησης και της τεχνητής νοημοσύνης, αφού οι μηχανές μπορούν να μαθαίνουν και να βελτιώνουν τις λειτουργίες τους με πιο αποτελεσματικό τρόπο. Με την άφιξη του Web 3.0, θα πραγματοποιηθούν περισσότερες εφαρμογές των τεχνολογιών του Σημασιολογικού Ιστού στην καθημερινότητα του ανθρώπου. Μέσα κοινωνικής δικτύωσης, συστήματα διαχείρισης επιχειρήσεων και υπηρεσίες υγείας είναι μερικά παραδείγματα ενσωμάτωσης των εφαρμογών. Η δυνατότητα της αυτόνομης επεξεργασίας των δεδομένων

θα συμβάλλει θετικά στη δημιουργία ενός ευφυούς διαδικτύου, δημιουργώντας νέες δυνατότητες και προκλήσεις για την τεχνολογία και την κοινωνία.

Συμπερασματικά, ο Σημασιολογικός Ιστός υπόσχεται να αλλάξει ριζικά τον τρόπο με τον οποίο ο άνθρωπος διαχειρίζεται τις πληροφορίες, δημιουργώντας έτσι ένα περιβάλλον στο οποίο οι μηχανές θα μπορούν να “συνεργάζονται” με έναν πιο φυσικό τρόπο με τους ανθρώπους, οδηγώντας σε μια νέα εποχή ψηφιακής καινοτομίας. [3] [4] [5] [6] [7]

Κεφάλαιο 2: RDF

Το Resources Description Framework (RDF) είναι ένα πρότυπο μοντέλο δεδομένων που χρησιμοποιείται για την ανταλλαγή δεδομένων και την αναπαράσταση πληροφορίας στον Παγκόσμιο Ιστό. Με άλλα λόγια, είναι ένας τρόπος οργάνωσης των δεδομένων στο διαδίκτυο, μετατρέποντας έτσι τα δεδομένα, να είναι πιο κατανοητά και εύκολα για τους υπολογιστές. Προτάθηκε από το W3C τον Φεβρουάριο του 1999 και καθορίζει το μοντέλο δεδομένων RDF και μια σύνταξη σε μορφή XML αρχείων για τη σειριοποίηση των δεδομένων αυτών. Το RDF, έχει την δυνατότητα να επικοινωνήσει και να κατανοήσει άλλα συστήματα τα οποία μπορεί να μην καταλαβαίνουν το συγκεκριμένο μοντέλο. Αυτό το καθιστά ένα πολύ χρήσιμο και ισχυρό εργαλείο για την αναζήτηση και κατανόηση πληροφοριών στον Παγκόσμιο Ιστό. Για να το πετύχει αυτό, χρησιμοποιεί έναν "προσαρμογέα" RDF που προσθέτει μοναδικές ταυτότητες (URIs) στα δεδομένα που δεν τις έχουν και μετά δημιουργεί δηλώσεις που περιγράφουν αυτά τα δεδομένα. Αυτή η χρήση των URIs στο μοντέλο RDF το κάνει να ξεχωρίζει συγκριτικά με το XML. Η επιλογή της XML ως συντακτικού είναι προφανής και η προδιαγραφή του RDF το χρησιμοποιεί. Ωστόσο, το μοντέλο δεδομένων RDF δεν είναι δεσμευμένο με μια συγκεκριμένη σύνταξη. Μπορεί να εκφραστεί σε οποιαδήποτε συντακτική αναπαράσταση και μπορεί επίσης να εξαχθεί από μη-RDF πηγές δεδομένων.

Το RDF επιτρέπει την περιγραφή πόρων (resources) υπό την μορφή τριπλετών. Η σύσταση του W3C, αναπαριστά τις πληροφορίες σε τριπλέτες, οι οποίες αποτελούνται από ένα υποκείμενο (περιγραφή του πόρου), ένα κατηγορημα (η ιδιότητα ή σχέση του πόρου) και ένα αντικείμενο (Η τιμή της ιδιότητας ή ο πόρος με τον οποίο σχετίζεται). Ωστόσο, το παραδοσιακό RDF δεν περιέχει χρονική διάσταση. Προκειμένου να αντιμετωπιστεί αυτό, οι ροές RDF ορίζονται ως ακολουθίες τριπλετών RDF με χρονικές "σφραγίδες", εφαρμόζοντας τεχνολογίες του σημασιολογικού ιστού σε δεδομένα ροής.

Ένας πόρος θεωρείται ένα αντικείμενο. Οι πόροι μπορεί να είναι συγγραφείς, βιβλία, τοποθεσίες, άνθρωποι, ερωτήματα αναζήτησης κλπ. Κάθε πόρος έχει ένα URI, ένα Καθολικό Αναγνωριστικό Πόρου. Ένα URI μπορεί να είναι ένα URL (διεύθυνση στον Παγκόσμιο Ιστό), ή κάποιο άλλο είδος μοναδικού αναγνωριστικού. Αυτή η μορφή απεικονίζει σύνθετες σχέσεις και την σύνδεση δεδομένων από διαφορετικές πηγές με τρόπο

συνεκτικό. Επίσης, αυτός ο τρόπος αναπαράστασης επιτρέπει την ανταλλαγή πληροφοριών στον Παγκόσμιο Ιστό με πολύ ευελιξία και ευκολία. [8] [12]

2.1 RDF Streaming

Ένα RDF Stream είναι μια ροή δεδομένων όπου κάθε στοιχείο δεδομένων είναι ένα RDF αντικείμενο. Θεωρούνται ατελείωτες ακολουθίες οι οποίες είναι ταξινομημένες χρονικά σε δεδομένα. Στον Σημασιολογικό Ιστό, το RDF Streaming είναι ένα σημαντικό και ενδιαφέρον πεδίο έρευνας το οποίο εστιάζει στην επεξεργασία και αναπαράσταση ροών δεδομένων. Συνδυάζει τις τεχνολογίες του Σημασιολογικού Ιστού με τις δυνατότητες των Data Stream Management Systems (DSMS) για να υπάρχει συνεχή επεξεργασία δεδομένων σε πραγματικό χρόνο. Η κοινότητα του Σημασιολογικού Ιστού έχει δώσει έμφαση στην ανάπτυξη τεχνικών όπως η RDF Stream Processing και η Stream Reasoning. Ωστόσο, υπάρχει έλλειψη μηχανισμών και πρωτοκόλλων για την ανταλλαγή RDF Streams. Η TripleWave είναι μία προσέγγιση ανοιχτού κώδικα για τη δημιουργία RDF Streams στον Παγκόσμιο Ιστό. Διευκολύνει τη διάδοση ροών στο διαδίκτυο, προσφέροντας έτσι ευέλικτες διαμορφώσεις από διάφορους τύπους πηγών δεδομένων και διαφορετικών τρόπων δημιουργίας ροών. Παράδειγμα, η μετατροπή (μέσω χαρτογραφήσεων) των ροών. Με αυτό τον τρόπο διευκολύνεται η διάδοση και η κατανάλωση των ροών αυτών. [9] [12]

Η RDF Streaming χρησιμοποιεί μια μορφή συνεχούς ροής δεδομένων. Τα δεδομένα διατυπώνονται ως ζεύγη τα οποία περιλαμβάνουν μία RDF τριπλέτα όπως και την αντίστοιχη χρονική σήμανση. Αυτό το μοντέλο δίνει την δυνατότητα διαχείρισης των δεδομένων τα οποία εξελίσσονται με το χρόνο, διατηρώντας παράλληλα τους σημασιολογικούς δεσμούς τους. Ένα RDF stream ορίζεται επίσης ως μια χρονικά διατεταγμένη ακολουθία ζευγών, όπου κάθε ζεύγος αποτελείται από μία τριπλέτα RDF και την χρονική του σήμανση. Υπάρχουν διάφοροι τρόποι επεξεργασίας των RDF streams. Οι πιο γνωστές είναι οι SPARQL, C-SPARQL, CQELS και Sparkwave. Κάθε σύστημα παρέχει διαφορετική προσέγγιση για την επεξεργασία των δεδομένων. Η Continuous SPARQL (C-SPARQL) είναι μια γλώσσα που χρησιμοποιείται για συνεχή ερωτήματα πάνω σε ροές δεδομένων RDF. Η C-SPARQL είναι η επέκταση της SPARQL προκειμένου να μπορεί να υποστηρίξει συνεχείς ροές δεδομένων χρησιμοποιώντας παράθυρα (windows). Αυτό έχει ως αποτέλεσμα να παρατηρεί τις πιο πρόσφατες τριπλέτες κατά την διάρκεια της ροής καθώς τα δεδομένα συνεχώς ρέουν. Η C-SPARQL αρχιτεκτονική περιλαμβάνει το "Query

Translator" που μεταφράζει και συνεπάγοντας, διαχειρίζεται τα συνεχόμενα μέρη της ερώτησης. Έπειτα, το "ContinuousEngine" που επεξεργάζεται σε πραγματικό χρόνο τις τριπλέτες RDF και το "SparqlEngine" που εκτελεί την SPARQL ερώτηση. Οι αναβαθμίσεις της C-SPARQL συμπεριλαμβάνουν νέους τελεστές για παράθυρα, επιτρέποντας την περιοδική αξιολόγηση των ερωτημάτων και τη διαχείριση πολλαπλών ροών.

Η RDF Streaming αξιοποιείται σε πολλές εφαρμογές όπως ανάλυση δεδομένων από αισθητήρες σε πραγματικό χρόνο, σε αστικούς υπολογισμούς καθώς και στην κοινωνική σημασιολογία. Ένα παράδειγμα ερωτήματος σε C-SPARQL μπορεί να επιστρέφει κάθε 10 λεπτά, τις μέσες τιμές της πίεσης του νερού που συλλέγονται την τελευταία ώρα. Η ανάγκη για διαχείριση μεγάλων όγκων δεδομένων, οδήγησε στην ανάπτυξη διαφόρων τεχνικών όπως οι μηχανισμοί δειγματοληψίας (sampling). Η προσθήκη των sampling operators στην C-SPARQL επιτρέπει την αφαίρεση των εισερχόμενων δεδομένων, διατηρώντας και παράλληλα τους σημασιολογικούς δεσμούς, βελτιώνοντας με αυτό τον τρόπο την αποδοτικότητα του συστήματος κατά την επεξεργασία υψηλών ρυθμών δεδομένων.

Η σημασία του RDF Streaming είναι μεγάλη καθώς έγκειται στη δυνατότητα του να επεξεργάζεται και να αναλύει σε πραγματικό χρόνο τεράστια σύνολα από δεδομένα, επιτρέποντας την ανάπτυξη και εξέλιξη ευφών εφαρμογών που μπορούν να αντιδρούν ακαριαία σε νέες πληροφορίες. Αυτό το καθιστά ένα σημαντικό και χρήσιμο εργαλείο για τη διαχείριση των συνεχώς αναπτυσσόμενων ροών δεδομένων στον σύγχρονο κόσμο. [10] [11] [12]

2.2 RDF streaming languages

Μία γλώσσα ερωτημάτων που αναπτύχθηκε για την RDF και αποτελεί πρότυπο του W3C είναι η SPARQL. Η γλώσσα αυτή, χειρίζεται και ανακτά τα δεδομένα που αποθηκεύονται σε μορφή τριπλετών. Παρόλα ταύτα, ο σχεδιασμός της SPARQL δεν υποστηρίζει συνεχή ροή δεδομένων, αλλά κυρίως στατικά δεδομένα. Αυτό είναι ένα μειονέκτημα καθώς δεν την καθιστά πρακτική για εφαρμογές που απαιτούν επεξεργασία πραγματικού χρόνου. Προκειμένου να επιλυθεί αυτό το πρόβλημα, αναπτύχθηκε η Continuous SPARQL (C-SPARQL), η οποία είναι μια επέκταση της SPARQL. Είναι σχεδιασμένη για να υποστηρίζει συνεχή ερωτήματα πάνω σε ροές δεδομένων RDF. Χρησιμοποιεί παράθυρα (windows) για να ξεχωρίζει τα δεδομένα που θα συμπεριληφθούν

στα ερωτήματα κατά τη διάρκεια της ροής των δεδομένων. Υπάρχουν όμως και άλλες γλώσσες και συστήματα που έχουν αναπτυχθεί για την επεξεργασία ροών δεδομένων RDF.

Η Continuous Query Evaluation over Linked Streams (CQUELS) είναι μια πανομοιότυπη γλώσσα ερωτημάτων με την C-SPARQL για την επεξεργασία συνεχών ερωτημάτων πάνω σε συνδεδεμένες ροές δεδομένων RDF. Και οι δύο γλώσσες χρησιμοποιούν ένα αποδοτικό μηχανισμό και παράθυρα (windows) για την εκτέλεση ερωτημάτων σε πραγματικό χρόνο. Η διαφορά της CQUELS είναι ότι μπορεί να υποστηρίξει την ενσωμάτωση και επεξεργασία πολύπλοκων ερωτημάτων σε σχέση με την C-SPARQL στην οποία, η αποδοτικότητά της μπορεί να μην είναι τόσο βελτιστοποιημένη για πολύ μεγάλα και σύνθετα δεδομένα. Παρόλο που είναι παρόμοιες, διαφέρουν στην προσέγγιση επίλυσης των ερωτημάτων και στην έμφαση που δίνουν στην αποδοτικότητα και επεκτασιμότητα.

Η RDF Stream Processing Query Language (RSP-QL) είναι μια προτεινόμενη γλώσσα από την κοινότητα W3C για την επεξεργασία ροών RDF. Υποστηρίζει την επεξεργασία δεδομένων μέσω παραθύρων (windows) και επιτρέπει την εκτέλεση ερωτημάτων σε πολλαπλές ροές RDF.

Οι παραπάνω γλώσσες έχουν αναπτυχθεί για να καλύψουν στον Σημασιολογικό Ιστό τις ανάγκες για επεξεργασία και ανάλυση δεδομένων σε πραγματικό χρόνο. Η ραγδαία ανάπτυξη των συγκεκριμένων γλωσσών έχει συμβάλει στην εξέλιξη έξυπνων εφαρμογών επιτρέποντας στην αποδοτική διαχείριση δεδομένων που προκύπτουν από νέα. [10] [12]

2.3 RDF Stream Processing Systems

Η ανάπτυξη των RDF Stream Processing Systems (Συστήματα Επεξεργασίας Ροών RDF) έχει πάρει μεγάλες διαστάσεις για την διαχείριση και επεξεργασία σε πραγματικό χρόνο ροών δεδομένων, αξιοποιώντας γλώσσες και τεχνολογίες του Σημασιολογικού Ιστού. Οι ροές δεδομένων είναι ακολουθίες από γραφήματα ή RDF δηλώσεις τα οποία με τον χρόνο μεταβάλλονται. Αυτά τα συστήματα είναι σημαντικά να χρησιμοποιούνται σε εφαρμογές όπου υπάρχει ταχύτητα και ποικιλία δεδομένων. Η Stream Reasoning κοινότητα, έχει συμβάλλει από το 2008 στην ανάπτυξη αλγορίθμων, μοντέλων δεδομένων και benchmarks για την Επεξεργασία Ροών RDF.

Το RSP-QL είναι ένα μοντέλο που αναπαριστά και επεξεργάζεται τις ροές RDF σε πραγματικό χρόνο σε δυναμικά περιβάλλοντα. Επεκτείνει την γλώσσα ερωτήσεων SPARQL για να επιτρέπει τη διαχείριση και την ανάλυση συνεχώς μεταβαλλόμενων RDF δεδομένων. Το μοντέλο SPARQL επεκτείνει δύο μοντέλα: Το CQL (Continuous Query Language) και το SECRET. Το CQL είναι μια συνεχής αναβάθμιση της SQL η οποία κάνει ερωτήματα σε μεταβαλλόμενα δεδομένα, ενώ το SECRET περιγράφει και αναλύει το τρόπο με τον οποίο τα συστήματα διαχειρίζονται τα παράθυρα δεδομένων. Τα παράθυρα (windows) είναι χωρισμένα σε τμήματα και εξετάζονται για μια συγκεκριμένη χρονική περίοδο.

Παρότι το RSP-QL παρέχει μια σταθερότητα για τις διαδικασίες επεξεργασίας ροών RDF, υπάρχουν αρκετές προκλήσεις. Μία από αυτές είναι η ενσωμάτωση της συλλογιστικής (reasoning) στα RSP συστήματα. Η συλλογιστική (reasoning) είναι η διαδικασία εξαγωγής συμπερασμάτων από δεδομένα. Χρησιμοποιείται για να διεξαχθούν περαιτέρω πληροφορίες που δεν είναι άμεσα εμφανείς από τα δεδομένα. Αυτή η δυνατότητα δεν είναι πλήρως ανεπτυγμένη και εφαρμοσμένη σε όλα τα συστήματα. Όμως υπάρχουν προσπάθειες να συνδυαστούν οι τεχνικές συλλογιστικής με την επεξεργασία ροών. Η συλλογιστική εφαρμόζει λογικούς κανόνες και αξιοποιεί τις υπάρχουσες οντολογίες, ώστε τα συστήματα να έχουν την δυνατότητα να καταλαβαίνουν και να επεξεργάζονται τα δεδομένα με μεγαλύτερη ακρίβεια και σε μεγαλύτερο βάθος.

Για παράδειγμα, το μοντέλο δεν λαμβάνει υπόψη τη διαδικασία εξαγωγής συμπερασμάτων από δεδομένα, κάτι το οποίο αποτελεί μειονέκτημα για την χρήση σημασιολογικών μοντέλων. Επίσης, τα IoT δεδομένα (Internet of Things) περιέχουν συχνά ανακριβή ή με σφάλματα δεδομένα, τα οποία μπορεί να επηρεάσουν την ορθότητα, την ποιότητα και την αξιοπιστία των αποτελεσμάτων από την επεξεργασία τους. [13] [14]

Κεφάλαιο 3: State of the art

Τα RDF είναι ένας τομέας που πλέον εξελίσσεται ραγδαία με πολλές ερευνητικές προσπάθειες και μελέτες να επικεντρώνονται στη βελτίωση της ακρίβειας και της αποδοτικότητας RDF Stream Processing συστημάτων (RSP). Ξεκινώντας από το 2008, έχουν δημιουργηθεί νέοι αλγόριθμοι, μοντέλα δεδομένων και benchmarks τα οποία έχουν συμβάλει στην βελτίωση επεξεργασιών και ανάλυσης των ροών δεδομένων. Οι νέοι αλγόριθμοι επικεντρώνονται στη βελτίωση της αποδοτικότητας, με σκοπό την ταχύτερη επεξεργασία τεράστιων όγκων δεδομένων σε πραγματικό χρόνο. Οι αλγόριθμοι αυτοί χρησιμοποιούν τεχνικές όπως τη παράλληλη επεξεργασία και τη προσαρμογή των παραθύρων (windows). Διάφορες γλώσσες επεξεργασίας ροών δεδομένων όπως η RSP-QL, έχουν επεκταθεί και εξελιχθεί για να μπορούν να υποστηρίξουν πιο σύνθετα ερωτήματα και να παρέχουν μεγαλύτερη ακρίβεια στην ανάλυση των δεδομένων.

Το C-SPARQL εξακολουθεί να είναι ένα από τα πιο δημοφιλή συστήματα RSP, με συνεχείς και περαιτέρω βελτιώσεις στην ευελιξία και την αποδοτικότητά του. Οι πιο πρόσφατες εκδόσεις του C-SPARQL έχουν ενσωματώσει τέτοιες δυνατότητες, που βελτιστοποιούν τη διαχείριση των παραθύρων δεδομένων και την υποστήριξη για πιο σύνθετα ερωτήματα. Ακόμα μερικά σημαντικά συστήματα στον τομέα της επεξεργασίας ροών RDF είναι το SPARQLStream και το CQELS. Προσφέρουν καινοτόμες προσεγγίσεις στην αξιολόγηση των ερωτημάτων σε πραγματικό χρόνο. Σε αυτά τα συστήματα χρησιμοποιούνται προηγμένες τεχνικές για την βελτίωση της απόδοσης.

Παρά το γεγονός ότι έχει χαράξει σημαντική πορεία η πρόοδος αυτού του τομέα, η έρευνα στην επεξεργασία ροών RDF αντιμετωπίζει κάποιες προκλήσεις. Η ενσωμάτωση λογικών κανόνων στα συστήματα για διεξαγωγή περαιτέρω πληροφοριών που δεν είναι άμεσα εμφανείς από τα δεδομένα (συλλογιστική), παραμένει ένα αναπάντητο ζήτημα. Η συλλογιστική μπορεί να βελτιώσει την ακρίβεια και την ποιότητα των αποτελεσμάτων. Επίσης, η διαχείριση δεδομένων με ανακρίβειες και σφάλματα, είναι μια άλλη σημαντική πρόκληση. Τα δεδομένα από IoT συσκευές είναι συχνά με σφάλματα, και η ανάπτυξη τεχνικών για τον φιλτράρισμα των δεδομένων είναι κρίσιμη για την βελτίωση της ποιότητας. Τέλος, τα benchmarks αποτελούν σημαντικά εργαλεία για την εξέλιξη της έρευνας και τη βελτίωση των συστημάτων. [13] [14]

3.1 Benchmarks

Τα benchmarks θεωρούνται σημαντικό εργαλείο για την αξιολόγηση διαφόρων συστημάτων και τεχνολογιών σύμφωνα με συγκεκριμένα χαρακτηριστικά, όπως αξιοπιστία, απόδοση, ασφάλεια. Είναι τυποποιημένες δοκιμές που σκοπός τους είναι να μετρούν τις επιδόσεις ενός συστήματος κάτω από συγκεκριμένες συνθήκες. Τα benchmarks ξεκίνησαν από την ανάγκη των επιστημόνων ώστε να έχουν έναν αντικειμενικό και σαφή τρόπο σύγκρισης διαφόρων τεχνολογιών. Στόχος είναι, να κατανοήσουν τις δυνατότητες και τις αδυναμίες τους. Ένα από τα πιο γνωστά benchmarks είναι το TPC-C, το οποίο αξιοποιείται για την αξιολόγηση της απόδοσης των συστημάτων που διαχειρίζονται βάσεις δεδομένων. Υπάρχουν διάφοροι τύποι των Benchmarks. Μερικοί από αυτούς είναι οι εξής:

→ **Specification-based Benchmarks:** Περιγράφουν όλες τις λειτουργίες που πρέπει να πραγματοποιηθούν, τις απαραίτητες παραμέτρους εισόδου καθώς και τα αποτελέσματα. Η υλοποίηση και ανάπτυξη αφήνεται στην ευχέρεια του χρήστη που εκτελεί το benchmark.

→ **Kit-based Benchmarks:** Οι προδιαγραφές και η υλοποίηση του benchmark θεωρούνται αναγκαίο μέρος της εκτέλεσης. Ο χρήστης δεν μπορεί να τροποποιήσει ή να επεξεργαστεί την εκτέλεση του benchmark.

→ **Hybrid Benchmarks:** Τα hybrid benchmarks συνδυάζουν στοιχεία από τα δύο παραπάνω είδη των benchmarks. Επιτρέπεται στον χρήστη να αναπτύξει και να τροποποιήσει μερικές λειτουργίες, όμως παρέχονται και συγκεκριμένες υλοποιήσεις που δεν μπορεί να τις αλλάξει σε κάποια κρίσιμα τμήματα της αξιολόγησης.

Για να δημιουργηθεί και να αναπτυχθεί σωστά ένα benchmark, πρέπει για αρχή, να καθοριστούν οι μεθοδολογίες που θα χρησιμοποιηθούν για την αξιολόγηση των συστημάτων. Έπειτα πρέπει να υπολογιστεί το workload προκειμένου να επιτευχθεί μία ισορροπία μεταξύ διαφόρων κριτηρίων σχεδίασης. Στο τελευταίο στάδιο, πραγματοποιούνται αυστηρές δοκιμές. Μέσω των επαναλαμβανόμενων δοκιμών, τα αποτελέσματα μπορούν να θεωρηθούν αξιόπιστα και τα benchmarks να γίνουν αποδεκτά για τυποποίηση.

Σκοπός των benchmarks είναι η ακριβή επαλήθευση, επιτρέποντας στους χρήστες να επιβεβαιώνουν την ορθότητα των αποτελεσμάτων. Επίσης, η ευχρηστία αποτελεί σημαντικό

παράγοντα διότι επιτρέπει την εύκολη εκτέλεση των δοκιμών στα περιβάλλοντα των χρηστών. [15]

3.2 RDF Stream Processing Benchmarks

Τα RDF Stream Processing Benchmarks θεωρούνται σύνολα μετρήσεων που σκοπό έχουν να αξιολογούν τις επιδόσεις των μηχανών επεξεργασίας RDF ροών και ερωτημάτων SPARQL. Σχεδιάστηκαν προκειμένου να βοηθούν τους ερευνητές να συγκρίνουν διαφορετικά RDF συστήματα σε τυποποιημένα σενάρια εφαρμογών, προκειμένου να καταλάβουν τις διαφορετικές δυνατότητές τους σε ποικίλες περιπτώσεις χρήσης. Η ανάπτυξη των RDF Stream Processing Benchmarks ξεκίνησε λόγω της ανάγκης να αξιολογούνται οι αποδόσεις των συστημάτων που διαχειρίζονται συνεχείς ροές δεδομένων. Σε αντίθεση με τα παραδοσιακά συστήματα διαχείρισης βάσεων δεδομένων, τα οποία δεν είχαν την δυνατότητα να επεξεργάζονται στατικά δεδομένα. Κάποια από τα πιο ευρέως χρησιμοποιούμενα RDF Benchmarks είναι:

→ **Lehigh University Benchmark (LUBM)**: Σχεδιάστηκε για να αξιολογήσει τις δυνατότητες λογικής και τους μηχανισμούς των συστημάτων βάσεων γνώσεων OWL. Βασίζεται σε μια υποθετική πανεπιστημιακή δομή και εξάγει μια σειρά από ερωτήματα που καλύπτουν διάφορες πτυχές της πανεπιστημιακής ζωής και ελέγχουν τις επιδόσεις και την επεκτασιμότητα των RDF.

→ **Berlin SPARQL Benchmark (BSBM)**: Χρησιμοποιείται κυρίως για την αξιολόγηση των επιδόσεων των RDF δεδομένων με επίκεντρο το ηλεκτρονικό εμπόριο. Προσφέρει μια σειρά από ερωτήματα που θυμίζουν προσομοίωση περιβάλλοντος ηλεκτρονικού εμπορίου. Τα δεδομένα μπορεί να είναι προϊόντα, παραγωγοί, καταναλωτές.

→ **SPARQL Performance Benchmark (SP2Bench)**: Σχεδιάστηκε βάσει των δεδομένων από την DBLP για την αξιολόγηση των επιδόσεων από SPARQL ερωτήματα. Περιέχει ένα σύνολο ερωτημάτων που αντικατοπτρίζουν πραγματικά σενάρια χρήσης.

Το **SRBench** είναι το πρώτο benchmark γενικού σκοπού που δημιουργήθηκε για την αξιολόγηση της απόδοσης των συστημάτων RSP ροών RDF/SPARQL δίνοντας έμφαση στη διαχείριση συνεχών δεδομένων εκτελώντας σε πραγματικό χρόνο ερωτήματα SPARQL. Χρησιμοποιεί αντιπροσωπευτικά δεδομένα σεναρίων που αντικατοπτρίζουν την

πραγματικότητα, όπως δεδομένα από αισθητήρες ή κοινωνικά δίκτυα. Η απόδοση υπολογίζεται σύμφωνα με τον χρόνο απόκρισης και την ορθότητα των αποτελεσμάτων. Το SRBench βασίζεται σε μια μακροχρόνια μελέτη στον τομέα DSMS, στα συστήματα strRS καθώς και στα σύνολα δεδομένων από το Linked Open Data cloud. Έτσι διασφαλίζεται ότι καλύπτει όλα τα σημαντικά χαρακτηριστικά της επεξεργασίας strRS. Τα RDF benchmarks, περιέχουν ένα επιγραμματικό αλλά πλήρες σύνολο ερωτημάτων το οποίο καλύπτει όλες τις βασικές αρχές επεξεργασίας ερωτημάτων strRS. Μπορεί να ικανοποιήσει απλά ερωτήματα αντιστοίχισης γραφημάτων μέχρι και ερωτήματα που απαιτείται λογική επεξεργασία σε πολλαπλά συνδεδεμένα σύνολα δεδομένων.

Το **SLD-Bench** είναι ένα άλλο benchmark που χρησιμοποιείται για την αξιολόγηση των RSP συστημάτων. Το συγκεκριμένο, δίνει προτεραιότητα στην επεξεργασία και ενσωμάτωση συνδεδεμένων δεδομένων με περιεχόμενο σημασιολογικό. Κάποια από τα κύρια χαρακτηριστικά του είναι η εστίαση στην ικανότητα των συστημάτων να ενοποιούν και να επεξεργάζονται δεδομένα από ποικίλες πηγές που είναι σημασιολογικά συνδεδεμένα. Επίσης, δοκιμάζει το πόσο είναι ικανά τα συστήματα να εκτελούν σύνθετα ερωτήματα που απαιτούν περίπλοκη λογική και συμπερασματολογία.

Το **TPC-H** benchmark είναι και αυτό ένα από τα πιο διαδεδομένα και σημαντικά benchmarks για την αξιολόγηση της απόδοσης συστημάτων διαχείρισης βάσεων δεδομένων. Δημιουργήθηκε από την Transaction Processing Performance Council (TPC). Η TPC είναι ένας οργανισμός που δημιουργεί πρότυπα benchmarks. Και αυτό με την σειρά του αποσκοπεί στην βέλτιστη αξιολόγηση της απόδοσης των συστημάτων επεξεργασίας διαχείρισης βάσεων δεδομένων.

Η αξιολόγηση των RSP συστημάτων παρουσιάζει αρκετά προβλήματα και εμπόδια στην επεξεργασία των δεδομένων εξαιτίας της πολυπλοκότητας τους. Μερικές από τις κύριες προκλήσεις είναι, η διαχείριση του τεράστιου όγκου δεδομένων, η χρονική επικαιρότητα και η ορθότητα των αποτελεσμάτων. Προκειμένου να αντιμετωπιστούν και να περιοριστούν οι υπάρχοντες προκλήσεις, προτείνεται από τους ειδικούς κάποιοι τρόποι αντιμετώπισης. Πρωτίστως, θα πρέπει να προσδιορίζονται οι βασικοί δείκτες απόδοσης (KPIs), δηλαδή ο χρόνος απόκρισης των ερωτημάτων, η πληρότητα και η ορθότητα των αποτελεσμάτων. Να χρησιμοποιούνται σενάρια που ανταποκρίνονται στην πραγματικότητα των καταστάσεων των εφαρμογών και χρήσεων των RSP συστημάτων. Τέλος, τα benchmarks να

αναπτύσσονται βάσει συγκριτικών αξιολογήσεων από διάφορα συστήματα, προκειμένου να αναδεικνύονται τα ισχυρά και τα αδύνατα σημεία τους.

Εν κατακλείδι, η ανάπτυξη των RDF Stream Processing Benchmarks είναι σημαντική και απαραίτητη για την εξέλιξη της τεχνολογίας στον κλάδο της επεξεργασίας ροών δεδομένων RDF. Τα benchmarks παρέχουν ένα μεγάλο οπλοστάσιο εργαλείων και μεθόδων για την αντικειμενική αξιολόγηση των επιδόσεων των συστημάτων. Έτσι διασφαλίζεται ότι μπορούν να ανταπεξέλθουν στις όλο και αυξανόμενες απαιτήσεις των σύγχρονων εφαρμογών. Συνολικά, τα RDF Stream Processing Benchmarks αποτελούν κομβικό σημείο για την επιστημονική πρόοδο και για ένα πληροφοριακά πλούσιο μέλλον. [16] [17]

3.3 Choke points and their meaning

Η έννοια των choke points ή αλλιώς ‘σημεία συμφόρησης’, αρχικά ξεκίνησε με το TPC-H benchmark, στο οποίο αναλύθηκαν 28 διαφορετικά choke points. Ο λόγος αυτής της ανάλυσης ήταν για να κατανοηθούν και να αντιμετωπιστούν διάφορα τεχνικά θέματα και προκλήσεις που αφορούσαν την απόδοση των συστημάτων διαχείρισης βάσεων δεδομένων. Τα choke points παριστούν σημαντικό παράγοντα σε ένα benchmark, καθώς προσδιορίζουν με ακρίβεια τα κρίσιμα σημεία που πρέπει να αντιμετωπιστούν και να εξαλειφθούν για την ορθή και αποτελεσματική λειτουργία του συστήματος. Η κάλυψη αυτών των προκλήσεων επιτυγχάνεται μέσω ενός περιορισμένου αριθμού ερωτήσεων στο workload. [18]

Οι έρευνες όλων αυτών των χρόνων έχουν επικεντρωθεί στον προσδιορισμό ποικίλων μεθοδολογιών για την αξιολόγηση της απόδοσης των συστημάτων RSP (Επεξεργασίας Ροών RDF), εντοπίζοντας τις κύριες προκλήσεις που πρέπει να αντιμετωπίζουν τα προτεινόμενα benchmarks. Εκτός αυτών, έχει θεσπιστεί ένα σύνολο βασικών δεικτών απόδοσης (KPIs) που πρέπει να μειωθούν. Βάσει αυτών των μεθόδων, παρουσιάζονται διάφορες διαστάσεις στις οποίες επικεντρώνονται τα υπάρχοντα benchmarks. Ο συνδυασμός ροών και στατικών δεδομένων έχει να κάνει με την ικανότητα του συστήματος να συνδυάζει στατικά και ροών δεδομένα. Με αυτό το τρόπο, παρουσιάζεται ταχεία επίδραση όσον αφορά την στρατηγική επεξεργασίας, καθώς τα στατικά δεδομένα πρέπει να είναι άμεσα προσβάσιμα και να συνδυάζονται γρήγορα με τις ροές δεδομένων. Η υποστήριξη λογικών συμπερασμάτων γίνεται πιο περίπλοκη και δύσκολη σε σενάρια με ροές δεδομένων, καθώς τα δεδομένα αλλάζουν με ραγδαίους ρυθμούς και πρέπει να

συνδυάζονται με τους κανόνες. Επιπρόσθετα, πρέπει να εφαρμόζονται σημασιολογικές επισημάνσεις στις ροές δεδομένων. Μία άλλη επίδραση που αφορά την απόδοση του συστήματος είναι το επιλεγμένο μοντέλο χρόνου. Οι σημασιολογίες που βασίζονται σε χρονικά διαστήματα είναι πιο δύσκολο να τις διαχειριστεί κανείς σε αντίθεση με τις σημασιολογίες που βασίζονται σε συγκεκριμένα σημεία χρόνου. Για την αποτελεσματική αντιμετώπιση της επεξεργασίας των συνεχών ερωτημάτων, πρέπει να επιλεγεί μια πρότυπη προσέγγιση για τη διαχείριση και οργάνωση των αποτελεσμάτων. Επίσης, μια πρόκληση που θεωρείται σημαντική, είναι στο να δοθεί προτεραιότητα και να εντοπιστούν τα σχετικά στοιχεία εντός της ροής δεδομένων για επεξεργασία. Η χρήση υπερβολικά ευρέων παραθύρων, μπορούν να έχουν αρνητικές επιπτώσεις στην απόδοση του συστήματος και την ποιότητα των παρεχόμενων υπηρεσιών με αποτέλεσμα να επιφέρουν αρνητικά αποτελέσματα.

Ένα σύστημα πρέπει να διατηρεί την ποιότητα, όταν υπάρχει μεγάλη ή απρόσμενη αύξηση στο ρυθμό εισόδου των δεδομένων. Αυτό το πρόβλημα είναι ιδιαίτερα σημαντικό για τα συστήματα επεξεργασίας ροών δεδομένων (RSP), τα οποία είναι σε μόνιμη λειτουργία. Ωστόσο, ο ρυθμός και το πλήθος των δεδομένων μπορεί να είναι απρόβλεπτος, όπως για παράδειγμα ο αριθμός των δημοσιεύσεων στα μέσα μαζικής ενημέρωσης, όπου ποικίλλει ανάλογα με τη δημοτικότητα ενός θέματος. Η διαχείριση αυτών των απρόβλεπτων αυξήσεων είναι δύσκολη και απαιτεί προσεκτικό σχεδιασμό των εργασιών επεξεργασίας. Παρακάτω γίνεται μία λεπτομερής ανάλυση για τα choke points όπως αναφέρονται στο άρθρο [19]:

- **CP1. Join Ordering:** Εξετάζει εάν το σύστημα μπορεί να αξιολογήσει τις αντισταθμίσεις ανάμεσα στο χρόνο που δαπανάται για την εύρεση της καλύτερης εκτέλεσης καθώς και της ποιότητας του τελικού αποτελέσματος. Η διάταξη των συζεύξεων είναι η σειρά με την οποία η μηχανή επεξεργάζεται τα διαφορετικά μέρη ενός ερωτήματος. Η σωστή διάταξη μπορεί να βελτιώσει σημαντικά την αποδοτικότητα.
- **CP2. Aggregation:** Τα aggregations υλοποιούνται με τη χρήση υπο-επιλογών στο ερώτημα SPARQL. Αφορούν τη συλλογή και συνοπτική παρουσίαση δεδομένων, όπως π.χ. ο υπολογισμός του μέσου όρου, εύρεση ενός συνόλου ή εύρεση μεγίστου. Το CP2 δοκιμάζει την ικανότητα της μηχανής να αναγνωρίζει και να επεξεργάζεται πρώτα αυτές τις λειτουργίες ώστε να βελτιστοποιήσει την απόδοση του ερωτήματος.

- **CP3. Optional & nested optional clauses:** Τα optional clauses σε ένα ερώτημα SPARQL επιτρέπουν την επιστροφή επιπλέον πληροφοριών αν αυτές είναι διαθέσιμες, χωρίς να είναι απαραίτητες για το αποτέλεσμα. Όταν εκτελείται ένα ερώτημα, η μηχανή πρέπει να αποφασίσει σε ποια σειρά θα εκτελέσει τα διάφορα μέρη του ερωτήματος. Το προαιρετικό φιλτράρισμα δεν μειώνει τον αριθμό των ενδιάμεσων αποτελεσμάτων που προκύπτουν από την επεξεργασία του βασικού ερωτήματος.
- **CP4. Reasoning:** Δοκιμάζει την ικανότητα του συστήματος να διαχειρίζεται αποτελεσματικά τις κατασκευές RDFS και OWL που εκφράζονται στο σχήμα. Το Reasoning αναφέρεται στην διαδικασία λογικής εξαγωγής νέας πληροφορίας από τα υπάρχοντα δεδομένα και κανόνες.
- **CP5. Parallel execution of joins:** Η παράλληλη εκτέλεση σημαίνει ότι το σύστημα μπορεί να επεξεργαστεί πολλές ενώσεις ταυτόχρονα. Αυτό βοηθάει στην ταχύτερη επεξεργασία των δεδομένων και στη βελτίωση της συνολικής απόδοσης του συστήματος. Το CP5 ελέγχει αν η μηχανή μπορεί να δημιουργήσει σχέδια εκτέλεσης που επιτρέπουν αυτή την παράλληλη επεξεργασία.
- **CP6. Filters:** Δοκιμάζει την ικανότητα των συστημάτων να εκτελούν όσο το δυνατόν νωρίτερα τα φίλτρα, ώστε να εξαλείψουν ένα ενδεχομένως μεγάλο πλήθος ενδιάμεσων αποτελεσμάτων. Τα φίλτρα χρησιμοποιούνται για να περιορίσουν τα δεδομένα που επεξεργάζεται η μηχανή, κρατώντας μόνο τα στοιχεία που πληρούν συγκεκριμένα κριτήρια και είναι απαραίτητα. Το CP6 ελέγχει αν η μηχανή μπορεί να εφαρμόσει αυτά τα φίλτρα όσο το δυνατόν νωρίτερα στη διαδικασία επεξεργασίας, ώστε να μειώσει τον όγκο των δεδομένων που χρειάζεται να επεξεργαστεί στα επόμενα βήματα. Αυτό βελτιώνει την απόδοση και μειώνει τον χρόνο επεξεργασίας.
- **CP7. Ordering:** Δοκιμάζει την ικανότητα του συστήματος να επιλέγει ερωτήματα που διευκολύνουν την ταξινόμηση των αποτελεσμάτων. Το CP7 ελέγχει αν η μηχανή μπορεί να επιλέξει στρατηγικές επεξεργασίας που καθιστούν την ταξινόμηση πιο αποτελεσματική. Αυτό σημαίνει ότι η μηχανή πρέπει να μπορεί να οργανώσει την εκτέλεση των ερωτημάτων με τέτοιο τρόπο ώστε η ταξινόμηση των τελικών αποτελεσμάτων να είναι γρήγορη και αποδοτική.

- **CP8. Geo-spatial predicates:** Δοκιμάζει την ικανότητα του συστήματος να χειρίζεται ερωτήματα για γεωχωρικά δεδομένα. Τα γεωχωρικά δεδομένα αφορούν πληροφορίες που έχουν να κάνουν με τοποθεσίες, όπως συντεταγμένες GPS ή χάρτες. Το CP8 ελέγχει αν το σύστημα μπορεί να επεξεργαστεί σωστά ερωτήματα που περιλαμβάνουν αυτές τις πληροφορίες, όπως την αναζήτηση σημείων σε συγκεκριμένη γεωγραφική περιοχή.
- **CP9. Full-text:** Η αναζήτηση πλήρους κειμένου αφορά την εύρεση συγκεκριμένων λέξεων ή φράσεων μέσα σε κείμενα. Το CP9 ελέγχει αν η μηχανή μπορεί να επεξεργαστεί ερωτήματα που περιλαμβάνουν κανονικές εκφράσεις π.χ. αναζήτηση για λέξεις που ταιριάζουν με ένα συγκεκριμένο μοτίβο στα δεδομένα που αποθηκεύει.
- **CP10. Duplicate elimination:** Δοκιμάζει την ικανότητα του συστήματος να εντοπίζει και να εξαλείφει διπλότυπες καταχωρήσεις κατά τη δημιουργία ενδιάμεσων αποτελεσμάτων. Κατά την επεξεργασία δεδομένων, μπορεί να εμφανιστούν διπλότυπες εγγραφές. Το CP10 ελέγχει αν το σύστημα μπορεί να αναγνωρίσει και να αφαιρέσει αυτές τις διπλότυπες καταχωρήσεις, εξασφαλίζοντας ότι τα τελικά αποτελέσματα είναι μοναδικά και χωρίς περιττά αντίγραφα.
- **CP11. Complex filter conditions:** Δοκιμάζει την ικανότητα της μηχανής να διαχειρίζεται αποδοτικά αρνήσεις, συνδυασμούς και διαζεύξεις. Σύνθετες συνθήκες φιλτραρίσματος περιλαμβάνουν λογικές πράξεις όπως η άρνηση (NOT), ο συνδυασμός (AND) και η διαζευκτική σύνδεση (OR). Το CP11 ελέγχει αν η μηχανή μπορεί να χειριστεί αυτές τις σύνθετες συνθήκες με αποδοτικό τρόπο. Για παράδειγμα, αν μπορεί να διασπάσει ένα σύνθετο φίλτρο σε μικρότερα, ανεξάρτητα φίλτρα και να τα εκτελέσει παράλληλα, βελτιώνοντας την απόδοση και την ταχύτητα επεξεργασίας των δεδομένων. [19]

3.4 Comparison of existing benchmarks based on choke points

Στο παρόν υποκεφάλαιο θα γίνει μία εκτενής ανάλυση και σύγκριση των choke points σε σύγκριση με τα υπάρχοντα RDF Benchmarks. Τα choke points αποτελούν σημαντικά σημεία για την απόδοση των συστημάτων επεξεργασίας δεδομένων. Η αξιολόγηση των RDF benchmarks απεικονίζει πόσο αποτελεσματικά είναι αυτά τα συστήματα στην αντιμετώπιση των διαφόρων προκλήσεων. Η παρακάτω ανάλυση, θα εστιάσει στη σε κάθε

benchmark εάν ικανοποιεί τα έντεκα προσδιορισμένα choke points που έχουν αναφερθεί στο προηγούμενο υποκεφάλαιο, προκειμένου να διαπιστωθεί η επάρκεια και η αποδοτικότητα των συστημάτων επεξεργασίας RDF δεδομένων.

Lehigh University Benchmark (LUBM):

Σχεδιάστηκε για να αξιολογήσει τις δυνατότητες λογικής και τους μηχανισμούς των συστημάτων βάσεων γνώσεων OWL. Βασίζεται σε μια υποθετική πανεπιστημιακή δομή και εξάγει μια σειρά από ερωτήματα που καλύπτουν διάφορες πτυχές της πανεπιστημιακής ζωής και ελέγχουν τις επιδόσεις και την επεκτασιμότητα των RDF.

- CP1. Join Ordering:

Το LUBM αποτελείται από σύνθετα ερωτήματα που απαιτούν την σωστή διάταξη των συζεύξεων προκειμένου να βελτιωθεί η απόδοση του συστήματος. Ως εκ τούτου, ικανοποιείται το συγκεκριμένο choke point.

- CP2. Aggregation:

Τα ερωτήματα στο LUBM δεν δίνουν έμφαση σε aggregations όπως μετρήσεις αθροισμάτων ή μέτρηση συνόλου. Επομένως, δεν ικανοποιείται πλήρως αυτό το choke point.

- CP3. Optional & Nested Optional Clauses:

Το LUBM δεν επικεντρώνεται στη χρήση προαιρετικών και ένθετων προαιρετικών συνθηκών/περιορισμών στις ερωτήσεις του. Ενδέχεται να περιέχονται κάποια τέτοια στοιχεία, όμως δεν είναι βασικός στόχος του benchmark. Συνεπώς, δεν ικανοποιείται πλήρως αυτό το choke point.

- CP4. Reasoning:

Το LUBM έχει σχεδιαστεί ώστε να αξιολογεί τις δυνατότητες reasoning των συστημάτων OWL, καθιστώντας το πλήρως κατάλληλο για τη δοκιμή της αποδοτικότητας στη διαχείριση των RDFS και OWL constructs. Συνεπώς, το LUBM ικανοποιεί το συγκεκριμένο choke point.

- CP5. Parallel Execution of Joins:

Το LUBM δεν δίνει έμφαση στην εκτέλεση ενώσεων. Παρόλο που έχει την δυνατότητα να δοκιμάζει την ικανότητα του συστήματος να αντιμετωπίζει σύνθετα ερωτήματα, δεν σχεδιάστηκε για την αξιολόγηση της παράλληλης εκτέλεσης ενώσεων. Επομένως, δεν ικανοποιεί πλήρως αυτό το choke point.

- CP6. Filters:

Τα queries του LUBM ενσωματώνουν διάφορα φίλτρα, αλλά δεν εκτελούνται όσο το δυνατόν νωρίτερα για την εξάλειψη μεγάλου αριθμού ενδιάμεσων αποτελεσμάτων. Συνεπώς, το LUBM δεν ικανοποιεί πλήρως αυτό το choke point.

- CP7. Ordering:

Το LUBM δεν έχει σχεδιαστεί για την ταξινόμηση των αποτελεσμάτων. Επομένως, δεν ικανοποιείται αυτό το choke point.

- CP8. Geo-spatial Predicates:

Το LUBM δεν δίνει καθόλου έμφαση στη διαχείριση γεωχωρικών δεδομένων όπως και δεν περιλαμβάνονται ερωτήματα που τα απαιτούν. Συνεπώς, δεν ικανοποιείται καθόλου το συγκεκριμένο choke point.

- CP9. Full-text:

Το LUBM δεν περιέχει ερωτήματα που απαιτούν κείμενο αναζητήσεων ή μεμονωμένες λέξεις στα δεδομένα. Επομένως, δεν ικανοποιείται καθόλου αυτό το choke point.

- CP10. Duplicate Elimination:

Η εξάλειψη διπλότυπων δεν είναι μία από τις προτεραιότητες του LUBM. Ενδέχεται να εμφανιστεί ως ζητούμενο σε κάποια ερωτήματα, όμως δεν αποτελεί κεντρικό σημείο του benchmark. Συνεπώς, δεν ικανοποιείται πλήρως αυτό το choke point.

- CP11. Complex Filter Conditions:

Τα queries του LUBM δεν εστιάζουν στη αντιμετώπιση πολύπλοκων συνθηκών φιλτραρίσματος, όπως σύζευξη, διάζευξη και άρνηση. Επομένως, το LUBM δεν ικανοποιεί αυτό το choke point.

→ Συνοπτικά, το LUBM δεν καλύπτει πλήρως πολλά από τα choke points που σχετίζονται με τη σύνθετη και αποδοτική επεξεργασία δεδομένων σε πραγματικό χρόνο. [20]

Berlin SPARQL Benchmark (BSBM):

Χρησιμοποιείται κυρίως για την αξιολόγηση των επιδόσεων των RDF δεδομένων με επίκεντρο το ηλεκτρονικό εμπόριο. Προσφέρει μια σειρά από ερωτήματα που θυμίζουν προσομοίωση περιβάλλοντος ηλεκτρονικού εμπορίου. Τα δεδομένα μπορεί να είναι προϊόντα, παραγωγοί, καταναλωτές.

- CP1. Join Ordering:

Το BSBM αξιολογεί την ικανότητα του συστήματος να βελτιστοποιεί τη σειρά εκτέλεσης των joins που απαιτούν περίπλοκες συνδέσεις μεταξύ διαφόρων οντοτήτων, όπως προϊόντα και κατασκευαστές. Συνεπώς ικανοποιείται το πρώτο choke point.

- CP2. Aggregation:

Τα ερωτήματα του BSBM περιέχουν συχνά aggregations, όπως υπολογισμούς μέσω όρων ή αθροισμάτων για αξιολογήσεις διαφόρων προϊόντων. Επομένως ικανοποιείται το συγκεκριμένο choke point.

- CP3. Optional & Nested Optional Clauses:

Τα ερωτήματα του BSBM περιλαμβάνουν optional clauses για να αντιμετωπίσουν περιπτώσεις όπου τα δεδομένα μπορεί να μην υπάρχουν. Συνεπώς ικανοποιείται αυτό το choke point.

- CP4. Reasoning:

Παρόλο που περιλαμβάνει queries που απαιτούν reasoning, όπως η ιεραρχία κατηγοριοποίησης προϊόντων, δεν επικεντρώνεται πολύ σε RDFS και OWL constructs. Επομένως το συγκεκριμένο choke point δεν ικανοποιείται πλήρως.

- CP5. Parallel Execution of Joins:

Το BSBM αξιολογεί τη δυνατότητα του συστήματος να εκτελεί παράλληλες ενώσεις (joins) μέσω ερωτημάτων, βελτιώνοντας έτσι την αποδοτικότητα. Συνεπώς ικανοποιείται το συγκεκριμένο choke point.

- CP6. Filters:

Περιλαμβάνει ερωτήματα με σύνθετα φίλτρα που εκτελούνται το συντομότερο δυνατό για να ελλατώσουν τον όγκο των ενδιάμεσων αποτελεσμάτων. Επομένως, ικανοποιείται αυτό το choke point.

- CP7. Ordering:

Το BSBM περιλαμβάνει τέτοιου είδους ερωτήματα, αξιολογώντας την ικανότητα του συστήματος να ταξινομεί τα αποτελέσματα.

- CP8. Geo-spatial Predicates:

Δεν περιέχει σε γεωχωρικά δεδομένα, επομένως δεν δοκιμάζει συγκεκριμένα την ικανότητα του συστήματος να διαχειρίζεται τέτοιου είδους δεδομένα. Συνεπώς δεν ικανοποιείται το συγκεκριμένο choke point.

- CP9. Full-text:

Δεν περιλαμβάνονται ερωτήματα που αξιολογούν τη δυνατότητα αναζήτησης πλήρους κειμένου ή λέξεων. Επομένως δεν ικανοποιείται το συγκεκριμένο choke point.

- CP10. Duplicate Elimination:

Το BSBM περιλαμβάνει ερωτήματα που αφαιρούν τα διπλότυπα αποτελέσματα. Συνεπώς ικανοποιείται αυτό το choke point.

- CP11. Complex Filter Conditions:

Εστιάζει στη αντιμετώπιση πολύπλοκων συνθηκών φιλτραρίσματος, όπως σύζευξη, διάζευξη και άρνηση. Επομένως ικανοποιείται το συγκεκριμένο choke point.

→ Συμπερασματικά, το BSBM καλύπτει τα περισσότερα choke points. Ωστόσο, δεν εστιάζει σε γεωχωρικά δεδομένα ή αναζήτηση κειμένων, περιορίζοντας έτσι την αξιολόγηση σε αυτά τα συγκεκριμένα σημεία. [21]

SPARQL Performance Benchmark (SP2Bench):

Σχεδιάστηκε βάσει των δεδομένων από την DBLP για την αξιολόγηση των επιδόσεων από SPARQL ερωτήματα. Περιέχει ένα σύνολο ερωτημάτων που αντικατοπτρίζουν πραγματικά σενάρια χρήσης.

- CP1. Join Ordering:

Το SP2Bench περιέχει ερωτήματα με σύνθετα joins, αξιολογώντας το σύστημα να βελτιστοποιεί τη σειρά εκτέλεσης των συνδέσεων για καλύτερη αποδοτικότητα. Συνεπώς, ικανοποιεί το συγκεκριμένο choke point.

- CP2. Aggregation:

Τα ερωτήματα στο SP2Bench δίνουν έμφαση σε aggregations όπως μετρήσεις αθροισμάτων ή μέτρηση συνόλου. Επομένως, ικανοποιείται πλήρως αυτό το choke point.

- CP3. Optional & Nested Optional Clauses:

Το SP2Bench ικανοποιεί το συγκεκριμένο choke point καθώς περιέχει ερωτήματα που απαιτούν από τα συστήματα να χειρίζονται προαιρετικές και ένθετες προαιρετικές συνθήκες/περιορισμούς.

- CP4. Reasoning:

Το SP2Bench δεν έχει σχεδιαστεί ώστε να αξιολογεί τις δυνατότητες reasoning των συστημάτων. Συνεπώς δεν ικανοποιεί πλήρως το συγκεκριμένο choke point.

- CP5. Parallel Execution of Joins:

Έχει σχεδιαστεί ώστε να δοκιμάζει την ικανότητα του συστήματος να εκτελεί παράλληλες συνδέσεις, αυξάνοντας την αποδοτικότητα της επεξεργασίας ερωτημάτων. Επομένως ικανοποιεί αυτό το choke point.

- CP6. Filters:

Τα queries του SP2Bench ενσωματώνουν διάφορα φίλτρα και εκτελούνται όσο το δυνατόν νωρίτερα για την εξάλειψη μεγάλου αριθμού ενδιάμεσων αποτελεσμάτων. Συνεπώς, ικανοποιεί πλήρως αυτό το choke point.

- CP7. Ordering:

Έχει σχεδιαστεί για την ταξινόμηση των αποτελεσμάτων από τα δεδομένα. Επομένως, ικανοποιείται αυτό το choke point.

- CP8. Geo-spatial Predicates:

Το SP2Bench δεν εστιάζει σε γεωχωρικά δεδομένα, επομένως δεν ικανοποιεί αυτό το choke point.

- CP9. Full-text:

Το SP2Bench δεν περιέχει ερωτήματα που απαιτούν κείμενο αναζητήσεων ή μεμονωμένες λέξεις στα δεδομένα. Επομένως, δεν ικανοποιείται καθόλου αυτό το choke point.

- CP10. Duplicate Elimination:

Περιέχει ερωτήματα που εντοπίζουν και εξαλείφουν τα διπλότυπα αποτελέσματα κατά τη δημιουργία ενδιάμεσων αποτελεσμάτων. Συνεπώς, ικανοποιείται αυτό το choke point.

- CP11. Complex Filter Conditions:

Εστιάζει στη αντιμετώπιση πολύπλοκων συνθηκών φιλτραρίσματος, όπως σύζευξη, διάζευξη και άρνηση. Επομένως, το SP2Bench ικανοποιεί το συγκεκριμένο choke point.

→ Συμπερασματικά: Το SP2Bench ικανοποιεί τα περισσότερα από τα choke points, εκτός από τα γεωχωρικά δεδομένα και την αναζήτηση κειμένου. [22]

TPC-H benchmark:

Είναι ένα από τα πιο διαδεδομένα και σημαντικά benchmarks για την αξιολόγηση της απόδοσης συστημάτων διαχείρισης βάσεων δεδομένων. Δημιουργήθηκε από την

Transaction Processing Performance Council (TPC). Η TPC είναι ένας οργανισμός που δημιουργεί πρότυπα benchmarks.

- CP1. Join Ordering:

Το TPC-H περιέχει ερωτήματα με σύνθετα joins, αξιολογώντας το σύστημα να βελτιστοποιεί τη σειρά εκτέλεσης των συνδέσεων για καλύτερη αποδοτικότητα. Συνεπώς, ικανοποιεί το συγκεκριμένο choke point.

- CP2. Aggregation:

Τα ερωτήματα στο TPC-H δίνουν έμφαση σε aggregations όπως μετρήσεις αθροισμάτων ή μέτρηση συνόλου. Επομένως, ικανοποιείται πλήρως αυτό το choke point.

- CP3. Optional & Nested Optional Clauses:

Το TPC-H δεν περιλαμβάνει optional clauses. Συνεπώς, δεν ικανοποιείται το choke point.

- CP4. Reasoning:

Το TPC-H δεν έχει σχεδιαστεί ώστε να αξιολογεί τις δυνατότητες reasoning των συστημάτων. Συνεπώς δεν ικανοποιείται πλήρως το συγκεκριμένο choke point.

- CP5. Parallel Execution of Joins:

Έχει σχεδιαστεί ώστε να δοκιμάζει την ικανότητα του συστήματος να εκτελεί παράλληλες συνδέσεις, αυξάνοντας την αποδοτικότητα της επεξεργασίας ερωτημάτων. Επομένως ικανοποιεί αυτό το choke point.

- CP6. Filters:

Τα queries ενσωματώνουν διάφορα φίλτρα και εκτελούνται όσο το δυνατόν νωρίτερα για την εξάλειψη μεγάλου αριθμού ενδιάμεσων αποτελεσμάτων. Συνεπώς, ικανοποιείται πλήρως αυτό το choke point.

- CP7. Ordering:

Έχει σχεδιαστεί για την ταξινόμηση των αποτελεσμάτων από τα δεδομένα. Επομένως, ικανοποιείται αυτό το choke point.

- CP8. Geo-spatial Predicates:

Το TPC-H δεν περιλαμβάνει γεωχωρικά δεδομένα. Επομένως δεν ικανοποιείται αυτό το choke point.

- CP9. Full-text:

Το TPC-H δεν περιλαμβάνει αναζητήσεις πλήρους κειμένου. Συνεπώς δεν ικανοποιείται αυτό το choke point.

- CP10. Duplicate Elimination:

Περιέχει ερωτήματα που εντοπίζουν και εξαλείφουν τα διπλότυπα αποτελέσματα κατά τη δημιουργία ενδιάμεσων αποτελεσμάτων. Συνεπώς, ικανοποιείται αυτό το choke point.

- CP11. Complex Filter Conditions:

Εστιάζει στη αντιμετώπιση πολύπλοκων συνθηκών φιλτραρίσματος, όπως σύζευξη, διάζευξη και άρνηση. Επομένως, το TPC-H ικανοποιεί το συγκεκριμένο choke point.

→ Συμπερασματικά: Το TPC-H πληροί σχεδόν όλα τα choke points για την αποδοτική επεξεργασία συνδεδεμένων δεδομένων, με εξαίρεση τα CP4, CP8, CP9.

[23]

Κεφάλαιο 4: Development of an RDF Streaming benchmark

Η ανάπτυξη ενός benchmark αποτελεί την βάση για την αξιολόγηση και τη βελτιστοποίηση διαφόρων συστημάτων που επεξεργάζονται RDF δεδομένα. Παρέχουν ένα συγκεκριμένο σύνολο δοκιμών με απώτερο σκοπό την μέτρηση της απόδοσης, της αποδοτικότητας και της επεκτασιμότητας σε ένα σύστημα. Στο παρακάτω κεφάλαιο, θα γίνει μία πλήρης ανασκόπηση για την τροποποίηση και ανάπτυξη ενός υπάρχοντος benchmark για RDF streaming, το οποίο λέγεται CityBench.

Το CityBench σχεδιάστηκε για να αξιολογεί συστήματα επεξεργασίας ροών δεδομένων που σχετίζονται με αστικά δεδομένα. Στην παρούσα εργασία, χρησιμοποιήθηκε ένα σύνολο δεδομένων το οποίο περιέχει δεδομένα μολύνσεων μονοξειδίου του άνθρακα που επικρατούν στην ατμόσφαιρα διαφόρων πόλεων. Βάσει το συγκεκριμένο dataset, υλοποιήθηκε ένα Benchmark προκειμένου να αναπτυχθούν νέα queries τα οποία δείχνουν διάφορα αποτελέσματα μολύνσεων σε ένα συγκεκριμένο σύνολο πόλεων εξετάζοντας ως απώτερο σκοπό την αποδοτικότητα του συστήματος.

4.1 Hardware and Software

Το παρών benchmark αναπτύχθηκε σε ένα φορητό υπολογιστή HP 15 ιντσών, ο οποίος διαθέτει επεξεργαστή AMD Ryzen 7 τετραπύρηνo, 16.0GB μνήμης RAM, SSD 256 GB και λειτουργικό σύστημα Windows 11. Το IDE (Ολοκληρωμένο Περιβάλλον Ανάπτυξης) που χρησιμοποιήθηκε είναι το Eclipse, το οποίο έχει αναπτυχθεί από το Eclipse Foundation, στην έκδοση 4.3.

4.2 Programming Languages and Libraries

Εκ των προτέρων της υλοποίησης, η έρευνα επικεντρώθηκε στην εύρεση τεχνολογιών πλούσιες στην καινοτομία οι οποίες θα πρέπει να είναι ικανές να υποστηρίξουν σύνθετα συστήματα όπως το συγκεκριμένο benchmark. Αυτές οι τεχνολογίες έπρεπε να δίνουν την δυνατότητα συντήρησης καθώς και μελλοντικών επεκτάσεων. Βασικός παράγοντας, ήταν η ύπαρξη μιας μεγάλης κοινότητας που προσφέρει λύσεις ανοιχτού κώδικα και προεγκατεστημένες λύσεις για πιθανές προκλήσεις που μπορεί να παρουσιάζονταν στην πορεία της υλοποίησης της εργασίας. Επιπλέον, αυτές οι τεχνολογίες έπρεπε να είναι

εύκολα προσαρμόσιμες από άλλους προγραμματιστές για κάποια πιθανή μελλοντική επέκταση. Οι επιλεγμένες τεχνολογίες για την υλοποίηση του συστήματος είναι:

- **Java:** Η κύρια γλώσσα προγραμματισμού που χρησιμοποιήθηκε είναι η JAVA για τη δημιουργία των κατάλληλων κλάσεων π.χ. Stream Generator και Stream Processing Solution καθώς και για τη διαχείριση των δεδομένων.
- **RSP-QL:** Το RSP-QL είναι ένα μοντέλο που αναπαριστά και επεξεργάζεται τις ροές RDF σε πραγματικό χρόνο σε δυναμικά περιβάλλοντα. Επεκτείνει την γλώσσα επερωτήσεων SPARQL για να επιτρέπει τη διαχείριση και την ανάλυση συνεχώς μεταβαλλόμενων RDF δεδομένων.
- **Rsp4j:** Η βιβλιοθήκη αυτή, προσφέρει εργαλεία για τη διαχείριση των ροών δεδομένων σε πραγματικό χρόνο, την επεξεργασία τους καθώς και την ενσωμάτωση με διάφορες πηγές δεδομένων. Υποστηρίζει τη γλώσσα RSP-QL, επιτρέποντας την εκτέλεση σύνθετων και συνεχόμενων επερωτήσεων πάνω σε ροές δεδομένων.

4.2.1 Java

Δημιουργός της γλώσσας Java είναι ο James Gosling με στήριξη της ομάδας του, Green Team της Sun Microsystems. Η JAVA έκανε την πρώτη της εμφάνιση και παρουσίαση το 1995. Σκοπός της σχεδίασης της ήταν να σχετίζεται με τις γλώσσες C και C++. Το πλεονέκτημα της όμως είναι, ότι είναι πολύ απλούστερη και ευκολότερη στη σύνταξη της και υποστηρίζει ένα αντικειμενοστραφές μοντέλο προγραμματισμού. Είναι διαφορετικά οργανωμένη, με αρκετά στοιχεία των C και C++ να παραλείπονται και εν αντιθέσει, μερικές ιδέες και τρόποι από άλλες γλώσσες να εμπεριέχονται. Η Java είναι τυποποιημένη με έναν ισχυρό και στατικό τρόπο. Με αυτήν την προδιαγραφή, διακρίνονται ξεκάθαρα τα σφάλματα που πρέπει να εντοπιστούν κατά τον χρόνο μεταγλώττισης, καθώς και εκείνων που εμφανίζονται κατά τον χρόνο εκτέλεσης. Ο πρωταρχικός σκοπός της δημιουργίας της ήταν για διαδραστικές τηλεοπτικές εφαρμογές, όμως με την πάροδο του χρόνου, επικεντρώθηκε και σε άλλους τομείς της τεχνολογίας. Ένας από αυτούς είναι το διαδίκτυο (δυναμικοί ιστότοποι και web εφαρμογές) λόγω της αυξανόμενης δημοφιλίας της.

Χρησιμοποιείται επίσης, ευρέως σε επιχειρήσεις λόγω της ασφάλειας που προσφέρει και της αξιοπιστίας της. Αξιοποιώντας την JAVA, έχουν αναπτυχθεί πολλές εφαρμογές Android. Τέλος, έχει χρησιμοποιηθεί σε big data frameworks.

Η γλώσσα προγραμματισμού Java είναι μια αντικειμενοστραφής γλώσσα γενικού σκοπού, η οποία βασίζεται σε κλάσεις. Είναι σχετικά υψηλού επιπέδου γλώσσα, διότι οι λεπτομέρειες της αναπαράστασης της μηχανής δεν είναι διαθέσιμες μέσω της γλώσσας. Περιέχει αυτόματη διαχείριση αποθήκευσης, συγκεκριμένα έναν garbage collector, για να αποφεύγονται τα προβλήματα ασφαλείας της περιττής δέσμευσης μεταβλητών και δεδομένων (όπως στο free της C ή στο delete της C++). Επιπρόσθετα, δεν περιλαμβάνει “επίφοβες” δημιουργίες πινάκων, όπως προσπελάσεις σε πίνακες χωρίς έλεγχο δεικτών. Αν συνέβαινε κάτι τέτοιο, θα μπορούσαν να προκληθούν ακαθόριστες συμπεριφορές και να χανόταν ο έλεγχος στο πρόγραμμα. Τέλος, οι μεταγλωττίσεις επιτυγχάνονται στο σύνολο εντολών bytecode και στη δυαδική μορφή που ορίζονται στην Προδιαγραφή της Εικονικής Μηχανής Java, Java SE 22 Edition. [24]

4.2.2 RspQL

Το RSP-QL είναι ένα μοντέλο που αναπαριστά και επεξεργάζεται τις ροές RDF σε πραγματικό χρόνο σε δυναμικά περιβάλλοντα οι οποίες αλλάζουν με την πάροδο του χρόνου. Αυτές οι ροές είναι κρίσιμες για εφαρμογές όπως η παρακολούθηση γεγονότων σε ρεαλιστικό χρόνο, η ανάλυση δεδομένων αισθητήρων και άλλα σενάρια που απαιτούν δυναμική διαχείριση δεδομένων. Μερικά παραδείγματα σεναρίων είναι η μέτρηση δεδομένων θερμοκρασίας ή ποιότητας αέρα. Παρακολούθηση και ανάλυση γεγονότων σε πλατφόρμες κοινωνικών δικτύων και ανάλυση συναλλαγών και τιμών καθώς συμβαίνουν. Η γλώσσα αυτή σχεδιάστηκε για να χειρίζεται συνεχή ερωτήματα που χρειάζονται δυναμική ανάλυση και επεξεργασία των δεδομένων που ρέουν.

Το RSP-QL επεκτείνει την γλώσσα ερωτήσεων SPARQL για να επιτρέπει τη διαχείριση και την ανάλυση συνεχώς μεταβαλλόμενων RDF δεδομένων καθώς και για να υποστηρίζει συνεχή ερωτήματα σε ροές RDF, προσφέροντας ταυτόχρονα συμβατότητα με τα υπάρχοντα ερωτήματα SPARQL. Το μοντέλο SPARQL επεκτείνει δύο μοντέλα: Το CQL (Continuous Query Language) και το SECRET. Το CQL είναι μια συνεχής αναβάθμιση της SQL η οποία κάνει ερωτήματα σε μεταβαλλόμενα δεδομένα, ενώ το SECRET περιγράφει και αναλύει το τρόπο με τον οποίο τα συστήματα διαχειρίζονται τα παράθυρα δεδομένων. Τα παράθυρα (windows) είναι χωρισμένα σε τμήματα και εξετάζονται για μια συγκεκριμένη χρονική περίοδο. [14]

4.2.3 Rps4j

Το RSP4J, είναι ένα ευέλικτο API για την επιτάχυνση της ανάπτυξης μηχανών και εφαρμογών RSP που ακολουθούν τη σημασιολογία RSP-QL. Οι χρήστες μπορούν να επέμβουν σε παραμετροποιήσεις προκειμένου να επιτεύξουν την ποικιλία συμπεριφορών των μηχανών RSP σε ένα παρόμοιο λογισμικό περιβάλλον. Για την προώθηση συστηματικής έρευνας, το RSP4J είναι ανοικτού κώδικα, παρέχει αναφορές, μόνιμους διαδικτυακούς αναγνωριστές και συνολική τεκμηρίωση για τους προγραμματιστές. Το RSP4J περιλαμβάνει το YASPER, μια βασική υλοποίηση μηχανής RSP που σκοπός της είναι να αξιοποιείται ως μελέτη περίπτωσης και εκπαιδευτικό εργαλείο. [25] [26]

4.3 Approach & Implementation

Σε αυτό το σημείο, θα γίνει μια διεξοδική ανάλυση και περιγραφή των βασικών πτυχών του συστήματος που υλοποιήθηκε. Αρχικά, θα γίνει μία περιγραφή του dataset και των τροποποιήσεων που έχουν γίνει σε αυτό καθώς και η παρουσίαση των βασικών αρχείων του πηγαίου κώδικα μαζί με την διαδικασία δημιουργίας των RDF ροών δεδομένων και των ερωτημάτων (queries). Τέλος, θα απεικονιστούν τα στιγμιότυπα εικόνων των αποτελεσμάτων από τις εκτελέσεις των ερωτημάτων από το περιβάλλον eclipse. Με αυτό τον τρόπο θα αξιολογηθεί η απόδοση του συστήματος βάσει των choke points και θα υπάρξει συγκριτικός πίνακας έναντι των ήδη υπαρχόντων benchmarks.

Το dataset που χρησιμοποιήθηκε για αυτό το σύστημα βασίζεται στο benchmark CityBench, το οποίο περιλαμβάνει διάφορους τύπους δεδομένων που αφορούν μολύνσεις στην ατμόσφαιρα, αστικά δεδομένα, δεδομένα κίνησης κλπ. ώστε να υποστηρίξουν τις πρωτοβουλίες για την δημιουργία έξυπνων πόλεων που ξεκίνησαν το 2014 στη Δανία. Το CityBench περιλαμβάνει πληροφορίες για τις διαθέσιμες θέσεις των χώρων στάθμευσης, τα επίπεδα ρύπανσης στην ατμόσφαιρα και δεδομένα κίνησης των πόλεων. Αυτά τα δεδομένα συλλέγονται μέσω αισθητήρων και συσκευών IoT που έχουν εγκατασταθεί σε όλη την πόλη.

Για τις ανάγκες της εργασίας, επιλέχθηκαν τα ποσοστά συγκέντρωσης μονοξειδίου του άνθρακα για κάποιες τυχαίες πόλεις του κόσμου που επιλέχθηκαν. Το ποσοστό του μονοξειδίου του άνθρακα δημιουργείται τυχαία με τιμές 1-100 σε κάθε ροή. Παρά το γεγονός ότι δεν υπάρχουν πραγματικά δεδομένα σε πραγματικό χρόνο ή τα δεδομένα είναι

πεπερασμένα, δεν υπάρχει ανησυχία για τον τελικό σκοπό αυτού του έργου, που είναι να εξετάσει πόσο καλά χειρίζεται τα ερωτήματα στον τελικό RDF γράφο και αν επιστρέφονται τα σωστά αποτελέσματα. Το dataset βρίσκεται σε αυτό τον σύνδεσμο στην παραπομπή υπ' αριθμόν [27].

Η εργασία αποτελείται από δύο κύρια Java αρχεία καθώς και από συμπεριλαμβανόμενες βιβλιοθήκες και πακέτα όπως το rsp4j. Το πρώτο αρχείο ονομάζεται "StreamGenerator2.java" και περιέχει έναν πίνακα με τυχαία ονόματα πόλεων από όλο τον κόσμο καθώς και μια ακέραια μεταβλητή που παράγεται τυχαία ανάμεσα στο εύρος 1-100, αντιπροσωπεύοντας τα ποσοστά μόλυνσης από το μονοξείδιο του άνθρακα στις πόλεις αυτές. Επίσης, έχει μια μεταβλητή που αντιπροσωπεύει τον τελικό RDF γράφο με τη τριπλέτα που δημιουργήθηκε σε κάθε παρατήρηση, καθώς και μεθόδους για τη δημιουργία αυτών των τριπλέτων, την εκκίνηση και τη διακοπή της ροής μετά από ένα συγκεκριμένο χρονικό διάστημα (20 δευτερόλεπτα). Το RDF (Resource Description Framework) βασίζεται στις τριπλέτες και αποτελείται από τρία μέρη:

- Υποκείμενο(**Subject**): Περιγραφή του πόρου.
- Κατηγορήμα(**Predicate**): Η ιδιότητα ή σχέση του πόρου.
- Αντικείμενο(**Object**): Η τιμή της ιδιότητας ή ο πόρος με τον οποίο σχετίζεται.

Έτσι, οι τριπλέτες που μελετώνται στην εργασία είναι της μορφής: " **Όνομα Πόλης** " (Υποκείμενο) " **έχει** " (Κατηγορήμα) " **Ποσοστό μόλυνσης επί τοις εκατό** " (Αντικείμενο). Παρακάτω απεικονίζεται ένα σύνολο τριπλετών που διαχειρίζεται το αρχείο "StreamGenerator2":

```
<http://rsp4j.io/pollutions/Amsterdam> <http://rsp4j.io/pollutions/has> "96" .  
<http://rsp4j.io/pollutions/Athens> <http://rsp4j.io/pollutions/has> "68" .  
<http://rsp4j.io/pollutions/Rome> <http://rsp4j.io/pollutions/has> "36" .  
<http://rsp4j.io/pollutions/Athens> <http://rsp4j.io/pollutions/has> "40" .  
<http://rsp4j.io/pollutions/Madrid> <http://rsp4j.io/pollutions/has> "34" .  
<http://rsp4j.io/pollutions/Athens> <http://rsp4j.io/pollutions/has> "2" .  
<http://rsp4j.io/pollutions/Rome> <http://rsp4j.io/pollutions/has> "94" .  
<http://rsp4j.io/pollutions/New_York> <http://rsp4j.io/pollutions/has> "27" .
```

Εικόνα 1: RDF τριπλέτες για τα δεδομένα μόλυνσης

Είναι σύνηθες το Αντικείμενο και το Κατηγορήμα να αναπαριστώνται ως ένα URI (Uniform Resource Identifier) επειδή αυτά τα δεδομένα συνηθίζεται να υπάρχουν σε πόρους όπως ιστοσελίδες ή σε διαδικτυακά αποθετήρια (repositories). Το αντικείμενο μπορεί να

είναι στη μορφή εκφρασμένη από μια κυριολεκτική τιμή (όπως μια συμβολοσειρά ή ένας αριθμός) είτε να εμφανίζεται ως ένα URI. Σε αυτό το παράδειγμα (και φυσικά σε όλες τις τριπλέτες), το ποσοστό μόλυνσης είναι μια τιμή ακέραιου αριθμού μετά την κατάλληλη τροποποίηση που χρειάζεται στο περιβάλλον rsp4j. Η παρακάτω εικόνα απεικονίζει τα δεδομένα και τις μεταβλητές του αρχείου StreamGenerator2, τα οποία κατασκευάζουν τη ροή δεδομένων. Αρχικά δηλώνεται ένας πίνακας από strings (συμβολοσειρές) με όνομα City, ο οποίος περιέχει τυχαία ονόματα πόλεων, η σταθερά prefix για την κατασκευή των URIs των τριπλετών, μια σταθερά timeout για την παύση του thread σε περίπτωση σφάλματος, και η μεταβλητή **pollutionStream** στην οποία θα αποθηκευτούν όλες οι τριπλέτες και θα προκύψει ο τελικός RDF γράφος. Επίσης, αποτελείται από μια boolean (true ή false) μεταβλητή που επιτρέπει την έναρξη ή τη διακοπή της ροής, και μια τελική σταθερά τύπου random της οποίας η χρήση της θα αιτιολογηθεί παρακάτω. Επίσης, υπάρχουν μερικές συναρτήσεις (getters) για τη ροή και για την απόκτηση ενός τυχαίου ονόματος πόλης από τον πίνακα.

```
public String[] City = {"Athens", "Paris", "Amsterdam", "Rome", "New_York", "Madrid"};

public StreamGenerator2() {
    this.pollutionStream = new DataStreamImpl<>(PREFIX + "pollutionStream");
    this.isStreaming = new AtomicBoolean(false);
    randomGenerator = new Random(1337);
}

public DataStream<Graph> getPollutionStream() {

    return pollutionStream;
}

public void startStreaming() {
    if (!this.isStreaming.get()) {
        this.isStreaming.set(true);
        Runnable task =
            () -> {
                long ts = 0;
                while (this.isStreaming.get()) {
                    long finalTs = ts;
                    if (randomGenerator.nextDouble() >= 0.5) {
                        pollutionStream.put(createCarbonMonoxide(), ts);
                    }
                    ts += 5*60*1000;
                    try {
                        Thread.sleep(TIMEOUT);
                    }
                    catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                }
            }
    }
}
```

Εικόνα 2: Δήλωση των μεταβλητών στο αρχείο StreamGenerator2.java

Μία από τις πιο σημαντικές ρουτίνες του αρχείου, αυτή που ενεργοποιεί τη διαδικασία δημιουργίας της ροής, είναι η μέθοδος **startStreaming()** η οποία έχει σχεδιαστεί για να ξεκινάει μια εργασία στο παρασκήνιο που προσομοιώνει μια διαδικασία ροής. Ξεκινά ελέγχοντας αν η ροή είναι επί του παρόντος ανενεργή χρησιμοποιώντας το `this.isStreaming.get()`. Αν η ροή δεν είναι ενεργή, θέτει την τιμή `isStreaming` σε `true` για να υποδείξει ότι η ροή θα ξεκινήσει. Στη συνέχεια, μια αυτοματοποιημένη διαδικασία `Runnable` ορίζεται χρησιμοποιώντας μια έκφραση `lambda`, η οποία θα εκτελεστεί σε ξεχωριστό νήμα. Μέσα στην διαδικασία, μια μεταβλητή χρονικού σημείου `ts` αρχικοποιείται στο 0. Η εργασία στη συνέχεια εισέρχεται σε ένα βρόχο `while` που συνεχίζεται όσο το `this.isStreaming.get()` επιστρέφει `true`, που σημαίνει ότι η ροή είναι ενεργή. Μέσα στον βρόχο, η τρέχουσα τιμή του χρονικού σημείου καταγράφεται σε `finalTs`, και ένας τυχαίος αριθμός παράγεται. Αν αυτός ο τυχαίος αριθμός είναι μεγαλύτερος ή ίσος με 0.5, η μέθοδος `pollutionStream.put(createCarbonMonoxide(),ts)` καλείται για να προσομοιώσει τα δεδομένα ροής. Μετά από αυτό, το χρονικό σημείο αυξάνεται κατά 5 λεπτά (σε χιλιοστά του δευτερολέπτου). Το νήμα στη συνέχεια κάνει παύση για μια διάρκεια που καθορίζεται από το `TIMEOUT`. Αν υπάρξει κάποιο σφάλμα κατά την εκτέλεση, το πρόγραμμα θα το εντοπίσει, θα το απομονώσει και θα συνεχίσει την εκτέλεση του. Τέλος, δημιουργείται ένα νέο νήμα για να εκτελέσει την ορισμένη εργασία, και το νήμα ξεκινά με το `thread.start()`, ενεργοποιώντας τη διαδικασία ροής στο παρασκήνιο. Αυτή η μέθοδος δημιουργεί μια απλή προσομοίωση ροής δεδομένων σε ένα περιβάλλον με πολλά νήματα, με περιοδικούς ελέγχους και διαστήματα για τον έλεγχο της διαδικασίας παραγωγής δεδομένων.

```

public Graph createCarbonMonoxide() {

    Random random = new Random();
    int carbon_monoxide = random.nextInt(101);
    int random_city = random.nextInt(6);
    String city = City[random_city];

    RDF instance = RDFUtils.getInstance();
    Graph graph = instance.createGraph();

    IRI cityIRI = instance.createIRI(pollutionStream + "/" + City[random.nextInt(City.length)]);
    IRI hasIRI = instance.createIRI(pollutionStream + "/" + "has");
    IRI cityNameIRI = instance.createIRI("http://example.org/ontology#city");
    IRI a = instance.createIRI("http://www.w3.org/1999/02/22-rdf-syntax-ns#type");

    Literal pollutionLiteral = instance.createLiteral(String.valueOf(carbon_monoxide));

    graph.add(instance.createTriple(cityIRI, hasIRI, pollutionLiteral));
    // for the last query:
    graph.add(instance.createTriple(cityIRI, a, cityNameIRI));
    // System.out.println(graph);

    return graph;
}

public void stopStreaming() {
    this.isStreaming.set(false);
}

public static void main(String[] args) {
    StreamGenerator2 generator = new StreamGenerator2();
    generator.startStreaming();
}
}

```

Εικόνα 3: Μέθοδος createCarbonMonoxide()

Η μέθοδος **createCarbonMonoxide()** του κώδικα έχει ως σκοπό τη δημιουργία ενός RDF γράφου (graph) που περιέχει πληροφορίες σχετικά με τα επίπεδα μονοξειδίου του άνθρακα σε μια τυχαία πόλη.

- Παράγεται ένας τυχαίος αριθμός carbon_monoxide με εύρος από 0 έως 100 για να αντιπροσωπεύει το επίπεδο του μονοξειδίου του άνθρακα σε ποσοστό.
- Παράγεται ένας τυχαίος αριθμός random_city που κυμαίνεται από 0 έως 5 για να επιλέξει μια τυχαία πόλη από τον πίνακα City. Ο τελικός γράφος δημιουργείται μέσω της RDFUtils.getInstance().createGraph() και αρχικά είναι άδειος μέχρι να εισαχθούν οι τριπλέτες.
- Δημιουργείται ένα IRI για την τυχαία πόλη (cityIRI) που επιλέχθηκε, χρησιμοποιώντας το όνομα του Stream (pollutionStream) και το όνομα της πόλης.
- Δημιουργείται ένα IRI για τη σχέση "has" (hasIRI) χρησιμοποιώντας το όνομα του Stream (pollutionStream) και την τιμή του μονοξειδίου του άνθρακα.

- Δημιουργείται ένα λεκτικό που περιέχει την τιμή του μονοξειδίου του άνθρακα (pollutionLiteral) σε μορφή συμβολοσειράς το οποίο μετατρέπεται σε ακέραιο.
- Προστίθεται μια τριπλέτα στη γραφή που συνδέει την πόλη με την τιμή του μονοξειδίου του άνθρακα μέσω της σχέσης "has".
- Τέλος, Εκτυπώνεται η τριπλέτα που δημιουργήθηκε.

Η μέθοδος **stopStreaming()** θέτει τη μεταβλητή `isStreaming` σε ψευδή τιμή (`false`), κάτι που θα σταματήσει τη διαδικασία ροής δεδομένων όταν ο βρόχος `while` της μεθόδου `startStreaming` ελέγξει αυτή τη μεταβλητή. Η κύρια μέθοδος (main method) λειτουργεί ως το σημείο εκκίνησης του προγράμματος. Μέσα στην κύρια μέθοδο, δημιουργείται ένα αντικείμενο της κλάσης `StreamGenerator2` καλώντας τον κατασκευαστή της (`StreamGenerator2 generator = new StreamGenerator2()`). Μόλις δημιουργηθεί το αντικείμενο, η μέθοδος `startStreaming` καλείται σε αυτό το αντικείμενο (`generator.startStreaming()`), η οποία ξεκινά τη διαδικασία ροής δεδομένων σε ένα νέο νήμα (`thread`).

```
public class QueryProcessingSolution2 {
    private StreamGenerator2 generator;
    private DataStream<Graph> pollutionStream;

    public QueryProcessingSolution2() throws InterruptedException {
        generator = new StreamGenerator2();
        pollutionStream = generator.getPollutionStream();
    }
}
```

Εικόνα 4: `QueryProcessingSolution2` μεταβλητές και constructor

Στην επόμενη ενότητα υπάρχει μια παρόμοια παρουσίαση για το δεύτερο κύριο αρχείο `java` με όνομα **QueryProcessingSolution2**. Στην πρώτη εικόνα υπάρχει η δήλωση των δύο βασικών μεταβλητών. Η μεταβλητή με όνομα "**generator**" είναι τύπου `StreamGenerator2`, το οποίο επιτρέπει τη σύνδεση με το αρχείο `StreamGenerator2` (όπου το τελικό `RDF` γράφημα αποθηκεύεται μετά την έναρξη της ροής). Η μεταβλητή με όνομα "**pollutionStream**" έχει την τιμή του `URL` της ροής "`http://rsp4j.io/pollutionStream/pollutions/`". Αυτά τα αντικείμενα δημιουργούνται στον κατασκευαστή του αρχείου.

```

public static void main(String[] args) throws InterruptedException {
    QueryProcessingSolution2 qps = new QueryProcessingSolution2();
    qps.queryAllPollution();
    qps.queryPollutionEqualFifty();
    qps.queryByCity();
    qps.queryPollutionUnderFifty();
    qps.queryPollutionOverSeventy();
    qps.queryPollutionBetweenTenAndForty();
    qps.queryPollutionBetweenRanges(); //(20-50 || 60-80)
    qps.queryComplexFilterCondition();
    qps.queryCountTownAmsterdam();
    qps.queryMaxPollution();
    qps.queryWithSubqueries();
    qps.queryReasoningTypeCity();
}

```

Εικόνα 5: Κύρια μέθοδος στο αρχείο QueryProcessingSolution2

Έπειτα, υπάρχει η κύρια συνάρτηση (main), η οποία είναι το σημείο εκκίνησης κάθε προγράμματος Java. Στην κύρια συνάρτηση υπάρχει η κλήση όλων των μεθόδων που είναι τα ερωτήματα προς το RDF γράφημα. Ένα αντικείμενο με όνομα "qps" τύπου QueryProcessingSolution2 είναι απαραίτητο για να καλέσει αυτές τις μεθόδους.

4.3.1 Explanation of the Benchmark Queries Based on the Choke Points

Σε αυτό το σημείο, θα γίνει ανάλυση και επεξήγηση του κάθε query που δημιουργήθηκε, καθώς και θα απεικονιστούν τα αποτελέσματα από το πρόγραμμα του eclipse.

```

public void queryAllPollution() throws InterruptedException
{
    ContinuousQuery<Graph, Graph, Binding, Binding> query =
        TPQueryFactory.parse(
            "PREFIX : <http://rsp4j.io/pollutions/> "
            + "PREFIX rsp4j: <http://rsp4j.io/> "
            + "PREFIX pollutionStream: <http://rsp4j.io/pollutions/pollutionStream/> "
            + "REGISTER RSTREAM <http://out/stream> AS "
            + "SELECT ?city ?has ?pollution "
            + "FROM NAMED WINDOW :window ON :pollutionStream [RANGE PT10M STEP PT1M] "
            + "WHERE {"
            + "    WINDOW :window {?city ?has ?pollution .}"
            + "}");

    TaskOperatorAPIImpl<Graph, Graph, Binding, Binding> t =
        new QueryTaskOperatorAPIImpl.QueryTaskBuilder().fromQuery(query).build();

    ContinuousProgram<Graph, Graph, Binding, Binding> cp =
        new ContinuousProgram.ContinuousProgramBuilder()
            .in(pollutionStream)
            .addTask(t)
            .out(query.getOutputStream())
            .addJoinAlgorithm(new HashJoinAlgorithm())
            .build();

    query.getOutputStream().addConsumer((el, ts) -> System.out.println(el + " @ " + ts));

    generator.startStreaming();

    Thread.sleep(20_000);
    generator.stopStreaming();
}

```

Εικόνα 6: Κοινή αρχιτεκτονική όλων των query

Κάθε μέθοδος θα αναλυθεί ξεχωριστά και θα παρουσιαστούν τα αποτελέσματα κάθε ερωτήματος. Η πλειονηφία όλων των μεθόδων έχουν την ίδια λογική, επομένως θα αναλυθεί μια φορά ο κώδικας των ερωτημάτων. Η μέθοδος **TPQueryFactory.parse** χρησιμοποιείται για να ορίσει ένα συνεχές ερώτημα σε RSP-QL. Το ερώτημα καταχωρεί ένα ρεύμα εξόδου (RSTREAM <http://out/stream>) και επιλέγει τριάδες με υποκείμενα (?city), ρήματα (?has), και αντικείμενα (?pollution) από ένα ορισμένο παράθυρο. Το παράθυρο ορίζεται πάνω από το ρεύμα με εύρος 10 λεπτά (PT10M) και βήμα 1 λεπτό (PT1M).

Ένα αντικείμενο **TaskOperatorAPIImpl** δημιουργείται χρησιμοποιώντας έναν constructor ερωτημάτων, ο οποίος παίρνει ως είσοδο το ορισμένο συνεχές ερώτημα. Ένα **ContinuousProgram** κατασκευάζεται για να συνδέσει το εισερχόμενο RDF ρεύμα (pollutionStream), τον τελεστή εργασίας (t), και το ρεύμα εξόδου του ερωτήματος. Ένας αλγόριθμος συνένωσης hash καθορίζεται για τις λειτουργίες συνένωσης εντός του συνεχούς προγράμματος.

Ένας καταναλωτής προστίθεται στο ρεύμα εξόδου του ερωτήματος για να επεξεργαστεί και να εκτυπώσει τα αποτελέσματα μαζί με τις χρονοσημάνσεις τους. Η μέθοδος **generator.startStreaming()** ξεκινά τη διαδικασία ροής δεδομένων. Η μέθοδος

`Thread.sleep(20_000)` επιτρέπει στη ροή να εκτελείται για 20 δευτερόλεπτα. Τέλος, η μέθοδος `generator.stopStreaming()` σταματά τη διαδικασία ροής δεδομένων.

- **1ο query:**

```
public void queryAllPolution() throws InterruptedException
{
    ContinuousQuery<Graph, Graph, Binding, Binding> query =
        TPQueryFactory.parse(
            "PREFIX : <http://rsp4j.io/pollutions/> "
            + "PREFIX rsp4j: <http://rsp4j.io/> "
            + "PREFIX pollutionStream: <http://rsp4j.io/pollutions/pollutionStream/> "
            + "REGISTER RSTREAM <http://out/stream> AS "
            + "SELECT ?city ?has ?pollution "
            + "FROM NAMED WINDOW :window ON :pollutionStream [RANGE PT10M STEP PT1M] "
            + "WHERE {"
            + " WINDOW :window {?city ?has ?pollution .}"
            + "}");
```

Εικόνα 7α: Εμφάνιση όλων των πόλεων με τυχαία Ποσοστά μόλυνσης

Το ερώτημα στην πρώτη μέθοδο επιστρέφει όλες τις πόλεις με τυχαία ποσοστά μόλυνσης χωρίς κανέναν περιορισμό του RDF γραφήματος μετά την εκκίνηση της ροής.

Τα αποτελέσματα παρουσιάζονται στην παρακάτω εικόνα ως RDF τριπλέτες με αντικείμενο (όνομα πόλης), κατηγορήμα (has) και υποκείμενο (ποσοστό μόλυνσης).

```
{city -> <http://rsp4j.io/pollutions/pollutionStream/Athens>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "60"} @ 600000
{city -> <http://rsp4j.io/pollutions/pollutionStream/Rome>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "43"} @ 600000
{city -> <http://rsp4j.io/pollutions/pollutionStream/Athens>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "60"} @ 900000
{city -> <http://rsp4j.io/pollutions/pollutionStream/New_York>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "11"} @ 900000
{city -> <http://rsp4j.io/pollutions/pollutionStream/Amsterdam>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "37"} @ 1500000
{city -> <http://rsp4j.io/pollutions/pollutionStream/Rome>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "95"} @ 2700000
{city -> <http://rsp4j.io/pollutions/pollutionStream/Athens>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "51"} @ 4800000
{city -> <http://rsp4j.io/pollutions/pollutionStream/Athens>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "69"} @ 5400000
{city -> <http://rsp4j.io/pollutions/pollutionStream/New_York>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "99"} @ 5700000
```

Εικόνα 7β: Εμφάνιση αποτελεσμάτων

- **2ο query:**

```
public void queryPollutionEqualFifty() throws InterruptedException
{
    ContinuousQuery<Graph, Graph, Binding, Binding> query =
        TPQueryFactory.parse(
            "PREFIX : <http://rsp4j.io/pollutions/> "
            + "PREFIX rsp4j: <http://rsp4j.io/> "
            + "PREFIX pollutionStream: <http://rsp4j.io/pollutions/pollutionStream/> "
            + "REGISTER RSTREAM <http://out/stream> AS "
            + "SELECT ?city ?has ?pollution "
            + "FROM NAMED WINDOW rsp4j:window ON :pollutionStream [RANGE PT10M STEP PT1M] "
            + "WHERE {"
            + "WINDOW rsp4j:window {?city ?has ?pollution .}"
            + "FILTER(?pollution = 50)"
            + "}");
}
```

Εικόνα 8α: Εμφάνιση όλων των πόλεων με ποσοστό μόλυνσης ίσο με 50%

Το ερώτημα στην δεύτερη μέθοδο επιστρέφει τις πόλεις με τον περιορισμό ότι η μόλυνση θα έχει ποσοστό 50%.

```
[city -> <http://rsp4j.io/pollutions/pollutionStream/Amsterdam>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "50"] @ 24600000
[city -> <http://rsp4j.io/pollutions/pollutionStream/New_York>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "50"] @ 47400000
[city -> <http://rsp4j.io/pollutions/pollutionStream/New_York>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "50"] @ 47700000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Paris>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "50"] @ 49500000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Paris>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "50"] @ 49800000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Paris>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "50"] @ 51900000
[city -> <http://rsp4j.io/pollutions/pollutionStream/New_York>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "50"] @ 57000000
```

Εικόνα 8β: Εμφάνιση αποτελεσμάτων

- **3ο query:**

```
public void queryByCity() throws InterruptedException
{
    ContinuousQuery<Graph, Graph, Binding, Binding> query =
        TPQueryFactory.parse(
            "PREFIX : <http://rsp4j.io/pollutions/> " // Adjusted prefix for pollution
            + "PREFIX rsp4j: <http://rsp4j.io/> "
            + "PREFIX pollutionStream: <http://rsp4j.io/pollutions/pollutionStream/> "
            + "REGISTER RSTREAM <http://out/stream> AS "
            + "SELECT ?city ?has ?pollution //s:subject, p:predicate, o:object
            + "FROM NAMED WINDOW rsp4j:window ON :pollutionStream [RANGE PT10M STEP PT1M] "
            + "WHERE {"
            + "WINDOW rsp4j:window {?city ?has ?pollution .}"
            + "FILTER(?city = pollutionStream:Amsterdam)"
            + "}");
}
```

Εικόνα 9α: Εμφάνιση τριπλετών με όνομα πόλης “Amsterdam”

Το ερώτημα στην τρίτη μέθοδο εμφανίζει τις τριπλέτες στις οποίες το όνομα πόλης είναι μόνο το “Amsterdam”.

```
[city -> <http://rsp4j.io/pollutions/pollutionStream/Amsterdam>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "99"] @ 600000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Amsterdam>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "99"] @ 900000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Amsterdam>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "84"] @ 900000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Amsterdam>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "73"] @ 5400000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Amsterdam>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "62"] @ 57000000
```

Εικόνα 9β: Εμφάνιση αποτελεσμάτων

- **4o query:**

```
public void queryPollutionUnderFifty() throws InterruptedException {
    ContinuousQuery<Graph, Graph, Binding, Binding> query =
        TPQueryFactory.parse(
            "PREFIX : <http://rsp4j.io/pollutions/> "
            + "PREFIX rsp4j: <http://rsp4j.io/> "
            + "PREFIX pollutionStream: <http://rsp4j.io/pollutions/pollutionStream/> "
            + "REGISTER RSTREAM <http://out/stream> AS "
            + "SELECT ?city ?has ?pollution "
            + "FROM NAMED WINDOW rsp4j:window ON :pollutionStream [RANGE PT10M STEP PT1M] "
            + "WHERE {"
            + "    WINDOW rsp4j:window {?city ?has ?pollution .}"
            + "    FILTER(?pollution < 50)"
            + "}");
}
```

Εικόνα 10α: Εμφάνιση πόλεων με ποσοστό μόλυνσης μικρότερο του 50%.

Στην τρίτη μέθοδο εμφανίζονται οι πόλεις οι οποίες έχουν ποσοστό μόλυνσης κάτω του 50%.

```
{city -> <http://rsp4j.io/pollutions/pollutionStream/Rome>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "25"} @ 900000
{city -> <http://rsp4j.io/pollutions/pollutionStream/Athens>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "5"} @ 6600000
{city -> <http://rsp4j.io/pollutions/pollutionStream/Madrid>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "22"} @ 7800000
{city -> <http://rsp4j.io/pollutions/pollutionStream/Athens>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "40"} @ 10500000
{city -> <http://rsp4j.io/pollutions/pollutionStream/Athens>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "40"} @ 10800000
{city -> <http://rsp4j.io/pollutions/pollutionStream/Athens>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "34"} @ 10800000
{city -> <http://rsp4j.io/pollutions/pollutionStream/Athens>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "34"} @ 11100000
{city -> <http://rsp4j.io/pollutions/pollutionStream/Rome>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "37"} @ 11100000
{city -> <http://rsp4j.io/pollutions/pollutionStream/Athens>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "9"} @ 11400000
```

Εικόνα 10β: Εμφάνιση αποτελεσμάτων

- **5o query:**

```
public void queryPollutionOverSeventy() throws InterruptedException {
    // Define the query to filter towns with pollution under fifty
    ContinuousQuery<Graph, Graph, Binding, Binding> query =
        TPQueryFactory.parse(
            "PREFIX : <http://rsp4j.io/pollutions/> " // Adjusted prefix for pollution
            + "PREFIX rsp4j: <http://rsp4j.io/> "
            + "PREFIX pollutionStream: <http://rsp4j.io/pollutions/pollutionStream/> "
            + "REGISTER RSTREAM <http://out/stream> AS "
            + "SELECT ?city ?has ?pollution " //s:subject, p:predicate, o:object
            + "FROM NAMED WINDOW rsp4j:window ON :pollutionStream [RANGE PT10M STEP PT1M] "
            + "WHERE {"
            + "    WINDOW rsp4j:window {?city ?has ?pollution .}"
            + "    FILTER(?pollution > 70)" // Filter towns with pollution under fifty
            + "}");
}
```

Εικόνα 11α: Εμφάνιση πόλεων με ποσοστό μόλυνσης άνω του 70%.

Η πέμπτη μέθοδος εμφανίζει τις πόλεις οι οποίες έχουν ποσοστό μόλυνσης μεγαλύτερο του 70%.

```
[city -> <http://rsp4j.io/pollutions/pollutionStream/Amsterdam>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "72"] @ 7800000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Paris>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "84"] @ 8100000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Paris>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "84"] @ 8400000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Athens>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "76"] @ 15600000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Athens>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "88"] @ 19800000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Amsterdam>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "76"] @ 21900000
[city -> <http://rsp4j.io/pollutions/pollutionStream/New York>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "83"] @ 22500000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Paris>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "84"] @ 24600000
[city -> <http://rsp4j.io/pollutions/pollutionStream/New York>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "99"] @ 33600000
```

Εικόνα 11β: Εμφάνιση αποτελεσμάτων

- **6ο query:**

```
public void queryPollutionBetweenTenAndForty() throws InterruptedException {
    ContinuousQuery<Graph, Graph, Binding, Binding> query =
        TPQueryFactory.parse(
            "PREFIX : <http://rsp4j.io/pollutions/> "
            + "PREFIX rsp4j: <http://rsp4j.io/> "
            + "PREFIX pollutionStream: <http://rsp4j.io/pollutions/pollutionStream/> "
            + "REGISTER RSTREAM <http://out/stream> AS "
            + "SELECT ?city ?has ?pollution "
            + "FROM NAMED WINDOW rsp4j:window ON :pollutionStream [RANGE PT10M STEP PT1M] "
            + "WHERE {"
            + "    WINDOW rsp4j:window {?city ?has ?pollution .}"
            + "    FILTER(?pollution >= 10 && ?pollution <= 40)"
            + "}");
```

Εικόνα 12α: Εμφάνιση πόλεων με ποσοστό μόλυνσης ανάμεσα σε 10 - 40%

Η έκτη μέθοδος εμφανίζει τις πόλεις οι οποίες έχουν ποσοστό μόλυνσης μεταξύ 10 έως και 40%.

```
[city -> <http://rsp4j.io/pollutions/pollutionStream/New York>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "37"] @ 16500000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Paris>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "39"] @ 19800000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Madrid>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "29"] @ 21000000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Rome>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "30"] @ 21300000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Paris>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "19"] @ 21900000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Paris>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "10"] @ 22800000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Amsterdam>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "33"] @ 34200000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Paris>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "30"] @ 37500000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Madrid>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "20"] @ 41100000
```

Εικόνα 12β: Εμφάνιση αποτελεσμάτων

- **7ο query:**

```
public void queryPollutionBetweenRanges() throws InterruptedException {
    ContinuousQuery<Graph, Graph, Binding, Binding> query =
        TPQueryFactory.parse(
            "PREFIX : <http://rsp4j.io/pollutions/> "
            + "PREFIX rsp4j: <http://rsp4j.io/> "
            + "PREFIX pollutionStream: <http://rsp4j.io/pollutions/pollutionStream/> "
            + "REGISTER RSTREAM <http://out/stream> AS "
            + "SELECT ?city ?has ?pollution "
            + "FROM NAMED WINDOW rsp4j:window ON :pollutionStream [RANGE PT10M STEP PT1M] "
            + "WHERE {"
            + "    WINDOW rsp4j:window {?city ?has ?pollution .}"
            + "    FILTER(?pollution >= 20 && ?pollution <= 50 || ?pollution >= 60 && ?pollution <= 80)"
            + "}");
```

Εικόνα 13α: Εμφάνιση πόλεων με μόλυνση 20 - 50% ή 60 - 80%.

Η έβδομη μέθοδος εμφανίζει τις πόλεις οι οποίες έχουν ποσοστό μόλυνσης μεταξύ 20 έως 50% ή 60 - 80%.

```
[city -> <http://rsp4j.io/pollutions/pollutionStream/Paris>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "69"] @ 9000000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Madrid>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "21"] @ 1500000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Amsterdam>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "21"] @ 2700000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Madrid>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "27"] @ 7800000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Amsterdam>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "48"] @ 8400000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Amsterdam>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "48"] @ 8700000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Amsterdam>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "34"] @ 10500000
```

Εικόνα 13β: Εμφάνιση αποτελεσμάτων

- **8ο query:**

```
public void queryComplexFilterCondition() throws InterruptedException {
    ContinuousQuery<Graph, Graph, Binding, Binding> query =
        TPQueryFactory.parse(
            "PREFIX : <http://rsp4j.io/pollutions/> "
            + "PREFIX rsp4j: <http://rsp4j.io/> "
            + "PREFIX pollutionStream: <http://rsp4j.io/pollutions/pollutionStream/> "
            + "REGISTER RSTREAM <http://out/stream> AS "
            + "SELECT ?city ?has ?pollution "
            + "FROM NAMED WINDOW rsp4j:window ON :pollutionStream [RANGE PT10M STEP PT1M] "
            + "WHERE {"
            + "    WINDOW rsp4j:window {?city ?has ?pollution .}"
            + "    FILTER(?city != pollutionStream:Athens && ?city != pollutionStream:Paris || ?pollution >= 20 && ?pollution <= 80)"
            + "}");
}
```

Εικόνα 14α: Εμφάνιση τριπλετών εκτός από Αθήνα ή Παρίσι με συγκεκριμένο εύρος μόλυνσης.

Η όγδοη μέθοδος εμφανίζει τις τριπλέτες οι οποίες εξαιρούν την πόλη Αθήνα ή το Παρίσι ή έχουν ποσοστό μόλυνσης μεταξύ 20 - 80% χρησιμοποιώντας την συνάρτηση FILTER().

```
[city -> <http://rsp4j.io/pollutions/pollutionStream/Madrid>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "86"] @ 300000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Rome>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "15"] @ 600000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Madrid>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "86"] @ 600000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Rome>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "15"] @ 900000
[city -> <http://rsp4j.io/pollutions/pollutionStream/New_York>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "44"] @ 900000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Madrid>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "58"] @ 1500000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Madrid>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "59"] @ 2700000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Amsterdam>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "62"] @ 4800000
```

Εικόνα 14β: Εμφάνιση αποτελεσμάτων

Οι Aggregate functions στην RSP-QL (RDF Stream Processing Query Language) χρησιμοποιούνται για να πραγματοποιούν υπολογισμούς σε σύνολα δεδομένων μιας RDF ροής. Αυτές οι συναρτήσεις περιλαμβάνουν:

- **COUNT:** Υπολογίζει τον αριθμό των στοιχείων σε ένα σύνολο δεδομένων ή ροής.
- **MAX:** Βρίσκει τη μέγιστη τιμή σε ένα σύνολο δεδομένων.
- **MIN:** Προσδιορίζει την ελάχιστη τιμή σε ένα σύνολο δεδομένων.
- **AVG:** Υπολογίζει τη μέση τιμή ενός αριθμητικού συνόλου δεδομένων.

Αυτές οι συναρτήσεις είναι απαραίτητες για την περίληψη και την ανάλυση συνεχών ροών δεδομένων σε πραγματικό χρόνο. Για αυτόν τον λόγο υπάρχει το ερώτημα στη μέθοδο `queryCountTownAmsterdam()`, στο οποίο υπολογίζεται πόσες φορές θα εμφανιστεί στις τριπλέτες η πόλη Amsterdam .

- **9ο query:**

```
public void queryCountTownAmsterdam() throws InterruptedException {
    ContinuousQuery<Graph, Graph, Binding, Binding> query =
        TPQueryFactory.parse(
            "PREFIX : <http://rsp4j.io/pollutions/> "
            + "PREFIX rsp4j: <http://rsp4j.io/> "
            + "PREFIX pollutionStream: <http://rsp4j.io/pollutions/pollutionStream/> "
            + "REGISTER RSTREAM <http://out/stream> AS "
            + "SELECT (COUNT(DISTINCT ?city) AS ?count) "
            + "FROM NAMED WINDOW rsp4j:window ON :pollutionStream [RANGE PT10M STEP PT1M] "
            + "WHERE {"
            + "    WINDOW rsp4j:window {?city ?has ?pollution .}"
            + "    FILTER(?city = pollutionStream:Amsterdam)"
            + "}");
}
```

Εικόνα 15α: μέτρηση εμφάνισης τριπλέτας με πόλη το Άμστερνταμ

Η ένατη μέθοδος μετράει πόσες φορές εμφανίστηκε στην τριπλέτα η πόλη Άμστερνταμ. Επειδή όμως κάθε φορά δημιουργείται καινούργιο παράθυρο και δεν αποθηκεύονται τα προηγούμενα δεδομένα, εμφανίζει κάθε φορά το count ίσο με 1.

```
{count -> <1>} @ 300000
{count -> <1>} @ 600000
{count -> <0>} @ 900000
{count -> <0>} @ 1500000
{count -> <0>} @ 2400000
{count -> <1>} @ 2700000
{count -> <0>} @ 4200000
{count -> <1>} @ 4800000
{count -> <1>} @ 5400000
{count -> <0>} @ 5700000
```

Εικόνα 15β: Εμφάνιση αποτελεσμάτων

- **10o query:**

```
public void queryMaxPollution() throws InterruptedException {
    ContinuousQuery<Graph, Graph, Binding, Binding> query =
        TPQueryFactory.parse(
            "PREFIX : <http://rsp4j.io/pollutions/> "
            + "PREFIX rsp4j: <http://rsp4j.io/> "
            + "PREFIX pollutionStream: <http://rsp4j.io/pollutions/pollutionStream/> "
            + "REGISTER RSTREAM <http://out/stream> AS "
            + "SELECT (MAX(?pollution) AS ?max) "
            + "FROM NAMED WINDOW rsp4j:window ON :pollutionStream [RANGE PT10M STEP PT1M] "
            + "WHERE { "
            + "    WINDOW rsp4j:window {?city ?has ?pollution .}"
            + "    FILTER(?city = pollutionStream:Paris)"
            + "}");
}
```

Εικόνα 16α: Υπολογισμός μέγιστης μόλυνσης για την πόλη Παρίσι

Η δέκατη μέθοδος υπολογίζει την μέγιστη μόλυνση μέσω aggregate function MAX(), όταν η πόλη είναι το Παρίσι. Δυστυχώς, όπως υποδεικνύει η παρακάτω εικόνα, υπάρχει πρόβλημα με τη συνάρτηση MAX (το ίδιο πρόβλημα υπάρχει και με τη συνάρτηση MIN) καθώς αυτές οι συναρτήσεις δεν βρίσκονται στη βιβλιοθήκη της rsp4j. Ωστόσο, η σύνταξη και η λογική προσέγγιση του ερωτήματος είναι σωστές.

```
[Thread-0] ERROR org.streamreasoning.rsp4j.operatorapi.ContinuousProgram - Function MAX not found in Registry!
[Thread-0] ERROR org.streamreasoning.rsp4j.operatorapi.ContinuousProgram - Function MAX not found in Registry!
[Thread-0] ERROR org.streamreasoning.rsp4j.operatorapi.ContinuousProgram - Function MAX not found in Registry!
[Thread-0] ERROR org.streamreasoning.rsp4j.operatorapi.ContinuousProgram - Function MAX not found in Registry!
[Thread-0] ERROR org.streamreasoning.rsp4j.operatorapi.ContinuousProgram - Function MAX not found in Registry!
[Thread-0] ERROR org.streamreasoning.rsp4j.operatorapi.ContinuousProgram - Function MAX not found in Registry!
```

Εικόνα 16β: Εμφάνιση αποτελεσμάτων

- **11o query:**

```
public void queryWithSubqueries() throws InterruptedException {
    ContinuousQuery<Graph, Graph, Binding, Binding> query =
        TPQueryFactory.parse(
            "PREFIX : <http://rsp4j.io/pollutions/> "
            + "PREFIX rsp4j: <http://rsp4j.io/> "
            + "PREFIX pollutionStream: <http://rsp4j.io/pollutions/pollutionStream/> "
            + "REGISTER RSTREAM <http://out/stream> AS "
            + "SELECT ?city ?has ?pollution "
            + "FROM NAMED WINDOW rsp4j:window ON :pollutionStream [RANGE PT10M STEP PT1M] "
            + "WHERE { "
            + "    WINDOW rsp4j:window { "
            + "        ?city ?has ?pollution ."
            + "    } "
            + "    SELECT ?city (COUNT(?pollution) AS ?count) "
            + "    WHERE { "
            + "        ?city ?has ?pollution "
            + "    } "
            + "    GROUP BY ?city "
            + "    } "
            + "    } "
            + "    }");
}
```

Εικόνα 17α: Εκτέλεση ερωτήματος με subquery

Η RSP-QL είναι μια γλώσσα ερωτημάτων όπως και η SPARQL. Για αυτό το λόγο είναι εφικτό να εκτελεστούν υποερωτήματα. Ένα υποερώτημα, γνωστό και ως εσωτερικό ερώτημα, περιλαμβάνεται μέσα σε ένα άλλο ερώτημα. Ο κύριος σκοπός ενός υποερωτήματος είναι να παρέχει ένα σύνολο αποτελεσμάτων που μπορούν να χρησιμοποιηθούν ως συνθήκη ή τιμή στο εξωτερικό ερώτημα. Αυτό το ερώτημα, που φαίνεται στην παραπάνω εικόνα, ανακτά τριπλέτες όπου το αντικείμενο είναι ?pollution. Επιπλέον, εκτελεί ένα υποερώτημα για να μετρήσει το πλήθος των μολύνσεων (?count) για κάθε υποκείμενο (?city) και ομαδοποιεί τα αποτελέσματα ανά ?city. Το τελικό αποτέλεσμα περιλαμβάνει τα υποκείμενα, τα κατηγορήματα και τα αντικείμενα μέσα στο καθορισμένο παράθυρο.

```
[city -> <http://rsp4j.io/pollutions/pollutionStream/Madrid>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "6") @ 600000  
[city -> <http://rsp4j.io/pollutions/pollutionStream/Rome>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "80") @ 600000  
[city -> <http://rsp4j.io/pollutions/pollutionStream/Amsterdam>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "82") @ 900000  
[city -> <http://rsp4j.io/pollutions/pollutionStream/Rome>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "80") @ 900000  
[city -> <http://rsp4j.io/pollutions/pollutionStream/Paris>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "59") @ 1500000  
[city -> <http://rsp4j.io/pollutions/pollutionStream/New_York>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "47") @ 2700000  
[city -> <http://rsp4j.io/pollutions/pollutionStream/Madrid>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "64") @ 4800000  
[city -> <http://rsp4j.io/pollutions/pollutionStream/Paris>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "5") @ 5400000
```

Εικόνα 17β: Εμφάνιση αποτελεσμάτων

- **12o query:**

```
public void queryReasoningTypeCity() throws InterruptedException {  
    ContinuousQuery<Graph, Graph, Binding, Binding> query =  
        TPQueryFactory.parse(  
            "PREFIX : <http://rsp4j.io/pollutions/> "  
            + "PREFIX rsp4j: <http://rsp4j.io/> "  
            + "PREFIX pollutionStream: <http://rsp4j.io/pollutions/pollutionStream/> "  
            + "REGISTER RSTREAM <http://out/stream> AS "  
            + "SELECT ?city ?has ?pollution "  
            + "FROM NAMED WINDOW rsp4j:window ON :pollutionStream [RANGE PT10M STEP PT1M] "  
            + " WHERE { "  
            + "   WINDOW rsp4j:window {?city ?has ?pollution .}"  
            + " ?city a <http://example.org/ontology#city> ."  
            + "}" );
```

Εικόνα 18α: Reasoning Query

Το τελευταίο ερώτημα ελέγχει αν ένα RDF literal εκφράζεται στο σχήμα αποτελεσματικά. Πολλές φορές υπάρχει η ανάγκη να εκφραστεί ένα στοιχείο (για παράδειγμα το αντικείμενο) ως ιδιότητα RDF, η οποία χρησιμοποιείται για να δηλώσει ότι ένας πόρος είναι μια περίπτωση μιας κλάσης. Το ερώτημα στην εικόνα το επιτυγχάνει αυτό, καθώς η μεταβλητή με

το όνομα "a" είναι μια ιδιότητα RDF που έχει οριστεί στο αρχείο StreamGenerator2. Ο σκοπός αυτού του ερωτήματος είναι να διασφαλίσει την αποφυγή δημιουργίας σφαλμάτων. Εάν δημιουργηθούν σφάλματα, δεν θα εμφανιστούν τα αποτελέσματα.

```
[city -> <http://rsp4j.io/pollutions/pollutionStream/Athens>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "3"] @ 11400000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Athens>, has -> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>, pollution -> <http://example.org/ontology#city>] @ 11400000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Paris>, has -> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>, pollution -> <http://example.org/ontology#city>] @ 11400000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Paris>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "50"] @ 11400000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Amsterdam>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "47"] @ 11700000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Amsterdam>, has -> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>, pollution -> <http://example.org/ontology#city>] @ 11700000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Paris>, has -> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>, pollution -> <http://example.org/ontology#city>] @ 11700000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Paris>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "50"] @ 11700000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Rome>, has -> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>, pollution -> <http://example.org/ontology#city>] @ 12900000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Rome>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "59"] @ 12900000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Athens>, has -> <http://rsp4j.io/pollutions/pollutionStream/has>, pollution -> "6"] @ 13500000
[city -> <http://rsp4j.io/pollutions/pollutionStream/Athens>, has -> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>, pollution -> <http://example.org/ontology#city>] @ 13500000
```

Εικόνα 18β: Εμφάνιση Αποτελεσμάτων

Κεφάλαιο 5: Conclusions

5.1 Conclusion

Η επεξεργασία ροών RDF (Resource Description Framework) είναι κρίσιμη, καθώς επιτρέπει την ανάλυση και συλλογιστική σε πραγματικό χρόνο πάνω σε συνεχείς ροές δεδομένων, κάτι που είναι απαραίτητο στον σημερινό κόσμο που βασίζεται στα δεδομένα. Μέσω της άμεσης επεξεργασίας και ενσωμάτωσης δυναμικών πληροφοριών, η επεξεργασία ροών RDF υποστηρίζει την έγκαιρη εξαγωγή πληροφοριών και τη λήψη αποφάσεων για εφαρμογές όπως τα συστήματα παρακολούθησης, το χρηματιστηριακό εμπόριο και η ανάλυση κοινωνικών μέσων. Επίσης, ενισχύει τη διαλειτουργικότητα και την ενσωμάτωση μεταξύ διαφορετικών πηγών δεδομένων, ακολουθώντας τα πρότυπα του Σημασιολογικού Ιστού, διευκολύνοντας τη συγχώνευση ετερογενών δεδομένων με συνεκτικό τρόπο.

Τα πρότυπα αξιολόγησης RDF είναι απαραίτητα εργαλεία για την αξιολόγηση της απόδοσης και της κλιμακωσιμότητας των συστημάτων διαχείρισης RDF δεδομένων. Παρέχουν τυποποιημένα σύνολα δεδομένων και ερωτήματα για την αξιολόγηση διαφόρων πτυχών, όπως η εισαγωγή δεδομένων, η επεξεργασία ερωτημάτων και οι δυνατότητες συλλογιστικής.

Σημαντικά πρότυπα αξιολόγησης περιλαμβάνουν το CityBench, το οποίο επικεντρώνεται σε εφαρμογές έξυπνων πόλεων και αξιολογεί την επεξεργασία ροών αστικών δεδομένων· το SRBench (Sensor RDF Benchmark), το οποίο έχει σχεδιαστεί για την αξιολόγηση μηχανών επεξεργασίας ροών RDF χρησιμοποιώντας δεδομένα αισθητήρων· και το LSBench (Linked Stream Benchmark), το οποίο ελέγχει συστήματα με συνδεδεμένες ροές δεδομένων σε κοινωνικά μέσα.

Σκοπός της παρούσας διπλωματικής εργασίας ήταν να αξιολογήσει αυτό το πρότυπο σε σχέση με τα κρίσιμα σημεία έναντι των άλλων προτύπων που έχουν ήδη αναπτυχθεί από άλλους επιστήμονες. Για αυτό το σκοπό, ο παρακάτω πίνακας παρουσιάζει μια σύγκριση των κρίσιμων σημείων που εντοπίζονται στα διάφορα πρότυπα (οι αριθμοί αντιπροσωπεύουν τον χρόνο που αξιολογείται ένα συγκεκριμένο κρίσιμο σημείο στο πρότυπο). Όπως αποδεικνύεται, τα optional clauses (CP3) και το reasoning (CP4) παραλείπονται από τα περισσότερα benchmarks και μόνο το SRBench τα τηρεί. Τα περισσότερα από αυτά μελετούν το join ordering (CP1), aggregation (CP2), τα φίλτρα (CP6), geospatial predicates (CP8) και τα σύνθετα φίλτρα (CP11), ενώ μόνο δύο από αυτά

μελετούν την ταξινόμηση των αποτελεσμάτων (CP7). Τέλος, όλα τα πρότυπα αγνοούν το parallel execution of joins (CP5), την αναζήτηση πλήρους κειμένου (CP9) και την εξάλειψη διπλότυπων (CP10), ανοίγοντας τον δρόμο για μελλοντικά πρότυπα στον τομέα αυτό.

	CP1	CP2	CP3	CP4	CP5	CP6	CP7	CP8	CP9	CP10	CP11
LSBench	7	2	0	0	0	2	1	0	0	0	1
SRBench	3	4	2	3	0	7	0	6	0	0	0
CSRBench	0	1	0	0	0	6	0	2	0	0	1
CityBench	5	3	0	0	0	2	2	6	0	0	1
YaBench	0	1	0	0	0	6	0	2	0	0	1

Εικόνα 19: Choke points & benchmarks

Στην παρακάτω εικόνα, απεικονίζεται το benchmark που δημιουργήθηκε για την παρούσα εργασία καθώς και ποια choke points ικανοποιεί:

	CP1	CP2	CP3	CP4	CP5	CP6	CP7	CP8	CP9	CP10	CP11
Paper's Benchmark	0	3	1	1	0	9	1	0	4	2	1

Εικόνα 20: Choke points and paper's benchmark

Το σημείο CP8 δεν ικανοποιείται, διότι δεν υπάρχει ανάγκη για χρήση τέτοιων δεδομένων (γεωγραφικό πλάτος και μήκος) για τις ανάγκες του άρθρου. Τα CP1 και CP5 (Διάταξη συνδέσεων και Παράλληλη εκτέλεση συνδέσεων) δεν μπορούν να ικανοποιηθούν, διότι δεν υπάρχει τρόπος να προσδιοριστεί πώς η μηχανή και ο βελτιστοποιητής ερωτημάτων υλοποιούν τις συνδέσεις στις ερωτήσεις, επομένως δεν μπορούν να αξιολογηθούν.

5.2 Future Work

Στο πεδίο των συστημάτων επεξεργασίας ροών RDF (RSP), υπάρχουν αρκετές προκλήσεις και ευκαιρίες για μελλοντική εργασία που ξεχωρίζουν. Ένα αξιοσημείωτο κενό εντοπίζεται στην απουσία ενός ισχυρού οργάνου τυποποίησης βασισμένου στην κοινότητα, γεγονός που οδηγεί σε κατακερματισμό στην υιοθέτηση γλωσσών ερωτημάτων. Πολλά συστήματα εξακολουθούν να βασίζονται σε ιδιόκτητες γλώσσες, και ενώ πρότυπα όπως τα

C-SPARQL και CQELS έχουν κερδίσει κάποια αποδοχή, η υιοθέτησή τους παραμένει περιορισμένη. Ωστόσο, η εισαγωγή του RSP-QL έχει προκαλέσει μια αλλαγή, με περισσότερα συστήματα να αρχίζουν να το υιοθετούν, σηματοδοτώντας μια κίνηση προς μεγαλύτερη τυποποίηση στον τομέα.

Όσον αφορά τις δυνατότητες των συστημάτων, ένας τομέας που απαιτεί σημαντική προσοχή είναι οι σταδιακές ενημερώσεις. Μόνο λίγα συστήματα RSP χειρίζονται αποτελεσματικά τις σταδιακές ενημερώσεις, αναδεικνύοντας μια ευκαιρία για περαιτέρω έρευνα και ανάπτυξη. Επιπλέον, πολλά υπάρχοντα συστήματα διαχωρίζουν την αξιολόγηση ερωτημάτων από τη συλλογιστική, βασιζόμενα σε διακριτά εξαρτήματα για κάθε εργασία. Η ενοποίηση αυτών των διαδικασιών θα οδηγούσε σε πιο προσαρμοστικά και αποδοτικά συστήματα. Η υποστήριξη συλλογιστικής είναι επίσης ένας ανεπαρκώς ανεπτυγμένος τομέας: τα λίγα συστήματα που την προσφέρουν παρέχουν συχνά μόνο βασικές δυνατότητες. Η διεύρυνση της εκφραστικότητας της συλλογιστικής και η εφαρμογή πιο προηγμένων μηχανισμών συνεπαγωγής πέρα από τις απλές προσεγγίσεις που βασίζονται σε γραφήματα, αντιπροσωπεύουν υποσχόμενες ερευνητικές κατευθύνσεις. Επιπλέον, η αξιοπιστία των δεδομένων που λαμβάνονται, μαζί με τις πολιτικές ανίχνευσης και επίλυσης συγκρούσεων, παραμένουν σε μεγάλο βαθμό μη αντιμετωπισμένα ζητήματα στον τομέα.

Η εξερεύνηση των κατανεμημένων συστημάτων στον τομέα RSP βρίσκεται ακόμη σε πρώιμο στάδιο, με μόνο λίγες λύσεις να έχουν προταθεί. Αυτός ο τομέας απαιτεί περισσότερη έρευνα, ειδικά όσον αφορά την ισορροπία μεταξύ κλιμάκωσης και εκφραστικότητας της συλλογιστικής. Για παράδειγμα, ενώ το CQELS Cloud εξετάζει το cloud computing, ο τομέας παραμένει ανεξερεύνητος. Σημειωτέον, μεγάλα συστήματα βάσεων δεδομένων RDF, όπως τα Stardog, GraphDB και Virtuoso, δεν υποστηρίζουν ακόμα γλώσσες συνεχών ερωτημάτων, δημιουργώντας μια ευκαιρία για μελλοντική εμπορική ανάπτυξη.

Ένα ακόμα πρόβλημα είναι η ανάπτυξη εργαλείων και πλατφορμών τα οποία θα είναι φιλικά προς τον χρήστη με σκοπό την καλύτερη ενσωμάτωση και ευρεία αποδοχή των RDF ροών στις ήδη υπάρχουσες υποδομές.

Η συγκριτική αξιολόγηση είναι ένας άλλος κρίσιμος τομέας που απαιτεί βελτίωση. Οι τρέχουσες μέθοδοι συγκριτικής αξιολόγησης στερούνται ανθεκτικότητας, ιδιαίτερα στην αξιολόγηση παράλληλης εκτέλεσης συνενώσεων, αναζήτησης πλήρους κειμένου και

εξάλειψης διπλοτύπων. Η ανάπτυξη καλύτερων και πιο ολοκληρωμένων μεθόδων συγκριτικής αξιολόγησης είναι απαραίτητη για την προώθηση του τομέα RSP και την εξασφάλιση αποτελεσματικής αξιολόγησης και σύγκρισης νέων συστημάτων.

References

- [1] T. B. Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, A. Secret, 1994, “The World Wide Web” <https://dl.acm.org/doi/pdf/10.1145/179606.179671>
- [2] T. B. Lee, R. Cailliau, “World-Wide Web”: <https://cds.cern.ch/record/245440/files/p69.pdf>
- [3] G. Antoniou, F Van Harmelen - 2004, “A Semantic Web Primer”: https://dlwqtxts1xzle7.cloudfront.net/30361978/a_semantic_web_primer-libre.pdf?1390885414=&response-content-disposition=inline%3B+filename%3DA_semantic_web_primer.pdf&Expires=1717695435&Signature=EQGems1sykpc98CJGDoZW1swcEUdM2KBdMQ5n0bx-0VrBpt3s4gFG8ibvY5XXGnP1N-bRVCviUJqbBN7PA8s1IceaBGZeoDJOnTwBGk13W53kzQnDjDQbqPrEPGLC3ngu~bnaIeqomBUD-bWcpEui~gJpcOTaXehe8T9WlljpaZc8hX0lvSfHIOfWetaZzjtWeyj3UOYikO~U1XDP0QlbNnHna4MNRpGbRt1smV4FYvufZezQTr3Wm8JtuzaPebDZnVE-KsLuHgmWWf9wNTdQHGbPjLu1M-JoVeGbnGbL5C0nBpTyZ21mnzNkOhLnl-A90HbweDgMJJo58XIdLkd9w__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA
- [4] CAMBRIDGE SEMANTICS, Boston <https://cambridgesemantics.com/blog/semantic-university/intro-semantic-web/>
- [5] Simple A LLC, All About The Semantic Web <https://simplea.com/Articles/semantic-web>
- [6] L. Ding, T. Finin, “Characterizing the Semantic Web on the Web”: https://link.springer.com/chapter/10.1007/11926078_18
- [7] Michael Uschold, AI Magazine 2003, “Where Are the Semantics in the Semantic Web?”: <https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/1716>
- [8] STEFAN DECKER, PRASENJIT MITRA, AND SERGEY MELNIK, Stanford University, “Framework for the Semantic Web: An RDF Tutorial”: <https://www.andrew.cmu.edu/user/mm6/95-733/PowerPoint/rdftutorial.pdf>
- [9] M. Andrea, Calbimonte, Jean-Paul, Dell’Aglia, Daniele, Balduini, Marco, Brambilla, Marco, Della Valle, Emanuele, Aberer, Karl “TripleWave: Spreading RDF Streams on the Web”, 2016: <https://www.zora.uzh.ch/id/eprint/132903/1/main.pdf>

[10] D. Francesco Barbieri, D. Braga, S. Ceri, E. Della Valle, M. Grossniklaus, “Querying RDF Streams with C-SPARQL”, Politecnico di Milano, Dipartimento di Elettronica e Informazione Milano, Italy, 2010:
<https://kops.uni-konstanz.de/server/api/core/bitstreams/77c7d3d5-ebcd-465f-bc24-59047a2a5cfb/content>

- [11] A. Fall Dia, Z. Kazi-Aoul, A. Boly, Y. Chabchoub, “C-SPARQL Extension for Sampling RDF Graphs Streams”, October 2017:
<https://hal.science/hal-01663811/file/AKDM-Dia%26al.pdf>
- [12] Daniele Dell’Aglia, Jean-Paul Calbimonte, E. Della Valle, O. Corcho, “Towards a Unified Language for RDF Stream Query Processing”, DEIB, Politecnico of Milano, Italy, January 2016: https://link.springer.com/chapter/10.1007/978-3-319-25639-9_48
- [13] R. Tommasini, P. Bonte, F. Ongenaes and E. Della Valle, University of Tartu Estonia, Politecnico di Milano Italy, Ghent University - imec Belgium, “RSP4J: An API for RDF Stream Processing”, 2021: <https://openreview.net/pdf?id=IbXJmD1i2WA>
- [14] Daniele Dell’Aglia, Emanuele Della Valle, Jean-Paul Calbimonte, Oscar Corcho, Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico of Milano, Milano, Italy, “RSP-QL Semantics: A Unifying Query Model to Explain Heterogeneity of RDF Stream Processing Systems” 2014:
https://re.public.polimi.it/bitstream/11311/964309/2/RSP-QL-Semantics-A-Unifying%20Query%20Model%20to%20Explain%20Heterogeneity_11311-964309_Della%20Valle.pdf
- [15] Jóakim v. Kistowski, Jeremy A. Arnold, Karl Huppler, Klaus-Dieter Lange, John L. Henning, Paul Cao, University of Würzburg, IBM Corporation, Hewlett-Packard Company, Oracle, “How to Build a Benchmark”, 2015:
<https://dl.acm.org/doi/pdf/10.1145/2668930.2688819>
- [16] Ying Zhang, Pham Minh Duc, Oscar Corcho, Jean-Paul Calbimonte, “SRBench: A Streaming RDF/SPARQL Benchmark”, Centrum Wiskunde & Informatica, Amsterdam, The Netherlands, Universidad Politecnica de Madrid, Spain, 2012:
https://oa.upm.es/20397/1/INVE_MEM_2012_134355.pdf
- [17] T. Scharrenbach, J. Urbani, A. Margara, E. Della Valle, “Seven Commandments for Benchmarking Semantic Flow Processing Systems”, A. Bernstein, University of Zurich, Switzerland, Vrije Universiteit Amsterdam, The Netherlands, Politecnico di Milano, Italy, May 2013: https://link.springer.com/chapter/10.1007/978-3-642-38288-8_21
- [18] Angles Rojas, R.; Boncz, P.A.; Larriba Pey, J.; Fundulaki, I.; Neumann, T.; Erling, O.; Neubauer, P.; Kotsev, V.; Toma, I., “The linked data benchmark council: a graph and RDF industry benchmarking effort”, May 2014:
<https://research.vu.nl/ws/portalfiles/portal/1133657/22545B.pdf>
- [19] Επιστημονικό άρθρο που επισύναψε ο Κ. Κονδυλάκης: “RDF Stream Processing: A survey”
- [20] Y. Guo, Z. Pan, and J. Heflin. "LUBM: A Benchmark for OWL Knowledge Base Systems." Journal of Web Semantics, 2005: <https://swat.cse.lehigh.edu/pubs/guo05a.pdf>

- [21] Christian Bizer, Andreas Schultz, “The Berlin SPARQL benchmark”, International Journal on Semantic Web and Information Systems, April 2009:
https://www.researchgate.net/publication/220123872_The_Berlin_SPARQL_benchmark
- [22] Schmidt, M., Hornung, T., Meier, M., Pinkel, C., & Lausen, G. (2009). SP2Bench: A SPARQL Performance Benchmark. In Proceedings of the 25th International Conference on Data Engineering (ICDE). IEEE.
<https://www.csd.uoc.gr/~hy561/papers/benchmarking/A%20SPARQL%20Performance%20Benchmark.pdf>
- [23] TPC BENCHMARK (Decision Support) Standard Specification Revision 2.18.0
https://www.tpc.org/tpc_documents_current_versions/pdf/tpc-h_v2.18.0.pdf
- [24] J. Gosling, B. Joy, G. Steele, G. Bracha, A. Buckley, D. Smith, G. Bierman, “The Java® Language Specification Java SE 22 Edition”, Oracle America, Inc, March 2024:
<https://docs.oracle.com/javase/specs/jls/se22/jls22.pdf>
- [25] “RSP4J: An API for RDF Stream Processing”, May 2021:
https://link.springer.com/chapter/10.1007/978-3-030-77385-4_34
- [26] Riccardo Tommasini, Pieter Bonte, “Web Stream Processing with RSP4J”, University of Tartu, Ghent University, June 2021:
https://www.researchgate.net/publication/352796292_Web_stream_processing_with_RSP4J
- [27] Citybench dataset Pollution
<https://github.com/CityBench/Benchmark/blob/master/streams/AarhusPollutionData158324.stream>