ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΡΟΣΔΙΟΡΙΣΜΟΣ ΠΑΡΑΘΥΡΩΝ ΣΕ ΡΕΥΜΑΤΑ ΔΕΔΟΜΕΝΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ηλία Τζωρτζακάκη

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2005

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΡΟΣΔΙΟΡΙΣΜΟΣ ΠΑΡΑΘΥΡΩΝ ΣΕ ΡΕΥΜΑΤΑ ΔΕΔΟΜΕΝΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ηλία Τζωρτζακάκη

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 19η Οκτωβρίου 2005.

.....	.....	.....
Τιμολέων Σελλής	Ιωάννης Βασιλείου	Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π	Επίκ. Καθηγητής Ε.Μ.Π

Αθήνα, Οκτώβριος 2005

.....

ΗΛΙΑΣ Γ. ΤΖΩΡΤΖΑΚΑΚΗΣ

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Πρόλογος

Σε πλήθος σύγχρονων εφαρμογών όπως δίκτυα αισθητήρων, παρακολούθηση κινούμενων αντικειμένων, χρηματιστηριακές εφαρμογές κ.ά. δημιουργούνται ιδιαίτερες συνθήκες ως προς την διαχείριση της εισερχόμενης πληροφορίας. Κοινά χαρακτηριστικά αυτών των εφαρμογών είναι η επεξεργασία δυναμικών ρευμάτων δεδομένων, η διατύπωση ερωτημάτων διαρκείας με έμφαση στην πιο πρόσφατη πληροφορία και η απαίτηση παροχής αποτελεσμάτων σε πραγματικό χρόνο. Αντί λοιπόν ο χρήστης να ζητάει και να «τραβάει» αποτελέσματα από τα υπάρχοντα δεδομένα, στο νέο μοντέλο επεξεργασίας, τα δεδομένα «σπρώχνουν» απαντήσεις στον χρήστη μέσω των ερωτημάτων διαρκείας που παραμένουν ενεργά. Τα συμβατικά συστήματα διαχείρισης βάσεων δεδομένων αδυνατούν να προσαρμοστούν στις νέες συνθήκες, γι' αυτό ο σχεδιασμός οδηγείται στην εξαρχής υλοποίηση συστημάτων διαχείρισης ρευμάτων δεδομένων.

Στις περισσότερες προτάσεις υλοποίησης τέτοιων συστημάτων, εισάγεται η έννοια των παραθύρων μέσω των οποίων επιτυγχάνεται η διαρκής απόσπαση ενός πρόσφατου και πεπερασμένου τμήματος από τα μέχρι τότε στοιχεία των ρευμάτων δεδομένων. Κατ' αυτόν τον τρόπο είναι δυνατόν να δίνονται προοδευτικά απαντήσεις από την επεξεργασία ερωτημάτων (λ.χ. συναθροιστικά), που κανονικά θα έπρεπε ν' αναμένουν την ολοκλήρωση του –πιθανόν ανεξάντλητου– ρεύματος πριν αρχίσουν να παράγουν αποτελέσματα. Τα παράθυρα εξυπηρετούν στην αντιμετώπιση των νέων απαιτήσεων αλλά η χρήση τους ενδέχεται να δημιουργήσει επιπτώσεις στην ακρίβεια των εξαγόμενων αποτελεσμάτων. Οι υπεισερχόμενες προσεγγίσεις προέρχονται από τον περιορισμό της επεξεργασίας στα περιεχόμενα των παράθυρων αντί για το σύνολο των στοιχείων των ρευμάτων.

Σε αυτήν την διπλωματική εργασία αναδεικνύεται η σημασία των παραθύρων στις εφαρμογές ρευμάτων δεδομένων ενώ αρχικά επιχειρείται παρουσίαση των διαφορετικών τύπων ερωτημάτων που επικρατούν σε αυτές τις εφαρμογές. Στην συνέχεια προτείνεται μία μέθοδος αναπαράστασης των ρευμάτων δεδομένων και βάσει αυτής επιδιώκεται η κατηγοριοποίηση και ο αλγεβρικός προσδιορισμός των κυριότερων τύπων παραθύρων. Ακολούθως, αναλύονται διεξοδικά τα κυρίαρχα ζητήματα που επικρατούν στην επεξεργασία πολλαπλών ερωτημάτων διαρκείας πάνω σε ρεύματα δεδομένων, μαζί με προτάσεις από την τρέχουσα βιβλιογραφία για την αντιμετώπιση τους. Τέλος, επιχειρείται η ανάπτυξη σε C++ ενός απλουστευμένου συστήματος διαχείρισης ρευμάτων δεδομένων όπου περιλαμβάνονται υλοποιήσεις παραθυρικών δομών και τελεστών για την επεξεργασία των ρευμάτων. Παρέχεται η δυνατότητα υποβολής προκαθορισμένων ερωτημάτων διαρκείας για διάφορους ρυθμούς άφιξης στοιχείων στα εισερχόμενα ρεύματα και κατά τον τρόπο αυτό διαπιστώνεται η ορθότητα των υλοποιημένων δομών.

Η παρούσα έκδοση του συστήματος ανοίγει προοπτικές για την περαιτέρω επέκτασή του είτε με προσθήκη περισσότερων τελεστών είτε αντιμετωπίζοντας θέματα όπως βελτιστοποίηση ερωτημάτων, αποδοτικότερη χρονοδρομολόγηση ή σχεδιασμός κατάλληλης διεπαφής για σύνθεση και υποβολή ερωτημάτων διαρκείας από τους χρήστες.

Οργάνωση του τόμου

Η συγκεκριμένη εργασία προσεγγίζει το θέμα του προσδιορισμού παραθύρων σε ρεύματα δεδομένων τόσο σε επίπεδο θεωρητικής μελέτης όσο και σε επίπεδο υλοποίησης. Έτσι τα κεφάλαια 2-4 καλύπτουν το θεωρητικό κομμάτι της εργασίας ενώ το κεφάλαιο 5 αναφέρεται στην παρούσα υλοποίηση. Συγκεκριμένα:

Στο 2ο **κεφάλαιο** αναλύονται οι ιδιαιτερότητες των ερωτημάτων που διατυπώνονται σε εφαρμογές ρευμάτων δεδομένων και επιδιώκεται η διάκριση τους σε κατηγορίες.

Στο 3ο **κεφάλαιο** παρουσιάζονται οι κυριότεροι τύποι παραθύρων που προτείνονται στην βιβλιογραφία. Κατατάσσονται σε κατηγορίες ανάλογα με την σημασιολογία τους και επιχειρείται αλγεβρικός προσδιορισμός των περιεχομένων τους. Ακόμη, επιδεικνύεται ένας τρόπος με τον οποίο μπορούν να ενσωματωθούν σε μία γλώσσα ερωταποκρίσεων μορφής SQL.

Στο 4ο **κεφάλαιο** αναλύονται τα βασικότερα ζητήματα της επεξεργασίας των ερωτημάτων διαρκείας. Παρουσιάζονται οι χρησιμοποιούμενοι τελεστές καθώς και τα προβλήματα που παρουσιάζουν κατά την προσαρμογή τους στο μοντέλο επεξεργασίας ρευμάτων δεδομένων. Επίσης, γίνεται ανασκόπηση των κυριότερων τεχνικών βελτιστοποίησης που παρουσιάζονται στην τρέχουσα βιβλιογραφία.

Το 5ο **κεφάλαιο** περιλαμβάνει την τεκμηρίωση και την επεξήγηση του κώδικα της παρούσας υλοποίησης. Παρουσιάζονται τόσο απαιτήσεις και γενικά στοιχεία λειτουργίας του συστήματος όσο και οι δομές που αναπτύχθηκαν. Το κεφάλαιο κλείνει με την παρουσίαση των ερωτημάτων διαρκείας που διατυπώθηκαν κατά την δοκιμαστική λειτουργία του συστήματος.

Στο 6ο **κεφάλαιο** γίνεται ανασκόπηση της παρούσας εργασίας φτάνοντας μέσα από μία λογική αλληλουχία στα συμπεράσματα που προέκυψαν. Ειδικά για το πλαίσιο της υλοποίησης παρουσιάζονται οι μελλοντικές δυνατότητες επέκτασης του συστήματος.

Ευχαριστίες

Ευχαριστώ τον καθηγητή μου κ. Τίμο Σελλή για το θερμό ενδιαφέρον του και τον πολύτιμο χρόνο που διέθεσε για τις συναντήσεις μας. Η καθοδήγηση του και ο σεβασμός στην προσπάθεια μου, υποστήριξαν σε σημαντικό βαθμό την επιτυχή ολοκλήρωση της διπλωματικής μου εργασίας.

Θέλω επίσης να εκφράσω τις ευχαριστίες μου στον Κώστα Πατρούμπα, η συμβολή και η διαθεσιμότητα του οποίου υπήρξαν καθοριστικές στην εκπόνηση της εργασίας αυτής. Με την άριστη γνώση του πάνω στο ευρύτερο πρόβλημα των ρευμάτων δεδομένων, με βοήθησε δίνοντας μου ορθές κατευθύνσεις σε κομβικά σημεία τόσο της θεωρητικής μελέτης όσο και της υλοποίησης του συστήματος.

Η προσωπική μου γνώμη είναι ότι μου δόθηκε η ευκαιρία να συνεργαστώ με δύο πολύ αξιόλογους ανθρώπους και νιώθω πραγματικά τυχερός γι' αυτό.

Σύνοψη

Προσδιορισμός παραθύρων σε ρεύματα δεδομένων

Σε πολλές σύγχρονες εφαρμογές, η πληροφορία λαμβάνει την μορφή ταχύτατα μεταβαλλόμενων ρευμάτων δεδομένων που διακινούνται μέσω δικτύων. Τα ρεύματα δεδομένων αποτελούνται από θεωρητικά άπειρο πλήθος στοιχείων συνήθως υπό την μορφή σχεσιακών πλειάδων με κάποια χρονική σήμανση. Οι πλειάδες του ρεύματος φτάνουν δυναμικά στο σύστημα με άγνωστο ρυθμό ή χρονική διάταξη και τροφοδοτούν ερωτήματα διαρκείας για την άμεση επεξεργασία τους. Γι' αυτό το λόγο, εφαρμόζονται παράθυρα πάνω σε ρεύματα δεδομένων με στόχο την διαρκή απόσπαση πεπερασμένου πλήθους πλειάδων εστιάζοντας στην πιο πρόσφατη πληροφορία, με πιθανές αλλά σαφώς προσδιορισμένες επιπτώσεις στην ακρίβεια των απαντήσεων.

Το ζήτημα του προσδιορισμού παραθύρων σε ρεύματα δεδομένων προσεγγίζεται τόσο σε επίπεδο θεωρητικής μελέτης όσο και σε επίπεδο υλοποίησης. Αρχικά επιδιώκεται η θεμελίωση, η κατηγοριοποίηση και ο αλγεβρικός προσδιορισμός της σημασιολογίας των παραθύρων. Κατόπιν επιχειρείται ο σχεδιασμός και η υλοποίηση ενός απλουστευμένου συστήματος διαχείρισης ρευμάτων δεδομένων με έμφαση στις παραθυρικές δομές και στους κυριότερους τελεστές (προβολή, επιλογή και σύνδεση διοχέτευση). Παρέχεται δυνατότητα διαμόρφωσης φυσικού προσχεδίου εκτέλεσης διαφόρων ερωτημάτων διαρκείας διασυνδέοντας κατάλληλα τις διαθέσιμες δομές, οι οποίες λειτουργούν ως αυτοτελείς οντότητες. Η άφιξη των στοιχείων από διάφορα ρεύματα δεδομένων προσομοιώνεται με ελεγχόμενο ρυθμό, ενώ η χρονοδρομολόγηση όλων των διεργασιών επαφίεται στο λειτουργικό σύστημα. Τέλος, η ορθή λειτουργία του συστήματος επιβεβαιώθηκε στην πράξη δοκιμάζοντας διαθέσιμα σύνολα δεδομένων.

Λέξεις κλειδιά:

Ερώτημα διαρκείας,
Παράθυρα,
Πλειάδα,
Προσχέδιο εκτέλεσης ερωτήματος,
Ρεύμα δεδομένων,
Τελεστής,
Χρονόσημο.

Abstract

Window specification over data streams

In many modern applications, information takes the form of rapidly changing data streams transferred through networks. Data streams consist of possibly unbounded data elements usually in the form of relational tuples with timestamps. Streaming tuples arrive to the system dynamically at irregular rates and unpredictable order and they are used for online processing of continuous queries. For this reason, several window types are applied over data streams in order to obtain a bounded set of tuples focusing on the most resent data, with possible though well determined consequences to the accuracy of the returned answer.

The issue of window specification over data streams is addressed both from a theoretical and an implementation point of view. Firstly, we aim at conceptual analysis, classification and algebraic specification of window semantics. Consequently, we set out to design and implement a simplified data stream management system, emphasizing on window types and basic operators (project, select and pipelined join). Various physical query plans may be constructed by properly combining available operators, each one implemented as an autonomous entity. The arrival rate of streaming elements is simulated with prespecified parameters, whereas the scheduling of all active threads is performed by the underlying operating system. Finally, the correctness of our approach was validated on several available data sets.

Keywords:

Continuous query,
Windows,
Tuple,
Query execution plan,
Data stream,
Operator,
Timestamp.

Πίνακας Περιεχομένων

Κεφάλαιο 1

Εισαγωγή στη διαχείριση ρευμάτων δεδομένων

1.1 Η ανάγκη διαχείρισης δυναμικών δεδομένων

Πλήθος σύγχρονων εφαρμογών εγείρουν την ανάγκη διαφορετικού τρόπου διαχείρισης της πληροφορίας που συλλέγεται σε σχέση με τα συμβατικά συστήματα βάσεων δεδομένων. Οι εφαρμογές αυτές εστιάζουν στις πιο πρόσφατες πληροφορίες που εισέρχονται στο σύστημα. Τυπικά παραδείγματα τέτοιων εφαρμογών αποτελούν δίκτυα αισθητήρων, παρακολούθηση κινούμενων αντικειμένων, εφαρμογές χρηματιστηρίου και εποπτεία κίνησης στο διαδίκτυο. Κοινά χαρακτηριστικά αυτών των εφαρμογών αποτελούν:

- η διαχείριση δυναμικά εισερχόμενης πληροφορίας με τη μορφή ***ρευμάτων δεδομένων*** (*data stream*), με έμφαση στην πιο πρόσφατη πληροφορία,
- η διατύπωση ***ερωτημάτων διαρκείας*** (*continuous queries*), τα οποία απαιτούν online επεξεργασία και πρέπει να επιστρέφουν απαντήσεις σε πραγματικό χρόνο.

Χωρικά κατανεμημένοι αισθητήρες μέτρησης της θερμοκρασίας στέλνουν τις χρονικά προσημασμένες μετρήσεις τους, σε ένα κεντρικό σύστημα όπου παρακολουθείται η εξέλιξη της θερμοκρασίας κατά την διάρκεια της ημέρας. Η διαρκής παρακολούθηση της τροχιάς ενός στόλου οχημάτων είναι απαραίτητη, για να μπορούν να απαντηθούν ερωτήματα εντοπισμού του πλησιέστερου σε κάποια περιοχή ενδιαφέροντος ασθενοφόρου, περιπολικού ή ταξί. Η ανάλυση και η επεξεργασία των διαρκώς μεταβαλλόμενων χρηματοοικονομικών δεικτών σε εφαρμογές χρηματιστηρίου μπορεί να αποτελέσει κριτήριο για αποφάσεις αγοραπωλησίας μετοχών. Στους κόμβους δρομολόγησης του διαδικτύου, η συνεχής επεξεργασία των δυναμικά μεταβαλλόμενων μετρήσεων για τα διακινούμενα πακέτα πληροφορίας, οδηγεί είτε σε αποφάσεις δρομολόγησης είτε στον εντοπισμό κρίσιμων καταστάσεων.

Στα παραπάνω παραδείγματα, οι μετρήσεις των αισθητήρων ή των κόμβων, η θέση των οχημάτων και οι πληροφορίες των μετοχών φτάνουν σε ένα κεντρικό σύστημα ως στοιχεία πληροφορίας κάποιου ρεύματος δεδομένων. Τα στοιχεία του ρεύματος λαμβάνουν συνήθως την μορφή σχεσιακών *πλειάδων* (*tuples*), οι οποίες συνήθως φέρουν κάποιο *χρονόσημο* (*timestamp*) που δηλώνει τον χρόνο παραγωγής τους. Το σύστημα καλείται να επεξεργαστεί αυτές τις πλειάδες, με

στόχο την έγκυρη και έγκαιρη παραγωγή απαντήσεων στα ερωτήματα διαρκείας που διατυπώνονται από τους χρήστες. Ωστόσο δεν υπάρχει άμεσος έλεγχος στην ποσότητα, ή τον ρυθμό άφιξης των στοιχείων από τα εισερχόμενα ρεύματα, γεγονός που δημιουργεί πλήθος επιπρόσθετων προβλημάτων στην επεξεργασία τους.

1.2 Ανεπάρκεια των συμβατικών ΣΔΒΔ

Το μοντέλο επεξεργασίας που επικρατεί στα γνωστά Συστήματα Διαχείρισης Βάσεων Δεδομένων αδυνατεί να ανταποκριθεί στις απαιτήσεις των εφαρμογών ρευμάτων δεδομένων. Σε αυτό το μοντέλο οι χρήστες «τραβάνε» (*pull model*) αποτελέσματα από τα υπάρχοντα δεδομένα στο σύστημα, διατυπώνοντας *ερωτήματα στιγμιότυπου* (*one-time ή snapshot queries*) σε αυτά. Στο νέο μοντέλο που διαμορφώνεται στις εφαρμογές ρευμάτων δεδομένων διακρίνονται δύο έντονες διαφοροποιήσεις:

- Τα δεδομένα δεν είναι εξ αρχής αποθηκευμένα στατικά στο σύστημα. Νέα στοιχεία πληροφορίας παράγονται διαρκώς με ρυθμό που καθορίζουν οι πηγές, πάνω στον οποίο ο χρήστης ή το σύστημα δεν έχει κανένα έλεγχο.
- Τα ερωτήματα διαρκείας που διατυπώνονται παραμένουν ενεργά επί μακρόν και καλούνται να παρέχουν σωστά αποτελέσματα σε κάθε άφιξη στοιχείων από τα εισερχόμενα ρεύματα.

Έτσι στο νέο μοντέλο, τα δεδομένα «σπρώχνουν» (*push model*) απαντήσεις στον χρήστη και στα ερωτήματα διαρκείας που ο ίδιος υπέβαλε. Για χαμηλούς ρυθμούς άφιξης στοιχείων υπό την μορφή ρεύματος, τα κλασικά ΣΔΒΔ θα αντιμετώπιζαν συμβατικά το πρόβλημα με την συντήρηση *υλοποιημένων όψεων* (*materialized views*) και επεξεργασία σε περιόδους χαμηλού φόρτου εργασίας. Ο χαμηλός ρυθμός άφιξης αποτελεί σημαντικό αλλά απαραίτητο περιορισμό μια και μεταφράζεται σε λιγότερο συχνές ενημερώσεις των υλοποιημένων όψεων.

Αρκετές όμως εφαρμογές παρουσιάζουν απαιτήσεις online επεξεργασίας, αφού είναι πολύ πιθανό να έχουν μικρά περιθώρια απόκρισης στην δημιουργία κρίσιμων συνθηκών που σηματοδοτούνται από τα εξαγόμενα αποτελέσματα. Έτσι δεν είναι αποδεκτή η λύση των υλοποιημένων όψεων, όπου ο χρόνος απόκρισης μπορεί να είναι αυθαίρετα μεγάλος ανάλογα με την χρονική στιγμή εμφάνισης χαμηλού φόρτου εργασίας. Μάλιστα τα επιδιωκόμενα περιθώρια απόκρισης του συστήματος μπορεί να είναι τόσο μικρά, ώστε να καθιστούν απαγορευτικό το κόστος αποθήκευσης πληροφορίας σε σκληρούς δίσκους. Άλλωστε το κόστος επεξεργασίας τέτοιων αποθηκών δεδομένων μπορεί από μόνο του να υπερβαίνει κατά πολύ τα περιθώρια της εφαρμογής.

Το γεγονός αυτό αλλάζει τις προδιαγραφόμενες απαιτήσεις ακρίβειας των απαντήσεων. Οι απαντήσεις δεν θα πρέπει να εξαρτώνται από το σύνολο των στοιχείων του ρεύματος, λόγω του κόστους αποθήκευσης και επεξεργασίας που δημιουργείται, έστω και αν αυτό προκαλέσει αρνητικές επιπτώσεις στην ακρίβεια των αποτελεσμάτων. Η εξάρτηση από κάποιο είδος *συνόψεων* (*synopses*) των παλαιοτέρων στοιχείων κρίνεται πολλές φορές αναγκαία, με στόχο την βελτίωση της υπεισερχόμενής προσέγγισης. Επιπρόσθετα η απαίτηση online επεξεργασίας προσανατολίζει τον σχεδιασμό του συστήματος στον κατά το δυνατό περιορισμό του στην κύρια μνήμη, κάτι το οποίο δεν αποτελεί ούτε στόχο ούτε έμφυτο χαρακτηριστικό των κλασικών ΣΔΒΔ.

Η παρουσία ερωτημάτων διαρκείας αλλάζει και τον τρόπο με τον οποίο επιχειρείται η βελτιστοποίηση των ερωτημάτων. Στις μέχρι τώρα προσεγγίσεις γινόταν συμψηφισμός στατιστικών στοιχείων των στατικά αποθηκευμένων σχέσεων και του κόστους επεξεργασίας κάθε τελεστή για την εξαγωγή του *βέλτιστου φυσικού προσχεδίου εκτέλεσης* (*physical query execution plan*) ξεχωριστά για κάθε ερώτημα. Στο μοντέλο που καθιερώνουν τα ρεύματα δεδομένων θα πρέπει να λαμβάνονται υπόψη και τα ερωτήματα διαρκείας τα οποία παραμένουν ενεργά στο σύστημα. Λ.χ. το σύστημα, είναι χρήσιμο να εκμεταλλευθεί κοινά τμήματα στα προσχέδια των διαφόρων ερωτημάτων. Η δράση του συστήματος σε τέτοιες περιπτώσεις είναι πιθανόν να επηρεάζει το προσχέδιο πολλαπλών ερωτημάτων, ενώ θα πρέπει να υποστηρίζεται η δυναμική αλλαγή των προσχεδίων εκτέλεσης, ανάλογα με τις υφιστάμενες συνθήκες (λ.χ. μεταβολές στον ρυθμό άφιξης των στοιχείων).

Κατά συνέπεια, γίνεται αντιληπτό ότι δεν είναι αρκετή οποιαδήποτε προσπάθεια προσαρμογής των κλασικών ΣΔΒΔ στις νέες απαιτήσεις. Ο σχεδιασμός τους πραγματοποιήθηκε με στόχο την αντιμετώπιση αρκετά διαφορετικών συνθηκών, ενώ αρκετές από τις θεμελιακές αρχές που επικρατούν έρχονται σε αντίφαση με το νέο μοντέλο επεξεργασίας που διαμορφώνεται (push model). Πραγματοποιείται λοιπόν στροφή στον σχεδιασμό εξ ολοκλήρου νέων συστημάτων υπό την γενικότερη κατηγορία Συστήματα Διαχείρισης Ρευμάτων Δεδομένων (ΣΔΡΔ), κληρονομώντας όπου χρειάζεται, βασικές αρχές επεξεργασίας της πληροφορίας από τα υπάρχοντα ΣΔΒΔ.

1.3 Γλώσσες ερωταποκρίσεων σε ρεύματα δεδομένων

Η διατύπωση των ερωτημάτων διαρκείας στις εφαρμογές ρευμάτων δεδομένων, συνήθως πραγματοποιείται μέσω κάποιας *δηλωτικής* (*declarative*) γλώσσας μορφής SQL, αφήνοντας στο σύστημα την επιλογή του κατάλληλου *φυσικού προσχεδίου εκτέλεσης* (*physical query plan*). Κατά τον τρόπο αυτό, καθορίζεται το είδος και η διάταξη των τελεστών που θα χρησιμοποιηθούν κατά την εκτέλεση του ερωτήματος. Εναλλακτικά, μπορεί να χρησιμοποιηθεί κάποια *διαδικαστική* (*procedural*) γλώσσα για τον άμεσο προσδιορισμό του φυσικού προσχεδίου εκτέλεσης από τον χρήστη, πιθανόν και με την βοήθεια κάποιας κατάλληλης *γραφικής διεπαφής χρήστη* (*Graphical User Interface*). Όπως όμως αναφέρθηκε, τα ερωτήματα διαρκείας εστιάζουν συνήθως στην επεξεργασία της πιο πρόσφατης πληροφορίας, εγείροντας την ανάγκη αφενός για χρονική σήμανση των στοιχείων, αφετέρου για την υλοποίηση κατάλληλων τελεστών που θα αποσπούν και θα επεξεργάζονται τα πλέον πρόσφατα στοιχεία. Η χρονική σήμανση επιτυγχάνεται με την προσθήκη ενός πεδίου *χρονοσήμου* (*timestamp*) στις πλειάδες των ρευμάτων, ενώ εισάγεται η έννοια των παραθύρων για να δηλωθεί η έμφαση των ερωτημάτων στην πιο πρόσφατη πληροφορία.

1.3.1 Παράθυρα σε ρεύματα δεδομένων

Ο ρόλος των παραθύρων συνίσταται στην απόσπαση πεπερασμένου πλήθους στοιχείων από το ρεύμα δίνοντας πρόσβαση διαρκώς σε περιορισμένο αλλά σαφώς προσδιορισμένο τμήμα του ρεύματος. Τα περιεχόμενα τους τοποθετούνται χρονικά κοντά στα πιο πρόσφατα στοιχεία των ρευμάτων, ενώ επαναπροσδιορίζονται με κάθε νέα πλειάδα που φτάνει σε αυτά, ανάλογα με τους κανόνες και τις παραμέτρους κάθε παραθύρου.

Οι παραθυρικοί τελεστές απαντώνται σε ερωτήματα διαρκείας με στόχο να τροφοδοτήσουν προβληματικούς τελεστές που χρησιμοποιούν όλα τα δεδομένα εισόδου τους για την επιστροφή πλήρους απάντησης, όπως λ.χ. οι *συναθροιστικοί* (*aggregate*) τελεστές *πλήθους* (*COUNT*), *αθροίσματος* (*SUM*), *μέσου όρου* (*AVG*) κ.τ.λ. Όμως όπως εξηγήθηκε σε προηγούμενη ενότητα, δεν είναι ούτε εφικτή ούτε επιθυμητή, η διατήρηση και η επεξεργασία όλων των δεδομένων. Η τροφοδότηση των προβληματικών τελεστών με παράθυρα έχει ως αποτέλεσμα την απεμπλοκή τους, με αντίτιμο την αποδοχή προσεγγίσεων στις επιστρεφόμενες απαντήσεις. Οι τελεστές αυτοί επεξεργάζονται πλέον τα στοιχεία που περιέχονται σε κάθε παράθυρο, εστιάζοντας και αυτοί με την σειρά τους στην πρόσφατη πληροφορία.

Οι εξαγόμενες απαντήσεις προκύπτουν έγκαιρα ως αποτέλεσμα της επεξεργασίας ενός σαφώς μικρότερου συνόλου δεδομένων, αλλά η ακρίβεια τους εξαρτάται από την σημασιολογία των ερωτημάτων. Αν τα ίδια τα ερωτήματα καθορίζουν τον περιορισμό της επεξεργασίας σε ένα κομμάτι της πιο πρόσφατης πληροφορίας, τότε οι απαντήσεις που προκύπτουν είναι ακριβείς και τα παράθυρα λειτουργούν ως μηχανισμός εξυπηρέτησης των απαιτήσεων του ερωτήματος. Στην περίπτωση αυτή (που είναι και η πλέον συνήθης σε αυτές τις εφαρμογές), δεν έχει νόημα η διατήρηση των παλαιοτέρων στοιχείων του ρεύματος καθώς αυτό εξελίσσεται. Αν αντίθετα τα ερωτήματα τίθενται χωρίς να ορίζουν κάποιο περιορισμό στο πλήθος των δεδομένων που επεξεργάζονται, τότε πρέπει να εφαρμοστούν μηχανισμοί διατήρησης *συνόψεων* (*synopses*) για τα παλαιότερα στοιχεία. Στην περίπτωση αυτή τα παράθυρα μαζί με τις συνόψεις οδηγούν στην εξαγωγή προσεγγιστικών απαντήσεων.

Σε κάθε περίπτωση, ο τύπος και οι παράμετροι κάθε παραθύρου προσδιορίζουν με αρκετά σαφή τρόπο τον βαθμό προσέγγισης που τυχόν υπεισέρχεται στις απαντήσεις. Με την χρήση τους

ωστόσο επιτυγχάνονται: η απαίτηση γρήγορης επεξεργασίας, ο περιορισμός σε σαφώς μικρότερο κομμάτι μνήμης και το φιλτράρισμα της εισερχόμενης πληροφορίας έτσι ώστε πάντοτε να στέλνεται για επεξεργασία ένα πρόσφατο κομμάτι της. Κατά τον τρόπο αυτό, αντισταθμίζονται επαρκώς, οι επιπτώσεις στην ακρίβεια των απαντήσεων που πιθανών να προκαλεί η χρήση των παραθύρων.

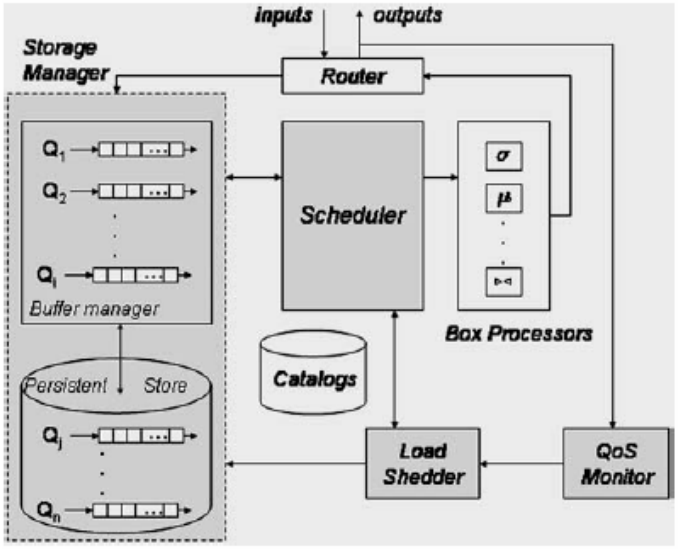
Η δημοφιλέστερη ίσως κατηγορία, είναι αυτή των *κυλιόμενων παράθυρων* (*sliding windows*) με τα οποία παρέχεται πρόσβαση στις πλειάδες που φέρουν χρονόσημο εντός κάποιας χρονικής έκτασης. Παρακάτω διατυπώνεται σε φυσική γλώσσα, ένα απλουστευμένο παράδειγμα ερωτήματος που χρησιμοποιεί έναν συναθροιστικό τελεστή και ένα κυλιόμενο παράθυρο με χρονική έκταση μισής ώρας.

Παράδειγμα 1.1: *Ζητείται η διαρκής επεξεργασία των μετρήσεων θερμότητας που στέλνει ένας αισθητήρας, για την παρακολούθηση του μέσου όρου θερμοκρασίας στο διάστημα της τελευταίας μισής ώρας, μέχρι να ζητηθεί ο τερματισμός αυτού του ερωτήματος.*

1.3.2 Προτεινόμενες γλώσσες ερωταποκρίσεων

Τα παράθυρα αποτελούν βασικό και καινοτόμο μηχανισμό των προτεινόμενων στην βιβλιογραφία ΣΔΡΔ (Aurora, Gigascope, STREAM, TelegraphCQ). Στα πλαίσια υλοποίησης αυτών των συστημάτων έχουν προταθεί οι εξής γλώσσες ερωταποκρίσεων:

- Aurora: SQuAl (**S**tream **Qu**ery **Al**gebra) **[ACC+03]**. Τα ερωτήματα που διατυπώνονται στην γλώσσα SQuAl μετασχηματίζονται σε γράφους κατάλληλα συνδεδεμένων *τόξων* και *κουτιών*. Οι γράφοι μπορούν εναλλακτικά να σχηματιστούν μέσω κατάλληλης γραφικής διεπαφής. Τα *κουτιά* αναπαριστούν τους παρεχόμενους τελεστές, ενώ τα *τόξα* αντιστοιχούν σε ρεύματα δεδομένων και χρησιμοποιούνται για τις συνδέσεις μεταξύ κουτιών. Οι τελεστές λειτουργούν με ρεύματα δεδομένων, ενώ η επεξεργασία στατικών σχέσεων υποστηρίζεται μόνο κατά έμμεσο τρόπο. Τα παράθυρα αποτελούν έμφυτο χαρακτηριστικό κάθε τελεστή που τα χρειάζεται και ορίζονται μέσω κατάλληλων παραμέτρων του τελεστή κατά την διατύπωση του ερωτήματος.
- Gigascope: GSQL **[CJSS03, JMSS05]**. Η γλώσσα αυτή έχει μορφή γλώσσας SQL, αλλά εφαρμόζεται αποκλειστικά σε ρεύματα δεδομένων. Κάθε ρεύμα θεωρείται ότι διαθέτει ένα ή περισσότερα γνωρίσματα βάσει των οποίων μπορεί να οριστεί κάποια χρονική διάταξη. Κατά την διατύπωση των ερωτημάτων ορίζονται *πρωτόκολλα* με την βοήθεια των οποίων: α) ερμηνεύονται τα περιεχόμενα των εισερχόμενων ρευμάτων, β) καθορίζεται ο τρόπος προσδιορισμού της διάταξης τους και γ) προσομοιώνεται η λειτουργία των παραθύρων. Η προσομοίωση των παραθύρων πραγματοποιείται με συνθήκες επιλογής πάνω στα γνωρίσματα που καθορίζουν την διάταξη των στοιχείων σύμφωνα με το επιλεγμένο *πρωτόκολλο*. Τέλος, λαμβάνεται ιδιαίτερη μέριμνα για την ενημέρωση των αποτελεσμάτων ακόμα και κατά την απουσία εισερχόμενων στοιχείων πληροφορίας.
- STREAM: CQL (**C**ontinuous **Q**uery **L**anguage) **[ABW03]**. Σε αυτήν την επέκταση της γλώσσας SQL υποστηρίζεται η επεξεργασία τόσο ρευμάτων δεδομένων όσο και στατικών σχέσεων. Η CQL αποτελείται από τρία δομικά στοιχεία: α) την σχεσιακή γλώσσα ερωταποκρίσεων SQL για τους τελεστές σχεσιακών πινάκων β) μία γλώσσα προσδιορισμού παραθύρων η οποία ακολουθεί συντακτικά στοιχεία της SQL-99 για την μετατροπή των ρευμάτων σε σχεσιακούς πίνακες και γ) τρεις ειδικούς τελεστές για την μετατροπή σχεσιακών πινάκων σε ρεύματα: ISTREAM για τις νέες πλειάδες του προσωρινού σχεσιακού πίνακα, DSTREAM για τις πλειάδες που διαγράφηκαν από τον πίνακα και RSTREAM για την παροχή όλων των πλειάδων του προσωρινού πίνακα.
- TelegraphCQ StreaQueL(**Strea**m **Que**ry **L**anguage) **[CCD+03, GO03]**: Πρόκειται για μία δηλωτική γλώσσα η οποία αποδίδει ιδιαίτερη έμφαση στον προσδιορισμό διαφόρων τύπων παραθύρων. Κατά βάση, στην τυπική σύνταξη ενός ερωτήματος σε SQL προσαρτάται μια επαναληπτική δομή (FOR-loop) δηλώνοντας την ακολουθία των παραθύρων που θα χρησιμοποιηθούν για να απομονώσουν τα στοιχεία των αντίστοιχων ρευμάτων. Πέρα από τα ρεύματα, υποστηρίζεται η επεξεργασία και σχεσιακών πινάκων.



Σχήμα 1.1: *Αρχιτεκτονική του συστήματος Διαχείρισης Ρευμάτων Δεδομένων* ***Aurora****. (Πηγή: [ACC+03])*

1.4 Κυριότερα Συστήματα Διαχείρισης Ρευμάτων Δεδομένων

Στην ενότητα αυτή αναφέρονται κάποια συνοπτικά στοιχεία για τα κυριότερα πρότυπα συστήματα διαχείρισης ρευμάτων δεδομένων (ΣΔΡΔ) που αναπτύσσονται από διάφορες ερευνητικές ομάδες. Οι σχετικές δημοσιεύσεις, αποτέλεσαν σημαντικό τμήμα της βιβλιογραφικής μελέτης που πραγματοποιήθηκε.

1.4.1 Aurora

Το Aurora είναι ένα γενικού σκοπού Σύστημα Διαχείρισης Ρευμάτων Δεδομένων που σχεδιάζεται και υλοποιείται από το 2001 με τη σύμπραξη των Πανεπιστημίων Brandeis και Brown καθώς και του M.I.T **[ACC+03, CCR+03, siteAur]**. Προτείνεται μία προσέγγιση βασισμένη σε ένα γραφικό περιβάλλον με «κουτιά» και «βέλη» θυμίζοντας ένα διάγραμμα ροής δεδομένων αναπαριστώντας το προσχέδιο εκτέλεσης κάποιου ερωτήματος διαρκείας.

Ο *χρονοπρογραμματιστής* (*scheduler*) αποτελεί το συνδετικό κρίκο όλων των τμημάτων της αρχιτεκτονικής του συστήματος, όπως εικονίζεται στο σχήμα 1.1. Αυτός αποφασίζει ποια «κουτιά» θα τρέξουν στη συνέχεια, προσδιορίζοντας τόσο τον αριθμό των πλειάδων που θα δοθούν για επεξεργασία σε κάποιο τελεστή, όσο και η πορεία των στοιχείων διαμέσου των τελεστών προς την έξοδο. Οι πλειάδες που καταλήγουν στην έξοδο παρακολουθούνται μονίμως από τον *Επόπτη Ποιότητας* (*QoS Monitor*), επισημαίνοντας στον χρονοπρογραμματιστή χρήσιμα στοιχεία για τις επιδόσεις του συστήματος.

Υπάρχει επίσης ένας *διαχειριστής αποθήκευσης* (*Storage Manager*) που αναλαμβάνει να οδηγήσει τις πλειάδες σε *ενδιάμεσες ουρές* (*buffer queues*). Αυτό θα συμβεί όταν διαπιστωθεί ότι εξαντλείται η διαθέσιμη ποσότητα μνήμης, κάτι που δεν μπορεί να αποκλειστεί όταν τα ιστορικά στοιχεία που συσσωρεύονται στα σημεία σύνδεσης διογκωθούν υπερβολικά.

Τα στοιχεία του ρεύματος διέρχονται μέσα από το δίκτυο των τελεστών, που μπορεί να θεωρηθεί ως ένας άκυκλος κατευθυνόμενος γράφος εκτελούμενων λειτουργιών. Σε κάθε εισερχόμενο στοιχείο προσδίδεται ένα μοναδικό αναγνωριστικό συστήματος τύπου χρονοσήμου (timestamp), το οποίο χρησιμοποιείται για εποπτεία της παρεχόμενης ποιότητας υπηρεσίας από το σύστημα. Τελικά τα στοιχεία καταλήγουν στην έξοδο, οι πλειάδες της οποίας τροφοδοτούν κάποια εφαρμογή κατά ασύγχρονο τρόπο, ανάλογα δηλαδή προς το ρυθμό παραγωγής τους.

Οι χρήστες έχουν την ευχέρεια να ορίσουν διάφορες απαιτήσεις σχετικά με την *ποιότητα υπηρεσιών* (*Quality of Service QoS*) των ερωτημάτων που θέτουν στο σύστημα. Το σύστημα με την σειρά του αντεπεξέρχεται στις απαιτήσεις αυτές με την δημιουργία και παρακολούθηση *διαγραμμάτων ποιότητας υπηρεσίας* (*QoS graphs*) όσον αφορά τους χρόνους απόκρισης, την ακρίβεια των αποτελεσμάτων και την βαρύτητα των εξαγόμενων απαντήσεων. Ο χρήστης έχει την δυνατότητα να μεταβάλλει την προτεραιότητα εκτέλεσης των ερωτημάτων, με στόχο την μεγιστοποίηση του αθροιστικού μεγέθους της *ποιότητας υπηρεσιών* (*QoS*) που λαμβάνεται, συνυπολογίζοντας όλους τους τελεστές («κουτιά»).

Οι προδιαγραφές των χρηστών και τα διαγράμματα ποιότητας υπηρεσίας χρησιμεύουν στη συνέχεια για να αποφασιστεί με ποιο τρόπο και σε ποιες περιστάσεις θα αποβληθεί κάποιο μέρος των δεδομένων, *ελαφρύνοντας το φόρτο του συστήματος* (*load shedding*) επηρεάζοντας την ακρίβεια των απαντήσεων. Έτσι, εάν το σύστημα διαπιστώσει ότι η τεχνική αυτή δεν αποδίδει αρκετά, μπορεί να προσπαθήσει να αναδιατάξει το δίκτυο των τελεστών, χρησιμοποιώντας γνωστές μεθόδους βελτιστοποίησης, όπως αυτές που στηρίζονται στην *αντιμεταθετικότητα* (*commutativity*) τελεστών.

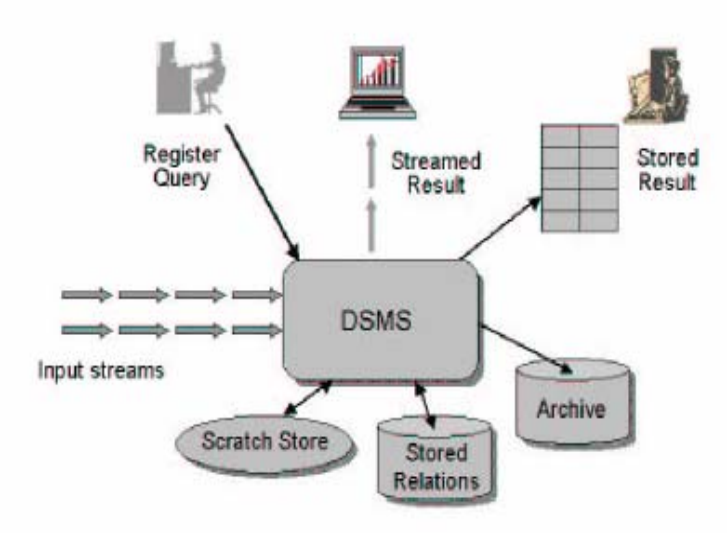
1.4.2 Gigascope

Πρόκειται για ένα σύστημα διαχείρισης ρευμάτων δεδομένων που αναπτύσσεται από την AT&T σε συνεργασία με το Πανεπιστήμιο Carnegie Mellon **[CJSS03, JMSS05]**. Το Gigascope χρησιμοποιείται στη διαχείριση δικτύου τηλεπικοινωνιών ή υπολογιστών και εφαρμόζεται (μέχρι στιγμής πειραματικά) στην εποπτεία δικτύων οπτικών ινών υψηλών ταχυτήτων, με ικανότητα μεταφοράς εκατομμυρίων πακέτων το δευτερόλεπτο. Το σύστημα έχει χρησιμοποιηθεί σε διάφορες εφαρμογές, όπως ανάλυση δικτύων, ανάλυση πρωτοκόλλων, ανίχνευση μη εξουσιοδοτημένης διείσδυσης σε δίκτυο, καθώς και ερευνητικούς σκοπούς (λ.χ. video streams). Στο σύστημα αυτό η διακινούμενη πληροφορία παίρνει την μορφή ρεύματος δεδομένων αποκλείοντας την επεξεργασία στατικών σχέσεων.

Προκειμένου να είναι εφικτή η επεξεργασία των ρευμάτων εισόδου και η συνακόλουθη μετατροπή τους σε ρεύματα εξόδου, σε όλες τις πλειάδες προσκολλάται χρονόσημο έπειτα από την εκτέλεση των σχετικών ερωτημάτων διαρκείας. Ειδικά τα ερωτήματα που άπτονται της ανάλυσης του δικτύου κάνουν ρητή αναφορά στο χρονικό προσδιορισμό των στοιχείων. Σε αντίθεση με παρόμοιες εφαρμογές τηλεπικοινωνιών που χρησιμοποιούν διαδικαστικές γλώσσες ερωταποκρίσεων, στο Gigascope προτιμήθηκε για λόγους ευελιξίας μια παραλλαγή της SQL (GSQL). Πρόκειται για μια συνεπτυγμένη μορφή της SQL (λ.χ. δεν επιτρέπονται outer joins), αλλά με ορισμένους πρόσθετους τελεστές (όπως ο τελεστής merge για τη συγχώνευση ρευμάτων δεδομένων, διατηρώντας όμως τη διάταξη των χρονοσήμων). Οι χρήστες, μέσω ενός μηχανισμού δήλωσης *όψεων* (*views*) έχουν τη δυνατότητα ορισμού συναρτήσεων ή τελεστών για εξειδικευμένες λειτουργίες, οι οποίες κατόπιν μπορούν να τεθούν στη διάθεση και των υπολοίπων χρηστών. Όταν υποβάλλονται ερωτήματα στο σύστημα, περνούν από τη διαδικασία *μεταγλώττισης* (*compiler*) και εξάγονται τμήματα κώδικα σε C και C++, τα οποία και τελικά εκτελούνται, επιστρέφοντας στους χρήστες τα εξαγόμενα ρεύματα δεδομένων. Τα αποτελέσματα των ερωτημάτων μπορούν να διοχετευτούν σε *αποθήκες δεδομένων* (*data warehouses*) για περαιτέρω επεξεργασία.

1.4.3 STREAM (STanford stREam datA Management)

Το STREAM είναι ένα Σύστημα Διαχείρισης Ρευμάτων Δεδομένων γενικού σκοπού, το οποίο αναπτύσσεται από το 2001 στο Πανεπιστήμιο Stanford **[ABB+03, ABW03, BBD+02, BW01, CCR+03, siteStr]**. Εγκαταλείποντας οποιαδήποτε απόπειρα προσαρμογής κάποιου υπάρχοντος ΣΔΒΔ ώστε να ανταποκριθεί στις απαιτήσεις που θέτουν τα χαρακτηριστικά των ρευμάτων δεδομένων και η φύση των ερωτημάτων διαρκείας, οι προσπάθειες επικεντρώθηκαν στη συγκρότηση ενός ολοκληρωμένου πρωτότυπου ΣΔΡΔ. Η ανάπτυξη έχει τρεις κυρίως στόχους: (1) Ένα ευέλικτο τρόπο *διεπαφής* (*interface*) προκειμένου να διευκολύνεται η ανάγνωση και η εγγραφή ρευμάτων δεδομένων, ως μέρος μιας ιεραρχικής διαχείρισης του αποθηκευτικού χώρου. (2) Την αποτελεσματική επεξεργασία των ερωτημάτων διαρκείας που διατυπώνονται σε SQL ή με τελεστές



Σχήμα 1.2: *Η αρχιτεκτονική του συστήματος STREAM (Πηγή: [ABB+03])*

της σχεσιακής άλγεβρας, συμπεριλαμβανομένων των *συναθροιστικών* (*aggregation*). (3) Ένα περιβάλλον API για την υποβολή των ερωτημάτων διαρκείας και τη λήψη των απαντήσεων σ' αυτά.

Στο σχήμα 1.2, παρουσιάζεται σε απλοποιημένη μορφή η αρχιτεκτονική σύλληψη του συστήματος. Παρόλο που βασική προτεραιότητα αποτελεί η online επεξεργασία των δεδομένων, δεν μπορεί να αποκλειστεί το ενδεχόμενο ορισμένες εφαρμογές να προϋποθέτουν *μόνιμη αρχειοθέτηση* (*Archive*) κάποιων στοιχείων για μεταγενέστερη επεξεργασία. Επιπλέον, διάφορα ερωτήματα διαρκείας τυπικά απαιτούν την τήρηση κάποιας *ενδιάμεσης κατάστασης* (*Scratch Store*), η οποία μπορεί να φυλάσσεται στη μνήμη (τακτική που ακολουθεί το STREAM) ή ακόμη και στο δίσκο.

Οι χρήστες έχουν δυνατότητα υποβολής ερωτημάτων διαρκείας, των οποίων η εκτέλεση παρατείνεται μέχρις ότου απενεργοποιηθούν. Τα αποτελέσματα που προκύπτουν μπορεί να διοχετεύονται σε εφαρμογές ως άλλα *ρεύματα δεδομένων* (*Streamed Result*), αλλά μπορεί να θεωρηθούν και ως κάποιας μορφής υλοποιημένες όψεις, δηλαδή σχεσιακοί πίνακες που ενημερώνονται με την πάροδο του χρόνου (*Stored Result*).

Τα ερωτήματα υποβάλλονται με χρήση της ειδικά διαμορφωμένης *δηλωτικής* (*declarative*) γλώσσας ερωταποκρίσεων *CQL* (*Continuous Query Language*). Συντακτικά, η CQL είναι υπερσύνολο της SQL, με προσθήκη εξειδικευμένων δομών για την υποστήριξη *κυλιόμενων παραθύρων* (*sliding windows*) και *δειγματοληψίας* δεδομένων (*sampling*). Σημαντικό στοιχείο της γλώσσας αποτελεί το γεγονός ότι η σημασιολογία των ερωτημάτων διαρκείας επιβάλλεται να αντιμετωπίζει με παρόμοιο τρόπο τόσο τα δεδομένα των ρευμάτων όσο κι εκείνα που αντλούνται από στατικές σχέσεις.

Όταν ένα ερώτημα διαρκείας υποβάλλεται στο σύστημα, μετασχηματίζεται στο κατάλληλο προσχέδιο εκτέλεσης, διαφορετικό για κάθε ερώτημα. Εναλλακτικά, τα προσχέδια μπορούν να υποβληθούν στο σύστημα απευθείας, με χρήση ενός γραφικού περιβάλλοντος, παρακάμπτοντας τη διατύπωση ερωτημάτων με την CQL. Αυτό το περιβάλλον προσφέρει τη δυνατότητα συγχώνευσης παρόμοιων προσχεδίων εκτέλεσης ή έστω κάποιων μερών τους. Το γραφικό περιβάλλον στηρίζεται στο γεγονός ότι τα ερωτήματα διαρκείας μπορούν να παρασταθούν με τη βοήθεια δομών στην κύρια μνήμη και να τοποθετηθούν σε αρχεία XML. Συνεπώς, κάποιοι ειδικά εξουσιοδοτημένοι χρήστες μπορούν να επέμβουν σ' αυτά τα αρχεία, δημιουργώντας, τροποποιώντας ή μεταφέροντας στοιχεία από το ένα στο άλλο, πριν τα θέσουν στο σύστημα.

Ένα προσχέδιο εκτέλεσης ερωτήματος αποτελείται από τελεστές αλληλοσυνδεόμενους με ουρές και τροφοδοτούμενους από συνόψεις δεδομένων. Οι τελεστές δέχονται δεδομένα από τις ουρές εισόδου, επεξεργάζονται τις πλειάδες βάσει της σημασιολογίας τους και παραδίδουν τις πλειάδες του αποτελέσματος σε μια - μοναδική για τον κάθε τελεστή - ουρά εξόδου. Εκτός από τους γνωστούς

σχεσιακούς τελεστές, για ορισμένους έχουν αναπτυχθεί και οι παραθυρικές εκδοχές τους (λ.χ. για τη σύνδεση ρευμάτων), καθώς και τελεστές δειγματοληψίας. Οι ενδιάμεσες ουρές καθορίζουν τα μονοπάτια που ακολουθούν οι πλειάδες κατά τη διάρκεια της εκτέλεσής τους. Τέλος, οι *συνόψεις* (*synopses*) δεδομένων χρησιμοποιούνται για να εξάγουν κάποιες περιλήψεις στοιχείων των ρευμάτων που έχουν παρέλθει από ορισμένους ενδιάμεσους τελεστές. Αυτές οι συνόψεις, που συνήθως βασίζονται σε κυλιόμενα παράθυρα, θα αξιοποιηθούν κατά το μελλοντικό υπολογισμό που θα διεξάγει ο τελεστής (λ.χ. εάν πρόκειται για τελεστή σύνδεσης, μπορεί να διατηρούνται συνοπτικά στοιχεία για το καθένα από τα δύο ρεύματα που πρόκειται να συσχετιστούν).

Η εκτέλεση των ερωτημάτων ρυθμίζεται από έναν καθολικό *χρονοπρογραμματιστή* (*global scheduler*). Στην τρέχουσα υλοποίηση του συστήματος, χρησιμοποιείται ένα σχήμα round-robin για να προχωρά απρόσκοπτα η εκτέλεση σε όσους τελεστές είναι έτοιμοι, αν και φυσικά μπορεί να υλοποιηθούν περισσότερο πολύπλοκες τεχνικές.

Η οπτικοποίηση πληροφοριών σχετικών με τα ερωτήματα, την εκτέλεσή τους και την κατανομή των πόρων του συστήματος, είναι πολύ σημαντική για τους διαχειριστές του συστήματος. Με αυτόν τον τρόπο, μπορούν να ρυθμίσουν κατάλληλα την απόδοση του ΣΔΡΔ, αν και το ίδιο από μόνο του θα πρέπει να ανταποκρίνεται σε μεταβαλλόμενες συνθήκες, που προκύπτουν τόσο από το πλήθος των ερωτημάτων, όσο και από τα χαρακτηριστικά των ρευμάτων. Οι παρεχόμενες δυνατότητες αναφέρονται στην τροποποίηση - κατά το χρόνο εκτέλεσης – της κατανομής της μνήμης (λ.χ. μεταξύ συνόψεων), της δομής των προσχεδίων εκτέλεσης (λ.χ. αλλάζοντας τον τύπο της σύνοψης που χρησιμοποιείται από κάποιον τελεστή σύνδεσης), καθώς και της πολιτικής χρονοδρομολόγησης ανάμεσα στις διαθέσιμες εναλλακτικές.

Η μνήμη του συστήματος μοιράζεται δυναμικά μεταξύ των συνόψεων και των ουρών στα σχέδια εκτέλεσης των ερωτημάτων, μαζί με την *ενδιάμεση μνήμη* (*buffers*) που διευκολύνουν το χειρισμό ρευμάτων που καταφτάνουν στο σύστημα, καθώς και μιας μορφής *λανθάνουσας μνήμης* (*cache*) που χρησιμοποιείται για δεδομένα που τηρούνται στο δίσκο.

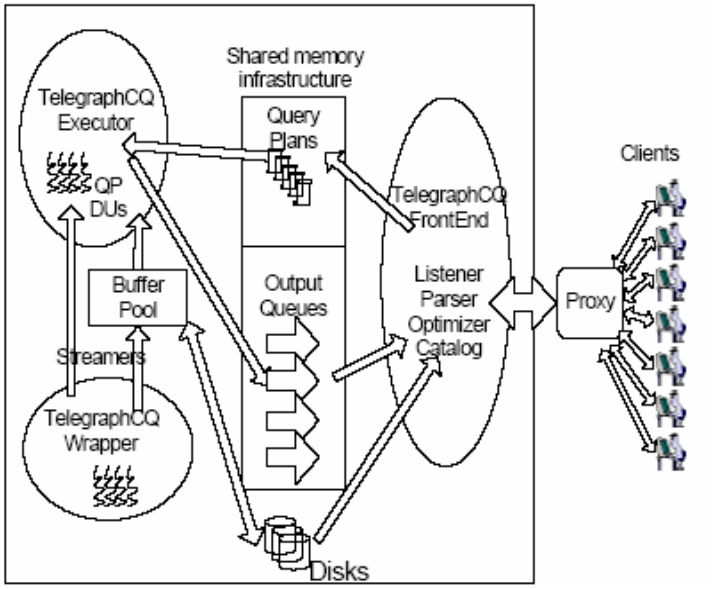
Συμπερασματικά, κεντρικό πρόβλημα στο σύστημα STREAM αποτελεί η αποτελεσματική εκτέλεση των ερωτημάτων διαρκείας σε καθεστώς περιορισμένης ποσότητας μνήμης. Ως επί το πλείστον, το ενδιαφέρον εστιάζεται στον υπολογισμό προσεγγιστικών απαντήσεων και στην ανάλυση των απαιτήσεων σε μνήμη των ερωτημάτων που τίθενται.

1.4.4 TelegraphCQ

Το σύστημα αυτό αποτελεί μετεξέλιξη του πρωτοτύπου Telegraph που συντονίζεται από το Πανεπιστήμιο Berkeley ήδη από το 2000, με κύριο στόχο την ανάπτυξη μιας αρχιτεκτονικής προσαρμοζόμενης στη ροή των δεδομένων κυρίως σε δικτυακά περιβάλλοντα, με έμφαση στα δίκτυα αισθητήρων **[CCD+03, siteTCQ]**. Ωστόσο, η αναγκαιότητα αντιμετώπισης των ζητημάτων που ανακύπτουν ως προς το χειρισμό ρευμάτων δεδομένων οδήγησε σε εξαρχής σχεδιασμό και (από το 2002) στην υλοποίηση του TelegraphCQ, διαχωρίζοντάς το από το ευρύτερο αντικείμενο του Telegraph. Η οπτική της προσέγγισης των ρευμάτων δεδομένων στο TelegraphCQ παρουσιάζει μια πολύ ενδιαφέρουσα πρωτοτυπία: θεωρείται ότι δεν είναι μόνο τα δεδομένα που εμφανίζουν μεταβλητότητα και ρέουν μέσα στο δίκτυο, αλλά και τα ίδια τα ερωτήματα διαρκείας μπορούν να παρομοιαστούν με ρεύματα, μιας και τόσο ο αριθμός τους όσο και η δομή τους αλλάζει κατά απρόβλεπτο τρόπο με την πάροδο του χρόνου.

Η κινητικότητα λοιπόν τόσο των δεδομένων όσο και των ερωτημάτων που τα αφορούν έδωσαν το έναυσμα ώστε η προσέγγιση που ακολουθείται από το TelegraphCQ να περιστραφεί γύρω από την *προσαρμοστικότητα τελεστών* (*adaptivity of operators*). Η διαχείριση των δεδομένων στο σύστημα αποβλέπει στην ικανότητά του να εξελίσσεται και να προσαρμόζεται ταχέως σε δραστικές αλλαγές που αναφέρονται στη διαθεσιμότητα δεδομένων, στο περιεχόμενο τους, στα χαρακτηριστικά του ίδιου του συστήματος ή του δικτύου που το τροφοδοτεί, και φυσικά στις απαιτήσεις των χρηστών.

Η ανάπτυξη στηρίχθηκε αρχικά στην προσαρμογή στοιχείων της αρχιτεκτονικής της PostgreSQL προγραμματίζοντας σε C/C++, ώστε να καταστεί εφικτή η από κοινού επεξεργασία ερωτημάτων διαρκείας επί ρευμάτων δεδομένων. Πολλά τμήματα του κώδικα της PostgreSQL χρησιμοποιήθηκαν, άλλα με ελάχιστες και άλλα με σημαντικές τροποποιήσεις, κυρίως αυτά που



Σχήμα 1.3: *Η αρχιτεκτονική του συστήματος TelegraphCQ (Πηγή: [CCD+03])*

αφορούν το *εξωτερικό μέρος* (*Front End*) της επικοινωνίας με τους χρήστες και τα ερωτήματα που θέτουν στο σύστημα. Στο σχήμα 1.3, με τις ελλείψεις απεικονίζονται οι κύριες διαδικασίες που συναποτελούν τον *εξυπηρετητή* (*server*) του TelegraphCQ (Front End, Executor, Wrapper), ενώ προβλέπεται ειδικός χώρος *κοινής μνήμης* (*shared memory*).

Στο εξωτερικό επίπεδο, ο Postmaster της PostgreSQL ξεκινάει κάποια νέα διεργασία μόλις αντιληφθεί αίτημα για νέα σύνδεση με το σύστημα. Επειδή κάθε σύνδεση μπορεί να έχει πολλούς ανοιχτούς *δρομείς* (*cursors*), χρησιμοποιείται ένας *αντιπρόσωπος* (*proxy*) για την συγκέντρωση των διάσπαρτων ερωτημάτων από τους χρήστες, οπότε είναι δυνατόν να ενεργοποιηθούν πολλοί *δρομείς* (*cursors*) με μια μόνο σύνδεση. Ο *Ακροατής* (Listener) υποδέχεται τα ερωτήματα διαρκείας, τα οποία *αναλύονται συντακτικά* (*Parser*) και *βελτιστοποιούνται* (*Optimizer*) προκειμένου να προκύψει ένα *προσαρμοζόμενο προσχέδιο εκτέλεσης* (*adaptive query plan*).

Κάθε προσχέδιο περιλαμβάνει και τελεστές που έχουν ήδη αναπτυχθεί στα πλαίσια του Telegraph, με κύριο στόχο την προσαρμοστικότητα στις εκάστοτε καταστάσεις. Έτσι, τα Eddies είναι τα *υποτμήματα* (*modules*) που αποφασίζουν με ποιο τρόπο τα δεδομένα πρέπει να διοχετεύονται αδιαλείπτως στους τελεστές των ερωτημάτων πλειάδα προς πλειάδα. Τα Flux (Fault-tolerant Load-balancing eXchange) δρομολογούν τις πλειάδες μεταξύ των επεξεργαστών μιας *συστοιχίας* (*cluster*) ώστε να επιτύχουν παραλληλισμό της εκτέλεσης με *εξισορρόπηση φόρτου* (*load balancing*) και *ανοχή σε σφάλματα* (*fault-tolerance*). Αυτά τα πρόσθετα *υποτμήματα* δεν μπορούν να ξεχωρίσουν αρχιτεκτονικά από τους άλλους συμβατικούς τελεστές, αφού απλώς απορροφούν και στη συνέχεια παράγουν πλειάδες. Ωστόσο, ο συνδετικός κρίκος τους είναι τα Fjords, μιας μορφής API που επιτρέπουν την ενδοεπικοινωνία μεταξύ αυτών των *τμημάτων*, ώστε να σχηματιστεί το τελικό προσχέδιο εκτέλεσης του ερωτήματος. Έπειτα, τα προσχέδια διοχετεύονται δυναμικά σε μια ουρά που έχει δημιουργηθεί στο κοινό τμήμα μνήμης του συστήματος. Απ' εκεί, ο *Εκτελεστής* (*Executor*) επιλέγει συνεχώς τα πλέον πρόσφατα προσχέδια και τα προσθέτει στα ήδη εκτελούμενα ερωτήματα. Τα αποτελέσματα της επεξεργασίας περιέρχονται σε ουρές εξόδου στο κοινό τμήμα της μνήμης, απ' όπου ο *Ακροατής* (*Listener*) τα κατευθύνει στον *αντιπρόσωπο* (*proxy*) για να διανεμηθούν τελικά στους χρήστες.

Εφόσον αναμένεται ότι το σύστημα θα υποστηρίξει ένα μεγάλο αριθμό ερωτημάτων, δεν κρίνεται σκόπιμο να δημιουργηθεί καινούργια διεργασία (*thread*) για το καθένα. Εν τούτοις, κρίνεται επιθυμητό η παρουσία πολλαπλών διεργασιών οι οποίες θα μπορούν να εκτελεστούν παράλληλα, να εντάσσεται στις δυνατότητες του TelegraphCQ. Κάθε διεργασία αποτελεί και ένα διαφορετικό «*Αντικείμενο Εκτέλεσης*» (*Execution Object*), που περιλαμβάνει έναν

χρονοπρογραμματιστή (*scheduler*), μια ή περισσότερες *ουρές γεγονότων* (*event queues*) και μια σειρά από μη προβλέψιμες αφηρημένες «*Ενότητες Αποστολής*» (*Dispatch Units*), δηλαδή οντότητες με κοινά στοιχεία που θα σταλούν προς εκτέλεση στο σύστημα. Γι' αυτό το λόγο, στον *Εκτελεστή* (*Executor*) τα ερωτήματα διακρίνονται σε κατηγορίες, ανάλογα με την ευχέρεια που εμφανίζουν για κοινή εκτέλεση με άλλα παρόμοια, βάσει επικαλύψεων στα ρεύματα δεδομένων και τους πίνακες που εμπλέκονται στο καθένα.

Τέλος, υπάρχει ο μηχανισμός του Wrapper στον οποίο ανατίθεται η προσκόμιση των δεδομένων των ρευμάτων στο σύστημα. Οι κύριες δυσκολίες στο σημείο αυτό, είναι αφενός μεν η αποφυγή ανασταλτικών φαινομένων, λ.χ. επιβράδυνση της εκτέλεσης εξαιτίας χρονοβόρων αναμονών για δεδομένα, αφετέρου δε η ελάττωση των προσπελάσεων στο δίσκο. Γι' αυτό, το συγκεκριμένο υποσύστημα του TelegraphCQ εκτελείται ως χωριστή διαδικασία, με χρήση ειδικών *μη* ανασταλτικών δομών, όπως τα Fjords, ενεργοποιώντας μια σειρά διεργασιών ώστε τα δεδομένα να εισάγονται και να εξάγονται αποτελεσματικά. Κατ' αυτόν τον τρόπο, επιτυγχάνεται δυνατότητα διαχείρισης διαφόρων τύπων πηγών δεδομένων, δηλαδή στοιχείων που είτε απαιτούνται από το σύστημα (pull model) είτε προσκομίζονται σ' αυτό (push model). Όταν τα δεδομένα εισέλθουν μέσω του Wrapper προωθούνται στον *Εκτελεστή* για επεξεργασία με τη διαμεσολάβηση των λεγόμενων Streamers. Οι τελευταίοι τα μετατρέπουν σε πλειάδες στις δομές της *ενδιάμεσης μνήμης* (*buffer pool*), κι αν ο χώρος δεν επαρκεί, τα καταχωρούν ακόμη και στο δίσκο. Οι πλειάδες που προκύπτουν προσπελαύνονται από τον *Εκτελεστή* μέσω ενός τελεστή *σάρωσης* (*scanner*) που η συμπεριφορά του ελέγχεται από τις δομές παραθύρων που έχουν ενσωματωθεί στα ερωτήματα.

1.5 Πλαίσιο εργασίας

Από το ευρύτερο πεδίο έρευνας στα ρεύματα δεδομένων, η συγκεκριμένη διπλωματική εργασία εστιάζει στα παράθυρα που συνοδεύουν την διατύπωση και την εκτέλεση ερωτημάτων διαρκείας. Οι κύριοι άξονες της παρούσας μελέτης περιλαμβάνουν:

- την κατηγοριοποίηση και τον αλγεβρικό προσδιορισμό των πιο διαδεδομένων τύπων παραθύρων.
- την υλοποίηση των κυριότερων παραθυρικών δομών χρησιμοποιώντας τους παραπάνω κανόνες επιδιώκοντας κατ' αυτόν τον τρόπο την επικύρωσή τους.
- την υλοποίηση επιλεγμένων τελεστών της σχεσιακής άλγεβρας κατά την προσαρμογή τους στο μοντέλο των ρευμάτων δεδομένων. Συγκεκριμένα θα υλοποιηθούν οι βασικότεροι τελεστές των κλασικών σχεσιακών βάσεων δεδομένων, δηλαδή *επιλογή* (*selection*), *προβολή* (*projection*) και *σύνδεση* (*join*).
- την δημιουργία φυσικών προσχεδίων εκτέλεσης ερωτημάτων διαρκείας με χρήση των παραπάνω δομών στα πλαίσια υλοποίησης ενός απλουστευμένου Συστήματος Διαχείρισης Ρευμάτων Δεδομένων
- προσομοίωση των ρευμάτων δεδομένων για τις ανάγκες της εφαρμογής. Η άφιξη των στοιχείων πρέπει να γίνεται με ελεγχόμενο ρυθμό για να προσομοιωθούν διαφορετικές συνθήκες λειτουργίας του συστήματος ενώ πρέπει να υποστηρίζονται οι βασικοί τύποι δεδομένων (ακέραιοι, δεκαδικοί, χαρακτήρες και συμβολοσειρές).
- αξιολόγηση και επικύρωση ορθής λειτουργίας του συστήματος ελέγχοντας τις απαντήσεις σε πρότυπα ερωτήματα διαρκείας πάνω σε ενδεικτικά διαθέσιμα σύνολα δεδομένων.

Κεφάλαιο 2

Ερωτήματα σε ρεύματα δεδομένων

2.1 Εισαγωγή

Στο κεφάλαιο αυτό θα αναδειχθούν οι ιδιαιτερότητες που παρουσιάζουν τα ερωτήματα πάνω σε ρεύματα δεδομένων και θα επιχειρηθεί η διάκριση τους σε κατηγορίες. Η κατηγοριοποίηση προχωράει κατά επίπεδα κάνοντας έναν αρχικό διαχωρισμό των ερωτημάτων σε ομοειδείς ομάδες, ενώ στην συνέχεια αναλύονται οι κυριότεροι τύποι ερωτημάτων.

Συνήθως, γίνεται διάκριση των ερωτημάτων σε δύο αρκετά γενικές κατηγορίες:

- *ερωτήματα στιγμιοτύπου* (*one-time ή snapshot queries*) και
- *ερωτήματα διαρκείας* (*continuous queries*)

Τα ερωτήματα στιγμιοτύπου απαντώνται κυρίως σε Συστήματα Διαχείρισης Βάσεων Δεδομένων. Διατυπώνονται από τον χρήστη, εφαρμόζονται πάνω στα τρέχοντα δεδομένα της βάσης και τερματίζουν αφού επιστρέψουν τις απαντήσεις τους. Δεν αποκλείεται ωστόσο και η παρουσία τους σε εφαρμογές ρευμάτων δεδομένων, οπότε εφαρμόζονται πάνω στο τρέχον *στιγμιότυπο* του ρεύματος. Ο όρος *στιγμιότυπο* του ρεύματος χρησιμοποιείται για να δηλώσει όλα τα στοιχεία του ρεύματος, τα οποία διατηρεί η εφαρμογή την χρονική στιγμή υποβολής του ερωτήματος. Έτσι, ένα ενδεικτικό ερώτημα στιγμιοτύπου, διατυπωμένο σε φυσική γλώσσα είναι το εξής:

Παράδειγμα 2.1: *Ζητείται ο υπολογισμός του τρέχοντος μέσου όρου θερμοκρασίας μίας πόλης, με βάση τις μετρήσεις χωρικά κατανεμημένων αισθητήρων.*

Στο παραπάνω παράδειγμα θεωρείται ότι κάθε αισθητήρας στέλνει στο σύστημα τις μετρήσεις του στην μορφή ξεχωριστού ρεύματος δεδομένων. Για τον υπολογισμό του ζητούμενου μέσου όρου, χρησιμοποιούνται οι πιο πρόσφατες μετρήσεις από τα ρεύματα των αισθητήρων που υπάγονται στην επικράτεια της πόλης ενδιαφέροντος.

Σε αντίθεση με τα ερωτήματα στιγμιοτύπου, τα ερωτήματα διαρκείας παραμένουν διαρκώς ενεργά στο σύστημα από την στιγμή υποβολής τους μέχρι τον τερματισμό τους από τον χρήστη. Αποτελούν την πιο συνήθη και ενδιαφέρουσα κατηγορία ερωτημάτων στα Συστήματα Διαχείρισης

Ρευμάτων Δεδομένων, ενώ υπολογίζονται διαρκώς καθώς νέα στοιχεία του ρεύματος φτάνουν στο σύστημα χωρίς να αποκλείεται ωστόσο και η χρήση σχεσιακών πινάκων κατά την επεξεργασία. Έτσι, σε μία εφαρμογή εποπτείας της κίνησης στο διαδίκτυο μπορεί να διατυπωθεί το ακόλουθο ερώτημα διαρκείας:

Παράδειγμα 2.2: *Ζητείται ο διαρκής υπολογισμός του πλήθους των χρηστών που συνδέονται σε έναν εξυπηρετητή, καθώς και η ενημέρωση για την περίπτωση όπου ο αριθμός αυτός υπερβεί ένα προκαθορισμένο κατώφλι.*

Για την εκτέλεση του παραπάνω ερωτήματος, πραγματοποιείται σε κάθε χρονική στιγμή υπολογισμός του ζητούμενου πλήθους και στην συνέχεια πραγματοποιείται σύγκριση με το ζητούμενο κατώφλι. Το αποτέλεσμα της σύγκρισης, μπορεί να ενεργοποιήσει την κλήση κάποιας συνάρτησης, για την ενημέρωση των διαχειριστών και την λήψη των απαραίτητων μέτρων.

Πέρα όμως από την διάρκεια, εξίσου σημαντική διαφορά των δύο κατηγοριών παρουσιάζεται στην μεταβλητότητα των συνθηκών που επικρατούν στο σύστημα κατά την επεξεργασία των ερωτημάτων. Στα *ερωτήματα στιγμιοτύπου* οι συνθήκες παραμένουν σταθερές σε αντίθεση με τα *ερωτήματα διαρκείας* όπου κατά την εξέλιξη των ρευμάτων μπορεί να μεταβληθούν: η διαθέσιμη μνήμη, ο ρυθμός άφιξης στοιχείων ή άλλα χαρακτηριστικά των ρευμάτων πάνω στα οποία εφαρμόζεται το ερώτημα. Έτσι στα *ερωτήματα διαρκείας* τόσο τα δεδομένα όσο και οι συνθήκες μεταβάλλονται δυναμικά ενώ τα *ερωτήματα στιγμιοτύπου* εφαρμόζονται σε πιο στατικό περιβάλλον.

Επίσης, μπορεί να πραγματοποιηθεί μία ευρύτερη κατηγοριοποίηση με βάση την χρονική στιγμή υποβολής των ερωτημάτων. Έτσι έχουμε ερωτήματα με χρονική στιγμή υποβολής :

- *προκαθορισμένη* (*predefined queries*) ή
- *περιστασιακή* (*ad hoc*)

Η κατηγοριοποίηση αυτή αφορά κυρίως τις διαδικασίες βελτιστοποίησης και χρονοδρομολόγησης, και για τον λόγο αυτό δεν αποτελεί αντικείμενο μελέτης της παρούσας εργασίας. Και στις δύο αυτές κατηγορίες μπορούν να ενταχθούν τόσο ερωτήματα στιγμιοτύπου όσο και ερωτήματα διαρκείας. Στην συνέχεια αναλύονται εκτενέστερα τα βασικά χαρακτηριστικά των ερωτημάτων διαρκείας

2.2 Ερωτήματα διαρκείας

Υπάρχουν διάφορα κριτήρια για την περαιτέρω κατηγοριοποίηση των ερωτημάτων διαρκείας όπως: ο τρόπος με τον οποίο υπολογίζονται, το επίπεδο ακρίβειας των επιστρεφόμενων απαντήσεων, η μορφή των εξαγόμενων απαντήσεων κ.α. Στην συνέχεια εφαρμόζοντας τα κριτήρια αυτά, πραγματοποιείται πολλαπλή κατηγοριοποίηση των ερωτημάτων διαρκείας, ενώ στις υποενότητες που ακολουθούν γίνεται εκτενέστερη αναφορά στις ομάδες που προκύπτουν.

Έτσι ανάλογα με τον χρόνο που ζητούνται απαντήσεις σε διατυπωμένα ερωτήματα έχουμε:

- *ενεργά ερωτήματα* (*active queries*) και
- *ανενεργά ερωτήματα* (*inactive queries*).

Ανάλογα με την μορφή των παρεχόμενων αποτελεσμάτων έχουμε:

- απαντήσεις σε μορφή ρευμάτων δεδομένων και
- απαντήσεις στην μορφή *υλοποιημένων όψεων* (*materialized views*).

Ο τρόπος υπολογισμού των απαντήσεων διακρίνει τα ερωτήματα σε:

- *μονότονα ερωτήματα* (*monotonic queries*) και
- *μη – μονότονα ερωτήματα* (*non – monotonic queries*).

Ανάλογα με το επίπεδο ακρίβειας των απαντήσεων διακρίνονται:

- ερωτήματα με ακριβείς απαντήσεις και
- ερωτήματα με προσεγγιστικές απαντήσεις.

Επισημαίνεται ότι στην περίπτωση που το σύστημα οδηγείται στην επιβολή προσεγγίσεων στις επιστρεφόμενες απαντήσεις, προκύπτει η ανάγκη προσδιορισμού των ερωτημάτων που θα υποστούν τις συνέπειες αυτές. Το ζήτημα αυτό αποκτά ιδιαίτερη σημασία όταν κάποια ερωτήματα κρίνονται πιο σημαντικά από τα υπόλοιπα και στα οποία υπάρχει περισσότερη ανάγκη για έγκυρες και έγκαιρες απαντήσεις. Αυτή η διαφορά κρισιμότητας που παρουσιάζουν τα ερωτήματα μεταξύ τους επισημαίνεται με τον μηχανισμό απόδοσης ***βαρών*** σε αυτά – μία διαδικασία η οποία μπορεί να πραγματοποιείται από τον χρήστη. Έτσι καθορίζονται προτεραιότητες εκτέλεσης μεταξύ των ερωτημάτων, περιορίζοντας σε σημαντικό βαθμό τις όποιες προσεγγίσεις προκύπτουν στα λιγότερο σημαντικά ερωτήματα.

2.2.1 Ενεργά και ανενεργά ερωτήματα

Τα ενεργά ερωτήματα διαρκείας είναι αυτά τα οποία έχουν τεθεί στο σύστημα και καλούνται διαρκώς να επιστρέφουν ενημερωμένες απαντήσεις, καθώς τα εισερχόμενα ρεύματα δεδομένων εξελίσσονται. Υπάρχουν ωστόσο και ερωτήματα διαρκείας η εκτέλεση των οποίων έχει προσωρινά ανασταλεί.

Αυτά τα ανενεργά ερωτήματα έχουν διατυπωθεί και ενυπάρχουν στο σύστημα παρά το γεγονός ότι δεν επεξεργάζονται δεδομένα και δεν παρέχουν απαντήσεις. Μπορούν ωστόσο να αλλάξουν κατάσταση και να τεθούν ενεργά οποιαδήποτε χρονική στιγμή, είτε προκαθορισμένη είτε έπειτα από απόφαση κάποιου χρήστη. Η ύπαρξή τους στο σύστημα όσο παραμένουν ανενεργά, μπορεί να επηρεάσει τις αποφάσεις του συστήματος τόσο για τα προσχέδια εκτέλεσης των υπολοίπων ενεργών ερωτημάτων όσο και για την κατανομή των διαθέσιμων πόρων.

2.2.2 Ρεύματα δεδομένων έναντι υλοποιημένων όψεων

Τα ερωτήματα διαρκείας μπορούν να παρέχουν τις απαντήσεις τους είτε σε μορφή ρευμάτων δεδομένων είτε ως *υλοποιημένες όψεις* (*materialized views*) υπό την μορφή σχεσιακών πινάκων που ανανεώνονται με τον χρόνο. Σημειώνεται ότι και οι δύο μορφές πρέπει να παρέχουν τις ίδιες ακριβώς απαντήσεις, ωστόσο δεν θεωρούνται ισότιμες. Οι ιδιαιτερότητες που παρουσιάζει η καθεμία, σε συνδυασμό με την αναμενόμενη συμπεριφορά της απάντησης του ερωτήματος οδηγεί στην σωστή επιλογή μεταξύ των δύο δυνατών μορφών.

Στην μορφή του ρεύματος δεδομένων πρέπει να ληφθεί ιδιαίτερη μέριμνα για την περίπτωση όπου η έλευση νέας πλειάδας προκαλεί την διαγραφή ή την ενημέρωση κάποιας από τις υπάρχουσες πλειάδες εξόδου. Η πιο ενδιαφέρουσα πρόταση για το πρόβλημα αυτό είναι η προσθήκη στο ρεύμα *αρνητικών πλειάδων* (*negative tuples*) **[GHM+05]** ενημερώνοντας τον χρήστη για αυτές τις αλλαγές. Το κόστος αυτής της προσέγγισης είναι ο χειρισμός περισσότερων πλειάδων, ενώ το πρόβλημα επιδεινώνεται σε περιπτώσεις συχνών διαγραφών ή ενημερώσεων. Αντίθετα η προσθήκη νέων πλειάδων στο ρεύμα της απάντησης δεν παρουσιάζει κάποια επιπλοκή.

Στις υλοποιημένες όψεις η άφιξη νέας πλειάδας στο ρεύμα οδηγεί είτε στον επαναπροσδιορισμό όλων των αποτελεσμάτων από την αρχή είτε στην *προοδευτική* (*incremental*) ενημέρωση της όψης, εξετάζοντας μόνο τις νέες πλειάδες. Η *προοδευτική* ενημέρωση είναι φυσικά επιθυμητή, όχι όμως πάντοτε εφικτή. Αυτό εξαρτάται από το ίδιο το ερώτημα και τους τελεστές που χρησιμοποιεί. Τα προβλήματα των υλοποιημένων όψεων, εστιάζονται κυρίως στην περίπτωση όπου η προοδευτική ενημέρωση δεν είναι εφικτή και υπάρχει συνεχής προσθήκη νέων απαντήσεων. Επίσης, ιδιαίτερης σημασίας είναι το θέμα αποθήκευσης ικανής ποσότητας πληροφορίας ώστε να είναι εφικτή η διατήρηση της όψης με όποιον τρόπο και αν υπολογίζεται. Ωστόσο η ενημέρωση ή η διαγραφή υπαρχόντων απαντήσεων δεν παρουσιάζει κάποιο επιπρόσθετο κόστος.

Έτσι γίνεται αντιληπτό, ότι η μορφή του ρεύματος συνηθίζεται σε περιπτώσεις όπου στην απάντηση του ερωτήματος παρουσιάζεται υψηλός ρυθμός προσθήκης νέων πλειάδων συγκριτικά με τον ρυθμό διαγραφής ή ενημέρωσης άλλων υφιστάμενων (παράδειγμα 2.3). Στην αντίθετη περίπτωση, όταν δηλαδή ο ρυθμός διαγραφών ή ενημερώσεων είναι υψηλότερος συγκριτικά με τον ρυθμό παραγωγής νέων πλειάδων, προτιμάται η μορφή της υλοποιημένης όψης (παράδειγμα 2.4).

Παράδειγμα 2.3: *Ζητείται ο διαρκής προσδιορισμός των 10 μετοχών με την μεγαλύτερη άνοδο, στο διάστημα των τελευταίων 10 λεπτών.*

Παράδειγμα 2.4: *Ζητείται κάθε μία ώρα η κατασκευή μιας λίστας με όλες τις μετοχές που διαπραγματεύθηκαν στο διάστημα αυτό.*

2.3 Μονοτονία ερωτημάτων διαρκείας

Η ύπαρξη μονοτονίας στα ερωτήματα αποτελεί μία ιδιαίτερα δολική ιδιότητα, η οποία δηλώνει τα εξής:

1) Στην απάντηση των ερωτημάτων καταγράφονται μόνο *προσθήκες πλειάδων* (*append only*) χωρίς να παρουσιάζεται ανάγκη για διαγραφές ή ενημερώσεις.
2) Σε κάθε χρονική στιγμή η απάντηση είναι υπερσύνολο των παρελθόντων στιγμιοτύπων της
3) Αρκεί σε κάθε χρονική στιγμή ο υπολογισμός του ερωτήματος πάνω στις νέες πλειάδες που προστέθηκαν στο ρεύμα εισόδου. Οι πλειάδες που προκύπτουν από την απάντηση απλώς προστίθενται στην έξοδο.

Αλγεβρικά η μονοτονία των ερωτημάτων εκφράζεται ως εξής :

$$A(Q,\tau) = \left(\bigcup_{t=1}^{\tau} (A(Q,t) - A(Q,t-1)) \right) \cup A(Q,0)$$

όπου με $A(Q,t)$ συμβολίζεται το σύνολο των πλειάδων της απάντησης του ερωτήματος διαρκείας Q την χρονική στιγμή t, με τ η παρούσα χρονική στιγμή, με 0 η χρονική στιγμή ενεργοποίησης του ερωτήματος ενώ με t-1 συμβολίζεται η προηγούμενη χρονική στιγμή. Στον παραπάνω τύπο φαίνεται ότι για τον υπολογισμό της απάντησης $A(Q,\tau)$ σε κάθε χρονική στιγμή τ, πρέπει αρχικά να υπολογιστεί το σύνολο της απάντησης $A(Q,0)$ σύμφωνα με τα τρέχοντα περιεχόμενα του ρεύματος. Ακολούθως, αρκεί ο εμπλουτισμός αυτού του συνόλου με τις απαντήσεις που προκύπτουν σε κάθε χρονική στιγμή t. Επισημαίνεται με τον τρόπο αυτό ότι, για κάθε πλειάδα που προστίθεται στο σύνολο της απάντησης, δεν θα προκύψει ποτέ ανάγκη διαγραφής ή γενικότερα τροποποίησης της.

Μία περαιτέρω εξειδίκευση της ιδιότητας της μονοτονίας προκύπτει όταν οι απαντήσεις σε μονότονα ερωτήματα διαρκείας μπορούν να υπολογίζονται *προοδευτικά* (*incrementally*). Έτσι για τον υπολογισμό του ερωτήματος σε κάθε χρονική στιγμή αρκεί να επεξεργαστούν μόνο οι νέες πλειάδες που προστέθηκαν στο εισερχόμενο ρεύμα από την τελευταία χρονική στιγμή υπολογισμού του ερωτήματος. Στα μονότονα ερωτήματα, η επεξεργασία των νέων πλειάδων που προστέθηκαν στο ρεύμα δεν αποκλείει την χρήση και παλαιοτέρων στοιχείων από το ίδιο ή άλλα εισερχόμενα ρεύματα.

Χαρακτηριστικό παράδειγμα προοδευτικά υπολογιζόμενου (άρα και μονότονου) ερωτήματος διαρκείας, αποτελεί ένα ερώτημα που περιλαμβάνει έναν μόνο τελεστή *προβολής* (*projection*) και τροφοδοτείται από ένα *σωρευτικό* (*append only*) ρεύμα δεδομένων. Σε κάθε χρονική στιγμή τα μόνα στοιχεία που απαιτεί η επεξεργασία του ερωτήματος είναι οι νέες πλειάδες που προστέθηκαν στο ρεύμα εισόδου του, από τις οποίες απλώς αφαιρούνται κάποια γνωρίσματα.

Αντίθετα ένα παράδειγμα μονότονου ερωτήματος διαρκείας αλλά όχι προοδευτικά υπολογιζόμενου, αποτελεί ένας τελεστής *σύνδεσης* (*join*) ο οποίος συνδέεται απευθείας σε δύο σωρευτικά ρεύματα δεδομένων. Σε κάθε χρονική στιγμή το ερώτημα πρέπει να υπολογιστεί για τις νέες πλειάδες που έφτασαν στα ρεύματα εισόδου του. Ωστόσο, για κάθε νέα πλειάδα από οποιοδήποτε ρεύμα εισόδου χρησιμοποιούνται κατά την επεξεργασία όλες οι πλειάδες – παλιές και νέες – του άλλου ρεύματος εισόδου. Έτσι το ερώτημα αυτό, παρότι μονότονο, δεν μπορεί να χαρακτηριστεί προοδευτικά υπολογιζόμενο. Τέτοιο ερώτημα διαρκείας όπως θα εξηγηθεί στο κεφάλαιο 4 παρουσιάζει αρκετά προβλήματα και δεν χρησιμοποιείται στην πράξη. Η χρήση του ωστόσο σε αυτό το σημείο αναδεικνύει την διαφορά μεταξύ μονότονων και προοδευτικά υπολογιζόμενων ερωτημάτων διαρκείας.

2.3.1 Μη μονότονα ερωτήματα

Ερωτήματα διαρκείας τα οποία δεν φέρουν την ιδιότητα της μονοτονίας, υπάγονται στην αρκετά γενική κατηγορία των *μη μονότονων* (*non – monotonic*) ερωτημάτων. Στα μη μονότονα ερωτήματα το σύνολο της απάντησης παύει να υπολογίζεται αποκλειστικά με προσθήκες νέων πλειάδων. Η έλευση κάποιας πλειάδας στο ρεύμα εισόδου ενός ερωτήματος μπορεί να οδηγήσει στην διαγραφή ή στην ενημέρωση υφιστάμενων απαντήσεων από το ρεύμα ή την υλοποιημένη όψη της εξόδου. Χαρακτηριστική ιδιότητα των ερωτημάτων αυτών είναι ότι σε κάθε χρονική στιγμή η απάντηση πιθανόν να χρειάζεται να υπολογιστεί από την αρχή.

Αλγεβρικά αυτό εκφράζεται ως εξής:

$$A(Q,\tau) = \bigcup_{t=0}^{\tau} A(Q,t)$$

όπου με $A(Q,t)$ συμβολίζεται το σύνολο των πλειάδων της απάντησης του ερωτήματος διαρκείας Q την χρονική στιγμή t, με τ η παρούσα χρονική στιγμή και με 0 η χρονική στιγμή ενεργοποίησης του ερωτήματος διαρκείας. Η διαφοροποίηση σχετικά με την αντίστοιχη έκφραση των μονότονων ερωτημάτων είναι εμφανής, δηλώνοντας την αδυναμία αποδοτικής αξιοποίησης των απαντήσεων προγενέστερων χρονικών στιγμών. Με αυτόν τον τρόπο εκφράζεται και η δυνατότητα διαγραφών και ενημερώσεων από το σύνολο της απάντησης.

Χαρακτηριστικές ομάδες μη μονότονων ερωτημάτων σύμφωνα με τους **[TGNO92]** είναι :

- Ερωτήματα που περιέχουν κριτήρια επιλογής βάσει της τρέχουσας χρονικής στιγμής. Συγκεκριμένα, πρόβλημα δημιουργείται όταν στην διατύπωση του ερωτήματος περιέχεται κάποια έκφρασή όπως:

```
SELECT...
FROM  ...
WHERE attr > getCurrentTime()
```

  όπου `attr` δηλώνει κάποιο γνώρισμα του ρεύματος εισόδου ενώ `getCurrentTime()` είναι μία συνάρτηση που επιστρέφει την τρέχουσα χρονική στιγμή. Με την πάροδο του χρόνου, καθώς αυξάνει η επιστρεφόμενη τιμή της `getCurrentTime()` προκύπτει η ανάγκη διαγραφής από την απάντηση ορισμένων πλειάδων που παύουν να πληρούν την συνθήκη επιλογής.
- Ερωτήματα στην διατύπωση των οποίων περιέχεται η έκφραση NOT EXISTS της SQL. Η σχέση ή το ρεύμα στο οποίο αναφέρεται η έκφραση NOT EXISTS μεταβάλλεται ανεξάρτητα από το ερώτημα, με αποτέλεσμα να προκύψει ανάγκη αντίστοιχης μεταβολής στην απάντηση του ερωτήματος είτε με προσθήκη είτε με διαγραφή στοιχείων.

Τα μη μονότονα ερωτήματα διαρκείας μπορούν να διακριθούν περαιτέρω με κριτήριο το μοτίβο που ακολουθείται κατά τις διαγραφές ή ενημερώσεις της απάντησης (*update pattern*) **[GO05]**. Το μοτίβο αναφέρεται τόσο στην σειρά με την οποία πραγματοποιούνται οι διαγραφές, όσο και στο ενδεχόμενο η διαγραφή να προϋποθέτει ή όχι την εισαγωγή *αρνητικών πλειάδων* (*negative tuples*) **[GHM+05]**. Ο ρόλος των αρνητικών πλειάδων έγκειται στην ακύρωση κάποιων ήδη υφιστάμενων πλειάδων της απάντησης, ενώ παρουσιάζονται διεξοδικότερα στο 4ο κεφάλαιο.

Διακρίνονται τρεις κατηγορίες τροποποιήσεων της απάντησης:

- Διαγραφές ή ενημερώσεις πλειάδων που πραγματοποιούνται με την ίδια σειρά με την οποία προστέθηκαν στην απάντηση οι πλειάδες αυτές. Δεν χρειάζεται αποθήκευση κάποιας επιπρόσθετης πληροφορίας για την πραγματοποίηση τους, αφού δεν μεταβάλλεται η διάταξη των στοιχείων της απάντησης και δεν απαιτείται η εισαγωγή αρνητικών πλειάδων. Παραδείγματα τέτοιων ερωτημάτων αποτελούν η *επιλογή* (*selection*) ή η *προβολή* (*projection*) πάνω σε στοιχεία ενός κυλιόμενου παραθύρου, το οποίο παρέχει διαρκώς πρόσβαση σε μία συγκεκριμένη χρονική έκταση του ρεύματος, ενώ τα περιεχόμενα του ανανεώνονται καθώς το ρεύμα εξελίσσεται. Τα ερωτήματα αυτής της κατηγορίας χαρακτηρίζονται ***ασθενέστερα μη–μονότονα ερωτήματα*** (***weakest non–monotonic queries***).

- Διαγραφές ή ενημερώσεις πλειάδων μπορεί να πραγματοποιούνται με διαφορετική σειρά από αυτήν με την οποία προστέθηκαν στην απάντηση, αλλά με την δυνατότητα πρόβλεψης του χρόνου διαγραφής τους, αποφεύγοντας την εισαγωγή αρνητικών πλειάδων. Παραδείγματα τέτοιων ερωτημάτων αποτελούν *η σύνδεση* (*join*) *ή η απαλοιφή διπλοτύπων* (*duplicate elimination*) πάνω από κυλιόμενα παράθυρα. Ερωτήματα αυτής της κατηγορίας χαρακτηρίζονται ***ασθενή μη μονότονα ερωτήματα*** (***weak non−monotonic queries***).
- Διαγραφές ή ενημερώσεις πλειάδων που πραγματοποιούνται σε απρόβλεπτες χρονικές στιγμές και με τυχαία σειρά. Χαρακτηριστικό είναι το παράδειγμα της συνολοθεωρητικής διαφοράς ανάμεσα στα περιεχόμενα δύο παραθύρων. Τα ερωτήματα αυτής της κατηγορίας χαρακτηρίζονται ***αυστηρά μη μονότονα ερωτήματα*** (***strict non−monotonic queries***).

Για την κατάταξη των μη μονότονων ερωτημάτων διαρκείας σε κάποια από αυτές τις κατηγορίες δεν αρκεί η γνώση των χρησιμοποιούμενων τελεστών. Οι ίδιοι τελεστές κατατάσσονται σε διαφορετικές κατηγορίες ανάλογα με την μορφή αποθήκευσης των δεδομένων που τους τροφοδοτούν. Έτσι π.χ. ο τελεστής σύνδεσης κατατάσσεται στην κατηγορία των *ασθενών μη μονότονων ερωτημάτων* όταν εφαρμόζεται στα περιεχόμενα κυλιόμενων παραθύρων, ενώ εντάσσεται στην κατηγορία των *αυστηρά μη μονότονων ερωτημάτων* όταν τροφοδοτείται από τουλάχιστον έναν σχεσιακό πίνακα δεδομένων ή υλοποιημένη όψη.

2.4 Ακριβείς και προσεγγιστικές απαντήσεις σε ερωτήματα διαρκείας

Στα κλασικά Συστήματα Διαχείρισης Βάσεων Δεδομένων η ακρίβεια των απαντήσεων θεωρείται δεδομένη. Σε αυτό συντελούν οι σταθερές συνθήκες που επικρατούν κατά την εκτέλεση των ερωτημάτων, το εκ των προτέρων γνωστό πλήθος των δεδομένων που επεξεργάζονται και το γεγονός ότι διατυπώνονται *ερωτήματα στιγμιοτύπου* αντί για *ερωτήματα διαρκείας*. Στις εφαρμογές ρευμάτων δεδομένων ωστόσο οι απαντήσεις στα ερωτήματα που τίθενται δεν είναι πάντοτε ακριβείς.

Αυτό οφείλεται στους ακόλουθους παράγοντες:

- Τα δεδομένα προς επεξεργασία δεν έχουν την μορφή στατικών σχέσεων αλλά ρευμάτων δεδομένων, με πιθανά απεριόριστο μέγεθος και με απρόβλεπτους ρυθμούς άφιξης στοιχείων.
- Η ακριβής απάντηση οποιουδήποτε τύπου ερωτήματος απαιτεί διαθέσιμα όλα τα στοιχεία των εισερχόμενων ρευμάτων.
- Οι περιορισμοί του συστήματος τόσο σε διαθέσιμη μνήμη όσο και σε επεξεργαστική ισχύ.
- Το ενδεχόμενο ταυτόχρονης εκτέλεσης πολλών τέτοιων ερωτημάτων
- Οι απαιτήσεις των εφαρμογών ρευμάτων δεδομένων για γρήγορη επεξεργασία και online απαντήσεις στα διατυπωθέντα ερωτήματα (λ.χ. χρηματιστήριο, παρακολούθηση δικτυακών δεδομένων κ.τ.λ.).

Καθοριστικοί λοιπόν παράγοντες για την ακρίβεια των απαντήσεων είναι: η διατύπωση του εκάστοτε ερωτήματος, οι υλοποιήσεις των τελεστών και οι διαθέσιμοι πόροι του συστήματος. Σε γενικές γραμμές είναι επιθυμητή μία διαρκής ανταλλαγή μεταξύ ακρίβειας αποτελεσμάτων και διαθέσιμων πόρων. Πρέπει είτε από την διατύπωση του ερωτήματος είτε από την υλοποίηση των τελεστών που συμμετέχουν σε αυτό να διασφαλίζεται ότι οι διαθέσιμοι πόροι του συστήματος επαρκούν για τον υπολογισμό της απάντησης με την μέγιστη δυνατή ακρίβεια.

Στις εφαρμογές ρευμάτων δεδομένων είναι αρκετά σύνηθες τα ερωτήματα που διατυπώνονται να εστιάζουν σε περιορισμένο τμήμα της πληροφορίας του ρεύματος. Έτσι είναι πιθανόν το ίδιο το ερώτημα (με την βοήθεια παραθυρικών δομών που αναπτύσσονται στα πλαίσια αυτής της εργασίας) να περιορίζει το πλήθος των στοιχείων που καλείται να επεξεργαστεί, καθιστώντας περισσότερο προσιτό τον στόχο για ακριβείς απαντήσεις.

Μία χαρακτηριστική κατηγορία ερωτημάτων που συνήθως μπορούν να απαντηθούν με ακρίβεια, είναι αυτή των *προοδευτικά* (*incrementally*) υπολογιζόμενων ερωτημάτων, τα οποία μπορούν να επεξεργάζονται κάθε νέα πλειάδα ξεχωριστά και ανεξάρτητα από τις υπόλοιπες. Η μόνη περίπτωση έλλειψης ακρίβειας προκύπτει όταν επικρατούν ακραίες συνθήκες στο σύστημα (λ.χ.

υψηλός ρυθμός άφιξης στοιχείων) μια και τα ερωτήματα αυτά γενικά προβάλλουν τις χαμηλότερες απαιτήσεις σε πόρους συστήματος.

Στις επόμενες υποενότητες παρουσιάζονται οι κυριότερες τεχνικές που αν και επιτυγχάνουν εξοικονόμηση πόρων συστήματος, οδηγούν ωστόσο σε προσεγγιστικές απαντήσεις. Για καθεμία από αυτές τις τεχνικές, παρουσιάζονται οι περιστάσεις υπό τις οποίες επιλέγονται αυτές έναντι των άλλων.

2.4.1 *Απόρριψη φόρτου* (*Load shedding*)

Η μέθοδος αυτή οδηγεί στην απόρριψη πλειάδων είτε από το πρωτογενές ρεύμα, είτε από ενδιάμεσα αποτελέσματα του ερωτήματος. Η χρήση αυτής της τεχνικής, ενδείκνυται σε δύο κυρίως περιπτώσεις:

- Όταν για αρκετό χρονικό διάστημα παρατηρείται υψηλός ρυθμός άφιξης νέων δεδομένων, που υπερβαίνει τις επεξεργαστικές δυνατότητες του συστήματος.
- Στην περίπτωση που κάποιος τελεστής βρεθεί με μεγάλο πλήθος πλειάδων στην είσοδο του

Η απόρριψη ορισμένων πλειάδων επιλέγεται είτε με στόχο την ανταπόκριση του συστήματος στις απαιτήσεις που θέτουν τα ερωτήματα για online επεξεργασία και απαντήσεις σε πραγματικό χρόνο, είτε για λόγους εξοικονόμησης διαθέσιμης μνήμης. Σε κάθε περίπτωση, η επιλογή των πλειάδων που θα απορριφθούν γίνεται είτε κατά τρόπο τυχαίο είτε βασισμένο σε κάποια κριτήρια για τον βαθμό ενδιαφέροντος των πλειάδων. Έτσι μπορεί να απορρίπτονται είτε οι παλαιότερες πλειάδες κάθε ρεύματος, είτε πλειάδες από λιγότερο σημαντικά ρεύματα δεδομένων όπως αυτά προσδιορίζονται από τον χρήστη μέσω κάποιου μηχανισμού απόδοσης βαρών.

Η μέθοδος απόρριψης φόρτου αντικατοπτρίζει την επιδίωξη ισοστάθμισης τόσο μεταξύ ακρίβειας αποτελεσμάτων και απόδοσης του συστήματος όσο και μεταξύ ακρίβειας και μνήμης. Τα εξαγόμενα αποτελέσματα υστερούν σε ακρίβεια μια και προέκυψαν από την επεξεργασία ενός υποσυνόλου των εισερχόμενων στοιχείων.

2.4.2 Επιβολή *παραθύρων* (*windows*)

Τα παράθυρα αποτελούν έναν μηχανισμό για την διαρκή απόσπαση πεπερασμένου πλήθους πλειάδων από κάποιο πιθανά απεριόριστο ρεύμα δεδομένων με κριτήρια άλλοτε χρονικά και άλλοτε αριθμητικά ανάλογα με τον τύπο του παραθύρου. Επισημαίνεται λοιπόν, ότι ο προσδιορισμός παραθύρων δεν συνεπάγεται απαραίτητα κάποιο επίπεδο προσέγγισης στις απαντήσεις. Το σύστημα ανταποκρίνεται σε τυχόν απαιτήσεις των ερωτημάτων για έμφαση στην πιο πρόσφατη πληροφορία, με την επιβολή παραθύρων κατάλληλου τύπου και παραμέτρων πάνω στα εισερχόμενα ρεύματα. Αυτή η περίπτωση είναι αρκετά συνήθης και δεν έχει επιπτώσεις στην ακρίβεια των αποτελεσμάτων. Ένα τέτοιο απλουστευμένο παράδειγμα χρήσης παραθύρου, για τις τελευταίες 10 μετρήσεις ενός δικτύου αισθητήρων υγρασίας, διατυπώνεται σε φυσική γλώσσα ως εξής:

Παράδειγμα 2.5: *Ζητείται ανά πάσα χρονική στιγμή η μέγιστη και η ελάχιστη τιμή υγρασίας ανάμεσα στις 10 τελευταίες μετρήσεις που έφτασαν στο σύστημα υπό την μορφή ρεύματος.*

Ωστόσο, πιθανή ανεπάρκεια των διαθέσιμων πόρων του συστήματος μπορεί να οδηγήσει στην επιβολή παραθύρων είτε πάνω στα πρωτογενή ρεύματα είτε στην είσοδο κάποιων προβληματικών τελεστών λ.χ. *σύνδεση ή συνάθροιση*. Ο στόχος πλέον είναι η ανταπόκριση του συστήματος στις απαιτήσεις των ερωτημάτων, με κόστος την επιβολή προσεγγίσεων στις απαντήσεις λόγω του μειωμένου πλήθους πλειάδων που οδηγήθηκε σε επεξεργασία. Σε αυτήν την περίπτωση η μέθοδος της επιβολής παραθύρων μπορεί να θεωρηθεί ως μία ειδική μορφή απόρριψης φόρτου, όπου απορρίπτονται κάθε φορά οι παλαιότερες πλειάδες των ρευμάτων. Πρέπει ωστόσο να σημειωθεί ότι το μέγεθος του υπεισερχόμενου σφάλματος εξαιτίας της επιβολής των παραθύρων είναι καλά προσδιορισμένο και αντιληπτό από τον χρήστη.

2.4.3 Διατήρηση *συνόψεων* (*synopses*)

Οι συνόψεις αποτελούν δομές οι οποίες επιτυγχάνουν την περιληπτική και ενδεχομένως προσεγγιστική αναπαράσταση της πληροφορίας που παρέχεται από ένα σύνολο πλειάδων για χρήση από συγκεκριμένα ερωτήματα. Τα ερωτήματα μπορούν να χρησιμοποιούν τα σαφώς πιο εύχρηστα περιεχόμενα των συνόψεων αντί για τις πρωτογενείς πλειάδες εξοικονομώντας μνήμη και χρόνο επεξεργασίας. Κατά τον τρόπο αυτό οι συνόψεις αποτελούν ένα μέσο ανταλλαγής μεταξύ μνήμης και ακρίβειας απαντήσεων. Ωστόσο, η κυριότερη δυσκολία εντοπίζεται στην online τήρηση συνόψεων καθώς τα περιεχόμενα του ρεύματος διαρκώς εξελίσσονται. Οι κυριότερες τεχνικές παραγωγής συνόψεων περιλαμβάνουν: *δειγματοληψία* (*sampling*), *σκίτσα* (*sketches*), *ιστογράμματα* (*histograms*), *κυματίδια* (*wavelets*) κ.α.

2.4.4 *Μαζική επεξεργασία* (*Batch processing*)

Η μέθοδος αυτή εφαρμόζεται μάλλον σπάνια, κυρίως σε περιπτώσεις όπου η επεξεργασία μίας πλειάδας αποτελεί χρονοβόρα διαδικασία προκειμένου να μην παρεμποδίζεται η πρόσληψη νέων πλειάδων σε συνθήκες υψηλού ρυθμού άφιξης στοιχείων. Σύμφωνα με την μέθοδο αυτή, οι νέες πλειάδες εισέρχονται απρόσκοπτα στο σύστημα, η επεξεργασία τους όμως καθυστερεί και πραγματοποιείται μαζικά για όλες τις νέες πλειάδες σε περιόδους ύφεσης του ρυθμού άφιξης νέων πλειάδων. Η προσέγγιση που υπεισέρχεται δεν έχει να κάνει με την ακρίβεια των απαντήσεων αλλά με το γεγονός ότι οι ακριβείς απαντήσεις δεν παρέχονται έγκαιρα. Οι απαιτήσεις κάθε εφαρμογής καθορίζουν τα αποδεκτά περιθώρια χρονικής υστέρησης της απάντησης, τα οποία αποτελούν ένα από τα βασικότερα κριτήρια για την επιλογή της μεθόδου αυτής.

Κεφάλαιο 3

Τύποι παραθύρων σε ερωτήματα διαρκείας

3.1 Εισαγωγή

Τα παράθυρα μπορούν να θεωρηθούν τελεστές οι οποίοι εφαρμόζονται πάνω σε ρεύματα δεδομένων, παρέχοντας διαρκώς πρόσβαση σε πεπερασμένο πλήθος στοιχείων πληροφορίας με έμφαση στα πιο πρόσφατα από αυτά. Καθώς το ρεύμα εξελίσσεται, τα περιεχόμενα των παραθύρων μεταβάλλονται κατά τρόπο που καθορίζεται από την σημασιολογία και τις παραμέτρους τους. Έτσι, ένα παράθυρο περιλαμβάνει ένα διαρκώς μεταβαλλόμενο υποσύνολο των στοιχείων του ρεύματος με το οποίο συσχετίζεται. Ακολουθούν μερικά παραδείγματα ερωτημάτων διαρκείας, από τα οποία διαφαίνονται οι διαφορετικοί τύποι παραθύρων που μπορούν να οριστούν:

Παράδειγμα 3.1: *Σε ένα δίκτυο χωρικά κατανεμημένων αισθητήρων οι οποίοι στέλνουν τις μετρήσεις τους σε τυχαίες χρονικές στιγμές, επιδιώκεται ο υπολογισμός του μέσου όρου θερμοκρασίας, με βάση τις 10 τελευταίες μετρήσεις.*

Παράδειγμα 3.2: *Σε εφαρμογές παρακολούθησης τροχιάς ενός στόλου οχημάτων, ζητείται ο διαρκής προσδιορισμός των δύο τελευταίων θέσεων κάθε οχήματος για την εκτίμηση της ταχύτητας καθενός από αυτά.*

Παράδειγμα 3.3: *Σε εφαρμογές χρηματιστηρίου οι χρήστες μπορεί να ενδιαφέρονται διαρκώς για την διακύμανση των μετοχών στο διάστημα των δύο τελευταίων ωρών.*

Παράδειγμα 3.4: *Χαλαρώνοντας τις απαιτήσεις του προηγούμενου παραδείγματος, οι χρήστες μπορεί να μην ενδιαφέρονται τόσο για την **διαρκή** ενημέρωση τους. Αντίθετα, πιθανόν να επιθυμούν κάθε δεκαπέντε λεπτά την παρουσίαση συγκεντρωτικών στατιστικών στοιχείων, για τις μετοχές που διακινήθηκαν στο τελευταίο τέταρτο.*

Παράδειγμα 3.5: *Επίσης μπορεί να διατυπωθεί ερώτημα για την παρακολούθηση της κίνησης κάποιων μετοχών, από κάποια χρονική στιγμή ενδιαφέροντος και μετά.*

Αν και κάθε παράθυρο εφαρμόζεται πάνω στα περιεχόμενα ενός και μόνο ρεύματος δεδομένων, είναι ωστόσο επιτρεπτό και σύνηθες, πολλαπλά παράθυρα να εφαρμόζονται πάνω στο ίδιο ρεύμα. Η χρήση τους μπορεί να ζητηθεί είτε για να εξυπηρετήσουν την απαίτηση των ερωτημάτων διαρκείας για απόσπαση της πιο πρόσφατης πληροφορίας, είτε για να περιορίσουν τις απαιτήσεις σε διαθέσιμους πόρους οδηγώντας κάποια ερωτήματα σε προσεγγιστικές απαντήσεις. Παρακάτω παρουσιάζονται κάποιες γενικές ιδιότητες των παραθύρων οι οποίες διαφοροποιούνται από τύπο σε τύπο. Έτσι κάθε παράθυρο διαθέτει:

- *αφετηρία* (*κάτω άκρο, lower bound*), η οποία αντιπροσωπεύει το παλαιότερο στοιχείο του ρεύματος που θα συμπεριληφθεί στο παράθυρο.
- *πέρας* (*άνω άκρο, upper bound*), που αντιστοιχεί στο τελευταίο στοιχείο που περιλήφθηκε στα περιεχόμενα του παραθύρου.
- *εύρος* (*width*), το οποίο άλλοτε εκφράζεται από το πλήθος των πλειάδων που περιλαμβάνει, άλλοτε δε από το *χρονικό διάστημα* (*range*) που αυτές καλύπτουν.
- ειδικό τρόπο μεταβολής των περιεχομένων του μέσω κατάλληλων μετατοπίσεων στα άκρα του.

Στην συνέχεια επιδεικνύεται ένας τρόπος για την αλγεβρική αναπαράσταση ενός ρεύματος δεδομένων, ο οποίος χρησιμοποιείται για την διατύπωση αλγεβρικών σχέσεων προσδιορισμού των περιεχομένων των διαφόρων τύπων παραθύρων. Ακολούθως επιχειρείται μια κατηγοριοποίηση των παραθύρων που αναφέρονται στην τρέχουσα βιβλιογραφία βασιζόμενη στις διαφορετικές μορφές που μπορεί να εμφανίζονται οι παραπάνω ιδιότητες.

3.2 Αλγεβρική αναπαράσταση ρεύματος δεδομένων

Συνήθως κάθε στοιχείο του ρεύματος εκλαμβάνεται ως μια σχεσιακή *πλειάδα γνωρισμάτων* (*tuple of attributes*), χωρίς να αποκλείονται άλλες αντικειμενοστρεφείς ή ιεραρχικές μορφές, λ.χ. XML έγγραφα όπως στο σύστημα NiagaraCQ. Κατ' αντιστοιχία προς όσα ισχύουν στο σχεσιακό μοντέλο, το σχήμα των πλειάδων μπορεί να οριστεί ως εξής:

> ***Σχήμα πλειάδων.*** Το σχήμα E των πλειάδων των δεδομένων αντιπροσωπεύεται από ένα σύνολο στοιχείων $\langle e_1, e_2, \ldots e_L \rangle$. Κάθε στοιχείο e_i λέγεται *γνώρισμα* (*attribute*), συμβολίζεται με A_i και οι τιμές του προέρχονται από ένα –πιθανόν άπειρο– ατομικό πεδίο ορισμού D_i. Ο πεπερασμένος αριθμός L των γνωρισμάτων λέγεται *βαθμός* (*arity*) του σχήματος. Κάθε *πλειάδα* (*tuple*) είναι ένα στιγμιότυπο του σχήματος και περιγράφεται από τις τιμές που λαμβάνει σε κάθε γνώρισμα.

Κάθε πλειάδα του ρεύματος συνοδεύεται από μια – συνήθως χρονική – ένδειξη. Τέτοια *χρονόσημα* (*timestamps*) μπορεί να δίνονται κατά την παραγωγή των στοιχείων στην πηγή (λ.χ. αισθητήρες), κάτι που πιθανόν να ζητείται κι από ερωτήματα που έχουν υποβληθεί (*χρόνος ισχύος*). Εναλλακτικά, οι χρονικές ενδείξεις (ή απλοί αύξοντες ακέραιοι αριθμοί) είναι δυνατόν να προσαρτώνται τεχνητά με τη σειρά στις πλειάδες μόλις φτάσουν στο σύστημα (*χρόνος συστήματος*). Το χρονόσημο εξυπηρετεί σημαντικά στη σύνθεση στοιχείων από πολλαπλά ρεύματα, όπως και στην διεκπεραίωση λειτουργιών πάνω σ' αυτά (λ.χ. συσχετισμοί συγχρονισμένων στοιχείων μεταξύ ρευμάτων). Καθώς τα παλαιότερα δεδομένα σταδιακά καθίστανται παρωχημένα με την άφιξη νέων πλειάδων, το χρονόσημο καθορίζει τη διαθεσιμότητα κάθε στοιχείου στο σύστημα και άρα τη συμμετοχή του στην επεξεργασία των ερωτημάτων διαρκείας. Σε κάθε περίπτωση, τα χρονόσημα προκύπτουν από ένα καθολικό πεδίο ορισμού *(Time Domain)*:

Πεδίο ορισμού χρόνου. Το πεδίο ορισμού του χρόνου T θεωρείται ως ένα διατεταγμένο, άπειρο σύνολο διακριτών τιμών $\tau \in T$, καθεμιά από τις οποίες καλείται *χρονική στιγμή* (*time instant*). Ένα *χρονικό διάστημα* $[\tau_1, \tau_2] \subset T$ συνίσταται από όλες τις διακριτές χρονικές στιγμές $\tau \in T$ όπου $\tau_1 \leq \tau \leq \tau_2$.

Ουσιαστικά, το πεδίο των χρονοσήμων είναι παραπλήσιο του συνόλου N των φυσικών αριθμών. Φυσικός αριθμός είναι επίσης το *εύρος* (*extent*) κάθε χρονικού διαστήματος, εφόσον καταμετρά όλα τα διακριτά χρονόσημα που το συγκροτούν. Η βασική παραδοχή είναι ότι το *απειράριθμο* ρεύμα δεδομένων περιλαμβάνει ένα *πεπερασμένο πολυσύνολο* (*multiset, bag*) στοιχείων με την ίδια χρονική ένδειξη $\tau \in T$. Επιτρέπονται *διπλότυπα* (*duplicates*), καθώς σε κάθε χρονόσημο είναι δυνατόν να καταγράφονται καμία, μία ή περισσότερες πλειάδες με ίδιες τιμές στα υπόλοιπα γνωρίσματα. Συνεπώς:

Ρεύμα δεδομένων. Ως ρεύμα δεδομένων S ορίζεται μια *απεικόνιση* $S : T \rightarrow 2^R$ από το πεδίο ορισμού του χρόνου T στο δυναμοσύνολο των πλειάδων R με κοινό σχήμα E. Το γνώρισμα A_τ χαρακτηρίζεται ως ***χρονόσημο*** των πλειάδων και λαμβάνει τιμές από το T.

Ως *τρέχον περιεχόμενο* $S(\tau_i)$ ενός ρεύματος δεδομένων S κατά τη χρονική στιγμή $\tau_i \in T$ θεωρείται το σύνολο των πλειάδων που φέρουν χρονόσημο $\tau \leq \tau_i$. Επίσης, ως *τρέχον στιγμιότυπο* $I(\tau_i)$ του ρεύματος S κατά τη χρονική στιγμή τ_i θεωρείται το σύνολο των πλειάδων με χρονόσημο τ_i. Τέλος, βάσει του ανωτέρω ορισμού, ο χρονισμός των πλειάδων είναι κοινός, αφού όλα τα χρονόσημα προέρχονται από το T. Λόγω των ιδιοτήτων του T, είναι εφικτή η *ολική χρονική διάταξη* (*temporal total order*) των πλειάδων:

Χρονική διάταξη. Η *χρονική διάταξη* ορίζεται ως μια (πολλά–προς–ένα) *απεικόνιση* $f_O : D_S \rightarrow T$ από το πεδίο των τύπων D_S των γνωρισμάτων που εμφανίζονται στις πλειάδες s του ρεύματος S προς το πεδίο του χρόνου T, με τις εξής ιδιότητες:

i) ***Ύπαρξη οροσήμων:*** $\forall\, s \in S,\ \exists\ \tau \in T,$ ώστε $f_O(s) = \tau$.

ii) ***Μονοτονία:*** $\forall\, s_1, s_2 \in S,$ αν για τα αντίστοιχα χρονόσημα ισχύει ότι $s_1.A_\tau \leq s_2.A_\tau$, τότε θα ισχύει και $f_O(s_1) \leq f_O(s_2)$.

Η χρονική διάταξη, εξασφαλίζει ότι τα στοιχεία θα δοθούν προς επεξεργασία με τη σειρά: ο επεξεργαστής ερωτημάτων δεν πρέπει να δεχτεί πλειάδες με μικρότερο χρονόσημο απ' όσες ήδη έχει λάβει, έτσι ώστε να παράγονται χρονικά συνεπή αποτελέσματα. Σε αντίθεση προς τα συμβατικά συστήματα βάσεων δεδομένων, τα στοιχεία του ρεύματος δεν θεωρούνται διαθέσιμα παρά μόνο προσωρινά και πάντως όχι στο σύνολό τους, αλλά τμηματικά. Επομένως, δεν είναι εφικτή κάποιας μορφής προεπεξεργασία που θα επιτρέψει την ευκολότερη ενσωμάτωση των στοιχείων, αφού υπονοείται ότι η αρχή και το πέρας της ακολουθίας των δεδομένων δεν είναι γνωστό.

3.3 Χαρακτηρισμός παραθύρων

Όπως αναφέρθηκε στην εισαγωγική ενότητα αυτού του κεφαλαίου, κάθε παράθυρο αποσπά πεπερασμένο πλήθος πλειάδων από το ρεύμα δεδομένων στο οποίο εφαρμόζεται, ακολουθώντας τους

δικούς του κανόνες. Αυτό μπορεί να εκφραστεί αλγεβρικά κατά τον ακόλουθο τρόπο **[PS05]**, όπου η συζευκτική συνθήκη E αντιπροσωπεύει τους κανόνες του εκάστοτε παραθύρου.

> ***Παράθυρο επί ρεύματος δεδομένων***. Έστω W_E ένα παράθυρο με συζευκτική συνθήκη E που εφαρμόζεται τη χρονική στιγμή $\tau_0 \in T$ στα στοιχεία ενός ρεύματος δεδομένων S, δηλαδή στα τρέχοντα περιεχόμενά του $S(\tau_0)$. Τότε:
>
> $\forall\ \tau_i \in T,\ \tau_i \geq \tau_0\ ,\ W_E\ (S(\tau_i)) = \{s \in S(\tau_i) : E(s)$ είναι αληθής$\}$,
>
> όπου $|W_E(S(\tau_i))| \leq n$,
>
> για κάποιο οσοδήποτε μεγάλο, αλλά πάντοτε πεπερασμένο $n \in N$.

Οι διαφορετικοί τύποι παραθύρων μπορούν να ενταχθούν σε ευρύτερες ομάδες με κριτήριο την ομοιότητα ως προς κάποια ιδιότητά τους. Στις επόμενες υποενότητες παρουσιάζονται τα κριτήρια που μπορεί να χρησιμοποιηθούν και προσδιορίζονται οι διαφορετικές ομάδες που προκύπτουν για καθένα από αυτά **[PS05]**.

3.3.1 Μετατόπιση άκρων

Το παράθυρο μπορεί να οριστεί ρητώς από τα άκρα του, και γι' αυτό χρησιμοποιούνται συνήθως οι χρονικές στιγμές αρχής και πέρατος του παραθύρου. Αναλόγως του κατά πόσον επιτρέπεται ν' αλλάζουν τα όριά τους, διακρίνονται παράθυρα με:

- *Σταθερά άκρα* (*fixed–size windows*), όπου τουλάχιστον το ένα άκρο του παραθύρου παραμένει αμετακίνητο. Τα *παράθυρα οροσήμου* (*landmark windows*) αποτελούν την χαρακτηριστικότερη δομή αυτής της κατηγορίας. Αν και τα δύο άκρα είναι στατικά, τότε πρόκειται για *πάγια παράθυρα ζώνης* (*fixed–band windows*).
- *Μεταβλητά άκρα* (*variable–size windows*). Συνήθως, προβλέπεται ότι και τα δύο άκρα του παραθύρου μεταβάλλονται με την πάροδο του χρόνου. Στην πλέον ενδιαφέρουσα περίπτωση, η αφετηρία και το πέρας του παραθύρου εξελίσσονται συντονισμένα με τον ίδιο ρυθμό, όπως συμβαίνει λ.χ. στα *κυλιόμενα παράθυρα* (*sliding windows*).

3.3.2 Μονάδα μέτρησης περιεχομένων

Εναλλακτικά, το παράθυρο προσδιορίζεται βάσει του μεγέθους του, θεωρώντας γνωστό ένα άκρο του, ώστε τελικά να μπορούν να εντοπιστούν οι πλειάδες που θα συμπεριληφθούν σ' αυτό. Η *εμβέλεια* (*scope*) του παραθύρου, δηλώνεται:

- σε «*λογικές*» μονάδες, δηλαδή από το χρονικό διάστημα βάσει των χρονοσήμων που καλύπτουν τα περιεχόμενά του. Αυτά ονομάζονται συνήθως *χρονικά παράθυρα* (*time-based windows*).
- σε «*φυσικές*» μονάδες, υπονοώντας το πλήθος των πλειάδων που εμπίπτουν εντός των άκρων του. Τα *παράθυρα πλειάδων* (*tuple-based windows*) και τα *μεριστικά παράθυρα* (*partitioned windows*) είναι τυπικές μορφές τέτοιων δομών.

Ως γνωστό άκρο του παραθύρου συνήθως θεωρείται το πέρας του, ερμηνεύοντάς το είτε ως τρέχον χρονόσημο είτε ως την πιο πρόσφατη πλειάδα του ρεύματος.

3.3.3 Βήμα προόδου

Εξαιρώντας την περίπτωση που τα άκρα του θεωρηθούν σταθερά, το παράθυρο ανανεώνει προοδευτικά τα περιεχόμενά του παρακολουθώντας την εξέλιξη του χρόνου ή την έλευση νέων πλειάδων του ρεύματος, ως εξής:

- Με *μοναδιαίο βήμα*, οπότε το παράθυρο προχωρεί για κάθε στοιχειώδη χρονική μονάδα (λ.χ. ανά δευτερόλεπτο) ή με κάθε πλειάδα που θα ενταχθεί στο ρεύμα. Χαρακτηριστικότερη περίπτωση είναι τα *κυλιόμενα παράθυρα* (*sliding windows*), όπου τα άκρα «μετακυλίονται» ομαλά, διατηρώντας σταθερό το μέγεθος του παραθύρου.
- *Κατά άλματα* (*hops*), συνήθως μη επικαλυπτόμενα. Π.χ. τα *επάλληλα παράθυρα* (*tumbling windows*) εξυπηρετούν όταν επιδιώκεται τα περιεχόμενα διαδοχικών στιγμιοτύπων τους να είναι ξένα μεταξύ τους, έτσι ώστε κάθε φορά να εξετάζονται εντελώς διαφορετικά στοιχεία του ρεύματος.

Βεβαίως, στην περίπτωση των κυλιόμενων παραθύρων είναι αναμενόμενο να υπάρχουν επικαλύψεις μεταξύ διαδοχικών στιγμιοτύπων. Όταν το μοναδιαίο βήμα εκφράζεται χρονικά, όλες οι πλειάδες με χρονόσημο μικρότερο από τη νέα αρχή του παραθύρου διαγράφονται, ώστε να ενταχθούν όσα στοιχεία (όχι κατ' ανάγκην ισάριθμα!) αναφέρονται στην τρέχουσα χρονική στιγμή. Αν πάλι το παράθυρο εκφράζεται ως αριθμός πλειάδων, τότε κάθε νεοεισερχόμενη πλειάδα συνεπάγεται την απόρριψη της παλαιότερης που υπάρχει στο παράθυρο. Άρα, καθώς το παράθυρο μετατοπίζεται, μεταβολές σημειώνονται μόνο στα άκρα του (διαγραφές στο κατώτερο, εισαγωγές στο ανώτερο), αφήνοντας άθικτα τα ενδιάμεσα στοιχεία.

Στην συνέχεια παρουσιάζονται οι κυριότεροι τύποι παραθύρων σύμφωνα με τους **[PS05]**. Για κάθε τύπο παραθύρου μπορεί να διατυπωθεί μία αλγεβρική σχέση ορισμού της συζευκτικής συνθήκης W_E μέσω της οποίας προσδιορίζονται τα περιεχόμενά του. Επίσης, κάθε παράθυρο συγκεκριμένου τύπου και παραμέτρων εντάσσεται σε μία ακριβώς από τις ομάδες κάθε κριτηρίου από αυτά που παρουσιάστηκαν παραπάνω. Επισημαίνεται ότι η χρησιμότητα των παραθύρων αναδεικνύεται από τον συνδυασμό τους με άλλους γνωστούς αλγεβρικούς τελεστές όπως θα αναπτυχθεί διεξοδικά στο κεφάλαιο 4. Ενδεικτικά παραδείγματα τέτοιων συνδυασμών αποτελούν: η *παραθυρική σύνδεση* (*windowed – join*), η *παραθυρική συνάθροιση* (*windowed – aggregate*), κλπ.

3.4 Τύποι φυσικών παράθυρων

Τέτοια παράθυρα συνήθως είναι κυλιόμενα, ώστε να επιστρέφεται ένας καθορισμένος αριθμός N πρόσφατων πλειάδων. Άλλωστε, σε πρακτικές εφαρμογές είναι μάλλον σπάνιο να ζητηθούν π.χ. τα N δεδομένα που έφτασαν μετά το k-στό στοιχείο, αφού αγνοείται αν και πότε έχει ληφθεί η συγκεκριμένη πλειάδα στη ροή του απειράριθμου ρεύματος. Σε μία τέτοια υποθετική περίπτωση τα άκρα του παραθύρου θα παγιώνονταν με την έλευση των N αυτών στοιχείων. Στην κατηγορία των φυσικών παραθύρων υπάγονται τα παράθυρα πλειάδων και τα μεριστικά παράθυρα.

3.4.1 *Παράθυρα πλειάδων* (*Tuple – based windows*)

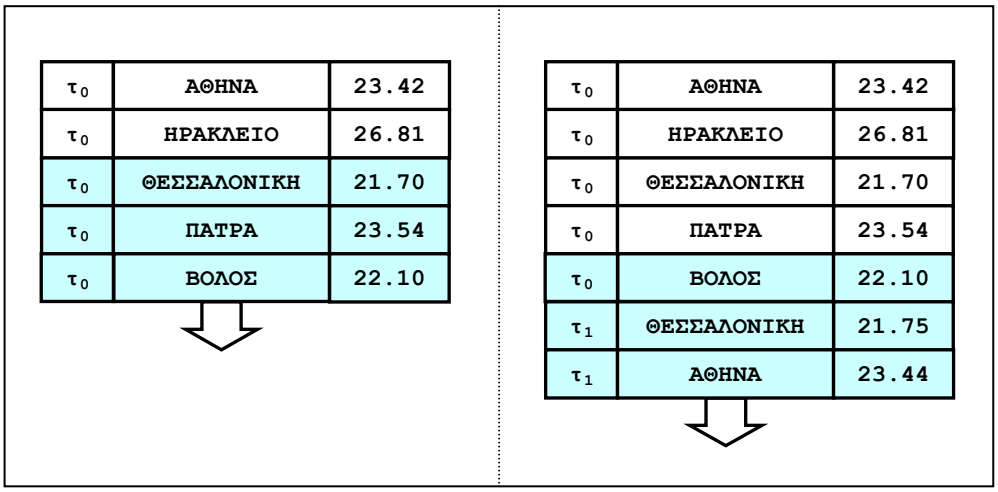
Τέτοια παράθυρα ανά πάσα χρονική στιγμή $\tau \in T$ εστιάζουν στις N τελευταίες πλειάδες του ρεύματος S. Σύμφωνα με την κατηγοριοποίηση που προτάθηκε στην ενότητα 3.3 παρουσιάζουν τις εξής ιδιότητες:

- Τα άκρα τους είναι και τα δύο μεταβλητά.
- Χρησιμοποιούν ως βάση προσδιορισμού των περιεχομένων τους το πλήθος των πλειάδων.
- Το βήμα τους είναι μοναδιαίο σε επίπεδο πλειάδας.

Τα στοιχεία εντός του παραθύρου υπολογίζονται ως εξής: ξεκινώντας από την παρούσα χρονική στιγμή και προχωρώντας προς το παρελθόν («*με την όπισθεν*»), επιλέγονται συνεχώς πλειάδες μέχρις ότου καλυφθεί ο ζητούμενος αριθμός N.

Ωστόσο, αυτή η μέθοδος κρύβει δύο πολύ λεπτά σημεία:

i) Πρώτον, δεν είναι σαφές τί θα συμβεί όταν ο αλγόριθμος, φτάνοντας στο απώτατο χρονικό ορόσημο που θα χρειαστεί να εξετάσει, διαπιστώσει ότι πρέπει να διαλέξει ανάμεσα σε περισσότερες πλειάδες συγκριτικά με όσες υπολείπονται για να συμπληρωθεί ο αριθμός N.



Σχήμα 3.1: *Παράδειγμα παραθύρου πλειάδων, με N = 3 πλειάδες*

ii) Επίσης, δεν είναι απλός ο *χειρισμός διπλοτύπων*, αν δηλαδή ο αλγόριθμος πρέπει να μετράει μία ή περισσότερες φορές τυχόν διπλότυπα καθώς υπολογίζει το παράθυρο. Συνήθως, τέτοια προβλήματα επιλύονται με *μη ντετερμινιστικό* τρόπο: λ.χ. στο σύστημα STREAM, τα αντίστοιχα παράθυρα (ROW-*based windows*) επιλέγουν τυχαία τις σχετικές πλειάδες, ενώ προσμετρούν τα διπλότυπα όσες φορές εμφανίζονται **[ABW03]**.

Στο σχήμα 3.1 απεικονίζονται δύο διαφορετικά στιγμιότυπα ενός ρεύματος δεδομένων το οποίο αναπαριστά την πληροφορία που στέλνουν στο σύστημα αισθητήρες μέτρησης θερμοκρασίας. Σε κάθε στοιχείο πληροφορίας περιέχεται ένα γνώρισμα χρονοσήμου (τ_0, τ_1, ..), το όνομα της πόλης όπου γίνεται η μέτρηση, ενώ η μετρούμενη θερμοκρασία αποτελεί το τελευταίο γνώρισμα κάθε πλειάδας. Στα στοιχεία αυτού του ρεύματος εφαρμόζεται παράθυρο πλειάδων μεγέθους N=3 ενώ σε κάθε στιγμιότυπο οι πλειάδες που περιέχονται στο παράθυρο διακρίνονται με διαφορετικό χρώμα.

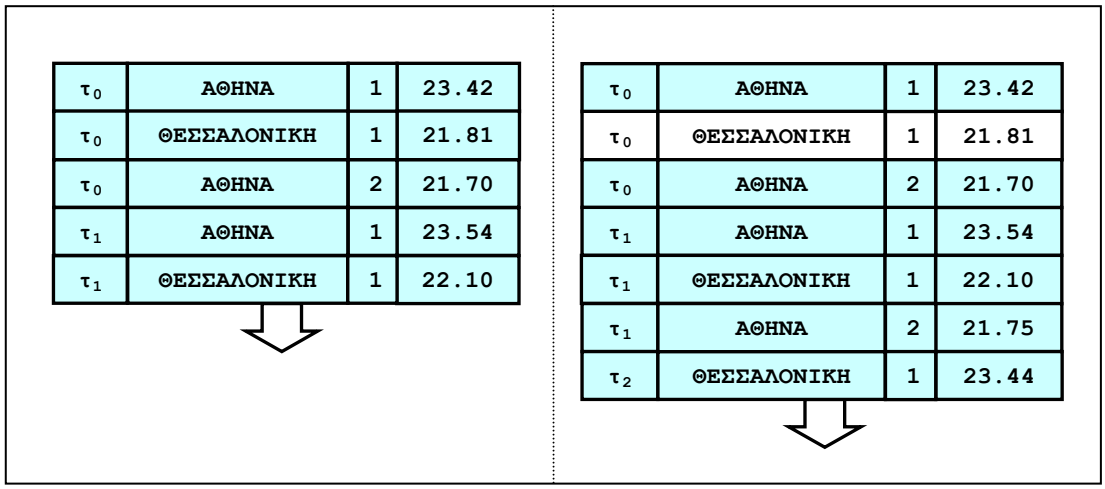
Η συζευκτική συνθήκη για τον προσδιορισμό των περιεχομένων αυτών των παραθύρων εκφράζεται αλγεβρικά ως εξής:

$$
\begin{aligned}
&W_{\mathrm{n}}(S, \tau, N) = \{ s \in S(\tau) : \\
&: \ \exists\ \tau_1 \in T\ (\ \tau_1 \le \tau\ \wedge\ |\{ s \in S(\tau) : \tau_1 \le s.A_\tau \le \tau \}| \le N\ ) \wedge \\
&\wedge\ \forall\ \tau_2 \in T\ (\ \tau_2 < \tau_1\ \wedge\ |\{ s \in S(\tau) : \tau_2 \le s.A_\tau \le \tau \}| > N\ )\ \}.
\end{aligned}
$$

Στην παραπάνω σχέση διακρίνονται δύο υποσυνθήκες. Η πρώτη προσδιορίζει το μέγιστο αριθμητικό πλήθος N των πλειάδων που θα περιέχονται στο παράθυρο W. Η δεύτερη υποσυνθήκη διασφαλίζει ότι το παράθυρο θα περιλαμβάνει N από τις πιο πρόσφατες πλειάδες του ρεύματος. Αντίστοιχες υποσυνθήκες απαντώνται και στις εκφράσεις της προτεινόμενης συζευκτικής συνθήκης για τους υπόλοιπους τύπους παραθύρων, με τις απαραίτητες πάντα τροποποιήσεις.

3.4.2 *Μεριστικά παράθυρα* (*Partitioned windows*)

Ένα μεριστικό παράθυρο επίσης εφαρμόζεται στις πρόσφατες πλειάδες του ρεύματος, αφού προηγουμένως τις ομαδοποιήσει βάσει των τιμών που εμφανίζουν για την λίστα των *γνωρισμάτων ομαδοποίησης* $L = \{A_i, A_j, ..., A_n\}$, όπως ο αντίστοιχος σχεσιακός τελεστής. Από κάθε προκύπτουσα ομάδα τιμών $<a_i, a_j, ..., a_n>$ λαμβάνονται τότε N πλειάδες και ακολουθεί συνένωση των επιμέρους αποτελεσμάτων. Αξίζει πάντως να σημειωθεί ότι μετά την ομαδοποίηση δεν εφαρμόζεται κάποια



Σχήμα 3.2: *Παράδειγμα μεριστικού παραθύρου, με N = 2 πλειάδες για κάθε ομάδα που ορίζεται από τα γνωρίσματα L={ cityName, sensorId }*

συνάρτηση συνάθροισης (λ.χ. SUM, AVG κ.ά.) όπως στη σχεσιακή άλγεβρα, αλλά απλώς καταμετρώνται οι πλειάδες κάθε ομάδας.

Τα παράθυρα αυτά παρουσιάζουν τις εξής χαρακτηριστικές ιδιότητες:

- Τα άκρα τους είναι και τα δύο μεταβλητά.
- Χρησιμοποιούν ως βάση προσδιορισμού των περιεχομένων τους το πλήθος των πλειάδων.
- Το βήμα τους είναι μοναδιαίο σε επίπεδο πλειάδας.

Τα στοιχεία εντός του παραθύρου υπολογίζονται ως εξής: ξεκινώντας από την παρούσα χρονική στιγμή και προχωρώντας προς το παρελθόν, επιλέγονται συνεχώς για κάθε διαφορετική τιμή της λίστας γνωρισμάτων ομαδοποίησης L οι πιο πρόσφατες N πλειάδες. Η λειτουργία του μεριστικού παραθύρου W μπορεί να περιγραφεί συμβολικά ως εξής :

$$W_{\mathrm{p}}(S, \tau, L, N) = \{ s \in S(\tau) : \forall\ A_{\mathrm{k}} \in L, s.A_{\mathrm{k}} = \alpha_{\mathrm{k}} \wedge \alpha_{\mathrm{k}} \in \mathrm{D}_{\mathrm{k}} \wedge$$
$$\wedge\ \exists\ \tau_1 \in T\ (\ \tau_1 \le \tau\ \wedge\ |\{ s \in S(\tau) : s.A_{\mathrm{k}} = \alpha_{\mathrm{k}} \wedge \tau_1 \le s.A_{\tau} \le \tau \}| \le N\ ) \wedge$$
$$\wedge\ \forall\ \tau_2 \in T\ (\ \tau_2 < \tau_1\ \wedge\ |\{ s \in S(\tau) : s.A_{\mathrm{k}} = \alpha_{\mathrm{k}} \wedge \tau_2 \le s.A_{\tau} \le \tau \}| > N\ )\ \}$$

Στο σχήμα 3.2 απεικονίζονται δύο διαφορετικά στιγμιότυπα ενός ρεύματος δεδομένων το οποίο αναπαριστά την πληροφορία που στέλνουν στο σύστημα αισθητήρες μέτρησης θερμοκρασίας. Σε κάθε πόλη (*cityName*) υπάρχουν πολλοί αισθητήρες που διακρίνονται μεταξύ τους από έναν αύξοντα ακέραιο (*sensorId*). Σε κάθε στοιχείο πληροφορίας περιέχεται ένα γνώρισμα χρονοσήμου (τ_0, τ_1, ..), το όνομα της πόλης *cityName*, ο ακέραιος *sensorId* και η αντίστοιχη μέτρηση θερμοκρασίας. Πάνω στα στοιχεία αυτού του ρεύματος εφαρμόζεται μεριστικό παράθυρο μεγέθους N=2, με λίστα γνωρισμάτων ομαδοποίησης L ={ *cityName, sensorId* }.

Ένα τέτοιο παράθυρο εξυπηρετεί στην διαρκή παρακολούθηση των 2 τελευταίων μετρήσεων που έστειλε κάθε αισθητήρας σε κάθε πόλη. Αν η λίστα γνωρισμάτων ομαδοποίησης διαμορφωνόταν ως L ={*cityName*}, τότε τα περιεχόμενα του παραθύρου θα αναπαριστούσαν τις N τελευταίες μετρήσεις θερμοκρασίας για κάθε πόλη, ανεξάρτητα από ποιους αισθητήρες της συγκεκριμένης πόλης εστάλησαν. Τα περιεχόμενα του παραθύρου σε κάθε στιγμιότυπο διακρίνονται ως χρωματισμένες πλειάδες, ενώ η διάταξη τους μέσα στο παράθυρο προκύπτει από την διάταξη των πλειάδων στο ρεύμα αφαιρώντας τις πλειάδες που δεν είναι χρωματισμένες.

Με το παράδειγμα αυτό αναδεικνύεται μία ιδιαιτερότητα των μεριστικών παραθύρων η οποία δεν παρουσιάζεται σε κανέναν άλλο τύπο. Όπως απεικονίζεται στο παράδειγμα, στο πρώτο

στιγμιότυπο του ρεύματος το παράθυρο περιέχει 2 μετρήσεις από τον αισθητήρα 1 της Θεσσαλονίκης. Στο δεύτερο στιγμιότυπο παρατηρείται η έλευση νέας μέτρησης αυτού του αισθητήρα η οποία πρέπει να συμπεριληφθεί στο παράθυρο ως πιο πρόσφατη (χρονόσημο τ_2). Όμως η παράμετρος μεγέθους N ορίστηκε ίση με 2, οπότε πρέπει να αφαιρεθεί από τα περιεχόμενα του παραθύρου η παλαιότερη μέτρηση αυτού του αισθητήρα. Αυτή η διαγραφή απεικονίζεται αποχρωματίζοντας την πλειάδα του αισθητήρα αυτού με το παλαιότερο χρονόσημο (εδώ χρονόσημο τ_0). Κατά συνέπεια, η θέση διαγραφής δεν είναι εκ των προτέρων γνωστή εμφανίζοντας εξάρτηση από τα εισερχόμενα δεδομένα, κάτι το οποίο δεν συμβαίνει με κανένα άλλο τύπο παραθύρου. Στα υπόλοιπα παράθυρα όλες οι διαγραφές πραγματοποιούνται από το *κάτω άκρο* του παραθύρου απορρίπτοντας – με τους ισχύοντες κανόνες – πάντοτε τα παλαιότερα στοιχεία του.

3.5 Τύποι λογικών παραθύρων

Τα λογικά παράθυρα προϋποθέτουν ρητό προσδιορισμό του χρονικού διαστήματος εντός του οποίου θα ερευνάται ποιες πλειάδες του ρεύματος εμπίπτουν. Αυτή η απαίτηση απλουστεύεται αρκετά αν οριστεί η έννοια της *εμβέλειας (scope)* κάθε παραθύρου ως μια *απεικόνιση* (*mapping*) από το πεδίο των χρονικών ορισμών στο πεδίο των διαστημάτων:

$$\text{scope: } T \rightarrow \{[\tau_1, \tau_2] : \tau_1, \tau_2 \in T,\ \tau_1 \leq \tau_2\}$$

Ουσιαστικά, για κάθε χρονική στιγμή η συνάρτηση εμβέλειας επιστρέφει τα χρονικά όρια (άκρα) του παραθύρου, συνεξετάζοντας τις παραμέτρους που ορίζουν τον τύπο του παραθύρου.

Στην κατηγορία των λογικών παραθύρων εντάσσονται τα *κυλιόμενα παράθυρα*, τα *επάλληλα παράθυρα*, τα *παράθυρα ορισμού* και τα *πάγια παράθυρα ζώνης*. Στα παραδείγματα που χρησιμοποιούνται κατά την παρουσίαση αυτών των τύπων, απεικονίζονται δύο στιγμιότυπα ενός ρεύματος δεδομένων, το οποίο αναπαριστά την πληροφορία που στέλνουν στο σύστημα αισθητήρες μέτρησης θερμοκρασίας. Τα δύο στιγμιότυπα διαφέρουν κατά μία μόνο πλειάδα, απεικονίζοντας την επίδραση που έχει στα περιεχόμενα του παραθύρου η άφιξη νέας πλειάδας στο ρεύμα. Κάθε στοιχείο πληροφορίας περιέχει: ένα γνώρισμα χρονοσήμου (τ_0, τ_1, ..), το όνομα της πόλης όπου γίνεται η μέτρηση και την μετρούμενη θερμοκρασία. Σε κάθε στιγμιότυπο οι πλειάδες που περιέχονται στο παράθυρο διακρίνονται με διαφορετικό χρώμα.

3.5.1 *Κυλιόμενα παράθυρα* (*Sliding windows*)

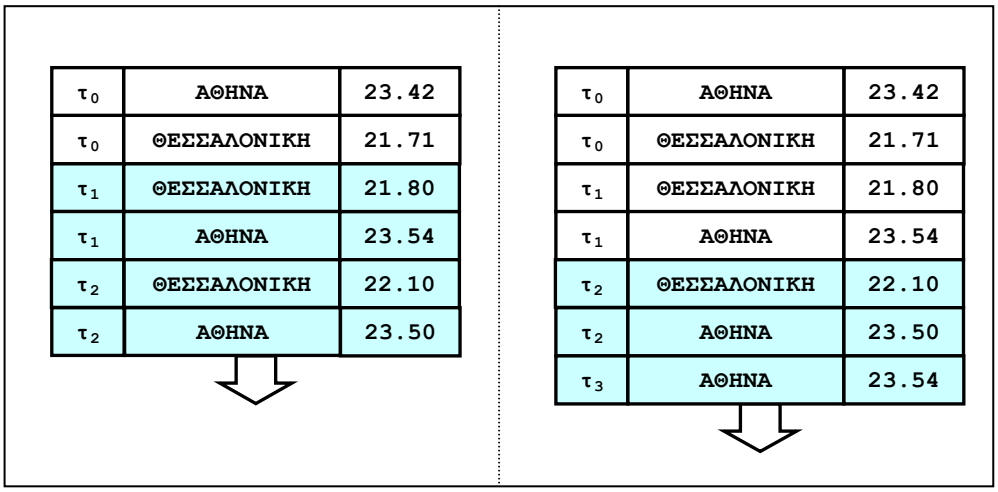
Πρόκειται για έναν σχετικά πολύπλοκο τύπο παραθύρων, ο οποίος όμως απαντάται αρκετά συχνά στις εφαρμογές ρευμάτων δεδομένων. Τα περιεχόμενα τους αναφέρονται σε συγκεκριμένη χρονική έκταση, ενώ γενικότερα τα παράθυρα αυτού του τύπου εμφανίζουν τις ακόλουθες ιδιότητες:

- τα άκρα τους είναι και τα δύο μεταβλητά,
- τα περιεχόμενά τους προσδιορίζονται *βάσει χρονοσήμου* και
- το *βήμα* μπορεί να είναι είτε μοναδιαίο είτε κατ' άλματα, πάντοτε όμως σε επίπεδο χρονοσήμου.

Για τον ορισμό τους χρησιμοποιούνται οι εξής παράμετροι:

- Παράμετρος *εύρους ή έκτασης* (*range*), με την οποία καθορίζεται η μέγιστη διαφορά χρονοσήμων που μπορεί να εμφανίζεται στις περιεχόμενες πλειάδες του παραθύρου.
- Παράμετρος *βήματος ή κύλισης* (*slide*), η οποία προσδιορίζει τον ρυθμό με τον οποίο ανανεώνονται τα περιεχόμενά του.

Έστω λοιπόν $\tau_0 \in T$ η χρονική στιγμή υποβολής του σχετικού ερωτήματος διαρκείας. Τότε η εμβέλεια ενός κυλιόμενου παραθύρου με εύρος ω και *βήμα* δ για κάθε $\tau \in T$ (με $\tau \geq \tau_0$) εκτείνεται:



Σχήμα 3.3: *Παράδειγμα κυλιόμενου παραθύρου, με βήμα $\delta = 1$ και κύλιση $\omega=2$ χρονόσημα*

$$\text{scope}_s(\tau, \omega, \delta) = \begin{cases} [\tau-\omega+1, \tau] & , \text{ αν } \tau \geq \tau_0+\omega \wedge \text{mod}((\tau-\tau_0), \delta)= 0 \\ \text{scope}_s(\tau-1, \omega, \delta) & , \text{ αν } \text{mod}((\tau-\tau_0), \delta) \neq 0 \\ [\tau_0, \tau] & , \text{ αν } \tau_0 \leq \tau < \tau_0+\omega \wedge \text{mod}((\tau-\tau_0), \delta)=0 \end{cases}$$

όπου τα μεγέθη τ_0, $\tau \in T$ εκφράζονται σε χρονικά ορόσημα και τα ω, $\delta \in N$ σε εύρος χρονικών διαστημάτων ($\omega, \delta > 0$). Για λόγους απλούστευσης, ο παραπάνω ορισμός υπονοεί ότι όλα τα χρονικά μεγέθη εκφράζονται ως φυσικοί αριθμοί, οπότε ο υπολογισμός της συνάρτησης εμβέλειας διεξάγεται σε διακριτές χρονικές στιγμές του T. Έτσι οι πλειάδες του παραθύρου προκύπτουν απ' τη σχέση:

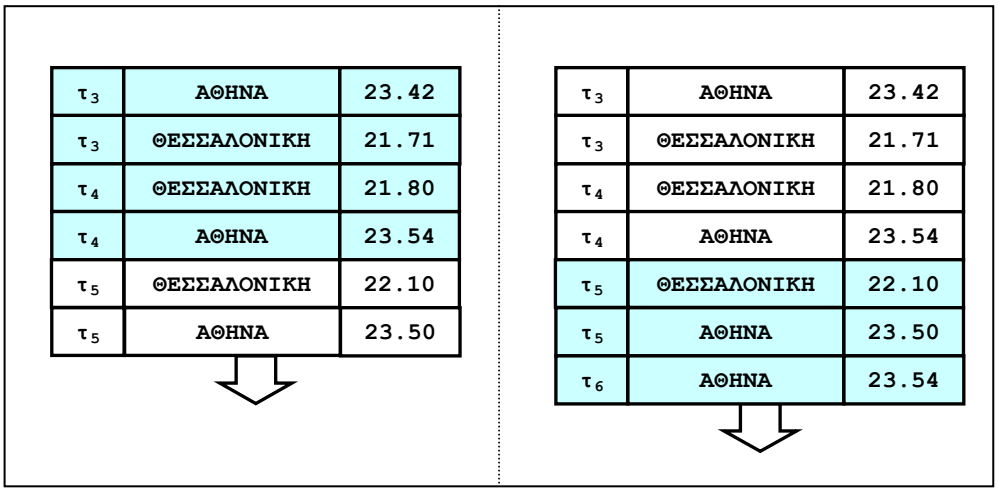
$$W_s(S, \tau, \omega, \delta) = \{s \in S(\tau) : s.A_\tau \in \text{scope}_s(\tau, \omega, \delta)\}$$

Επειδή γενικά ισχύει $\delta < \omega$, τα περιεχόμενα δύο συναπτών στιγμιοτύπων του κυλιόμενου παραθύρου εμφανίζουν επικάλυψη. Στο μεταξύ, τα περιεχόμενά του μένουν αναλλοίωτα μέχρι η συνάρτηση να εφαρμοστεί ξανά στον επόμενο παλμό, μετά από δ χρονικές στιγμές. Αυτό εκφράζεται από την αναδρομική έκφραση στον μεσαίο κλάδο της συνάρτησης εμβέλειας: τα άκρα του παραθύρου δεν μεταβάλλονται παρά μόνο στις χρονικές στιγμές που προδιαγράφει το βήμα δ. Το τρίτο σκέλος της συνάρτησης προνοεί για το ενδεχόμενο αρχικών «λειψών» παραθύρων αμέσως μετά την υποβολή του ερωτήματος, όταν η εμβέλεια υπερβαίνει τη χρονική έκταση των τρεχόντων περιεχομένων του ρεύματος. Τέλος, αφού η συνάρτηση εμβέλειας είναι *μονότονη* (εφόσον η χρονική εξέλιξη συνεπάγεται ομόλογη μετακύλιση των διαστημάτων) και μπορεί να οριστεί ακόμη και για μελλοντικές στιγμές, καλύπτονται όλα τα προσεχή στοιχεία του ρεύματος, ασχέτως πότε και αν τελικά εμφανιστούν.

Συνήθως, το βήμα δ έχει μέγεθος όσο κι η *μονάδα μέτρησης* του χρόνου (λ.χ. δευτερόλεπτο), ώστε η πρόοδος του παραθύρου να συμβαδίζει απολύτως με την αντίστοιχη χρονική (σχήμα 3.3). Έτσι με την εφαρμογή ενός τέτοιου κυλιόμενου παραθύρου αποσπώνται διαρκώς από το ρεύμα οι πλειάδες των ω τελευταίων χρονικών στιγμών.

3.5.2 *Επάλληλα παράθυρα* (*Tumbling windows*)

Τα επάλληλα παράθυρα μπορούν να θεωρηθούν συγγενικά των κυλιόμενων. Στα κυλιόμενα παράθυρα η συνάρτηση εμβέλειας, εφαρμόζεται για τιμές βήματος δ εν γένει μικρότερες του εύρους ω. Ωστόσο, η συνάρτηση εμβέλειας των κυλιόμενων χρονικών παραθύρων ισχύει αυτούσια ακόμα και



Σχήμα 3.4: *Παράδειγμα επάλληλου παραθύρου, με βήμα δ = 2 και κύλιση ω=2 χρονόσημα*

στην περίπτωση όπου το χρονικό βήμα θεωρείται μεγαλύτερο ή ίσο του εύρους. Στην περίπτωση αυτή, τα περιεχόμενα του ρεύματος θα επιστρέφονται κατά κύματα, όπως συμβαίνει με τα επάλληλα παράθυρα. Βασικός στόχος των παραθύρων αυτού του τύπου είναι η αποφυγή των επικαλύψεων σε διαδοχικά στιγμιότυπα των περιεχόμενων τους, διαφοροποιώντας τα κατά τον τρόπο αυτό από τα κυλιόμενα.

Στην πλέον συνήθη περίπτωση εφαρμογής επάλληλων παραθύρων, το άλμα δ θεωρείται ίσο προς το εύρος $\omega \in N$ του παραθύρου. Με την συνθήκη αυτή διασφαλίζεται ότι όλες οι πλειάδες του ρεύματος θα περιληφθούν κάποια στιγμή στα περιεχόμενα του παραθύρου. Αυτή η περίπτωση απεικονίζεται στο σχήμα 3.4 με εύρος ω και βήμα δ τέτοια ώστε $\delta = \omega = 2$.

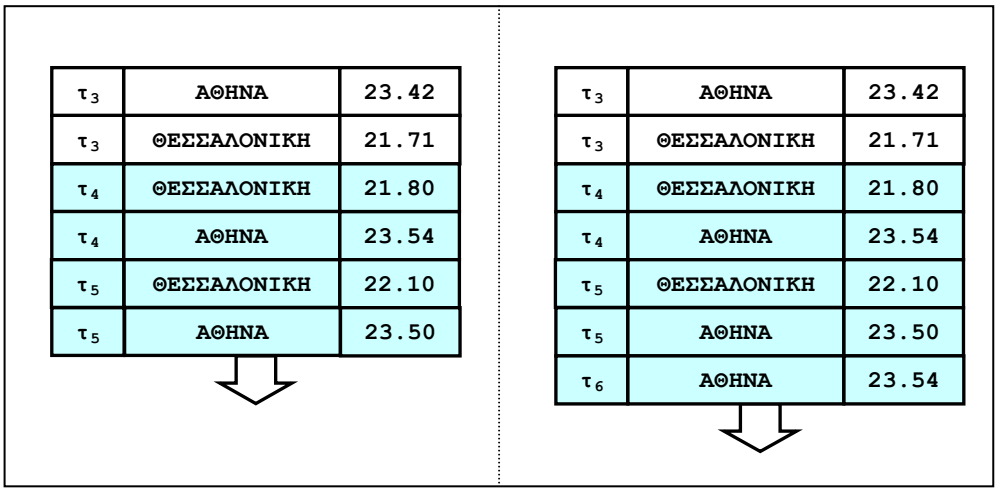
Για τον προσδιορισμό των περιεχομένων ενός επάλληλου παραθύρου W με $\delta = \omega$ εφαρμόζεται η ακόλουθη συζευκτική συνθήκη, αφού πρώτα προηγηθεί υπολογισμός της συνάρτησης εμβέλειας $\text{scope}_S(\tau, \omega, \omega)$ που ορίστηκε στα κυλιόμενα παράθυρα. Έτσι οι πλειάδες του παραθύρου για κάθε χρονική στιγμή $\tau \in T$ δίνονται απ' τη σχέση:

$$W_t(S, \tau, \omega, \omega) = \{s \in S(\tau) : s.A_\tau \in \text{scope}_s(\tau, \omega, \omega)\}$$

Τα επάλληλα παράθυρα αποδεικνύονται ιδιαιτέρως χρήσιμα όταν ζητούνται συναθροιστικά αποτελέσματα σε διαδοχικά, μη επικαλυπτόμενα τμήματα του ρεύματος **[ACC+03]**. Υπάρχει βεβαίως η δυνατότητα να οριστούν επάλληλα παράθυρα με βήμα δ μεγαλύτερο από το εύρος ω αλλά η περίπτωση αυτή οδηγεί σε απώλεια πληροφορίας. Συγκεκριμένα αν με τ_{END} συμβολίζεται το μεγαλύτερο χρονόσημο που εμφανίζεται κάθε φορά στα περιεχόμενα του παραθύρου, τότε στην επόμενη ανανέωσή του, θα χαθούν όλα εκείνα τα στοιχεία του ρεύματος που φέρουν χρονόσημο εντός του διαστήματος $[\tau_{END}+1, \tau_{END}+\delta-\omega]$. Ένα τέτοιο παράθυρο με παραμέτρους $\delta = 10$ και $\omega = 2$ λεπτά, χρησιμοποιείται για να παρέχει κάθε 10 λεπτά πρόσβαση στα περιεχόμενα των 2 τελευταίων λεπτών.

3.5.3 *Παράθυρα οροσήμου* (*Landmark windows*)

Τέτοιας μορφής παράθυρα διατηρούν πάντοτε σταθερό ένα από τα δύο άκρα τους, «σημαδεύοντας» αυθαίρετα ένα χρονικό σημείο ενδιαφέροντος. Τα περιεχόμενά τους προσδιορίζονται βάσει χρονοσήμου, ενώ το βήμα τους είναι μοναδιαίο σε επίπεδο πλειάδας. Για να οριστεί το χρονικό διάστημα εφαρμογής, δηλαδή η *εμβέλεια* του παραθύρου, διακρίνονται οι εξής περιπτώσεις:



τ_3	ΑΘΗΝΑ	23.42
τ_3	ΘΕΣΣΑΛΟΝΙΚΗ	21.71
τ_4	ΘΕΣΣΑΛΟΝΙΚΗ	21.80
τ_4	ΑΘΗΝΑ	23.54
τ_5	ΘΕΣΣΑΛΟΝΙΚΗ	22.10
τ_5	ΑΘΗΝΑ	23.50

τ_3	ΑΘΗΝΑ	23.42
τ_3	ΘΕΣΣΑΛΟΝΙΚΗ	21.71
τ_4	ΘΕΣΣΑΛΟΝΙΚΗ	21.80
τ_4	ΑΘΗΝΑ	23.54
τ_5	ΘΕΣΣΑΛΟΝΙΚΗ	22.10
τ_5	ΑΘΗΝΑ	23.50
τ_6	ΑΘΗΝΑ	23.54

Σχήμα 3.5: *Παράδειγμα παράθυρου οροσήμου, με αφετηρία την χρονική στιγμή : $\tau_l = \tau_4$*

- Η *αφετηρία* τ_l του παραθύρου θεωρείται γνωστή, οπότε για κάθε χρονική στιγμή $\tau \in T$ που αποτελεί το *πέρας* του («*ανοιχτό άκρο*») και ισχύει $\tau \geq \tau_0$, η εμβέλεια προκύπτει από την εξής συνάρτηση:

$$\text{scope}_l(\tau, \tau_l) = \begin{cases} \varnothing & , \text{ αν } \tau_0 \leq \tau < \tau_l \\ [\tau_l, \tau] & , \text{ αν } \tau \geq \tau_l \geq \tau_0 \end{cases}$$

Τα περιεχόμενα του παραθύρου προκύπτουν καθώς τα χρονόσημα των πλειάδων του ρεύματος ελέγχονται ως προς την εμβέλεια (σχήμα 3.5). Θέτοντας ως αφετηρία $\tau_l \in T$, το παράθυρο οροσήμου που εφαρμόζεται στο ρεύμα δεδομένων S κατά τη στιγμή τ είναι:

$$W_l(S, \tau, \tau_l) = \{s \in S(\tau) : s.A_\tau \in \text{scope}_l(\tau, \tau_l)\}$$

Στην προκειμένη περίπτωση, το παράθυρο θα συνεχίσει να παρέχει πλειάδες επ' αόριστον, μέχρις ότου το σχετικό ερώτημα διαρκείας καταργηθεί (ή το ρεύμα εξαντληθεί).

- Η περίπτωση να είναι δεσμευμένο το *πέρας* τ_u του παραθύρου είναι λιγότερο συχνή: καθώς η αφετηρία διατρέχει το χρόνο, αναπόφευκτα το άνω όριο κάποτε θα υπερκαλυφθεί και το παράθυρο θα παγιωθεί (θα «κλείσουν» τα περιεχόμενά του). Αν $\tau_0 \in T$ η χρονική στιγμή υποβολής του ερωτήματος διαρκείας, τότε για κάθε χρονική στιγμή $\tau \in T$ (με $\tau \geq \tau_0$) η αντίστοιχη συνάρτηση εμβέλειας είναι:

$$\text{scope}_u(\tau, \tau_u) = \begin{cases} [\tau_0, \tau] & , \text{ αν } \tau_0 \leq \tau \leq \tau_u \\ [\tau_0, \tau_u] & , \text{ αν } \tau > \tau_u \geq \tau_0 \end{cases}$$

Προφανώς, δεν έχει έννοια ο προσδιορισμός παραθύρων με πέρατα χρονικές στιγμές στο παρελθόν ($\tau_u < \tau_0$). Άρα, τα περιεχόμενα του παραθύρου με πέρας $\tau_u \in T$ προκύπτουν κατά τη χρονική στιγμή τ ως εξής:

$$W_u(S, \tau, \tau_u) = \{s \in S(\tau) : \; s.A_\tau \in \text{scope}_u(\tau, \tau_u)\}$$

Από τον ανωτέρω ορισμό της συνάρτησης εμβέλειας, είναι σαφές ότι τα άκρα του παραθύρου θα παγιωθούν όταν $\tau > \tau_u$. Για το μοντέλο των *σωρευτικών* (*append-only*) ρευμάτων δεδομένων, τα περιεχόμενα του παραθύρου τότε θα «παγώσουν» και η χρησιμότητά τους έκτοτε θα είναι παραπλήσια μ' εκείνη των συμβατικών σχεσιακών πινάκων (για όσο χρόνο τα στοιχεία κρατηθούν στη μνήμη του συστήματος).

3.5.4 *Πάγια παράθυρα ζώνης* (*Fixed–band windows*)

Όταν και τα δύο άκρα τ_l, $\tau_u \in T$ του παραθύρου οροσήμου θεωρηθούν μονίμως *φραγμένα*, τότε για κάθε $\tau \in T$ το χρονικό διάστημα της εμβέλειάς του δίνεται από τη συνάρτηση:

$$\text{scope}_f(\tau, \tau_l, \tau_u) = \begin{cases} \varnothing & , \text{ αν } \tau < \tau_l \\ [\tau_l, \tau] & , \text{ αν } \tau_l \le \tau \le \tau_u \\ [\tau_l, \tau_u] & , \text{ αν } \tau > \tau_u \end{cases}$$

Τα περιεχόμενα του παραθύρου επίτηδες δεν συσχετίζονται με την στιγμή τ_0 υποβολής του ερωτήματος, ώστε θεωρητικά να μπορούν να καλυφθούν αυθαίρετες χρονικές ζώνες στο παρελθόν. Τότε ουσιαστικά πρόκειται για πάγια *παράθυρα ζώνης* (*band windows*). Συμβολικά:

$$W_f(S, \tau, \tau_u, \tau_l) = \{s \in S(\tau) : \; s.A_\tau \wedge \tau_l \le \tau_u \wedge \tau_l \le s.A_\tau \le \tau_u\}$$

Από σημασιολογική άποψη, επιστρέφονται όσες πλειάδες φέρουν χρονόσημα που εμπίπτουν στην πάγια εμβέλεια του παραθύρου. Αν για την τρέχουσα χρονική στιγμή τ (το πιο πρόσφατο χρονόσημο) ισχύει $\tau \le \tau_l$, τότε δεν θα βρεθεί κανένα στοιχείο. Όσο ισχύει $\tau_l \le \tau \le \tau_u$, θα επιστρέφονται πλειάδες, εφόσον βέβαια υπάρχουν κάποιες με χρονόσημα εντός εμβέλειας. Απ' τη στιγμή τ_u κι έπειτα, δηλαδή όταν $\tau \ge \tau_u$, τα περιεχόμενα του παραθύρου θα παραμένουν αναλλοίωτα, κάνοντας φυσικά την υπόθεση ότι είναι δυνατόν να τηρούνται στη μνήμη επ' αόριστον, χωρίς να χρειαστεί να διαγραφούν.

3.6 Συντακτικές δομές προσδιορισμού παραθύρων

Στην ενότητα αυτή περιγράφεται ένας τρόπος σύνταξης των εκφράσεων που επιτρέπουν τον προσδιορισμό των παραπάνω τύπων παραθύρων σε γλώσσες μορφής SQL αντίστοιχων της CQL **[ABW03]**. Κάθε παράθυρο αναφέρεται σε ένα μόνο ρεύμα και δηλώνεται στην πρόταση FROM του ερωτήματος. Έτσι ορίζονται:

- *Παράθυρα πλειάδων*: Οι N πιο πρόσφατες πλειάδες του ρεύματος S επιλέγονται με την έκφραση S [ROWS N]. Ουσιαστικά, εξετάζονται οι πρόσφατες πλειάδες ως προς το χρονόσημο που φέρουν, ξεκινώντας από την πιο πρόσφατη τιμή του και βαδίζοντας αντίθετα προς την εξέλιξη του χρόνου, μέχρις ότου ληφθεί ο καθορισμένος αριθμός πλειάδων. Ενδέχεται βέβαια το πλήθος των πλειάδων με ίδιο χρονόσημο να υπερβαίνει ή ακόμη και να υπολείπεται του αριθμού των πλειάδων N που προκαθορίστηκε, οπότε η επιλογή θα πρέπει να γίνει κατά μη ντετερμινιστικό τρόπο. Λ.χ. αν υπάρχουν 24 πλειάδες με ίδιο χρονόσημο και το παράθυρο προσδιορίζει την επιλογή μόνο 20 εξ αυτών ή υπολείπονται ακόμη 20 για να συμπληρωθεί ο επιδιωκόμενος αριθμός, οι πλειάδες αυτές θα πρέπει να επιλεγούν αυθαίρετα.

- *Μεριστικά παράθυρα*: Αυτά διαμερίζουν το ρεύμα *S* σε υπορεύματα βάσει της λίστας γνωρισμάτων ομαδοποίησης $L=\{A_1,A_2, ..., A_k\}$. Για κάθε υπορεύμα, υπολογίζεται ένα κυλιόμενο παράθυρο μεγέθους *N* πλειάδων και τα επιμέρους αποτελέσματα συνενώνονται για να παραχθούν τα τελικά περιεχόμενα του παραθύρου. Κατ' ουσίαν, κάποια πλειάδα του ρεύματος με συγκεκριμένες τιμές για τα *γνωρίσματα ομαδοποίησης* θα συμμετέχει στο παράθυρο, μόνον εφόσον το χρονόσημο την κατατάσσει μεταξύ των πλέον πρόσφατων πλειάδων της ομάδας με ίδιες τιμές στα συγκεκριμένα γνωρίσματα. Συνολικά, η έκφραση είναι:
  S [PARTITION BY $A_1, A_2,..., A_k$ ROWS N]

- *Χρονικά παράθυρα*: Με την έκφραση *S* [RANGE *T* SLIDE *D*], μπορούν να οριστούν *κυλιόμενα παράθυρα* πάνω στα περιεχόμενα του ρεύματος *S*. Αν τυχόν T=1, το παράθυρο αναφέρεται μόνο στις πλειάδες με το πιο πρόσφατο χρονόσημο, σύνταξη που ισοδυναμεί με την έκφραση *S* [*NOW*]. Επίσης, αν στο ερώτημα χρειαστεί να γίνει αναφορά σε όλες τις πλειάδες του ρεύματος τότε $T = \infty$, γεγονός που αποδίδεται από την έκφραση *S* [RANGE UNBOUNDED], απαραίτητο δάνειο από την SQL-99. Στην ιδιαίτερη περίπτωση των *επάλληλων παραθύρων* η πρόταση διαμορφώνεται ως *S* [RANGE *T* SLIDE *T*], ορίζοντας χρονικές περιόδους μεγέθους *T*.

- *Παράθυρα ορισήμου*, όπου συνήθως η αφετηρία παραμένει σταθερή στο χρόνο, ενώ το πέρας είναι πάντοτε η τρέχουσα χρονική στιγμή. Επιπλέον, δεν έχει νόημα ο ορισμός τους βάσει αριθμού πλειάδων, εφόσον δεν μπορεί να είναι γνωστό το πλήθος των πλειάδων που θα εισρεύσει στο σύστημα την επόμενη χρονική στιγμή. Συνεπώς, η σύνταξη *S* [AFTER *t*], θυμίζει αρκετά την αντίστοιχη των χρονικών παραθύρων, αντί όμως για χρονικό διάστημα προσδιορίζεται μια συγκεκριμένη χρονική στιγμή *t*, ασχέτως αν υπάρχει παρόμοιο χρονόσημο για οποιαδήποτε πλειάδα του ρεύματος *S*. Κατ' ανάλογο τρόπο μπορεί να αντιμετωπιστεί η περίπτωση όπου ζητούνται όλες οι πλειάδες του ρεύματος πριν μια καθορισμένη χρονική στιγμή *t*, ορίζοντας ένα παράθυρο της μορφής *S* [BEFORE *t*], μολονότι ένα τέτοιο ενδεχόμενο κρίνεται μάλλον εξεζητημένο. Οι δύο δομές μπορούν να συνδυαστούν για να συγκροτήσουν *πάγιο παράθυρο ζώνης* καθορίζοντας τη σχετική χρονική περίοδο ενδιαφέροντος ως: *S* [AFTER t_1 AND BEFORE t_2], με $t_1 \leq t_2$.

Αν τυχόν δεν προσδιορίζεται οποιασδήποτε μορφής παράθυρο, τότε υποτίθεται ότι το ερώτημα ή το υποερώτημα αναφέρεται σε όλα τα στοιχεία του ρεύματος που έχουν εισέλθει στο σύστημα. Στην περίπτωση αυτή, το ερώτημα θα πρέπει να διατυπώνεται προσθέτοντας την έκφραση *S* [ROWS UNBOUNDED] ή ισοδύναμα *S* [RANGE UNBOUNDED]. Τέλος επισημαίνεται ότι στο κεφάλαιο 5 θα γίνει χρήση τέτοιων συντακτικών δομών, για την διατύπωση χαρακτηριστικών ερωτημάτων διαρκείας.

Κεφάλαιο 4

Επεξεργασία ερωτημάτων διαρκείας

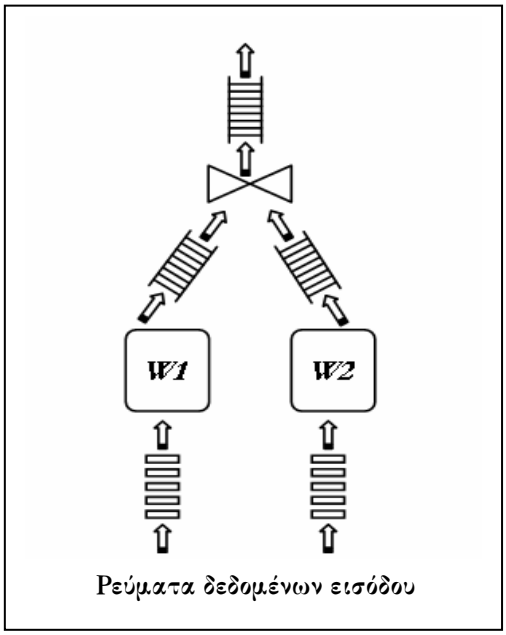
4.1 Εισαγωγή

Τα ερωτήματα διαρκείας διατυπώνονται από τον χρήστη είτε με την βοήθεια κάποιας γλώσσας παραπλήσιας με την SQL, είτε μέσω κάποιου γραφικού περιβάλλοντος που παρέχεται από το Σύστημα Διαχείρισης Ρευμάτων Δεδομένων. Ακολουθεί η επιλογή ενός κατάλληλου *φυσικού προσχεδίου εκτέλεσης* (*physical query plan*) από το σύστημα για την επεξεργασία τους, όπου καθορίζεται το είδος και η διάταξη των τελεστών και των παραθύρων που θα χρησιμοποιηθούν κατά την εκτέλεση του ερωτήματος (σχήμα 4.1). Ωστόσο, αρκετοί τελεστές που χρησιμοποιούνται κατά την διατύπωση των ερωτημάτων διαρκείας, αν και συνήθως προέρχονται από τροποποιήσεις ανάλογων τελεστών της σχεσιακής άλγεβρας, παρουσιάζουν ιδιαίτερα προβλήματα κατά την προσαρμογή τους στο περιβάλλον επεξεργασίας ρευμάτων δεδομένων.

Κατά την επιλογή κατάλληλων προσχεδίων για τα ερωτήματα όσο και κατά την εκτέλεση τους, προκύπτουν επιπρόσθετοι προβληματισμοί από παράγοντες που έχουν να κάνουν με την αχανή φύση των ρευμάτων, τους περιορισμούς στους διαθέσιμους πόρους του συστήματος, τις απαιτήσεις των ερωτημάτων διαρκείας και εν τέλει, το νέο μοντέλο επεξεργασίας που διαμορφώνεται (*push model*). Έτσι ανακύπτουν ζητήματα σχετικά με:

- την υλοποίηση των τελεστών,
- την διάταξη των στοιχείων από τα εισερχόμενα ρεύματα,
- την πολιτική χρονοδρομολόγησης των τελεστών,
- τον υπολογισμό του κόστους των ερωτημάτων και
- την διαδικασία βελτιστοποίησης.

Στις ενότητες που ακολουθούν αναλύονται διεξοδικά τα ζητήματα αυτά και παρουσιάζονται λύσεις που έχουν προταθεί από διαφορές ερευνητικές ομάδες για την ικανοποιητική αντιμετώπιση τους. Παράλληλα περιγράφεται η μορφή των τελεστών που υλοποιήθηκαν καθώς και ο βασικός άξονας της επεξεργασίας των σχετικών ερωτημάτων.



Σχήμα 4.1: *Παράδειγμα φυσικού προσχεδίου εκτέλεσης ενός ερωτήματος διαρκείας*

4.2 Τελεστές ερωτημάτων διαρκείας

Οι τελεστές που απαιτούνται για την επεξεργασία ερωτημάτων διαρκείας σε ρεύματα δεδομένων, αντιστοιχούν συνήθως σε ανάλογους τελεστές της σχεσιακής άλγεβρας που χρησιμοποιούνται στα συστήματα Βάσεων Δεδομένων. Ωστόσο οι διαφορετικές συνθήκες κάτω από τις οποίες καλούνται να λειτουργήσουν, θέτουν σε νέα βάση τον σχεδιασμό και την υλοποίησή τους.

Τα κυριότερα νέα ζητήματα συνδέονται με τα εξής:

1) Η πιθανή έλλειψη διάταξης ή συγχρονισμού των εισερχόμενων ρευμάτων.
2) Η συνεχής άφιξη νέων στοιχείων πληροφορίας, ενώ το σύστημα ήδη επεξεργάζεται άλλα προγενέστερα δεδομένα.
3) Το πιθανόν άπειρο πλήθος στοιχείων προς επεξεργασία, σε συνδυασμό με τους περιορισμούς του συστήματος σε διαθέσιμους πόρους μνήμης και επεξεργαστικής ισχύος.
4) Οι τυχόν διαγραφές στοιχείων από τα αποτελέσματα, καθώς με την πάροδο του χρόνου και την εξέλιξη των ρευμάτων, καθίστανται παρωχημένα.
5) Την απαιτούμενη υποστήριξη διοχέτευσης των αποτελεσμάτων τους.

Τελεστές όπως η *προβολή* (*projection*) και η *επιλογή* (*selection*) προσαρμόζονται χωρίς ιδιαίτερη δυσκολία στις παραπάνω συνθήκες. Οι τελεστές αυτοί μπορούν να εφαρμοστούν απευθείας πάνω σε κάποιο ρεύμα δεδομένων παράγοντας ένα νέο ρεύμα εξόδου. Κάθε νέο στοιχείο της εισόδου τους, υπόκειται την επεξεργασία του τελεστή και τα αποτελέσματα προστίθενται στο ρεύμα εξόδου. Έτσι οι τελεστές αυτοί δουλεύουν καλά με *σωρευτικά* (*append – only*) ρεύματα δεδομένων στα οποία προστίθενται διαρκώς νέα στοιχεία χωρίς ποτέ να διαγράφονται. Ωστόσο, ακόμα και στις προτεινόμενες στρατηγικές διαγραφής στοιχείων από ρεύματα δεδομένων, οι τελεστές αυτοί διατηρούν ένα ιδιαίτερα κρίσιμο χαρακτηριστικό: επεξεργάζονται κάθε στοιχείο της εισόδου τους ανεξάρτητα από τα υπόλοιπα χωρίς να χρειάζεται να τηρούν καμία πρόσθετη πληροφορία.

Αυτή την ιδιότητα δεν την εμφανίζουν αρκετοί άλλοι τελεστές, οι οποίοι υπάγονται είτε στην κατηγορία των ανασταλτικών τελεστών (*blocking operators*) είτε στην ευρύτερη κατηγορία των τελεστών διατήρησης κατάστασης (*stateful operators*) **[TMSF02]**. Ανασταλτικοί θεωρούνται οι τελεστές που, σύμφωνα με το σχεσιακό μοντέλο επεξεργασίας, καταναλώνουν όλα τα δεδομένα

εισόδου τους προτού αρχίσουν να παράγουν τα αποτελέσματα τους. Στην κατηγορία αυτή υπάγονται τελεστές όπως:

- *ομαδοποίηση* (*group – by*)
- *ταξινόμηση* (*sorting*)
- *διαφορά* (*difference*).

Οι τελεστές διατήρησης κατάστασης απαιτούν την διατήρηση πληροφορίας για όλα τα στοιχεία των εισόδων τους. Η πληροφορία αυτή πέρα από την απλή λύση της διατήρησης όλων των πλειάδων, μπορεί να έχει την μορφή κατάλληλα κατασκευασμένων περιλήψεων ή συνόψεων (summaries ή synopses). Στη κατηγορία αυτή υπάγονται οι τελεστές:

- *σύνδεση* (*join*)
- *απαλοιφή διπλοτύπων* (*duplicate elimination*)
- *τομή* (*intersection*).

Κοινά χαρακτηριστικά των τελεστών που ανήκουν στις δύο κατηγορίες, είναι η αδυναμία τους να επεξεργαστούν κάθε στοιχείο ανεξάρτητα από τα υπόλοιπα, καθώς και η απαίτηση απεριόριστων διαθέσιμων πόρων (μνήμη και διαθέσιμος χρόνος επεξεργασίας). Η λύση που συνήθως προτείνεται για την απεμπλοκή τους είναι η επιβολή παραθύρων στα ρεύματα δεδομένων που καλούνται να επεξεργαστούν. Οι τελεστές πλέον εφαρμόζονται πάνω στα περιεχόμενα των παραθύρων επιτυγχάνοντας τον περιορισμό του πλήθους των στοιχείων που επεξεργάζονται και κατά συνέπεια του απαιτούμενου χρόνου επεξεργασίας. Μάλιστα, αυτός ο συνδυασμός παραθύρων και τελεστών, μπορεί να θεωρηθεί ότι οδηγεί στον ορισμό νέων τύπων τελεστών όπως λ.χ. *παραθυρική σύνδεση* (*windowed – join*) ή *παραθυρική συνάθροιση* (*windowed – aggregation*).

Το κόστος αυτής της λύσης είναι η εισαγωγή προσεγγίσεων στις απαντήσεις λόγω του μειωμένου αριθμού στοιχείων που οδηγήθηκαν σε επεξεργασία. Ωστόσο, το μέγεθος της υπεισερχόμενης προσέγγισης προσδιορίζεται με σαφήνεια από τον τύπο του επιλεγόμενου παραθύρου και τις παραμέτρους του. Άλλωστε, συμβαίνει συχνά το ίδιο το ερώτημα να εστιάζει στην επεξεργασία των πιο πρόσφατων στοιχείων πληροφορίας, με τα παράθυρα να λειτουργούν ως μηχανισμός εξυπηρέτησης της απαίτησης αυτής, χωρίς επιπτώσεις στην ακρίβεια των απαντήσεων.

Ωστόσο η χρήση των παραθύρων επιδεινώνει το πρόβλημα διαγραφών ή ενημερώσεων από το σύνολο της απάντησης των ερωτημάτων, αυξάνοντας την συχνότητα με την οποία συμβαίνουν. Συγκεκριμένα, η διαρκής άφιξη νέων στοιχείων στο ρεύμα προκαλεί την ανανέωση των περιεχομένων του παραθύρου που τροφοδοτεί κάποιον από τους παραπάνω προβληματικούς τελεστές. Στην περίπτωση όπου ο τελεστής υπολογίζει από την αρχή όλο το αποτέλεσμα της απάντησης υπό την μορφή υλοποιημένης όψης, τότε δεν δημιουργείται κάποιο επιπρόσθετο κόστος από την εισαγωγή παραθύρων. Αν όμως η επιστρεφόμενη απάντηση διατηρεί την μορφή ρεύματος, τότε πέρα από τις νέες πλειάδες της απάντησης που ούτως ή άλλως θα προστεθούν, θα πρέπει να διαγραφούν με κάποιο τρόπο εκείνες οι πλειάδες του ρεύματος απάντησης, που προήλθαν από στοιχεία τα οποία εκπίπτουν από την εμβέλεια του εκάστοτε παραθύρου. Αυτή λοιπόν η διαρκής μεταβολή του κάτω άκρου του επιβαλλόμενου παραθύρου, θα αυξήσει το πλήθος των επιβαλλόμενων διαγραφών ή ενημερώσεων. Σε κάθε περίπτωση όμως, η εισαγωγή παραθύρων κρίνεται αναγκαία, εφόσον οι πιθανές συνέπειες της χρήσης τους, αντισταθμίζονται ικανοποιητικά από το κέρδος της απεμπλοκής των προβληματικών τελεστών. Σε αυτό το συμπέρασμα συνηγορεί και το γεγονός της ενσωμάτωσης τους στα κυριότερα πρότυπα συστήματα διαχείρισης ρευμάτων δεδομένων (Aurora, Gigascope, STREAM, TelegraphCQ).

Τέλος, όλοι οι τελεστές επιβάλλεται να υποστηρίζουν την λειτουργία *διοχέτευσης* (*pipelining*) των αποτελεσμάτων τους, παρέχοντας άμεσα τις παραγόμενες πλειάδες τους, στους τελεστές που ακολουθούν στο επιλεγμένο προσχέδιο εκτέλεσης (σχήμα 4.1). Άλλωστε στις εφαρμογές ρευμάτων δεδομένων δεν μπορεί να σταθεί αποδοτικά η εναλλακτική τεχνική της *αποθήκευσης* (*materialization*) των απαντήσεων, οι οποίες γίνονται διαθέσιμες μόνο όταν ολοκληρωθεί η επεξεργασία όλων των στοιχείων εισόδου. Σε αυτό συνηγορούν οι τρεις παρακάτω λόγοι:

1) Το πλήθος των στοιχείων των ρευμάτων είναι απεριόριστο, με συνέπεια κάθε τελεστής να παραμένει διαρκώς σε κατάσταση αναμονής νέων στοιχείων από το ρεύμα.

2) Ακόμα και αν το παραπάνω πρόβλημα αντιμετωπιστεί, π.χ. με την εφαρμογή παραθύρων στις εισόδους των τελεστών, η υπεισερχόμενη καθυστέρηση στην παραγωγή απαντήσεων είναι αρκετά μεγάλη για να ικανοποιήσει τις απαιτήσεις έγκαιρων απαντήσεων που προβάλλουν τα ερωτήματα διαρκείας.
3) Σε οποιοδήποτε σενάριο *αποθήκευσης* των αποτελεσμάτων των τελεστών προκύπτει σημαντικό πρόβλημα με την διαθέσιμη κύρια μνήμη. Οι τελεστές θα πρέπει να αποθηκεύουν το πιθανότατα μεγάλο πλήθος των απαντήσεων τους μέχρι να επεξεργαστούν όλα τα στοιχεία τους, ενώ αυτά θα ήταν προτιμότερο να προωθηθούν για κατανάλωση στους επόμενους τελεστές το συντομότερο δυνατό.

Η βέλτιστη εφαρμογή της τεχνικής διοχέτευσης προβλέπει την άμεση διάθεση των απαντήσεων του τελεστή, όπως προκύπτουν από την επεξεργασία κάθε πλειάδας. Κατά τον τρόπο αυτό, τα αποτελέσματα αποδεσμεύονται από τους τελεστές το συντομότερο δυνατό, ενώ οι απαντήσεις αρχίζουν να παράγονται σε εύλογο χρονικό διάστημα. Ωστόσο, με αυτήν την τεχνική δυσχεραίνονται οι διαγραφές και οι ενημερώσεις των προηγούμενων αποτελεσμάτων, αφού μετά την αποδέσμευση των πλειάδων απάντησης, οι τελεστές δεν έχουν άμεση πρόσβαση σε αυτές. Στην υλοποίηση του συστήματος που παρουσιάζεται στο επόμενο κεφάλαιο, ακολουθήθηκε η τεχνική της διοχέτευσης των αποτελεσμάτων των τελεστών, χωρίς ωστόσο να αντιμετωπιστεί το πρόβλημα των ενημερώσεων του συνόλου της απάντησης.

Στην συνέχεια παρουσιάζονται οι κυριότεροι παραθυρικοί τελεστές και ο επιδιωκόμενος τρόπος λειτουργίας του. Για την αλγεβρική αναπαράσταση των ρευμάτων χρησιμοποιείται η περιγραφή που δόθηκε στο προηγούμενο κεφάλαιο, ενώ προτάσεις για την βελτιστοποίηση της λειτουργίας των τελεστών παρέχονται σε επόμενη ενότητα αυτού του κεφαλαίου.

4.2.1 Παραθυρική Προβολή

Ο τελεστής προβολής μπορεί να εφαρμοστεί απευθείας πάνω σε κάποιο ρεύμα δεδομένων χωρίς να προϋποθέτει την εφαρμογή παραθύρου στην είσοδό του. Ωστόσο, στην παραθυρική εκδοχή του, ορίζεται ο τελεστής παραθυρικής προβολής $\pi_{W,L}$ ο οποίος εφαρμόζει ένα παράθυρο W στο εισερχόμενο ρεύμα S και διατηρεί για κάθε πλειάδα του παραθύρου ένα συγκεκριμένο σύνολο γνωρισμάτων που ορίζεται από την λίστα $L = \{A_1, A_2, ..., A_k\}$. Συμβολικά:

$$\pi_{W,L}(S(\tau)) = \{ \langle s.A_1, s.A_2, ..., s.A_k, s.A_\tau \rangle : s \in W(S(\tau)) \}$$

Στην παραπάνω σχέση αξίζει να σημειωθεί ότι στις πλειάδες εξόδου προσαρτάται πάντοτε το πεδίο χρονοσήμου A_τ που χρησιμοποιείται για την διάταξη των στοιχείων της εξόδου του τελεστή. Η εφαρμογή αυτού του τελεστή οδηγεί σε εξοικονόμηση μνήμης – μικρής συνήθως κλίμακας – ως αποτέλεσμα της απόρριψης ορισμένων γνωρισμάτων που δεν χρησιμοποιούνται από τα ερωτήματα.

4.2.2 Παραθυρική Επιλογή

Όπως και ο τελεστής προβολής, έτσι και ο τελεστής επιλογής μπορεί να εφαρμοστεί απευθείας πάνω σε κάποιο ρεύμα δεδομένων. Ορίζεται ωστόσο και η παραθυρική εκδοχή του $\sigma_{W,F}$ όπου εφαρμόζεται μία συνθήκη επιλογής F στα περιεχόμενα ενός παραθύρου W για το εισερχόμενο ρεύμα S. Συμβολικά:

$$\sigma_{W,F}(S(\tau)) = \{ s \in W(S(\tau)) \wedge F(s) : \text{αληθής} \}$$

Η συνθήκη επιλογής F μπορεί να έχει τις εξής μορφές:

- $s.A_k = a_k$, οπότε εφαρμόζεται κάποιο κατηγόρημα επιλογής ανάμεσα σε ένα γνώρισμα και μία τιμή από το πεδίο ορισμού αυτού του γνωρίσματος ($a_k \in D_k$) ή

- $s.A_i = s.A_j$, οπότε εφαρμόζεται μία συνθήκη μεταξύ δύο οποιωνδήποτε γνωρισμάτων κάθε πλειάδας πλην του χρονοσήμου.

Σημειώνεται ότι μπορεί να οριστεί και πιο σύνθετη συνθήκη επιλογής F αποτελούμενη από τον συνδυασμό απλών συνθηκών της παραπάνω μορφής στις οποίες δεν είναι απαραίτητη η χρήση συνθήκης ισότητας (π.χ. $>$, $\leq$ κ.α.). Ο τελεστής επιλογής μπορεί να οδηγήσει σε σημαντική μείωση της πληροφορίας που εξάγεται απ' αυτόν, ανάλογα με το πλήθος των πλειάδων που ικανοποιούν την συνθήκη F.

4.2.3 Παραθυρική Σύνδεση

Ένα ενδεικτικό παράδειγμα χρήσης του τελεστή παραθυρικής σύνδεσης με το οποίο αποσαφηνίζεται διαισθητικά η λειτουργία του δίνεται από το εξής ερώτημα:

Παράδειγμα 4.1: *Από ένα δίκτυο χωρικά κατανεμημένων αισθητήρων λαμβάνονται μετρήσεις σε δύο ρεύματα δεδομένων: το ένα με σχήμα < χρονόσημο, πόλη, θερμοκρασία > και το άλλο με σχήμα < χρονόσημο, πόλη, υγρασία >. Για λόγους απλότητας θεωρείται ότι σε κάθε πόλη υπάρχει ένας αισθητήρας θερμότητας και ένας αισθητήρας υγρασίας. Με την εφαρμογή κυλιόμενων παραθύρων στα ρεύματα αυτά επιδιώκεται η σύνδεσή τους με βάση το γνώρισμα της πόλης για την παραγωγή πλειάδων με σχήμα < χρονόσημο, πόλη, υγρασία, θερμοκρασία >.*

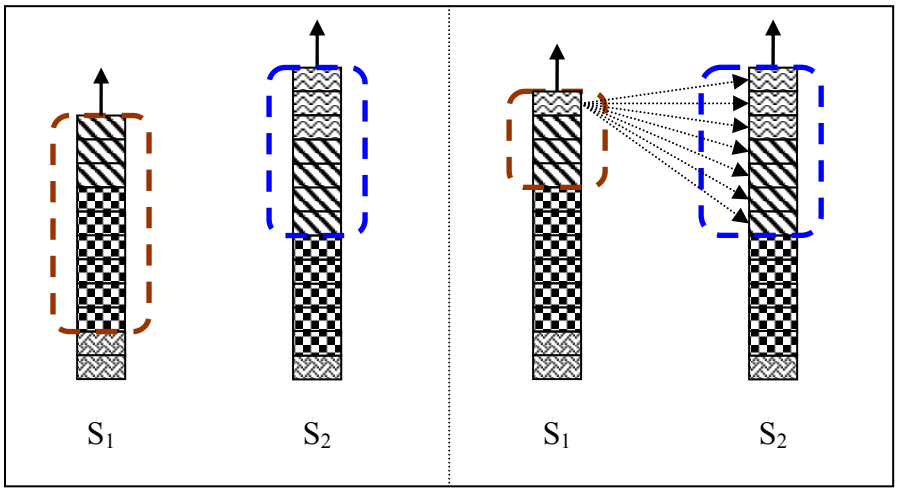
Ο σύνθετος αυτός τελεστής, περιλαμβάνει έναν συμμετρικό τελεστή σύνδεσης είτε μεταξύ δύο ρευμάτων S_1 και S_2 είτε μεταξύ ρεύματος S και σχεσιακού πίνακα R. Η τροφοδότηση του τελεστή με ρεύματα γίνεται με την διαμεσολάβηση κατάλληλων παραθύρων, (ένα για κάθε ρεύμα) χωρίς όμως να είναι αναγκαίο τα παράθυρα να έχουν την ίδια μονάδα, την ίδια εμβέλεια ή τον ίδιο τύπο. Για κάθε χρονική στιγμή $\tau \in T$, ο τελεστής *σύνδεσης παραθύρων* επιστρέφει τη *συνένωση* (*concatenation*) ζευγών πλειάδων που εμφανίζονται στα αντίστοιχα στιγμιότυπα των παραθύρων σύμφωνα με την ακόλουθη σχέση:

$$S_1(\tau) \underset{W}{\bowtie} S_2(\tau) = \{ \langle s_1, s_2, \tau_m \rangle : s_1 \in W_1(S_1(\tau)),\ s_2 \in W_2(S_2(\tau)) \ \wedge \ \wedge \ E(s_1, s_2) \ \wedge \ \tau_m = \max(s_1.A_\tau, s_2.A_\tau) \}$$

Κάθε νεοεισερχόμενη πλειάδα του ρεύματος S_1 ελέγχεται ως προς τη συνθήκη σύνδεσης E με όλα τα τρέχοντα περιεχόμενα του κυλιόμενου παραθύρου W_2 του ρεύματος S_2. Αν βρεθούν στοιχεία που να ταιριάζουν, τότε τα εξαγόμενα αποτελέσματα λαμβάνουν το πιο πρόσφατο χρονόσημο που εμφανίζεται στο ζεύγος των πρωτογενών πλειάδων. Αυτή είναι άλλωστε η χρονική ένδειξη που αποκαθιστά τη διάταξη των τελικών αποτελεσμάτων του εξαγομένου ρεύματος δεδομένων **[BBD+02, AN04]**. Εναλλακτικά, μια δεύτερη πρόταση για την σήμανση των αποτελεσμάτων θα ήταν η απόδοση της χρονικής στιγμής παραγωγής κάθε πλειάδας από τον τελεστή σύνδεσης **[BBD+02]**, κάτι που πιθανόν δημιουργεί επιπλοκές σε περιπτώσεις πολλαπλών συνδέσεων. Αντίστοιχη διαδικασία πραγματοποιείται για κάθε νέα πλειάδα του ρεύματος S_2.

Επισημαίνεται ότι υπάρχουν αρκετοί δυνατοί αλγόριθμοι για την εφαρμογή της συνθήκης σύνδεσης Ε οι κυριότεροι από τους οποίους είναι οι :

- *Παραθυρική Σύνδεση Διοχέτευσης* (*Pipelined Join*): σε αυτήν την εκδοχή η συνθήκη σύνδεσης ελέγχεται για όλα τα δυνατά ζεύγη μεταξύ των νέων πλειάδων κάθε παραθύρου και όλων των υπαρχουσών πλειάδων του άλλου παραθύρου.
- *Παραθυρική Σύνδεση Ενθέτου Βρόχου* (*Nested-Loop Window Join*): σε κάθε βήμα αυτού του αλγορίθμου υπολογίζεται η συνθήκη σύνδεσης για όλα τα δυνατά ζεύγη πλειάδων μεταξύ των περιεχομένων των δύο παραθύρων.



Σχήμα 4.2: *Σύνδεση ρευμάτων δεδομένων με χρήση κυλιόμενων παραθύρων*

Στις παραπάνω μορφές είναι εφικτή η χρήση κάποιου hash ευρετηρίου για την γρηγορότερη εύρεση των ζευγών που ικανοποιούν την συνθήκη σύνδεσης. Στις εφαρμογές ρευμάτων δεδομένων αποφεύγεται η χρήση του παραθυρικού συνδέσμου ενθέτου βρόχου, μια και συνήθως οδηγεί σε σημαντικές επικαλύψεις μεταξύ των διαδοχικών αποτελεσμάτων του. Εξαίρεση αποτελεί η περίπτωση όπου ο τελεστής τροφοδοτείται με επάλληλα παράθυρα, τότε όμως η λειτουργία του εξομοιώνεται με αυτήν της παραθυρικής σύνδεσης διοχέτευσης. Κατά την υλοποίηση αυτού του τελεστή επιλέχθηκε η μορφή της παραθυρικής σύνδεσης διοχέτευσης (ενότητα 5.5.3).

Η πιο αξιοσημείωτη υλοποίηση παραθυρικής σύνδεσης διοχέτευσης είναι αυτή που προκύπτει από την εφαρμογή *κυλιόμενων παραθύρων* (*sliding-window join*), χωριστά για κάθε ρεύμα. Ένα τέτοιο παράδειγμα με χρήση δυο κυλιόμενων παραθύρων απεικονίζεται στο σχήμα 4.2, όπου διακρίνονται δύο διαδοχικά στιγμιότυπα των ρευμάτων και των αντίστοιχων παραθύρων που τροφοδοτούν τον τελεστή. Κάθε διαφορετικό μοτίβο συμβολίζει πλειάδες του ιδίου χρονοσήμου ενώ η εμβέλεια των παραθύρων διακρίνεται με διακεκομμένες γραμμές. Και τα δύο παράθυρα έχουν μοναδιαίο βήμα και εμβέλεια 2 χρονοσήμων. Τα δύο στιγμιότυπα διαφέρουν ως προς την έλευση μίας πλειάδας στο ρεύμα S_1, η οποία μεταβάλει τα περιεχόμενα του αντίστοιχου παραθύρου, ενώ με βέλη υποδεικνύονται τα ζεύγη μεταξύ των οποίων θα επιχειρηθεί σύνδεση.

4.2.4 Παραθυρική Συνάθροιση

Ένα παράδειγμα χρήσης αυτού του τελεστή παρέχεται στο εξής ερώτημα:

Παράδειγμα 4.2: *Ζητείται η διαρκής επεξεργασία των μετρήσεων που στέλνουν αισθητήρες από διάφορες πόλεις, για την παρακολούθηση του μέσου όρου θερμοκρασίας σε κάθε πόλη κατά το διάστημα της τελευταίας μισής ώρας.*

Όπως και για τον ανάλογο τελεστή της εκτεταμένης σχεσιακής άλγεβρας, προηγείται ομαδοποίηση των πλειάδων εντός του παραθύρου που εφαρμόζεται στα στοιχεία εισόδου του. Η ομαδοποίηση γίνεται σύμφωνα με τις τιμές των στοιχείων στα γνωρίσματα του σχήματος του ρεύματος S όπως προσδιορίζονται στη *λίστα των γνωρισμάτων ομαδοποίησης* του τελεστή. Για κάθε ομάδα, δηλαδή συνδυασμό τιμών των γνωρισμάτων ομαδοποίησης, εφαρμόζεται η *συνάρτηση συνάθροισης* (COUNT, SUM, MIN, MAX, AVG) και ως χρονική ένδειξη του τελικού αποτελέσματος προσαρτάται το τρέχον χρονόσημο τ. Στο παράδειγμα 4.2 η λίστα γνωρισμάτων περιλαμβάνει μόνο το όνομα της πόλης, οπότε ο ζητούμενος μέσος όρος υπολογίζεται ομαδοποιώντας τους αισθητήρες και τις μετρήσεις τους με βάση την πόλη από την οποία προέρχονται.

Και στα ερωτήματα συνάθροισης κυριαρχεί η εφαρμογή *κυλιόμενων παραθύρων* (*sliding-window aggregates*), μορφή που είχε εμφανιστεί ήδη στην SQL-99 και σε εμπορικά συστήματα (Oracle 9i), λ.χ. με τη μορφή κινούμενων μέσων όρων για τη διευκόλυνση αναλυτικών (OLAP) λειτουργιών. Πρακτικά, η ομαδοποίηση εφαρμόζεται εκ νέου στα κυμαινόμενα περιεχόμενα του παραθύρου, αφού τα προηγούμενα αποτελέσματα συνήθως δεν ανταποκρίνονται πια στη νέα κατάσταση πλειάδων του παραθύρου.

4.2.5 Συνολοθεωρητικοί τελεστές

Οι γνωστές συνολοθεωρητικές πράξεις όταν εφαρμοστούν σε στοιχεία παραθύρων, υπακούουν στη σημασιολογία *πολυσυνόλων* (*bag semantics*). Γι' αυτό, ο συμβολισμός των πλειάδων επεκτείνεται μ' ένα επιπλέον «πλασματικό» γνώρισμα που δηλώνει το *πλήθος εμφανίσεων* k κάθε πλειάδας εντός του παραθύρου. Έτσι θεωρώντας δύο ρεύματα δεδομένων S_1 και S_2 επί των οποίων εφαρμόζονται τα παράθυρα W_1 και W_2 αντιστοίχως ορίζονται:

- Ο τελεστής *παραθυρικής ένωσης* (*windowed-union*), ο οποίος πρέπει να διατηρεί στην απάντηση, κάθε πλειάδα που περιέχεται στα παράθυρα W_1 και W_2. Το πλήθος k σε αυτήν την περίπτωση θα ισούται με το άθροισμα των εμφανίσεων της πλειάδας αυτής σε κάθε παράθυρο.
- Ο τελεστής *παραθυρικής τομής* (*windowed-intersection*). Στο σύνολο της απάντησης περιλαμβάνεται κάθε πλειάδα που περιέχεται και στα δύο παράθυρα, ενώ το πλήθος k αντιστοιχεί στο πλήθος των κοινών εμφανίσεων κάθε πλειάδας στα W_1 και W_2.
- Ο τελεστής *παραθυρικής διαφοράς* (*windowed-difference*), ο οποίος πρέπει να διατηρεί στην απάντηση κάθε πλειάδα που περιέχεται στο παράθυρο W_1 και δεν περιλαμβάνεται στο παράθυρο W_2. Σε αυτήν την περίπτωση, το πλήθος k κάθε πλειάδας που περιέχεται στο σύνολο της απάντησης, ισούται με την διαφορά μεταξύ του πλήθους των εμφανίσεων της στο παράθυρο W_1 και του αντιστοίχου πλήθους στο παράθυρο W_2.

4.3 Διάταξη και συγχρονισμός στοιχείων σε ρεύματα δεδομένων

Η χρονική διάταξη των στοιχείων κάθε ρεύματος και ο συγχρονισμός των εισερχόμενων ρευμάτων μεταξύ τους, αποτελούν ιδιαίτερα σημαντικά ζητήματα στις εφαρμογές ρευμάτων δεδομένων. Το πρόβλημα που δημιουργείται από την έλλειψη διάταξης ή συγχρονισμού εντοπίζεται κυρίως στον προσδιορισμό και την σήμανση των αποτελεσμάτων παραθυρικών τελεστών και ερωτημάτων διαρκείας γενικότερα. Έτσι, *λογικά παράθυρα* που προσδιορίζουν τα περιεχόμενά τους βάσει χρονοσήμου, μπορεί εσφαλμένα να ανανεώσουν τα περιεχόμενα τους λόγω έλευσης κάποιου στοιχείου με μεγαλύτερο χρονόσημο. Επιπρόσθετα, λόγω αυτών των λαθών που προκύπτουν στην τροφοδότηση παραθυρικών τελεστών, παράγονται διαφορετικά αποτελέσματα από τα αναμενόμενα, τόσο ως προς το περιεχόμενο όσο και ως προς την χρονική σήμανσή τους. Για παράδειγμα, ένας συναθροιστικός τελεστής που δέχεται μία πλειάδα μεταγενέστερου χρονοσήμου, μπορεί εσφαλμένα να θεωρήσει ότι έχει επεξεργαστεί όλα τα διαθέσιμα στοιχεία των πηγών μέχρι αυτό το χρονόσημο, παράγοντας εσφαλμένα αποτελέσματα. Η έλλειψη συγχρονισμού μεταξύ των εισερχόμενων ρευμάτων επηρεάζει τελεστές που εφαρμόζονται πάνω σε δύο ρεύματα δεδομένων, όπως π.χ. σε έναν τελεστή σύνδεσης που επιδιώκει την συσχέτιση χρονικά ομοίων στοιχείων από τις δύο εισόδους του. Καθώς τα δύο ρεύματα μπορεί να εξελίσσονται με διαφορετικούς ρυθμούς ή ένα από αυτά πιθανόν προσωρινά να μην παράγει στοιχεία, δημιουργείται πρόβλημα συσσώρευσης πλειάδων σε κάποια από τις εισόδους του, και καθυστέρησης στην εξαγωγή απαντήσεων.

Συνήθως στις εφαρμογές ρευμάτων δεδομένων, οι πηγές πληροφορίας είναι χωρικά κατανεμημένες και τα στοιχεία των ρευμάτων φθάνουν στο σύστημα μέσω ενσύρματου ή ασύρματου δικτύου. Σε ένα τέτοιο περιβάλλον οι κυριότεροι παράγοντες απώλειας της διάταξης ή του συγχρονισμού των στοιχείων των ρευμάτων **[SW04]** είναι οι εξής:

- Διαφορετικοί ρυθμοί παραγωγής στοιχείων από τις πηγές.
- Διαφορετικές καθυστερήσεις τις οποίες υπόκεινται τα στοιχεία των ρευμάτων λόγω π.χ. διαφορετικών διαδρομών στο δίκτυο.

- Απώλειες στοιχείων κατά την αποστολή τους στο σύστημα.
- Αποστολή στοιχείων μέσω διαύλου επικοινωνίας που δεν εγγυάται την διάταξη.
- Έλλειψη συγχρονισμού μεταξύ των ρολογιών των πηγών.

Η απλή λογική της απόρριψης πλειάδων με χρονόσημο μικρότερο από το μέγιστο που έχει ληφθεί για όλα τα ρεύματα, δεν μπορεί να θεωρηθεί αποδοτική, αφού επηρεάζουν αισθητά την ακρίβεια των απαντήσεων. Η έλλειψη διάταξης μπορεί να αντιμετωπιστεί μερικά, με την προσωρινή αποθήκευση των στοιχείων των ρευμάτων σε έναν κεντρικό *καταχωρητή* (*buffer*). Όταν ο καταχωρητής γεμίσει, θα επιτελεί ταξινόμηση των ληφθέντων στοιχείων, τροφοδοτώντας στην συνέχεια τα ερωτήματα με χρονικά διατεταγμένες πλειάδες. Ωστόσο, για υψηλούς ρυθμούς άφιξης στοιχείων και υψηλό βαθμό έλλειψης διάταξης, πιθανόν να μην υπάρχουν τα απαιτούμενα χρονικά περιθώρια για ταξινόμηση των στοιχείων πριν αυτά τροφοδοτήσουν τα ερωτήματα. Επιπλέον δεν επιλύεται το πρόβλημα του συγχρονισμού μεταξύ των ρευμάτων.

Στο **[SW04]** επεκτείνεται η παραπάνω λογική με την παραγωγή πλειάδων *παλμού* (*heartbeats*) για την διασφάλιση της διάταξης και την επίτευξη συγχρονισμού μεταξύ των ρευμάτων. Τα στοιχεία κάθε ρεύματος κρατούνται σε έναν κεντρικό *διαχειριστή εισόδου* (*input manager*) για χρονικό διάστημα που καθορίζεται από τα στοιχεία που έχουν ληφθεί και μετρήσεις που έγιναν για καθέναν από τους παράγοντες έλλειψης διάταξης και συγχρονισμού που προαναφέρθησαν. Ο *διαχειριστής εισόδου* αρχικά μαζεύει για κάθε ρεύμα πλειάδες κοινού χρονοσήμου τ_{old}. Με την έλευση πλειάδας νέου χρονοσήμου τ_{new}, χρησιμοποιούνται οι προαναφερθείσες μετρήσεις για να καθοριστεί το χρονικό διάστημα που χρειάζεται να περιμένει το σύστημα για καθυστερημένες πλειάδες χρονοσήμου τ_{old}. Μόλις παρέλθει αυτό το διάστημα θεωρείται ότι έφτασαν όλες οι πλειάδες με τ_{old} και το σύστημα εφεξής απορρίπτει τυχόν καθυστερημένες πλειάδες με τέτοιο χρονόσημο, πιθανόν ενημερώνοντας κάποια από τα στατιστικά του στοιχεία. Η διαδικασία αυτή συνεχίζεται μέχρι να ληφθούν όλα τα στοιχεία χρονοσήμου τ_{old} από όλα τα εισερχόμενα ρεύματα. Τότε ο *διαχειριστής εισόδου* προωθεί στα ερωτήματα τις πλειάδες χρονοσήμου τ_{old} μαζί με μία ειδική πλειάδα *παλμού* (*heartbeat*), η οποία δηλώνει το τέλος των στοιχείων αυτού του χρονοσήμου. Τυχόν απουσία νέων στοιχείων από κάποια πηγή, αντιμετωπίζεται με τον ορισμό ενός μέγιστου χρονικού διαστήματος, πέρα από το οποίο το σύστημα θεωρεί ότι η πηγή δεν παρήγαγε άλλα στοιχεία με το εκάστοτε αναμενόμενο χρονόσημο τ_{old}. Οι πλειάδες *παλμού* τυγχάνουν ειδικής μεταχείρισης από τους τελεστές μια και πρέπει πάντα να προωθούνται στην έξοδό τους για να τροφοδοτήσουν και τους επόμενους τελεστές στο επιλεγμένο προσχέδιο εκτέλεσης.

Μάλιστα, η μέθοδος των πλειάδων παλμού μπορεί να θεωρηθεί μία εναλλακτική πρόταση για την απεμπλοκή των προβληματικών τελεστών, αφού αυτοί τροφοδοτούνται σταδιακά με το σαφώς μικρότερο πλήθος των στοιχείων ενός συγκεκριμένου χρονοσήμου. Ωστόσο, οι απαιτήσεις μνήμης που προβάλει ιδίως ο *διαχειριστής εισόδου* παραμένουν απροσδιόριστες και απεριόριστες. Δημιουργείται επίσης πρόβλημα με την υπεισερχόμενη καθυστέρηση στην εξαγωγή απαντήσεων, η οποία μπορεί να μην είναι αποδεκτή για ορισμένα ερωτήματα.

Μία διαφορετική προσέγγιση στο πρόβλημα της διάταξης παρέχεται στο **[ACC+03]**, όπου γίνεται αρχικά διαχωρισμός μεταξύ *τελεστών που εξαρτώνται ή μη από την διάταξη* (*order-agnostic* και *order-sensitive operators*). Τελεστές αντίστοιχοι με την προβολή, την επιλογή, και την ένωση υπάγονται στην κατηγορία των τελεστών που δεν εξαρτώνται από την διάταξη. Αντίθετα, τελεστές όπως η ταξινόμηση, η συνάθροιση και η σύνδεση παρουσιάζουν ευαισθησία στην διάταξη των στοιχείων τα οποία επεξεργάζονται. Στους προβληματικούς αυτούς τελεστές παρέχονται κάποια περιθώρια έλλειψης διάταξης στα εισερχόμενα στοιχεία μέσω μίας παραμέτρου *αργοπορίας* (*slack*). Η παράμετρος αυτή καθορίζει το μέγιστο πλήθος διαδοχικών στοιχείων για τα οποία η έλλειψη διάταξης θα αντιμετωπιστεί επιτυχώς από τον τελεστή. Τα εξαγόμενα αποτελέσματα από αυτούς τους τελεστές παρουσιάζουν κατ' ελάχιστο τοπική διάταξη ίση με αυτήν που καθορίζει η παράμετρος *αργοπορίας*, χωρίς ωστόσο να παρέχονται εγγυήσεις για την συνολική διάταξη. Τέλος, για την μερική αντιμετώπιση του προβλήματος έλλειψης συγχρονισμού μεταξύ των ρευμάτων παρέχεται η υλοποίηση του τελεστή Resample με την οποία επιδιώκεται η χρονική ευθυγράμμιση των δύο ρευμάτων εισόδου του.

Σημειώνεται ότι στην παρουσίαση των παραθύρων και των τελεστών που προηγήθηκε, η διάταξη των στοιχείων σε κάθε ρεύμα θεωρήθηκε δεδομένη. Η υπόθεση αυτή εκφράστηκε στο

προηγούμενο κεφάλαιο (ενότητα 3.3) κατά τον ορισμό της αλγεβρικής αναπαράστασης που χρησιμοποιήθηκε για τα ρεύματα. Επίσης οι υλοποιημένοι τελεστές διατηρούν την διάταξη στα στοιχεία που παράγουν.

4.4 Πολιτικές χρονοδρομολόγησης τελεστών

Η σειρά με την οποία καλούνται οι τελεστές των διαφόρων ερωτημάτων διαρκείας και ο χρόνος επεξεργασίας που δίνεται σε αυτούς αποτελούν αντικείμενο της εφαρμοζόμενης *πολιτικής χρονοδρομολόγησης* (*scheduling policy*). Η ύπαρξη πολλαπλών ερωτημάτων με απαιτήσεις online απαντήσεων, οι διαρκείς διακυμάνσεις και οι πιθανές εξάρσεις στον ρυθμό άφιξης στοιχείων πληροφορίας αναδεικνύουν την σημασία αυτού του ζητήματος στις εφαρμογές ρευμάτων δεδομένων.

Ένα σημαντικό ζήτημα που τίθεται είναι εάν η χρονοδρομολόγηση πραγματοποιείται σε επίπεδο πλειάδων, σε επίπεδο τελεστών ή ακόμα και σε επίπεδο ερωτημάτων. Μάλιστα στην δεύτερη περίπτωση προκύπτει προβληματισμός για κάθε επιλεγμένο τελεστή, σχετικά με το αν δίνεται μέγιστος χρόνος επεξεργασίας ή μέγιστο πλήθος πλειάδων για επεξεργασία. Στην υλοποίηση του απλουστευμένου συστήματος που πραγματοποιήθηκε, ακολουθήθηκε χρονοδρομολόγηση σε επίπεδο ερωτημάτων την οποία μάλιστα αναλαμβάνει το λειτουργικό σύστημα.

Οι απλούστερες δυνατές πολιτικές χρονοδρομολόγησης που μπορούν να ακολουθηθούν προκύπτουν από την προσαρμογή των γνωστών αλγορίθμων Round–Robin και FIFO. Αναλυτικά:

- Round – Robin : Σε αυτήν την στρατηγική δίνεται κυκλικά σειρά επεξεργασίας σε ένα πλήθος ενεργών τελεστών για συγκεκριμένο χρονικό διάστημα. Κάθε τελεστής είτε εξαντλεί το διάστημα αυτό είτε επεξεργάζεται όλα τα στοιχεία του και στην συνέχεια δίνει την σειρά του στον επόμενο τελεστή. Χαρακτηρίζεται από δικαιοσύνη μεταξύ των τελεστών αλλά δεν παρουσιάζει καλές επιδόσεις σε εξάρσεις στον ρυθμό άφιξης στοιχείων, ενώ δεν λαμβάνει μέτρα για την ελαχιστοποίηση της χρησιμοποιούμενης μνήμης.
- FIFO : Εδώ κάθε πλειάδα οδηγείται προς επεξεργασία ανάλογα με την σειρά άφιξης της. Η επεξεργασία κάθε πλειάδας είναι πλήρης, υπό την έννοια ότι η επόμενη πλειάδα λαμβάνεται μόνο όταν παραχθούν τα αποτελέσματα των ερωτημάτων που αντιστοιχούν στην έλευση της προηγούμενης. Παρουσιάζει τα ίδια προβλήματα με την στρατηγική Round – Robin ενώ σε γενικές γραμμές εμφανίζει χαμηλότερες επιδόσεις.

Στις παραπάνω μεθόδους παραβλέπεται η έννοια της *επιλεκτικότητας* (*selectivity*) των τελεστών. Η παράμετρος επιλεκτικότητας για έναν τελεστή, δηλώνει τον μέσο αριθμό των παραγόμενων πλειάδων που προκύπτουν από την επεξεργασία ενός μόνο στοιχείου πληροφορίας από το ρεύμα δεδομένων. Έτσι ο τελεστής της *προβολής* παρουσιάζει επιλεκτικότητα 1 ενώ η *επιλογή* συνήθως έχει παράμετρο επιλεκτικότητας μικρότερο της μονάδας. Αντίθετα, τελεστές όπως η *σύνδεση*, μπορεί από την επεξεργασία μίας πλειάδας να παράγουν πολλές πλειάδες απάντησης, παρουσιάζοντας επιλεκτικότητα μεγαλύτερη της μονάδας. Η παράμετρος επιλεκτικότητας μπορεί να αποτελέσει κριτήριο για την ευνοϊκότερη χρονοδρομολόγηση τελεστών που παρουσιάζουν χαμηλή επιλεκτικότητα, αποσκοπώντας στην μείωση της διακινούμενης πληροφορίας στο σύστημα και κατά συνέπεια σε εξοικονόμηση διαθέσιμης μνήμης.

Στην συνέχεια παρέχεται συνοπτική περιγραφή του τρόπου με τον οποίο αντιμετωπίζεται το θέμα της χρονοδρομολόγησης, στα πρότυπα συστήματα Aurora, STREAM και TelegraphCQ.

4.4.1 Aurora : Train Scheduling

Στην προσέγγιση της ερευνητικής ομάδας του Aurora **[ACC+03]** τα ερωτήματα σχηματίζονται διασυνδέοντας *βέλη* και *κουτιά* που αντιστοιχούν σε ουρές πλειάδων και κουτιά τελεστών αντίστοιχα. Για κάθε ερώτημα μπορεί να προσδιοριστεί το πολύ ένα *υπέρ–κουτί* (*superbox*) το οποίο σχηματίζεται από την ομαδοποίηση ορισμένων τελεστών–κουτιών του ερωτήματος. Τα υπέρ–κουτιά μπορούν είτε να καθορίζονται στατικά είτε να διαμορφώνονται δυναμικά κατά την διάρκεια εκτέλεσης. Επίσης, για κάθε ερώτημα μπορούν διατηρούνται ένα ή περισσότερα κανονικοποιημένα *διαγράμματα ποιότητας υπηρεσίας* (***Q**uality **o**f **S**ervice – QoS*

graphs) για τους χρόνους απόκρισης, την ακρίβεια των αποτελεσμάτων και την βαρύτητα των εξαγόμενων απαντήσεων με στόχο την εξυπηρέτηση προτεραιότητας ερωτημάτων.

Ο χρονοδρομολογητής επιλέγει ένα από τα διαθέσιμα κουτιά (ή υπέρ–κουτιά) των ερωτημάτων καθώς και το πλήθος των πλειάδων που θα του δοθούν για επεξεργασία. Οι πλειάδες αυτές σχηματίζουν *συρμούς πλειάδων* (*tuple trains*) απ' όπου προέκυψε και η ονομασία αυτής της πολιτικής χρονοδρομολόγησης: *Train Scheduling* **[CCR+03]**. Η χρονοδρομολόγηση πολλαπλών ερωτημάτων πραγματοποιείται σε δύο επίπεδα. Αρχικά επιλέγεται το ερώτημα που θα εκτελεστεί ανάλογα με τα διαθέσιμα διαγράμματα ποιότητας υπηρεσίας. Στην συνέχεια, επιλέγεται ο τρόπος με τον οποίο θα εξυπηρετηθεί το ερώτημα αυτό αποφασίζοντας για την διαδοχή με την οποία θα κληθούν τα κουτιά (ή το υπέρ–κουτί) του ερωτήματος καθώς και οι *συρμοί πλειάδων* που θα σταλούν σε αυτά. Η διαδοχή με την οποία θα κληθούν τα κουτιά ενός υπέρ–κουτιού επιλέγεται, αφενός βάσει στατιστικών στοιχείων για το κόστος επεξεργασίας και την επιλεκτικότητα των κουτιών, αφετέρου βάσει του επιδιωκόμενου στόχου. Έτσι για κάθε ερώτημα μπορεί να επιδιώκεται :

- ελάχιστο κόστος, όπως αυτό προσδιορίζεται από το πλήθος κλήσεων κουτιών ανά παραγόμενη πλειάδα αποτελέσματος,
- ελάχιστη καθυστέρηση στην έναρξη παραγωγής αποτελεσμάτων και
- μέγιστη αποδέσμευση μνήμης στην μονάδα του χρόνου.

Το γεγονός ότι οι αποφάσεις του χρονοδρομολογητή βασίζονται στα διαγράμματα ποιότητας υπηρεσίας, παρέχει την ελευθερία στο σύστημα να προσαρμόσει την στρατηγική χρονοδρομολόγησης προς την αντιμετώπιση διαφόρων καταστάσεων που προσδιορίζονται από την παρακολούθηση αυτών των διαγραμμάτων. Επιπλέον, η δυνατότητα για ομαδοποίηση κουτιών και μαζική επεξεργασία πλειάδων επιτυγχάνει μείωση αφενός της επιπρόσθετης επιβάρυνσης χρονοδρομολόγησης, αφετέρου του μέσου χρόνου επεξεργασίας ανά πλειάδα των τελεστών. Τέλος, η εκμετάλλευση στατιστικών στοιχείων επιλεκτικότητας και κόστους επεξεργασίας των τελεστών, αποτελεί ένα ακόμη πλεονέκτημα αυτής της προσέγγισης.

4.4.2 STREAM : Chain Scheduling

Στην πρόταση για *χρονοδρομολόγηση αλυσίδας* (*chain scheduling*) **[BBDM03]**, **[MWA+03]** της ερευνητικής ομάδας του STREAM, αντιμετωπίζεται με αρκετά καλά αποτελέσματα το ζήτημα της μείωσης της απαιτούμενης μνήμης για την επεξεργασία των ερωτημάτων. Χρησιμοποιούνται στατιστικά στοιχεία για τους τελεστές σχετικά με τον μέσο χρόνο επεξεργασίας ανά πλειάδα και την επιλεκτικότητα τους. Τα στοιχεία αυτά είτε θεωρούνται γνωστά και σταθερά, είτε υπολογίζονται σε τακτά χρονικά διαστήματα από το σύστημα. Σε κάθε ερώτημα αντιστοιχεί ένα μοναδικό *μονοπάτι τελεστών* (*operator path*) το οποίο θα ακολουθήσουν οι πλειάδες που θα επεξεργαστεί αυτό το ερώτημα. Ακολούθως, για κάθε ερώτημα χαράσσεται το *διάγραμμα προόδου* (*progress chart*) του, ενσωματώνοντας τόσο την πληροφορία του επιλεγμένου μονοπατιού όσο και των στατιστικών στοιχείων που προαναφέρθηκαν. Ο χρονοδρομολογητής ομαδοποιεί μη επικαλυπτόμενα τμήματα από τα μονοπάτια τελεστών σε αλυσίδες διαδοχικών τελεστών. Στην συνέχεια, επιλέγει τόσο την επόμενη αλυσίδα τελεστών στην οποία θα δοθεί χρόνος επεξεργασίας όσο και την διάρκεια του χρόνου αυτού βασιζόμενος στα διαθέσιμα διαγράμματα προόδου και το πλήθος των στοιχείων προς επεξεργασία που διαθέτει κάθε τελεστής.

Ένα σημαντικό πλεονέκτημα αυτής της πρότασης είναι η χαμηλή *επιβάρυνση χρονοδρομολόγησης* (*scheduling overhead*) μιας και τα *διαγράμματα προόδου* είτε θεωρούνται στατικά, είτε ανανεώνονται σε τακτά αλλά αραιά χρονικά διαστήματα. Επίσης, με την βοήθεια των διαγραμμάτων προόδου παρέχεται ευνοϊκότερη μεταχείριση σε τελεστές χαμηλής επιλεκτικότητας, επιτυγχάνοντας την μείωση των απαιτήσεων διαθέσιμής μνήμης. Ωστόσο, δεν αντιμετωπίζονται αποδοτικά τα προβλήματα που δημιουργούνται λόγω εξάρσεων στον ρυθμό άφιξης στοιχείων στο σύστημα, οδηγώντας είτε σε άνιση μεταχείριση μεταξύ των τελεστών είτε σε μεγάλες χρονικές καθυστερήσεις των απαντήσεων. Επίσης, δεν έχουν αναπτυχθεί μέθοδοι για την κοινή χρήση μνήμης και επεξεργασίας από ερωτήματα που εμφανίζουν ομοιότητες στα *μονοπάτια τελεστών* τους. Στο **[MWA+03]** ωστόσο, εκφράζονται προβληματισμοί για την εφαρμογή αυτής της

βελτιστοποίησης. Μέχρι στιγμής παραμένει ακόμη ανοιχτό το ζήτημα της ενσωμάτωσης ερωτημάτων με βάρη στα κριτήρια για τις αποφάσεις του χρονοδρομολογητή.

4.4.3 TelegraphCQ : Eddies

Στο TelegraphCQ **[AH00, CCD+03, siteTCQ]** αποφεύγεται η λογική της κατασκευής στατικών προσχεδίων εκτέλεσης που ακολουθούνται από όλες τις πλειάδες στις οποίες αναφέρεται το εκάστοτε ερώτημα. Έχοντας σαν κύριο στόχο την προσαρμοστικότητα στις μεταβαλλόμενες συνθήκες, χρησιμοποιούνται οι οντότητες Eddies για την δρομολόγηση κάθε πλειάδας στα παρεχόμενα *υποτμήματα* (*modules*) τελεστών. Σε κάθε Eddie συνδέονται οι είσοδοι και οι έξοδοι ενός πλήθους άλλων οντοτήτων, ακόμα και άλλων οντοτήτων Eddie. Κάθε πλειάδα που εισέρχεται σε έναν Eddie, δρομολογείται με πιθανόν διαφορετικό τρόπο από τις υπόλοιπες πλειάδες του ιδίου ρεύματος, και φεύγει από αυτόν όταν υποβληθεί στην επεξεργασία όλων των συνδεόμενων οντοτήτων του.

Για κάθε πλειάδα πρέπει να παρέχεται κατ' ελάχιστο, η πληροφορία των υποτμημάτων που έχει επισκεφθεί. Επιπλέον, ανάλογα με την επιθυμητή πολιτική χρονοδρομολόγησης, προκύπτει η ανάγκη διατήρησης περαιτέρω στοιχείων είτε για κάθε πλειάδα ξεχωριστά είτε για ένα σύνολο πλειάδων. Η απόφαση για την δρομολόγηση των πλειάδων στην είσοδο ενός Eddie βασίζεται κυρίως στις προτεραιότητες που τυχόν έχουν αποδοθεί στις πλειάδες και στην διαρκή παρακολούθηση του ρυθμού με τον οποίο τα υποτμήματα των τελεστών καταναλώνουν και παράγουν πλειάδες. Έτσι ευνοούνται κατά την χρονοδρομολόγηση, οι πλειάδες υψηλής προτεραιότητας και οι γρήγοροι και επιλεκτικοί τελεστές. Κατ' αυτόν τον τρόπο πραγματοποιείται δρομολόγηση τόσο σε επίπεδο πλειάδων όσο και σε επίπεδο τελεστών.

Η παρεχόμενη προσαρμοστικότητα αυτής της πολιτικής χρονοδρομολόγησης είναι αρκετά ικανοποιητική. Επίσης, θετικό είναι το γεγονός ότι δεν απαιτούνται εκτιμήσεις επιλεκτικότητας ή χρόνου επεξεργασίας των τελεστών, μια και αυτά τα μεγέθη προσδιορίζονται συνεχώς παρακολουθώντας τους ρυθμούς κατανάλωσης και παραγωγής στοιχείων από τους τελεστές. Ωστόσο σημαντικό μειονέκτημα αυτής της προσέγγισης, αποτελεί η υψηλή επιβάρυνση των αποφάσεων χρονοδρομολόγησης. Για την αντιμετώπιση αυτού του προβλήματος μελετώνται τεχνικές μείωσης του μέσου χρόνου επεξεργασίας ανά πλειάδα των τελεστών μέσω αλλαγών στο πλήθος και στην διάταξη των τελεστών που δρομολογούνται με κάθε απόφαση χρονοδρομολόγησης. Προς τον ίδιο στόχο προσανατολίζονται και οι προσπάθειες για μαζική επεξεργασία πλειάδων από τα υποτμήματα τελεστών.

4.5 Προσδιορισμός κόστους σε ερωτήματα διαρκείας

Ο προσδιορισμός του κόστους των ερωτημάτων διαρκείας παρουσιάζει σημαντικές διαφοροποιήσεις από τις μεθόδους που ακολουθούνται στα κλασικά Συστήματα Διαχείρισης Βάσεων Δεδομένων. Στις μέχρι τώρα προσεγγίσεις το κόστος υπολογιζόταν ως το πλήθος των απαιτούμενων αναγνώσεων / εγγραφών (I/O) από ή προς τον σκληρό δίσκο κάνοντας συμψηφισμό στατιστικών στοιχείων των στατικά αποθηκευμένων σχέσεων και του κόστους επεξεργασίας κάθε τελεστή. Βάσει αυτών των υπολογισμών γινόταν η επιλογή του βέλτιστου προσχεδίου εκτέλεσης.

Στα Συστήματα Διαχείρισης Ρευμάτων Δεδομένων αλλάζει τόσο ο τρόπος υπολογισμού και οι χρησιμοποιούμενες μετρικές όσο και η πολυπλοκότητα της εύρεσης βέλτιστων προσχεδίων εκτέλεσης. Πλέον το κόστος δεν μετράται σε πλήθος I /O του σκληρού δίσκου μια και η χρήση του αποφεύγεται. Μερικές από τις πιθανές μετρικές που μπορούν να χρησιμοποιηθούν για τον υπολογισμό του κόστους είναι:

- Το επίπεδο ακρίβειας των απαντήσεων. Η επιρροή των συνόψεων, της απόρριψης φορτίου και της επιβολής παραθύρων ως προσεγγιστική μέθοδος αναμένεται να αποτελέσουν τους κύριους ρυθμιστές αυτής της παραμέτρου.
- Ο ρυθμός παραγωγής πλειάδων στην απάντηση (λ.χ. εξαγόμενες πλειάδες ανά δευτερόλεπτο). Μετρώντας τους ρυθμούς άφιξης πλειάδων και τους ρυθμούς παραγωγής αποτελεσμάτων από

τους τελεστές είναι εφικτή η αναδιάταξη τους με στόχο την βελτιστοποίηση του ρυθμού παραγωγής απαντήσεων.
- Ο ρυθμός παραγωγής σημαντικών πλειάδων όπως αυτές καθορίζονται από την απόδοση βαρών στα ενεργά ερωτήματα.
- Το ποσοστό απασχόλησης των διαθέσιμων πόρων του συστήματος.

Τέλος πρέπει επίσης να σημειωθεί ότι ο υπολογισμός του κόστους δεν μπορεί να υποτεθεί στατικός, μια και οι εφαρμογές ρευμάτων δεδομένων χαρακτηρίζονται από την δυναμική και ρευστή φύση τους. Έτσι χρειάζεται επαναπροσδιορισμός του κόστους κάθε ερωτήματος όταν ενεργοποιούνται νέα ερωτήματα, όταν παρατηρούνται έντονες διαταραχές στον ρυθμό άφιξης των στοιχείων των ρευμάτων ή όποτε αυξάνει ο βαθμός απασχόλησης των πόρων του συστήματος. Για τον λόγο αυτό, οι προτεινόμενες τεχνικές βελτιστοποίησης εστιάζουν περισσότερο στην εξοικονόμηση διαθέσιμων πόρων παρά στην εύρεση κάποιου βέλτιστου προσχεδίου εκτέλεσης.

4.6 Τεχνικές βελτιστοποίησης της εκτέλεσης των ερωτημάτων

Πέρα από τα ζητήματα χρονοδρομολόγησης, διάταξης και συγχρονισμού των στοιχείων, η επεξεργασία των ερωτημάτων παρουσιάζει περιθώρια βελτιστοποίησης σε θέματα που αφορούν:
- την λειτουργία προβληματικών τελεστών (λ.χ. συνάθροιση και σύνδεση),
- την αξιοποίηση οποιασδήποτε γνώσης για τα περιεχόμενα των ρευμάτων,
- την διαδικασία διαγραφών και ενημερώσεων από το ρεύμα της απάντησης.

Στις υποενότητες που ακολουθούν παρουσιάζονται οι προτάσεις που επικρατούν στην τρέχουσα βιβλιογραφία σχετικά με τα παραπάνω ζητήματα.

4.6.1 Πολλαπλές παραθυρικές συνδέσεις πάνω σε κοινά ρεύματα δεδομένων

Οι τελεστές παραθυρικής σύνδεσης θεωρούνται ιδιαίτερα «ακριβοί» τελεστές τόσο ως προς τον απαιτούμενο χρόνο επεξεργασίας ανά πλειάδα, όσο και ως προς τις απαιτήσεις μνήμης που εμφανίζουν. Αυτό οφείλεται αφενός στο πλήθος των ζευγών που πρέπει να ελεγχθούν ως προς την συνθήκη σύνδεσης για κάθε πλειάδα, αφετέρου στο μέγεθος της πληροφορίας που πρέπει να τηρείται για τις εισερχόμενες πλειάδες. Συνήθως χρησιμοποιούνται κυλιόμενα παράθυρα για τα ρεύματα εισόδου, ενώ συχνή είναι και η χρήση στατικής σχέσης για μία από τις δύο εισόδους.

Ιδιαίτερο ενδιαφέρον εντοπίζεται στις δυνατότητες βελτιστοποίησης του υπολογισμού πολλαπλών παραθυρικών συνδέσεων πάνω σε κοινά ρεύματα δεδομένων με κοινή *συνθήκη σύνδεσης* (*join predicate*) και πιθανόν διαφορετικές παραμέτρους παραθύρων. Η περίπτωση αυτή δεν θεωρείται καθόλου σπάνια, αλλά η απλοϊκή λύση του υπολογισμού κάθε παραθυρικού τελεστή σύνδεσης ανεξάρτητα από τους υπόλοιπους παρουσιάζει αδιαμφισβήτητα υψηλό κόστος. Το ερέθισμα για την προσπάθεια αποδοτικότερης αντιμετώπισης τέτοιων καταστάσεων, δίνεται από την παρατήρηση ότι τυχόν επικαλύψεις μεταξύ των παραθύρων τροφοδότησης αυτών των τελεστών, οδηγούν σε αντίστοιχες επικαλύψεις μεταξύ των απαντήσεών τους.

Στην στρατηγική που ακολουθείται από τους **[HFAE03]** δημιουργείται ένα κοινό προσχέδιο εκτέλεσης για την εξυπηρέτηση όλων των παρόμοιων παραθυρικών τελεστών σύνδεσης. Μάλιστα, καθένας από αυτούς εφαρμόζει όμοια κυλιόμενα χρονικά παράθυρα στα δύο ρεύματα εισόδου του. Στον προτεινόμενο μηχανισμό, διακρίνονται το *τμήμα σύνδεσης* (*join part*) και το *τμήμα δρομολόγησης* (*routing part*). Αρχικά, οδηγούνται στο *τμήμα σύνδεσης* όλες οι πλειάδες που εμπίπτουν στην εμβέλεια των παραθύρων τροφοδότησης όλων των τελεστών, απαλείφοντας τις επαναλαμβανόμενες πλειάδες από τυχόν επικαλυπτόμενα παράθυρα. Το *τμήμα σύνδεσης* δέχεται αυτές τις πλειάδες και πραγματοποιεί μαζικά την λειτουργία σύνδεσης για όλους τους αντίστοιχους τελεστές. Στην συνέχεια, το *τμήμα δρομολόγησης* αναλαμβάνει να τροφοδοτήσει τις εξόδους των διαφορετικών τελεστών με τα αποτελέσματα που τους αντιστοιχούν, από το υπερσύνολο των αποτελεσμάτων που παράγονται στο *τμήμα σύνδεσης*. Κατά τον τρόπο αυτό επιτυγχάνεται σημαντική εξοικονόμηση μνήμης και επεξεργαστικού φόρτου: αφενός δεν διατηρούνται χωριστές

ουρές εισόδου για τους διαφορετικούς τελεστές σύνδεσης, αφετέρου κάθε πλειάδα οδηγείται μόνο μία φορά στο *τμήμα σύνδεσης*. Προκύπτει όμως θέμα επιλογής της μέθοδο που θα ακολουθηθεί από το *τμήμα σύνδεσης* για την επεξεργασία των στοιχείων που οδηγούνται σε αυτό.

Η απλούστερη μέθοδος αντιστοιχεί στην επεξεργασία των στοιχείων του μεγαλύτερου παραθύρου (**L**argest **W**indow **O**nly – **LWO**) το οποίο άλλωστε περιλαμβάνει τα περιεχόμενα των παραθύρων των υπολοίπων τελεστών. Η μέθοδος (**S**mallest **W**indow **F**irst – **SWF**) προβλέπει την απόδοση προτεραιότητας στην εξυπηρέτηση των μικρότερων παραθύρων, με στόχο την ανταπόκριση στον γρηγορότερο ρυθμό ανανέωσης τους, συγκριτικά με τα υπόλοιπα παράθυρα. Τέλος προτείνεται μία μέθοδος όπου πρωτίστως εξυπηρετούνται οι πλειάδες για τις οποίες υπολογίζεται ότι θα εξυπηρετήσουν περισσότερα ερωτήματα (**M**aximum **Q**uery **T**hroughput – **MQT**). Από την συγκριτική αξιολόγηση των τεχνικών, προκύπτει ανωτερότητα της μεθόδου MQT ακολουθούμενη από την SWF με χειρότερη την LWO.

Τέλος αξίζει να σημειωθεί, ότι για να δημιουργηθούν οι κατάλληλες συνθήκες ύπαρξης παρόμοιων τελεστών σύνδεσης θα πρέπει αυτοί να τοποθετούνται πριν από τους υπόλοιπους τελεστές του επιλεγμένου προσχεδίου εκτέλεσης κάθε ερωτήματος. Έτσι σε αντίθεση με τις κλασσικές μεθόδους βελτιστοποίησης των τελεστών σύνδεσης, οι τελεστές επιλογής και προβολής πρέπει να τοποθετούνται μετά από το τμήμα δρομολόγησης του προτεινόμενου μηχανισμού.

4.6.2 Panes για την βελτιστοποίηση συναθροιστικών τελεστών

Η χρήση χρονικά κυλιόμενων παραθύρων αποτελεί την πιο συνήθη επιλογή για την τροφοδότηση των συναθροιστικών τελεστών. Ο απλούστερος τρόπος υλοποίησης τέτοιων τελεστών συνίσταται στον διαρκή επαναπροσδιορισμό των απαντήσεων τους, λαμβάνοντας κάθε φορά υπόψη όλες τις πλειάδες που εμπίπτουν μέσα στην εμβέλεια του παραθύρου. Στην μέθοδο αυτή διακρίνονται δύο κυρίως προβλήματα:

- *Απεριόριστες απαιτήσεις μνήμης*. Κάθε χρονική στιγμή απαιτούνται για την απάντηση, όλες οι πλειάδες με χρονόσημο εντός της εμβέλειας του παραθύρου – πλήθος το οποίο εν γένει είναι άγνωστο και απεριόριστο.
- *Υψηλό κόστος επεξεργασίας*. Κάθε πλειάδα χρησιμοποιείται περισσότερες από μία φορές στις διαδοχικές χρονικές στιγμές απάντησης του ερωτήματος, ανάλογα με την *έκταση* (*range*) και την *κύλιση* (*slide*) του παραθύρου. Αυτό συμβαίνει λόγω της επικάλυψης των περιεχομένων του παραθύρου καθώς αυτό κυλά στον χρόνο, ενώ για την απάντηση πρέπει να χρησιμοποιούνται όλα τα περιεχόμενα του παραθύρου. Έτσι, καθώς αυξάνει ο λόγος έκταση/κύλιση, αυξάνει παράλληλα και ο αριθμός των φορών που χρησιμοποιείται κάθε πλειάδα. Μόνο στην περίπτωση χρήσης επάλληλων παράθυρων δεν υφίσταται πρόβλημα επικαλύψεων.

Εναλλακτική πρόταση υπολογισμού των συναθροιστικών ερωτημάτων στα κυλιόμενα παράθυρα με στόχο την αντιμετώπιση αυτών των προβλημάτων, είναι η μέθοδος που προτείνεται στο **[LMT+05]**. Στην μέθοδο αυτή ο υπολογισμός της απάντησης επιτυγχάνεται σε δύο βήματα, με τον υπολογισμό δύο υποερωτημάτων.

Στο πρώτο βήμα εκτελείται ένα υποερώτημα όμοιο με το αρχικό, αλλά με παραμέτρους *έκτασης* και *κύλισης* ίσες μεταξύ τους και διαφορετικές από το αρχικό. Έτσι, το αρχικό παράθυρο διασπάται σταδιακά σε μη επικαλυπτόμενα τμήματα, για τα οποία υπολογίζεται το μερικό συναθροιστικό αποτέλεσμα του πρωταρχικού ερωτήματος για κάθε στοιχείο ομαδοποίησης. Αυτά τα μερικά αποτελέσματα διατηρούνται με την μορφή πλειάδων (τα επονομαζόμενα *panes*). Με αυτήν την μέθοδο, ο τελεστής επεξεργάζεται κάθε πλειάδα εισόδου του μία και μόνο φορά για την παραγωγή του αντίστοιχου μερικού αποτελέσματος. Επίσης επιτυγχάνεται συμπύκνωση της πληροφορίας των αρχικών δεδομένων σε *panes*, από την οποία επωφελείται το δεύτερο υποερώτημα καθώς το πλήθος των παραγόμενων ενδιάμεσων πλειάδων είναι συνήθως σημαντικά μικρότερο.

Στο δεύτερο βήμα υπολογίζεται η τελική απάντηση εκτελώντας ένα υποερώτημα πανομοιότυπο με το πρωταρχικό του συναθροιστικού τελεστή, και μάλιστα με τις ίδιες παραμέτρους κυλιόμενου παραθύρου. Η μοναδική διαφοροποίηση είναι ότι αυτό το ερώτημα δεν αναφέρεται στα πρωταρχικές πλειάδες εισόδου του τελεστή, αλλά στις ενδιάμεσες πλειάδες των *panes* που παρήχθησαν στο προηγούμενο βήμα. Καθώς το παράθυρο κυλά στον χρόνο, εξακολουθεί να

παρουσιάζεται επικάλυψη σε επίπεδο όμως ενδιάμεσων πλειάδων. Στις περισσότερες περιπτώσεις αυτό συνεπάγεται σημαντικά μικρότερο κόστος επεξεργασίας, μια και συνήθως αναφέρεται σε σαφώς μικρότερο και οπωσδήποτε πεπερασμένο πλήθος πλειάδων. Ο υπολογισμός αυτού του υποερωτήματος επιδέχεται σε ορισμένες περιπτώσεις και περαιτέρω βελτιστοποίηση. Συγκεκριμένα, η ενημέρωση των αποτελεσμάτων μπορεί να πραγματοποιηθεί χρησιμοποιώντας τα τελευταία αποτελέσματα του συνολικού ερωτήματος και έναν περιορισμένο αριθμό ενδιάμεσων πλειάδων από το πρώτο υποερώτημα, αντί για όλες όσες εμπίπτουν στην εμβέλεια του παραθύρου.

Όπως φάνηκε και παραπάνω, κάθε πρωταρχική πλειάδα εισόδου του τελεστή οδηγείται μόνο μία φορά προς επεξεργασία, ενώ αποδεσμεύεται μόλις υπολογιστεί το *pane* στο οποίο μετέχει. Κατά τον τρόπο αυτό επιτυγχάνεται σημαντική αύξηση στον ρυθμό κατανάλωσης των στοιχείων που οδηγούνται στον συναθροιστικό τελεστή. Το χαρακτηριστικό αυτό πλεονέκτημα οδηγεί στην μείωση του απαιτούμενου χώρου αποθήκευσης, σε κλίμακα η οποία προσδιορίζεται από τις παραμέτρους του ερωτήματος και τον ρυθμό άφιξης των δεδομένων. Ένα επίσης σημαντικό πλεονέκτημα αυτής της μεθόδου, είναι ότι δεν χρησιμοποιεί νέους τελεστές για την παραγωγή των ενδιάμεσων ή των τελικών αποτελεσμάτων. Αντίθετα, χρησιμοποιεί παρόμοια συναθροιστικά ερωτήματα με το πρωταρχικό με κατάλληλες τροποποιήσεις είτε στις παραμέτρους τους είτε στα δεδομένα εισόδου. Ωστόσο, υπάρχουν κάποιες συνθήκες κάτω από τις οποίες η μέθοδος αυτή δεν επιφέρει τις επιδιωκόμενες βελτιώσεις όπως όταν :

- αυξάνεται το πλήθος των στοιχείων ομαδοποίησης που αντιστοιχεί σε κάθε *pane*,
- μειώνεται ο μέσος αριθμός των πλειάδων που αντιστοιχούν σε κάθε *pane* κάτι το οποίο αντιστοιχεί σε χαμηλό ρυθμό εισόδου των δεδομένων στο σύστημα,
- μειώνεται το πλήθος των *panes* που εμπίπτουν μέσα στην εμβέλεια του παραθύρου.

4.6.3 Προσθήκη στίξεων στα ρεύματα δεδομένων

Ο μηχανισμός των *στίξεων* (*punctuations*) αποτελεί μέθοδο έκφρασης και αξιοποίησης οποιασδήποτε διαθέσιμης πληροφορίας για το περιεχόμενο των ρευμάτων. Εμφανίζονται με την μορφή εμβόλιμων πλειάδων και εκφράζουν συνθήκες οι οποίες επικρατούν σε όλα τα επόμενα στοιχεία του ρεύματος, όπως για παράδειγμα ότι το χρονόσημο των επόμενων πλειάδων θα είναι μεγαλύτερο από το φέρον χρονόσημο της πλειάδας στίξης. Ωστόσο επισημαίνεται ότι η χρήση τους δεν περιορίζεται σε συνθήκες χρονοσήμων. Αντίθετα, μπορούν να εκφράζονται συνθήκες πάνω σε οποιοδήποτε γνώρισμα των πλειάδων του ρεύματος, υποδιαιρώντας το εκάστοτε ρεύμα σε μικρότερα υπορεύματα. Οι πλειάδες στίξης δηλώνουν το τέλος ενός τέτοιου υπορεύματος και παράγονται κατά κύριο λόγο από τις πηγές των ρευμάτων. Επιβάλλεται επίσης να τυγχάνουν ειδικής μεταχείρισης από τους τελεστές. Κάθε τελεστής που επεξεργάζεται μία πλειάδα στίξης πρέπει να παράγει στην έξοδο του μία αντίστοιχη πλειάδα με στόχο την ενημέρωση και των υπόλοιπων τελεστών για την μεταφερόμενη πληροφορία.

Το σύστημα μπορεί να εκμεταλλευθεί τις λαμβανόμενες στίξεις για τους εξής σκοπούς:

- Αποκατάσταση της χρονικής διάταξης των στοιχείων.
- Απεμπλοκή ανασταλτικών τελεστών όπως οι συναθροιστικοί τελεστές.
- Ασφαλής αποδέσμευση μνήμης που χρησιμοποιείται από *τελεστές διατήρησης κατάστασης*.

Πλειάδες στίξης που φέρουν πληροφορία σχετική με το χρονόσημο των επόμενων πλειάδων του ρεύματος, μπορούν να χρησιμοποιηθούν για την διασφάλιση της διάταξης των στοιχείων κάθε ρεύματος. Οι εισερχόμενες πλειάδες κάθε ρεύματος μπορούν να αποθηκεύονται σε κάποιον προσωρινό καταχωρητή, μέχρι να ληφθεί κάποια πλειάδα στίξης. Τότε όλες οι πλειάδες αυτού του ρεύματος με χρονόσημο ίσο με αυτό της πλειάδας στίξης, δίνονται στα ερωτήματα προς επεξεργασία. Τυχόν πλειάδες που για οποιοδήποτε λόγο φθάνουν μετά την αντίστοιχη πλειάδα στίξης απορρίπτονται από το σύστημα. Ωστόσο με αυτόν τον τρόπο δεν επιλύεται το πρόβλημα συγχρονισμού των ρευμάτων, ενώ δημιουργείται έντονο πρόβλημα σε περίπτωση απώλειας κάποιας πλειάδας στίξης από το δίκτυο μεταφοράς.

Στην προσέγγιση που γίνεται στο **[LMP+05]** χρησιμοποιείται ως υπόβαθρο η μελέτη των **[MLT+05]** για την υλοποίηση παραθυρικών τελεστών, για να επιδειχθεί ο τρόπος με τον οποίο ο μηχανισμός στίξεων μπορεί να χρησιμοποιηθεί για την απεμπλοκή παραθυρικών συναθροιστικών

τελεστών. Συγκεκριμένα, προτείνεται ένας μηχανισμός ο οποίος, εκμεταλλευόμενος την παρουσία των στίξεων, κατορθώνει την μείωση τόσο του απαιτούμενου χώρου προσωρινής αποθήκευσης, όσο και του απαιτούμενου χρόνου επεξεργασίας. Μάλιστα, ως περαιτέρω μέθοδος βελτιστοποίησης προτείνεται η ενσωμάτωση των *panes* που παρουσιάστηκαν στην προηγούμενη υποενότητα.

Επίσης στο **[MLT+05]** γίνεται λόγος για την χρήση των πλειάδων στίξης ως μέσο ασφαλούς αποδέσμευσης μέρους της μνήμης που χρησιμοποιείται από τελεστές διατήρησης κατάστασης. Τόσο το επίπεδο ασφάλειας αυτής της λειτουργίας όσο και η ακρίβεια των απαντήσεων στα ερωτήματα, εξαρτώνται από την ακρίβεια της πληροφορίας που αναπαριστούν οι στίξεις και την ορθότητα της διάταξης με την οποία έφτασαν στο σύστημα. Μερικά από τα ζητήματα που δημιουργούνται για τον μηχανισμό των πλειάδων στίξης είναι τα εξής:

- Σε ποιο βαθμό η χρήση συγκεκριμένων στίξεων εξυπηρετεί την εκτέλεση των ερωτημάτων;
- Μπορεί να βρεθεί ένας αποδοτικός τρόπος για την εύρεση της κατάλληλης μορφής στίξεων;
- Είναι εφικτή ή έστω σκόπιμη η κατασκευή στίξεων για την περιγραφή της πληροφορίας που φέρουν οι μελλοντικές πλειάδες (*forward-looking punctuation*) ;

Τέλος επισημαίνεται, ότι οι πλειάδες *παλμών* (*heartbeats*) **[SW04]** μπορούν να θεωρηθούν εξειδίκευση των πλειάδων στίξεως, αφού ουσιαστικά αποτελούν στίξεις με πληροφορία χρονοσήμου. Ωστόσο, στον μηχανισμό παλμών καταβάλλεται περισσότερη προσπάθεια για την σωστή και έγκυρη παραγωγή αυτών των εξειδικευμένων στίξεων, σε αντίθεση με την παρούσα προσέγγιση όπου το ενδιαφέρον εντοπίζεται στην αξιοποίηση των γενικευμένων στίξεων που κατά κύριο λόγο παρέχουν οι πηγές. Τέλος, στο **[JMSS05]** προτείνεται ένας μηχανισμός όπου κατά κάποιο τρόπο πραγματοποιείται συγχώνευση των δύο εννοιών.

Στην υλοποίηση του συστήματος που παρουσιάζεται στο επόμενο κεφάλαιο έγινε χρήση των πλειάδων στίξης με στόχο την ασφαλή διαγραφή στοιχείων από τις ουρές των τελεστών.

4.6.4 Αρνητικές πλειάδες στην επεξεργασία ερωτημάτων διαρκείας

Οι απαντήσεις ερωτημάτων διαρκείας που διατυπώνονται πάνω σε παράθυρα μεταβλητών άκρων (π.χ. κυλιόμενα παράθυρα), δεν μπορούν να παραχθούν εύκολα στην μορφή σωρευτικού ρεύματος δεδομένων. Ο λόγος είναι ότι προκύπτει ανάγκη διαγραφής κάποιων πλειάδων από το ρεύμα εξόδου, αφού οι πρωτογενείς πλειάδες από τις οποίες προήλθαν καθίστανται σταδιακά παρωχημένες καθώς εκπίπτουν από την εμβέλεια των εφαρμοζόμενων παραθύρων. Επιπλέον, με την κλασική προσέγγιση ενεργοποίησης των τελεστών, μόνο όταν υπάρχουν στοιχεία στις εισόδους τους (Input Triggered), παρατηρείται αδυναμία στην παραγωγή έγκυρων και έγκαιρων απαντήσεων κατά τα διαστήματα απουσίας εισερχόμενων στοιχείων. Έτσι, είναι δυνατό κάποια από τις πλειάδες εξόδου του τελεστή να εκπίπτει από την χρονική εμβέλεια που ορίζει το ερώτημα αλλά να παραμένει στο σύνολο της απάντησης λόγω απουσίας πλειάδων εισόδου στον τελεστή που θα ενεργοποιούσαν την διαγραφή της. Σε γενικές γραμμές, αυτό το πρόβλημα μειώνεται με την αύξηση του ρυθμού εισερχόμενων πλειάδων.

Στο **[GHM+05]** προτείνεται η έννοια των *αρνητικών πλειάδων* (*negative tuples*) για την έγκαιρη σήμανση κάθε ακυρωμένης πλειάδας του ρεύματος εξόδου, επιτυγχάνοντας τον *προοδευτικό* (*incremental*) υπολογισμό τέτοιων ερωτημάτων. Κάθε νέα πλειάδα στα περιεχόμενα ενός παραθύρου ή στο ρεύμα εξόδου ενός τελεστή σημειώνεται με θετικό πρόσημο. Οι αρνητικές πλειάδες παράγονται είτε από τα παράθυρα δηλώνοντας την απομάκρυνση μιας πλειάδας από τα περιεχόμενά τους, είτε από τους τελεστές για την ακύρωση κάποιας θετικής πλειάδας από την έξοδο τους. Μάλιστα, στα κυλιόμενα παράθυρα μπορεί να προβλεφθεί η χρονική στιγμή παραγωγής κάθε αρνητικής πλειάδας, προσθέτοντας στο χρονόσημο της αντίστοιχης θετικής την εμβέλεια του παραθύρου. Οι τελεστές προβολής και επιλογής επεξεργάζονται κατά τον ίδιο τρόπο όλες τις πλειάδες, είτε θετικές είτε αρνητικές. Ωστόσο για τους τελεστές διατήρησης κατάστασης και τους ανασταλτικούς τελεστές, η έλευση αρνητικής πλειάδας αντιμετωπίζεται με ξεχωριστό τρόπο για καθένα από αυτούς. Ειδικά για τον «ακριβό» τελεστή σύνδεσης προτείνεται μία βελτιστοποίηση χάρη στην οποία επιτυγχάνεται πολύ γρήγορη επεξεργασία των αρνητικών πλειάδων, μέσω της διατήρησης κάποιας επιπρόσθετης πληροφορίας.

Το μειονέκτημα από την εφαρμογή της μεθόδου των αρνητικών πλειάδων εντοπίζεται στον διπλασιασμό της διακινούμενων πλειάδων στο σύστημα, αφού σε κάθε εισερχόμενη πλειάδα αντιστοιχεί μία θετική και μία αρνητική. Για το λόγο αυτό προτείνεται μία μέθοδος για την διαρκή εναλλαγή μεταξύ της τεχνικής των αρνητικών πλειάδων και της κλασικής τεχνικής ενεργοποίησης των τελεστών κατά την ανίχνευση πλειάδων στην είσοδο τους. Έτσι, σε συνθήκες ιδιαίτερα υψηλού φόρτου επιλέγεται η κλασική μέθοδος αποφεύγοντας την χρήση αρνητικών πλειάδων. Αντίθετα, όταν το σύστημα μπορεί να ανταποκριθεί στην επιβάρυνση των αρνητικών πλειάδων ή όταν η εγκυρότητα των απαντήσεων είναι ιδιαίτερα κρίσιμη, μπορεί να υιοθετηθεί η τεχνική των αρνητικών πλειάδων.

Κεφάλαιο 5

Υλοποίηση

5.1 Στοιχεία υλοποίησης

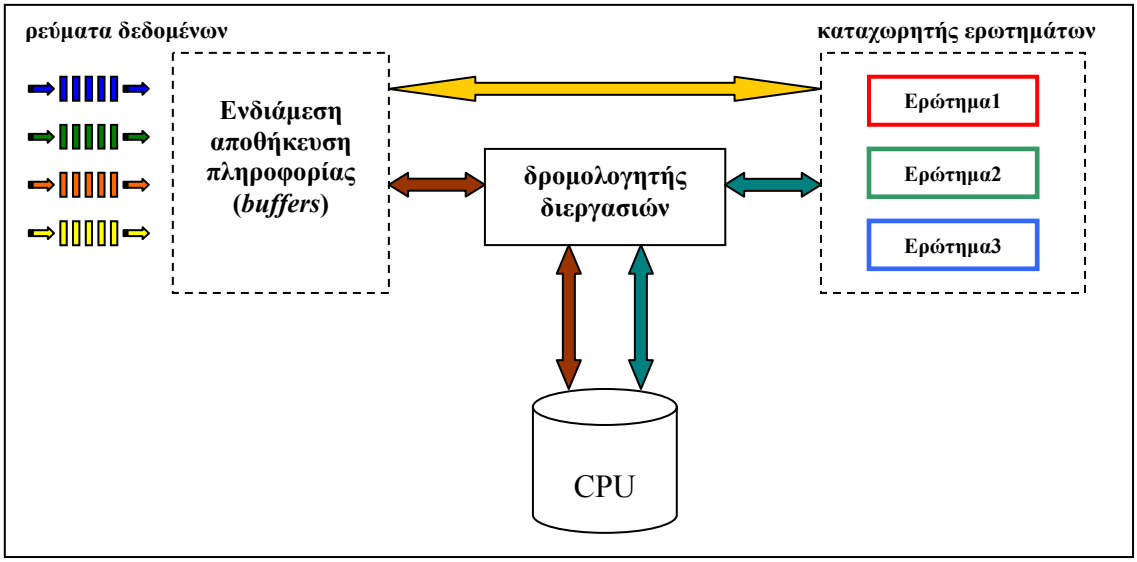
Η υλοποίηση των παραθύρων και των τελεστών έγινε σε γλώσσα C++ ενώ για την συγγραφή του κώδικα χρησιμοποιήθηκε το προγραμματιστικό εργαλείο Visual Studio .Net σε περιβάλλον Windows XP. Υπήρξε ιδιαίτερη μέριμνα ώστε να μην χρησιμοποιηθούν έτοιμα πακέτα ή κλάσεις λογισμικού με στόχο η συγγραφή του κώδικα να ξεκινήσει από μηδενική βάση και στην τελική του μορφή να περιλαμβάνει αποκλειστικά στοιχεία της standard έκδοσης της C++. Ο στόχος αυτός επετεύχθη απόλυτα, με την εξαίρεση ότι χρησιμοποιήθηκε έτοιμη η υλοποίηση των *διεργασιών* (*threads*) που παρέχεται μαζί με το λειτουργικό σύστημα Linux (RedHat 9) ακολουθώντας το πρότυπο POSIX.

5.2 Προδιαγραφές συστήματος

Η υλοποίηση του συστήματος καλείται να ικανοποιεί τις παρακάτω απαιτήσεις:

- δημιουργία ρευμάτων δεδομένων κατά τρόπο ελεγχόμενο, με στόχο την προσομοίωση ποικίλων ρυθμών εισροής δεδομένων στο σύστημα για διαφορετικά ρεύματα,
- δημιουργία κατάλληλων τελεστών για τον σχηματισμό ερωτημάτων διαρκείας που θα εφαρμοστούν στα παραπάνω ρεύματα δεδομένων,
- υποστήριξη βασικών τύπων δεδομένων όπως ακέραιοι (*integer*), δεκαδικοί (*double*), χαρακτήρες (*char*), και συμβολοσειρές (*charn*),
- ελαχιστοποίηση της επαναλαμβανόμενης πληροφορίας στο μέτρο του δυνατού,
- υποστήριξη εκτέλεσης πολλαπλών ερωτημάτων.

Στο διάγραμμα που ακολουθεί δίνεται η γενική εικόνα των δυνατοτήτων που πρέπει να προσομοιώνει το σύστημα.



Σχήμα 5.1: *Διάγραμμα λειτουργίας συστήματος*

Στο διάγραμμα αυτό σκόπιμα δεν παρουσιάζονται οι χρήστες, για να τονιστεί ότι το σύστημα δεν προορίζεται σε αυτό το στάδιο να επιτελεί *ανάλυση ερωτημάτων* (*query analyzer*) ή δημιουργία *προσχεδίων εκτέλεσης* (*query execution plans*).

Έτσι για την επίτευξη των στόχων του συστήματος έγιναν οι εξής υποθέσεις εργασίας:

1) Δεν αποτελεί στόχο, στο παρόν στάδιο, η υλοποίηση συντακτικού *αναλυτή* (*parser*) ερωτημάτων που θα τίθεται από τον χρήστη κατά την λειτουργία του συστήματος. Ο χρήστης επιλέγει μία φορά από ένα σύνολο προκατασκευασμένων σε επίπεδο φυσικού προσχεδίου εκτέλεσης ερωτημάτων (ενότητα 5.6).
2) Δεν αποτελεί στόχο, στην παρόν στάδιο, η αντιμετώπιση προβλημάτων χρονικής διάταξης των στοιχείων στα ρεύματα. Η ύπαρξη χρονικής διάταξης αποτελεί βασική παραδοχή στην υλοποίηση του συστήματος.
3) Το χρονόσημο κάθε πλειάδας είναι ένας ακέραιος, τον οποίο είτε τον καθορίζει το σύστημα είτε περιέχεται ως πληροφορία μέσα στα *γνωρίσματα* (*attributes*) κάθε πλειάδας, αναπαριστώντας τον χρόνο παραγωγής της.
4) Δεν θα χρησιμοποιηθεί ο σκληρός δίσκος σε περιπτώσεις όπου η κύρια μνήμη δεν επαρκεί. Θεωρείται ότι υπάρχει επάρκεια διαθέσιμης κύριας μνήμης για την εκτέλεση των ερωτημάτων και τους *καταχωρητές* (*buffers*) προσωρινής αποθήκευσης των ρευμάτων.
5) Δεν τίθεται στόχος κατασκευής *συνόψεων* (*synopses*).
6) Η έξυπνη χρονοδρομολόγηση των διεργασιών δεν αποτελεί βασικό στόχο της παρούσας εργασίας. Σε πρώτο στάδιο θα αρκεστούμε στην δίκαιη χρονοδρομολόγηση διεργασιών σε επίπεδο ερωτημάτων.

Στις ενότητες που ακολουθούν αρχικά παρουσιάζονται βασικές δομές και κλάσεις που χρησιμοποιεί το σύστημα για: την προσομοίωση των ρευμάτων δεδομένων (ενότητα 5.3), την υλοποίηση των επιλεγμένων παραθύρων (ενότητα 5.4) και την υλοποίηση των επιλεγμένων τελεστών (ενότητα 5.5). Στην ενότητα 5.6 επιδεικνύεται η σύνδεση όλων των δομών για την εκπλήρωση των απαιτήσεων του παραπάνω διαγράμματος.

5.3 Δομές προσομοίωσης ρευμάτων δεδομένων

Τα ρεύματα δεδομένων προσομοιώνονται διαβάζοντας δεδομένα από πρότυπα ASCII αρχεία εισόδου με ελεγχόμενο ρυθμό. Η εργασία αυτή επιτελείται με την βοήθεια της κλάσης `Scanner`

```
int, int, charn 15,timestamp ,charn 30,double
1,97,KAVALA,0,'2004-01-31 22:00:00 EET',27.60
2,2,ATHENS,0,'2004-01-31 22:00:00 EET',27.60
3,58,VOLOS,0,'2004-01-31 22:00:00 EET',31.85
4,364,CORFU,0,'2004-01-31 22:00:00 EET',26.54
5,717,HERAKLION,0,'2004-01-31 22:00:00 EET',25.48
6,701,LARISSA,0,'2004-01-31 22:00:00 EET',27.60
.....
```

Εικόνα 5.2: *Παράδειγμα αρχείου εισόδου ενός ρεύματος δεδομένων*

ενώ κάθε αρχείο περιέχει πληροφορία για το σχήμα του ρεύματος. Τα στοιχεία του ρεύματος διατηρούνται σε κόμβους (κλάση `Node`) και κόμβοι του ιδίου ρεύματος ομαδοποιούνται σε δομές ουρών (κλάση `Queue`). Κάθε ρεύμα λοιπόν προσομοιώνεται ως ένα αντικείμενο `Queue` το οποίο αποτελείται από `Nodes` και γεμίζει μέσω της `Scanner`. Παρακάτω αναλύονται αυτές οι κλάσεις όπως και οι υπέρ-κλάσεις `UnaryOperator` και `BinaryOperator` οι οποίες αποτελούν τον βασικό σκελετό για παράθυρα και τελεστές ενός ρεύματος και δύο ρευμάτων εισόδου αντίστοιχα.

5.3.1 Κλάση `Node`

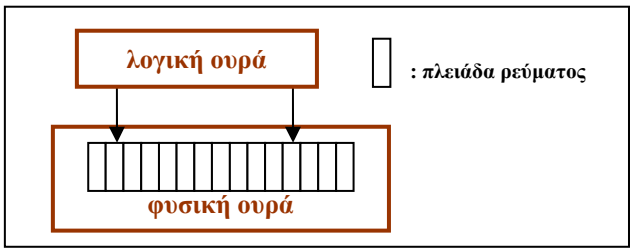
Κάθε πλειάδα διατηρεί την πληροφορία της σε ένα *αντικείμενο* (*object*) *κόμβου* (*Node*) της αντίστοιχης κλάσης. Η πληροφορία αποθηκεύεται σε δυαδική μορφή (bytes) και ο ίδιος ο κόμβος δεν μπορεί να γνωρίζει αν περιλαμβάνει ακέραιους, δεκαδικούς ή χαρακτήρες. Την ερμηνεία των περιεχομένων bytes την πραγματοποιεί η *ουρά* (*Queue*) στην οποία ανήκει, όπως εξηγείται παρακάτω. Ο σχεδιασμός αυτός είχε σαν στόχο την συγκέντρωση της κοινής πληροφορίας για το σχήμα των κόμβων κάθε ρεύματος, στα αντικείμενα ουρών που αναφέρονται στο αντίστοιχο ρεύμα, αποφεύγοντας άσκοπες επαναλήψεις σε κάθε κόμβο.

Πέρα από τα bytes πληροφορίας, κάθε κόμβος περιλαμβάνει έναν ακέραιο ο οποίος αναπαριστά το *χρονόσημο* κάθε πλειάδας και μία Bool μεταβλητή που δηλώνει αν η πλειάδα αυτή αποτελεί πλειάδα του ρεύματος ή *πλειάδα στίξης* (*punctuation*). Οι πλειάδες στίξης προστίθενται στο ρεύμα από το σύστημα σηματοδοτώντας το τέλος των στοιχείων ενός συγκεκριμένου χρονοσήμου. Τέλος, κάθε κόμβος περιλαμβάνει δείκτες για τον επόμενο (για την σύνδεση των στοιχείων κάθε ουράς) και τον προηγούμενο κόμβο του ρεύματος (λόγω απαιτήσεων των μεριστικών παραθύρων – ενότητα 5.4.2) συνδέοντας κατά τον τρόπο αυτό τα στοιχεία κάθε ρεύματος.

5.3.2 Κλάση `Scanner`

Βασικά, αυτός ο τελεστής αντλεί στοιχεία και τα προωθεί στο σύστημα για περαιτέρω επεξεργασία. Ο ρόλος κάθε αντικειμένου αυτής της κλάσης είναι η μετατροπή ενός ASCII αρχείου εισόδου σε ρεύμα δεδομένων. Κάθε γραμμή αποτελεί μία πλειάδα του ρεύματος ενώ η πρώτη γραμμή κάθε αρχείου δίνει την πληροφορία του σχήματος των πλειάδων που θα ακολουθήσουν. Ένα ενδεικτικό παράδειγμα τέτοιου αρχείου με μετρήσεις από αισθητήρες θερμοκρασίας φαίνεται στην εικόνα 5.2

Ο *κατασκευαστής* (*constructor*) της κλάσης `Scanner` καλεί έναν από τους κατασκευαστές της κλάσης `Queue` με όρισμα αυτήν την γραμμή σχήματος. Στην επόμενη ενότητα για την κλάση `Queue` αναλύεται ο τρόπος επεξεργασίας της γραμμής αυτής που έχει ως αποτέλεσμα τον καθορισμό του σχήματος του αντίστοιχου ρεύματος δεδομένων. Στον κατασκευαστή της κλάσης `Scanner` αρχικοποιείται επίσης μία παράμετρος `rate` για κάθε αντικείμενο, η οποία δηλώνει το πλήθος των πλειάδων που επιλέγεται να διαβαστούν στη διάρκεια ενός δευτερολέπτου. Κατά τον τρόπο αυτό προσομοιώνεται ο *ρυθμός άφιξης* στοιχείων στο αντίστοιχο ρεύμα. Την λειτουργία διαβάσματος πλειάδων με συγκεκριμένο ρυθμό την αναλαμβάνει η μέθοδος `scan()` κάθε αντικειμένου `Scanner`.



Σχήμα 5.3: *Φυσικές και λογικές ουρές*

Έτσι πολλαπλά ρεύματα δεδομένων προσομοιώνονται κατά τον ακόλουθο τρόπο :

- Αρχικά καλείται ο κατασκευαστής της κλάσης `Scanner` για καθένα από τα ρεύματα με τις δικές του παραμέτρους: για το όνομα του αρχείου, τον χαρακτήρα που διαχωρίζει τα γνωρίσματα (*attributes*) κάθε πλειάδας μέσα στο αρχείο (στην περίπτωση του σχήματος 5.2 το κόμμα ",") και τον ακέραιο που καθορίζει τον ρυθμό των άφιξης των δεδομένων.
- Ακολούθως, η μέθοδος `scan()` κάθε αντικειμένου `Scanner` που υλοποιήθηκε παραπάνω ανατίθεται σε ξεχωριστή *διεργασία* (*thread*) και επαφίεται στο λειτουργικό σύστημα να καθορίσει την σειρά και τον χρόνο εκτέλεσής της.

5.3.3 Κλάση `Queue`

Τα αντικείμενα *ουρών* (*Queue*) όπως φάνηκε από την παραπάνω ενότητα είναι ικανά να αναπαραστήσουν τον ενδιάμεσο χώρο αποθήκευσης που θα χρησιμοποιηθεί από κάθε πρωτογενές ρεύμα δεδομένων που εισέρχεται στο σύστημα. Επίσης, σε αντικείμενα αυτής της κλάσης ανατίθεται ο ρόλος της διασύνδεσης των παραθύρων και των τελεστών μεταξύ τους κατά τον σχηματισμό των *προσχεδίων εκτέλεσης ερωτημάτων* (*query execution plans*). Βασικά στοιχεία των ουρών είναι οι δείκτες στους κόμβους *αρχής* (`head`) και *τέλους* (`tail`) οι οποίοι χρησιμοποιούνται για αναφορά στις πλειάδες του ρεύματος που οριοθετούν κάθε αντικείμενο ουράς.

Διακρίνονται δύο είδη ουρών:

- οι **φυσικές** ουρές, που περιέχουν άμεσες αναφορές σε πραγματικές πλειάδες δεδομένων προσθέτοντας και διαγράφοντας αντικείμενα κόμβων
- οι **λογικές** ουρές, που περιέχουν αναφορές σε *φυσικές* ουρές χωρίς να επαναλαμβάνουν ή να επηρεάζουν πλειάδες που περιέχονται στην φυσική ουρά.

Αυτή η διάκριση φαίνεται πιο παραστατικά στο σχήμα 5.3 και γίνεται για να αποφευχθεί η επαναλαμβανόμενη πληροφορία που διατηρείται στο σύστημα όπου αυτό είναι εφικτό. Τέλος είναι δυνατή η ύπαρξη και *λογικής* ουράς με αναφορά σε άλλη επίσης *λογική* ουρά κ.ο.κ. καταλήγοντας σε κάθε περίπτωση σε κάποια *φυσική*.

Η κλάση αυτή περιλαμβάνει ξεχωριστούς κατασκευαστές και μεθόδους εισαγωγής ή διαγραφής στοιχείων για τους δύο τύπους ουρών. Η σωστή χρήση τους είναι ευθύνη του προγραμματιστή. Παρακάτω αναλύονται διεξοδικά οι ξεχωριστές λειτουργίες κάθε τύπου ουράς.

Για την *φυσική ουρά* παρέχονται οι εξής λειτουργίες:

- Παρέχονται δύο μέθοδοι εισαγωγής κόμβων: η `insertScannedNode()` και η `insertNodeBytes()`. Η πρώτη χρησιμοποιείται για την μετατροπή μίας γραμμής από το αρχείο εισόδου σε αντικείμενο κόμβου το οποίο προστίθεται στο τέλος της ουράς. Η μέθοδος αυτή χρησιμοποιεί την πληροφορία της ουράς για το σχήμα για να περάσει τα κατάλληλα bytes πληροφορίας στον κατασκευαστή του νέου κόμβου. Η δεύτερη μέθοδος επίσης προσθέτει ένα αντικείμενο κόμβου στο τέλος της ουράς αλλά δέχεται σαν όρισμα έτοιμα τα bytes της πληροφορίας που τελικά θα αποθηκευτούν στον νέο κόμβο. Και οι δύο μέθοδοι δημιουργούν αποκλειστικά κόμβους ρεύματος.
- Για τους *κόμβους στίξης* (*punctuation Nodes*) υπάρχει ξεχωριστή μέθοδος η `insertPunctuation()` η οποία δέχεται ως όρισμα έναν ακέραιο. Η μέθοδος αυτή δημιουργεί και εισάγει στο τέλος της ουράς έναν κόμβο στίξης με χρονόσημο ίσο με τον ακέραιο που

περάστηκε σαν όρισμα, δηλώνοντας το τέλος των στοιχείων του ρεύματος με αυτήν την τιμή χρονοσήμου. Καλείται κατά την δημιουργία των πρωτογενών ρευμάτων μόλις ανιχνευτεί αλλαγή χρονοσήμου κατά την ανάγνωση του αντίστοιχου αρχείου, και εισάγεται στην φυσική ουρά του ρεύματος πριν από τον πρώτο κόμβο με το νέο χρονόσημο. Η υπόθεση εργασίας για χρονικά διατεταγμένα ρεύματα εξασφαλίζει ότι η θέση στην οποία τοποθετείται ο κόμβος στίξης είναι πράγματι η σωστή.

- Για την διαγραφή στοιχείων από την ουρά παρέχεται η μέθοδος `deleteNode()` η οποία σε κάθε κλήση της διαγράφει την παλαιότερη πλειάδα της ουράς

Για την *λογική ουρά* παρέχονται οι εξής λειτουργίες:

- Παρέχεται η μέθοδος `insertExistingNode()` για την εισαγωγή νέου κόμβου στο τέλος της ουράς. Η μέθοδος αυτή δεν δημιουργεί νέο κόμβο αλλά αντίθετα αρκείται στην μεταβολή του δείκτη τέλους της ουράς ώστε να περιλάβει έναν ακόμη κόμβο. Στο σημείο αυτό πραγματοποιείται έλεγχος για την περίπτωση που ο νέος κόμβος είναι *κόμβος στίξης* οπότε και δεν πρέπει να μεταβληθούν τα στατιστικά στοιχεία της ουράς.
- Η διαγραφή κόμβων από αυτήν την ουρά δεν συνοδεύεται από φυσική καταστροφή κάποιου κόμβου. Αυτό αποτελεί δυνατότητα και λειτουργία των φυσικών ουρών. Η απλή αφαίρεση του παλαιότερου στοιχείου από την λογική ουρά γίνεται με την μέθοδο `deleteLogicalNode()`.

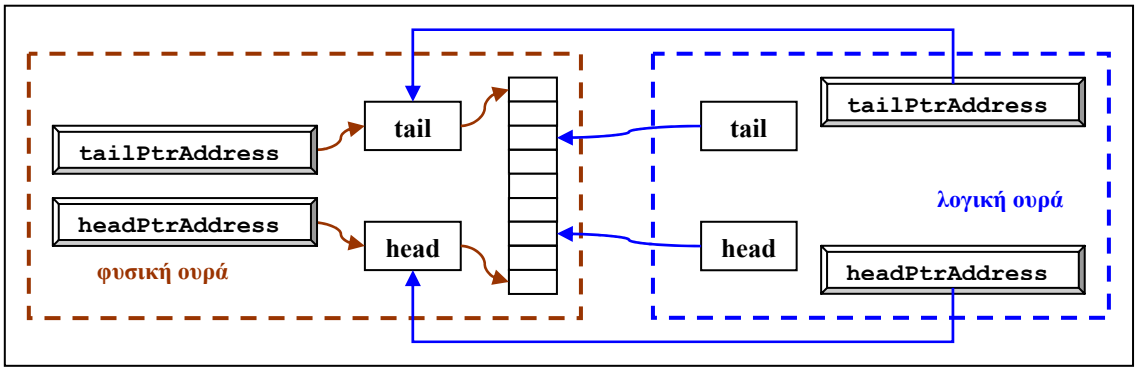
Υπάρχουν δύο κατασκευαστές για τις ουρές. Και οι δύο δημιουργούν άδειες ουρές και έχουν σαν κύριο στόχο να αρχικοποιήσουν την πληροφορία που φυλάσσεται για το σχήμα των περιεχόμενων πλειάδων.

Ο πρώτος εκ των δύο είναι αυτός που αναφέρθηκε στην προηγούμενη ενότητα ότι καλείται από τον κατασκευαστή της κλάσης Scanner. Δέχεται ως όρισμα την γραμμή σχήματος από τα αρχεία εισόδου των ρευμάτων, καθώς και τον χαρακτήρα που χρησιμοποιείται σε αυτό το αρχείο για διαχωρισμό των *γνωρισμάτων* κάθε πλειάδας. Από την επεξεργασία αυτής της γραμμής προκύπτει πληροφορία για το πλήθος των *γνωρισμάτων* και το συνολικό πλήθος των bytes που θα έχει κάθε πλειάδα του ρεύματος, όπως επίσης και για τον *τύπο* (*type*), το *μέγεθος* σε bytes (*size*), και την *θέση του πρώτου byte* (*offset*) κάθε *γνωρίσματος ξεχωριστά*. Η πληροφορία αυτή διατηρείται αποκλειστικά στις δομές ουρών καθιστώντας τες το μοναδικό μέσο ερμηνείας της πληροφορίας που κατέχουν οι κόμβοι. Η γραμμή σχήματος μπορεί να δηλώσει την λέξη `timestamp` ως τύπο ενός *γνωρίσματος*. Η σήμανση αυτή δηλώνει ότι αυτό το πεδίο είναι ένας ακέραιος, ο οποίος θα πρέπει να αποτελέσει το χρονόσημο της πλειάδας αυτής, το οποίο αποθηκεύεται ως ξεχωριστό πεδίο σε κάθε κόμβο. Αν δεν δηλωθεί τέτοιος τύπος τότε θα πρέπει για κάθε εισερχόμενο κόμβο να παρέχεται και η πληροφορία του χρονόσημου που θα έχει. Ο δεύτερος κατασκευαστής αποτελεί έναν `copy constructor` ο οποίος αντιγράφει τα δεδομένα σχήματος από την ουρά που δέχεται ως όρισμα.

Όλες οι ουρές διατηρούν διαρκώς ενήμερα κάποια στατιστικά στοιχεία για τα περιεχόμενα τους. Συγκεκριμένα κάθε ουρά γνωρίζει το πλήθος των κόμβων που περιέχονται μεταξύ των δεικτών *αρχής* και *τέλους* της ουράς. Επίσης κρατάει την πληροφορία του πιο πρόσφατου και του πιο παλαιού χρονοσήμου των περιεχομένων της. Στα στοιχεία αυτά δεν συμμετέχουν οι κόμβοι στίξεως αφού αυτοί προστέθηκαν από το σύστημα για τους δικούς του σκοπούς και δεν φέρουν την πληροφορία κάποιας πλειάδας του ρεύματος.

Οι ουρές κρατούν και κάποια ιδιαίτερα σημαντικά πεδία, τα οποία επιτρέπουν στους τελεστές που τις χρησιμοποιούν να ακολουθούν την εξέλιξη των αντίστοιχων φυσικών ουρών. Τα πεδία αυτά είναι οι διπλοί δείκτες (δείκτης σε δείκτη) `headPtrAddress` και `tailPtrAddress`. Στις λογικές ουρές το περιεχόμενο αυτών των διπλών δεικτών είναι οι διευθύνσεις των δεικτών `head` και `tail` της αναφερόμενης φυσικής ή λογικής ουράς. Στην περίπτωση των φυσικών ουρών αυτοί οι διπλοί δείκτες περιέχουν τις διευθύνσεις στις οποίες αποθηκεύονται οι δείκτες `head` και `tail` της ιδίας. Δίχως την ύπαρξή τους οι τελεστές που χρησιμοποιούν κάποια λογική ουρά δεν θα μπορούσαν να «δουν» τις αλλαγές της φυσικής ουράς στην οποία αναφέρονται. Στο σχήμα 5.4 απεικονίζονται αυτά τα πεδία.

Τέλος θα πρέπει να σημειωθεί ότι κατά την αρχικοποίηση και των δύο τύπων ουρών εισάγεται σε αυτές ένας «ψευδο-κόμβος», ο οποίος εξυπηρετεί στην σωστή αρχικοποίηση των πεδίων τους και πιο συγκεκριμένα των διπλών δεικτών `headPtrAddress` και `tailPtrAddress`. Στον



Σχήμα 5.4: *Γραφική απεικόνιση διπλών δεικτών σε φυσικές και λογικές ουρές*

κόμβο αυτό ανατίθεται η άκυρη τιμή χρονοσήμου -1 όπως επίσης και σε όλα τα πεδία του. Δόθηκε ιδιαίτερη προσοχή έτσι ώστε ο κόμβος αυτός να μην προσμετράται στα στατιστικά στοιχεία που κρατάει η κάθε ουρά και να μην αποτελεί κόμβο επεξεργασίας για τα παράθυρα και τους τελεστές.

5.3.4 Υπέρ-κλάση `UnaryOperator`

Ο ρόλος αυτής της *αφηρημένης υπέρ-κλάσης* (*abstract super-class*) είναι να δημιουργήσει μία κοινή *διεπαφή* (*interface*) την οποία θα αποκτήσουν μέσω *κληρονομικότητας* όλα τα παράθυρα και οι τελεστές που έχουν μία ουρά εισόδου. Οι δομές αυτές θα αναφέρονται με τον όρο *υποκλάσεις* (*subclass*) της `UnaryOperator`.

Ο *κατασκευαστής* αυτής της κλάσης καλείται από τον *κατασκευαστή* της υποκλάσης του και δέχεται ως όρισμα την ουρά που θα αποτελέσει την είσοδο στο αντικείμενο αυτής της υποκλάσης. Δημιουργείται ένα λογικό αντίγραφο αυτής της ουράς και οι δείκτες *head* και *tail* τίθενται ίσοι με τους αντίστοιχους δείκτες της ουράς που περάστηκε σαν όρισμα, ενώ ρυθμίζονται και τα στατιστικά στοιχεία της νέας ουράς. Η λογική ουρά που προέκυψε είναι προσπελάσιμη με το όνομα `inputQueue` από οποιαδήποτε υποκλάση και αν κάλεσε τον κατασκευαστή. Αντίστοιχα, η ουρά εξόδου, αν και δεν αρχικοποιείται στο σημείο αυτό, θα είναι γνωστή με το όνομα `outputQueue`. Εδώ αρχικοποιείται και ο δείκτης `workingNode` που θα έχει κάθε υποκλάση δείχνοντας στην αρχή της ουράς εισόδου. Ο δείκτης αυτός χρησιμοποιείται για να σημαδεύει τον κόμβο της ουράς εισόδου στον οποίο φτάνει μέχρι στιγμής η επεξεργασία.

Η επιλογή να υλοποιηθεί η `inputQueue` ως *λογική* και όχι ως *φυσική* ουρά εξηγείται από το γεγονός ότι ο ρόλος της ουράς εισόδου σε κάθε υποκλάση είναι να παρέχει στο αντικείμενο κάποιες πλειάδες, οι οποίες πιθανόν να προκαλέσουν κάποιο αποτέλεσμα στην `outputQueue`. Δεν υπάρχει κανένας λόγος να αλλάξει η σειρά η διάταξη ή το σχήμα των πλειάδων εισόδου μια και χρησιμοποιούνται μόνο για ανάγνωση κάποιων πεδίων. Με την επιλογή αυτή γίνεται ένα σημαντικό βήμα ως προς τον στόχο ελαχιστοποίησης της επαναλαμβανόμενης πληροφορίας στο σύστημα.

Όλες οι υποκλάσεις κληρονομούν την μέθοδο `getCount()`που επιστρέφει το πλήθος των στοιχείων της `outputQueue`. Υποχρεώνονται όμως να υλοποιήσουν τις δικές τους μεθόδους:

- `operate()`, η οποία καλείται να καταναλώνει τα στοιχεία της ουράς εισόδου και να επιτελεί την λειτουργία κάθε τελεστή ή παραθύρου όπως ορίζεται στην αντίστοιχη υποκλάση.
- `printContents()`, η οποία δημιουργήθηκε με στόχο να βοηθήσει στο στάδιο εξακρίβωσης ορθής λειτουργίας, εκτυπώνοντας σε κάθε κλήση της όλους τους κόμβους που περιέχονται στην `outputQueue` του τελεστή ή του παραθύρου.

5.3.5 Υπέρ-κλάση `BinaryOperator`

Αυτή η *αφηρημένη υπέρ-κλάση* δημιουργείται για να αποτελέσει την κοινή *διεπαφή* την οποία θα αποκτήσουν μέσω *κληρονομικότητας* όλα τα παράθυρα και οι τελεστές που έχουν δύο ουρές

εισόδου. Οι δομές αυτές θα αναφέρονται με τον όρο *υποκλάσεις* της `BinaryOperator`. Παρέχει αντίστοιχη λειτουργικότητα με αυτήν της *διεπαφής* `UnaryOperator`, ενώ στην τρέχουσα έκδοση του συστήματος, η μοναδική υποκλάση της είναι ο τελεστής παραθυρικής σύνδεσης διοχέτευσης.

5.4 Υλοποιήσεις παραθύρων

Τα παράθυρα που υλοποιήθηκαν είναι τα εξής:

- *παράθυρα πλειάδων* (*tuple-based windows*),
- *μεριστικά παράθυρα* (*partitioned windows*) με ένα γνώρισμα ομαδοποίησης,
- *κυλιόμενα παράθυρα* (*sliding windows*),
- *επάλληλα παράθυρα* (*tumbling windows*) ως ειδική περίπτωση των κυλιόμενων,
- *παράθυρα οροσήμου* (*landmark windows*) με σταθερό το κάτω άκρο.

Όλα τα παράθυρα κληρονομούν από την κλάση `UnaryOperator` και κατά συνέπεια διαχειρίζονται τα δικά τους αντικείμενα `inputQueue` και `outputQueue`. Η υλοποίηση των μεθόδων `operate()` και `printContents()` είναι παρόμοια για όλα τα παράθυρα και για το λόγο αυτό αναλύεται παρακάτω. Πέρα όμως από τις μεθόδους αυτές που καλούνται να υλοποιήσουν λόγω της υπέρ-κλάσης τους, όλα τα παράθυρα υλοποιούν με τους δικούς τους κανόνες και τις μεθόδους `checkNode()` και `initializeOutput()`.

Η μέθοδος `checkNode()`, παίρνοντας ως όρισμα έναν κόμβο κάνει τους απαραίτητους ελέγχους για να διαπιστωθεί αν πρέπει να αλλάξει κατ' οποιοδήποτε τρόπο την `outputQueue`.Κάθε παράθυρο, όταν δέχεται μία *πλειάδα στίξης* ως παράμετρο στην μέθοδο `checkNode()` την μεταβιβάζει στην έξοδο του, χωρίς να ελέγχονται οι κανόνες του παραθύρου αλλά και χωρίς να επηρεάζονται τα στατιστικά στοιχεία της ουράς εξόδου.

Η μέθοδος `initializeOutput()`εφαρμόζει κάποιους επιπρόσθετους έλεγχους για την εισαγωγή του πρώτου στοιχείου της ουράς εξόδου στον κόμβο που σημαδεύει ο δείκτης `workingNode`.Αν ο κόμβος περάσει τους ελέγχους αυτούς με επιτυχία, το επόμενο βήμα είναι η κλήση της `checkNode()` με όρισμα τον κόμβο αυτόν.

Σε κάθε κλήση της μεθόδου `operate()` κάθε παραθύρου γίνεται επεξεργασία το πολύ ενός κόμβου από την ουρά εισόδου, επηρεάζοντας ανάλογα την ουρά εξόδου. Έτσι στην `outputQueue` παρέχεται συνεχώς εκείνο το στιγμιότυπο του ρεύματος που καθορίζεται από την σημασιολογία του παραθύρου και τα μέχρι τότε επεξεργασμένα στοιχεία της ουράς εισόδου. Αν ανιχνευτεί νέος κόμβος στην ουρά εισόδου, ακολουθεί έλεγχος αν η `outputQueue` περιέχει στοιχεία. Αν είναι άδεια, καλείται η μέθοδος `initializeOutput()` του παραθύρου. Σε αντίθετη περίπτωση καλείται απευθείας η μέθοδος `checkNode()` με όρισμα τον νέο κόμβο εισόδου στον οποίο φροντίζουμε να αναφέρεται ο δείκτης `workingNode`.

Η μέθοδος `printContents()` εκτυπώνει όλες τις περιεχόμενες πλειάδες της `outputQueue` κάθε παραθύρου και καλείται σε κάθε αλλαγή της ουράς αυτής. Αποτελεί το βασικό εργαλείο μέσω του οποίου πραγματοποιήθηκε ο έλεγχος της ορθότητας λειτουργίας του συστήματος.

Ακολουθεί παρουσίαση των ιδιαιτεροτήτων της υλοποίησης κάθε παραθυρικής δομής.

5.4.1 *Παράθυρα πλειάδων* (*Tuple-based windows*)

Παράθυρα πλειάδων δημιουργούνται με την βοήθεια της κλάσης `TupleWindow`.Ο *κατασκευαστής* αυτής της κλάσης δέχεται ως ορίσματα ένα δείκτη στην ουρά εισόδου, και δύο ακεραίους `size` και `start`.

- Η παράμετρος ουράς παρέχεται στον κατασκευαστή της υπέρ-κλάσης `UnaryOperator` επιστρέφοντας ένα λογικό αντίγραφο της στην `inputQueue`.
- Η παράμετρος `size` καθορίζει το μέγεθος του παραθύρου, δηλώνοντας το μέγιστο πλήθος των στοιχείων που θα περιέχει η `outputQueue`.
- Η παράμετρος `start` καθορίζει το χρονόσημο εκείνο έπειτα από το οποίο θα εφαρμοστεί το παράθυρο στα δεδομένα της ουράς εισόδου. Όλες οι πλειάδες με χρονόσημο μικρότερο από αυτήν

την τιμή δεν διέρχονται ποτέ στην outputQueue και ο έλεγχος αυτός πραγματοποιείται στην μέθοδο initializeOutput().

Επίσης ο *κατασκευαστής* δημιουργεί μια άδεια λογική ουρά εξόδου καλώντας τον αντίστοιχο Queue *κατασκευαστή* με σκοπό να αντιγράψει στην outputQueue την πληροφορία του σχήματος της ουράς εισόδου. Η επιλογή υλοποίησης της ουράς εξόδου ως *λογικής* δικαιολογείται από το γεγονός ότι η μοναδική λειτουργία που καλείται να κάνει το παράθυρο πλειάδων είναι ένας περιορισμός σε συγκεκριμένο τμήμα *N* πλειάδων του ρεύματος εισόδου. Κάθε ζητούμενο κομμάτι εξόδου δεν διαφέρει στα περιεχόμενα στην διάταξη ή στο σχήμα από το αντίστοιχο τμήμα στην είσοδο, υποδεικνύοντας έτσι την επιλογή λογικής ουράς εξόδου.

Η checkNode() αρχικά ελέγχει αν ο κόμβος που καλείται να εξετάσει αποτελεί κόμβο δεδομένων. Αν αυτό ισχύει, τότε πρέπει να περάσει τον κόμβο στην έξοδο με την μέθοδο insertExistingNode() της outputQueue.Ακολούθως, συγκρίνοντας το νέο πλήθος των στοιχείων που περιέχονται στην έξοδο με την μεταβλητή μέγιστου μεγέθους (maxSize) του παραθύρου εξετάζει αν πρέπει να διαγραφούν κόμβοι από την ουρά εξόδου. Την λειτουργία αυτή επιτελεί όποτε κριθεί σκόπιμο η μέθοδος deleteNodes() του παραθύρου, η οποία εφόσον εφαρμόζεται σε λογική ουρά δεν προκαλεί πραγματική διαγραφή κάποιου κόμβου, αλλά αρκείται σε μεταβολές δεικτών και στατιστικών στοιχείων της ουράς.

5.4.2 *Μεριστικά παράθυρα* (*Partitioned windows*)

Η κλάση PartitionedWindow χρησιμοποιείται για την δημιουργία μεριστικών παραθύρων με ένα μόνο *γνώρισμα ομαδοποίησης* (*grouping attribute*). Η κλάση αυτή συνοδεύεται από τις συμπληρωματικές κλάσεις GroupSubQueue και GroupNode με τις οποίες δημιουργείται ένα είδος ευρετηρίου στους κόμβους της outputQueue.

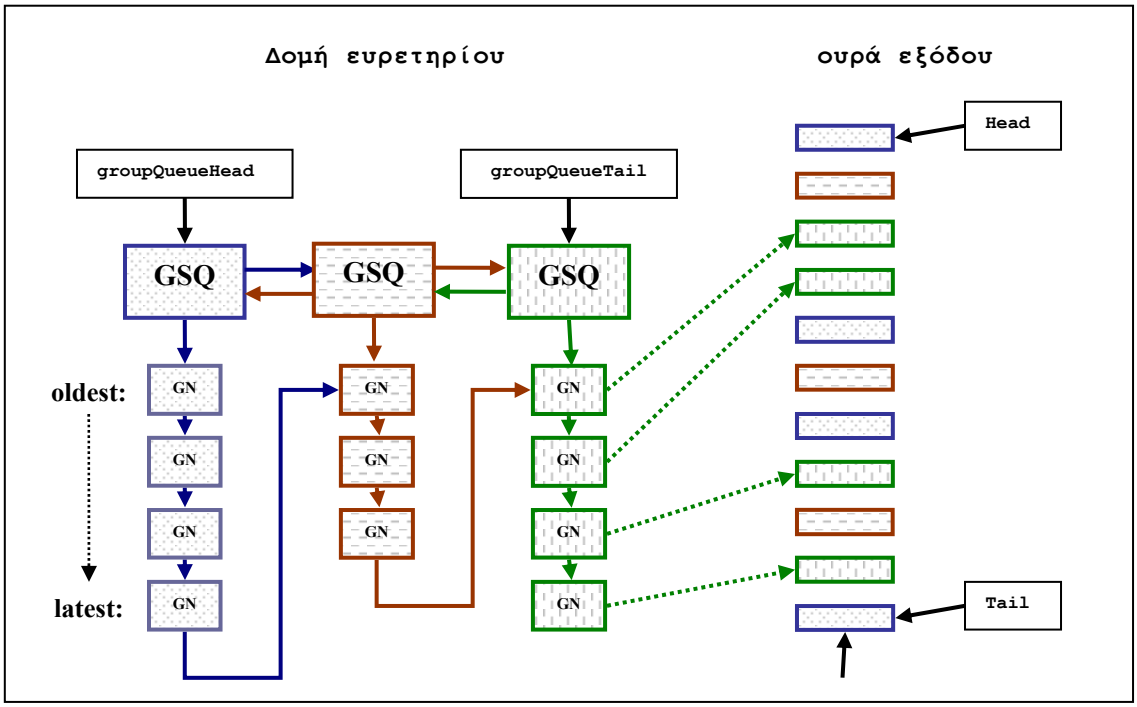
Ο *κατασκευαστής* της κλάσης PartitionedWindow δέχεται ως ορίσματα ένα δείκτη στην ουρά εισόδου, και τρεις ακεραίους n, attr και start:

- Η παράμετρος ουράς παρέχεται στον κατασκευαστή της υπέρ-κλάσης UnaryOperator επιστρέφοντας ένα λογικό αντίγραφο της στην inputQueue.
- Η παράμετρος n καθορίζει το πλήθος των στοιχείων που θα διατηρούνται στην ουρά εξόδου από κάθε διαφορετική τιμή του *γνωρίσματος ομαδοποίησης*.
- Η παράμετρος attr δηλώνει ποιο γνώρισμα χρησιμοποιηθεί για την ομαδοποίηση. Θεωρείται ότι δίνεται σωστή τιμή στο πεδίο αυτό παραθύρου και τέτοιες είναι οι ακέραιες τιμές [0 .. (πλήθος ιδιοτήτων πλειάδας -1)].
- Η παράμετρος start καθορίζει το χρονόσημο εκείνο έπειτα από το οποίο θα εφαρμοστεί το παράθυρο στα δεδομένα της ουράς εισόδου. Όλες οι πλειάδες με χρονόσημο μικρότερο από αυτήν την τιμή δεν διέρχονται ποτέ στην outputQueue και ο έλεγχος αυτός πραγματοποιείται στην μέθοδο initializeOutput().

Στον *κατασκευαστή* δημιουργείται μια άδεια φυσική ουρά εξόδου καλώντας τον copy constructor της κλάσης Queue με όρισμα την ουρά εισόδου και σκοπό την δημιουργία outputQueue ιδίου σχήματος. Η ουρά εξόδου υλοποιήθηκε ως *φυσική* ουρά, εφόσον η έλευση κάθε νέας πλειάδας μπορεί να προκαλέσει διαγραφή σε τυχαίο σημείο της ουράς εξόδου αλλάζοντας την διάταξή της.

Δύο ήταν τα κύρια ζητήματα που ανέκυψαν κατά την υλοποίηση αυτού του παραθύρου:

- Ποτέ δεν είναι εκ των προτέρων γνωστός ο κόμβος που θα φύγει από την ουρά στην επόμενη διαγραφή. Αυτός θα καθοριστεί από τα δεδομένα που θα έρθουν στην ουρά εισόδου. Στα υπόλοιπα παράθυρα δεν υφίσταται τέτοιο θέμα αφού πάντοτε οι τυχόν διαγραφές γίνονται ξεκινώντας από το παλαιότερο στοιχείο της outputQueue.
- Όπως σε όλα τα υπόλοιπα παράθυρα και τους τελεστές έτσι και εδώ θα πρέπει να ληφθεί ιδιαίτερη μέριμνα ώστε η έξοδος να παραμένει χρονικά διατεταγμένη. Αυτή η παρατήρηση αποκλείει κάποιες σκέψεις που έγιναν για ουρά εξόδου με τις ομάδες να ακολουθούν η μία την άλλη έχοντας όλα τα στοιχεία τους συγκεντρωμένα κατά το μοτίβο xxxxyyyzrrrr.



Σχήμα 5.5: *Δομή ευρετηρίου και ουρά εξόδου για μεριστικό παράθυρο με παράμετρο N = 4*

Η λύση που δόθηκε είναι επιγραμματικά η εξής: Αφού η είσοδος είναι χρονικά διατεταγμένη και η έξοδος (ιδίου σχήματος με την είσοδο) παρουσιάζει επίσης αυτήν την απαίτηση, επιλέγεται κάθε πλειάδα εισόδου που οδηγείται σε επεξεργασία, να περνάει άμεσα στην έξοδο, ενώ μέσω μιας βοηθητικής δομής ευρετηρίου βρίσκεται ο κόμβος της outputQueue που πιθανόν να πρέπει να διαγράφεί. Διαγραφή πρέπει να προκύψει όταν σε μία ομάδα από αυτές που ορίζει το γνώρισμα ομαδοποίησης έρθει το N+1 στοιχείο της. Το στοιχείο που πρέπει να διαγραφεί είναι το παλαιότερο της συγκεκριμένης ομάδας και έτσι διατηρείται η χρονική διάταξη της εξόδου.

Με την βοήθεια των κλάσεων GroupSubQueue και GroupNode δημιουργήθηκε αυτή η δομή ευρετηρίου, για να υπάρχει η δυνατότητα ανεύρεσης στην outputQueue κάθε στοιχείου οποιασδήποτε ομάδας, γνωρίζοντας ταυτόχρονα και την χρονική διάταξη του στοιχείου αυτού εντός της ομάδας του. Κατά τον τρόπο αυτό μπορούμε να εντοπίζουμε στην ουρά εξόδου το παλαιότερο στοιχείο κάθε ομάδας, ακριβώς δηλαδή αυτό που τυχόν να χρειαστεί να διαγραφεί. Ωστόσο, για την διαγραφή ενός στοιχείου από μία ουρά δεν αρκεί η γνώση μόνο του στοιχείου αυτού, εκτός και αν το στοιχείο γνωρίζει τόσο το επόμενο όσο και το προηγούμενο του στην ουρά. Για τον λόγο αυτό επιλέχθηκε να κρατιέται σε κάθε κόμβο, ένας επιπλέον δείκτης για τον προηγούμενο κόμβο του. Αυτόν τον δείκτη δεν τον χρησιμοποιεί κανένα άλλο παράθυρο ή τελεστής και γι' αυτό στον *κατασκευαστή* των κόμβων αρχικοποιείται σε NULL. Στην υλοποίηση αυτού του παραθύρου λαμβάνεται ιδιαίτερη μέριμνα για την διατήρηση ορθών περιεχομένων σε αυτόν τον δείκτη.

Στο σχήμα 5.5 απεικονίζεται ένα παράδειγμα της δομής αυτού ευρετηρίου για παράμετρο παραθύρου N = 4. Με GSQ σημειώνονται αντικείμενα GroupSubQueue, ενώ με GN σημειώνονται αντικείμενα GroupNode. Τέλος κάθε χρώμα ή μοτίβο δηλώνει ομοιογένεια ως προς το γνώρισμα ομαδοποίησης.

Το ευρετήριο είναι μία ουρά από στοιχεία GSQ με την αρχή του στον δείκτη groupQueueHead και το πέρας του στον δείκτη groupQueueTail. Δημιουργείται και εισάγεται σε αυτήν την ουρά ένα αντικείμενο GSQ για κάθε νέα τιμή του γνωρίσματος ομαδοποίησης – στο σχήμα, κάθε διαφορετικό χρώμα / μοτίβο πλειάδας στην ουρά εξόδου. Κάθε αντικείμενο GSQ αποτελεί μία *υπο-ουρά* στοιχείων GN, μεγέθους το πολύ N. Δεν πρόκειται να προκύψει διαγραφή κάποιου στοιχείου GSQ, αφού κάτι τέτοιο δεν προβλέπεται από την σημασιολογία του παραθύρου.

Από την στιγμή που θα δημιουργηθεί μία νέα ομάδα στην ουρά εξόδου και μετά, θα διατηρούνται διαρκώς στοιχεία αυτής της ομάδας στην έξοδο.

Για κάθε κόμβο της ουράς εξόδου υπάρχει στο ευρετήριο ένα αντικείμενο GN κάτω από το αντίστοιχο αντικείμενο GSQ. Στο παράδειγμα, για λόγους απλότητας του σχήματος, η σχέση αυτή απεικονίζεται χρησιμοποιώντας διακεκομμένα βέλη μόνο για τα αντικείμενα πρασίνου χρώματος. Κάθε αντικείμενο GN περιέχει μόνο δύο δείκτες. Έναν προς το αντικείμενο κόμβου της ουράς εξόδου στον οποίο αντιστοιχεί και έναν προς το επόμενο αντικείμενο GN.

Η πορεία ενός κόμβου δεδομένων της εισόδου από την στιγμή που περνάει σαν όρισμα στην `checkNode()` έχει ως εξής:

- Αρχικά διαβάζεται η τιμή του γνωρίσματος ομαδοποίησης του κόμβου και διατρέχονται τα GSQ στοιχεία της ουράς του ευρετηρίου για να διαπιστωθεί αν υπάρχει διαθέσιμη ομάδα στην οποία θα ενταχθεί ο κόμβος αυτός. Ο έλεγχος γίνεται σε επίπεδο byte, ανεξαρτήτως τύπου του γνωρίσματος ομαδοποίησης.
- Αν δεν υπάρχει αντίστοιχο GSQ αντικείμενο, δημιουργείται ένα για την νέα τιμή του γνωρίσματος ομαδοποίησης και εντάσσεται στο τέλος της GSQ ουράς του ευρετηρίου. Αν υπάρχει σχετικό GSQ αντικείμενο τότε αυτό απλά εντοπίζεται. Σε κάθε περίπτωση λοιπόν στο επόμενο βήμα είναι γνωστή η GSQ ομάδα του ευρετηρίου στην οποία ανήκει ο νέος κόμβος.
- Ο κόμβος προστίθεται στο πέρας της ουράς εξόδου του παραθύρου.
- Δημιουργείται ένα GN αντικείμενο που δείχνει στον νέο κόμβο της ουράς εξόδου.
- Το νέο GN αντικείμενο εντάσσεται στο τέλος της αντίστοιχης GSQ υπο-ουράς αυξάνοντας κατά ένα τον μετρητή των στοιχείων της.
- Εξετάζεται αν η τιμή αυτού του μετρητή υπερβαίνει την παράμετρο *N* του παραθύρου.
- Αν ο παραπάνω έλεγχος ήταν αρνητικός η εισαγωγή ολοκληρώθηκε και η `checkNode()` τερματίζει.
- Σε αντίθετη περίπτωση εντοπίζεται από την GSQ ομάδα του νέου κόμβου το παλαιότερο στοιχείο GN μέσω του οποίου παρέχεται πρόσβαση στο στοιχείο της `outputQueue` που πρέπει να διαγραφεί. Αφαιρείται ο κόμβος από την ουρά εξόδου με την βοήθεια των δεικτών `prevPtr` και `nextPtr`, και στην συνέχεια αφαιρείται το GN αντικείμενο από την δομή του ευρετηρίου. Για την αφαίρεση αυτή ενημερώνεται και ο δείκτης προς το επόμενο GN αντικείμενο από το τελευταίο αντικείμενο GN της προηγούμενης GSQ ομάδας. Η εισαγωγή πλέον ολοκληρώθηκε και η `checkNode()` τερματίζει.

5.4.3 *Κυλιόμενα παράθυρα* (*Sliding windows*)

Για την υλοποίηση κυλιόμενων παραθύρων δημιουργήθηκε η κλάση `SlidingWindow`.Ο κατασκευαστής αυτής της κλάσης δέχεται ως ορίσματα ένα δείκτη στην ουρά εισόδου, και τρεις ακεραίους `rangeInput`, `slideInput` και `startInput`.

- Η παράμετρος ουράς παρέχεται στον κατασκευαστή της υπέρ-κλάσης `UnaryOperator` επιστρέφοντας ένα λογικό αντίγραφο της στην `inputQueue`.
- Η παράμετρος `rangeInput` καθορίζει την εμβέλεια (*range*) του παραθύρου. Αφού η χρονική διάταξη των στοιχείων διατηρείται μέσα στο σύστημα, αρκεί ο έλεγχος του πρώτου και του τελευταίου στοιχείου της ουράς εξόδου για την εξακρίβωση της χρονικής έκτασης των περιεχομένων της.
- Η παράμετρος `slideInput` καθορίζει το *βήμα προόδου* (*slide*) του παραθύρου αναφορικά με το χρονόσημο των περιεχομένων του. Αν η τιμή αυτή είναι μεγαλύτερη από την τιμή της παραμέτρου `rangeInput` το νέο παράθυρο οδηγεί σε απώλειες πλειάδων. Αυτό μπορεί να θεωρηθεί ακόμη και θεμιτό για υψηλούς ρυθμούς δεδομένων ως μία μέθοδος μείωσης του φόρτου εργασίας, με επίπτωση βέβαια στην ακρίβεια των αποτελεσμάτων. Κάτι τέτοιο πάντως δεν αποτελεί την πλέον συνήθη περίπτωση.
- Η παράμετρος `startInput` καθορίζει το χρονόσημο εκείνο έπειτα από το οποίο θα εφαρμοστεί το παράθυρο στα δεδομένα της ουράς εισόδου. Όλες οι πλειάδες με χρονόσημο μικρότερο από αυτήν την τιμή δεν φτάνουν ποτέ στην `outputQueue` και ο έλεγχος αυτός πραγματοποιείται στην μέθοδο `initializeOutput()`.

Επίσης ο *κατασκευαστής* δημιουργεί μια άδεια λογική ουρά εξόδου καλώντας τον αντίστοιχο `Queue` *κατασκευαστή* με σκοπό να αντιγράψει στην `outputQueue` την πληροφορία του σχήματος της ουράς εισόδου. Η επιλογή υλοποίησης της ουράς εξόδου ως *λογικής* δικαιολογείται από το γεγονός ότι το παράθυρο πρέπει να παρέχει πρόσβαση σε συγκεκριμένο τμήμα της ουράς εισόδου δίχως να αλλάζει τα περιεχόμενα την διάταξη ή σχήμα του αντίστοιχου κομματιού εισόδου.

Η `checkNode()` αρχικά ελέγχει αν ο κόμβος που καλείται να εξετάσει αποτελεί κόμβο δεδομένων. Αν αυτό ισχύει τότε εφαρμόζει τους αλγεβρικούς κανόνες που αναφέρθηκαν στην ενότητα 3.5.1, και ανάλογα ρυθμίζει τα περιεχόμενα της `outputQueue`. Στο τέλος της `checkNode()`, αφού ρυθμιστούν τυχόν αλλαγές που προκάλεσε η επεξεργασία του νέου κόμβου, ελέγχεται αν συντρέχει λόγος διαγραφής στοιχείων από την ουρά εξόδου, υπολογίζοντας την τρέχουσα χρονική έκταση των περιεχομένων της και συγκρίνοντας με την προκαθορισμένη *εμβέλεια* (*range*). Όποτε απαιτούνται διαγραφές, πραγματοποιούνται από την μέθοδο `deleteNodes()`, η οποία αρκείται σε μεταβολές δεικτών και στατιστικών στοιχείων της ουράς εξόδου χωρίς να επιχειρεί πραγματική καταστροφή κάποιου κόμβου.

5.4.4 *Επάλληλα παράθυρα* (*Tumbling windows*)

Για τα επάλληλα παράθυρα δεν υλοποιήθηκε κάποια ξεχωριστή κλάση. Η λειτουργία τους προσομοιώνεται με την βοήθεια της κλάσης `SlidingWindow`, με τον περιορισμό η παράμετρος `slideInput` να είναι μεγαλύτερη ή ίση από την παράμετρο `rangeInput`. Για την ακρίβεια η ισότητα αποτελεί συνήθως την επιδιωκόμενη συνθήκη, αφού μόνο τότε ένα επάλληλο παράθυρο δεν προκαλεί απώλεια πληροφορίας. Ο κύριος στόχος τους είναι να φροντίζουν έτσι ώστε διαδοχικές αλλαγές στην ουρά εξόδου τους να αποτελούν στιγμιότυπα που δεν παρουσιάζουν επικάλυψη.

5.4.5 *Παράθυρα οροσήμου* (*Landmark windows*)

Γι' αυτά τα παράθυρα δημιουργήθηκε η κλάση `LandMarkWindow`. Υποστηρίζονται μόνο παράθυρα οροσήμου με σταθερό κάτω άκρο. Ο *κατασκευαστής* αυτής της κλάσης δέχεται ως ορίσματα ένα δείκτη στην ουρά εισόδου και έναν ακέραιο `startInput`.

- Η παράμετρος ουράς παρέχεται στον κατασκευαστή της υπέρ-κλάσης `UnaryOperator` επιστρέφοντας ένα λογικό αντίγραφο της στην `inputQueue`.
- Η παράμετρος `startInput` καθορίζει το χρονόσημο εκείνο έπειτα από το οποίο θα εφαρμοστεί το παράθυρο στα δεδομένα της ουράς εισόδου. Όλες οι πλειάδες με χρονόσημο μικρότερο από αυτήν την τιμή δεν φτάνουν ποτέ στην `outputQueue` και ο έλεγχος αυτός πραγματοποιείται στην μέθοδο `initializeOutput()`.

Η `outputQueue` υλοποιήθηκε επίσης ως *λογική* ουρά ενώ η λειτουργία της `checkNode()` πραγματοποιείται απλά επισυνάπτοντας στην έξοδο όλες τις πλειάδες με χρονόσημο μεγαλύτερο ή ίσο από το κάτω άκρο του παραθύρου.

5.5 Υλοποιήσεις τελεστών

Οι τελεστές που υλοποιήθηκαν είναι οι εξής:

- *προβολή* (*projection*)
- *επιλογή* (*selection*)
- *παραθυρική σύνδεση διοχέτευσης* (*windowed pipelined join*)

Οι τελεστές projection και selection κληρονομούν από την κλάση `UnaryOperator` και κατά συνέπεια διαχειρίζονται τα δικά τους αντικείμενα `inputQueue` και `outputQueue`. Αντίθετα, ο τελεστής pipelined join έχοντας δύο ουρές εισόδου, κληρονομεί από την κλάση `BinaryOperator`.

Έτσι οι τρεις τελεστές παρουσιάζουν τις ακόλουθες ομοιότητες:

- Υλοποιούν τις μεθόδους: `checkNode()` για επεξεργασία του κόμβου που δίνεται ως παράμετρος, `operate()` για κατανάλωση κόμβων από την ουρά εισόδου και κλήση της `checkNode()`, και `printContents()` για την εκτύπωση όλων των περιεχομένων της `outputQueue` με στόχο την εξακρίβωση ορθής λειτουργίας.
- Υλοποιούν *λογικές* ουρές εισόδου. Κανένας τελεστής δεν πρέπει να επηρεάζει τα δεδομένα τα οποία καλείται να επεξεργαστεί και αυτό γιατί τα ίδια δεδομένα πιθανόν να χρησιμοποιούνται από άλλον τελεστή ή ακόμα και από άλλο ερώτημα. Έτσι, μια και η είσοδος χρησιμοποιείται μόνο για ανάγνωση κόμβων παραμένοντας άθικτη, δεν υπάρχει λόγος επανάληψης της πληροφορίας αυτής.
- Αντίθετα, η ουρά εξόδου υλοποιείται εμφανώς ως *φυσική* ουρά αφού είναι διαφορετικού σχήματος, διάταξης ή περιεχομένων σε σχέση με την ουρά εισόδου.

Τέλος, όσον αφορά την `outputQueue`, οι τελεστές πρέπει να μεριμνούν για την χρονική διάταξη των κόμβων της και την εισαγωγή κόμβων στίξης σε κατάλληλες θέσεις. Παρακάτω ακολουθεί η περιγραφή των υλοποιήσεων αυτών των τελεστών.

5.5.1 *Προβολή* (*Projection*)

Για την υλοποίηση του τελεστή προβολής δημιουργήθηκε η κλάση `Projection`. Ο *κατασκευαστής* αυτής της κλάσης δέχεται τις εξής παραμέτρους:

- ένα δείκτη στην ουρά εισόδου που τον περνάει στον κατασκευαστή της υπέρ-κλάσης `UnaryOperator` επιστρέφοντας ένα λογικό αντίγραφο της στην `inputQueue` του τελεστή.
- μία συμβολοσειρά με όνομα `bitmap` αποτελούμενη από τους χαρακτήρες 0 και 1. Το πλήθος των χαρακτήρων πρέπει να είναι κατ' ελάχιστο ίσο με το πλήθος των γνωρισμάτων κάθε πλειάδας εισόδου. Περισσότεροι χαρακτήρες επιτρέπονται, αλλά αγνοούνται. Λιγότεροι χαρακτήρες ή διαφορετικοί από 0 και 1 δημιουργούν πρόβλημα. Υποτίθεται πάντοτε ότι ο κατασκευαστής καλείται με το `bitmap` όρισμα του στην σωστή μορφή.

Ο *κατασκευαστής* δημιουργεί έναν πίνακα από Boolean τιμές μεγέθους ίσου με το πλήθος των γνωρισμάτων των πλειάδων της ουρών εισόδου. Χρησιμοποιώντας το `bitmap` όρισμα του ανά χαρακτήρα, αρχικοποιεί τις αντίστοιχες τιμές του πίνακα σε TRUE για χαρακτήρα 1 ή FALSE για χαρακτήρα 0. Έτσι ο τελεστής διατηρεί με τον πίνακα αυτό μία *μάσκα* (*mask*) την οποία θα χρησιμοποιεί τόσο για την δημιουργία του σχήματος της `outputQueue` όσο και για την επιλογή των byte από κάθε κόμβο εισόδου που θα αποτελέσουν μέρος στον αντίστοιχο κόμβο της ουράς εξόδου. Για την δημιουργία ουράς εξόδου κατάλληλου σχήματος χρησιμοποιείται η μάσκα σε συνδυασμό με την πληροφορία σχήματος της ουράς εισόδου. Στο στάδιο αυτό δημιουργείται μία συμβολοσειρά αντίστοιχη με την πρώτη γραμμή που συναντάμε σε κάθε ASCII αρχείο εισόδου που προσομοιώνει κάποιο ρεύμα δεδομένων (εικόνα 5.2). Στην συνέχεια καλείται ο αντίστοιχος *κατασκευαστής* για την δημιουργία κενής `outputQueue` με το επιθυμητό σχήμα.

Η μέθοδος `operate()` διαβάζει όλες τις νέες πλειάδες που προστέθηκαν στην ουρά εισόδου καλώντας για καθεμία από αυτές την συνάρτηση `checkNode()`. Αξίζει να τονιστεί μία σημαντική διαφοροποίηση από την λειτουργία της μεθόδου αυτής στις παραθυρικές δομές. Πλέον σε κάθε κύκλο εργασίας της `operate()` καταναλώνονται όλα τα νέα στοιχεία της ουράς εισόδου, ενώ τα παράθυρα σε ανάλογη περίπτωση καταναλώνουν μία το πολύ πλειάδα από την εισερχόμενη ούρα. Αυτός ο τρόπος λειτουργίας της `operate()` των παραθύρων έχει σαν στόχο να διασφαλιστεί ότι δεν θα παραλειφθούν στοιχεία της ουράς εισόδου από την έξοδο των παραθύρων. Αντίθετα οι τελεστές δεν έχουν να μεριμνήσουν για κάτι τέτοιο αφού δεν διαγράφουν αποτελέσματα από την έξοδό τους.

Τέλος για την μέθοδο `checkNode()` διακρίνονται οι παρακάτω περιπτώσεις ανάλογα με το είδος του κόμβου που δέχεται σαν όρισμα:

- Αν πρόκειται για κόμβο στίξεως, τότε καλείται η μέθοδος `insertPunctuation()` της `outputQueue` με όρισμα την τιμή χρονοσήμου του κόμβου εισόδου. Εισάγεται κατά τον τρόπο αυτό ένας κόμβος στίξης στην ουρά εξόδου και η `checkNode()` τερματίζει.

- Αν πρόκειται για κόμβο δεδομένων πρέπει να δημιουργηθεί νέος κατάλληλος κόμβος στην ουρά εξόδου. Με την βοήθεια της μάσκας αποθηκεύονται προσωρινά τα bytes των κατάλληλων ιδιοτήτων από κάθε πλειάδα και μεταβιβάζονται μαζί με το χρονόσημο της πλειάδας σαν ορίσματα στην μέθοδο `insertNodeBytes ()` της `outputQueue`.Ένας νέος κόμβος προστίθεται στην `outputQueue` και η `checkNode()` τερματίζει.

5.5.2 *Επιλογή* (*Selection*)

Ο τελεστής selection για ένα ερώτημα, υλοποιείται ως αντικείμενο της κλάσης selection. Ο *κατασκευαστής* αυτής της κλάσης δέχεται τις εξής παραμέτρους:

- ένα δείκτη στην ουρά εισόδου που τον περνάει στον κατασκευαστή της υπερ-κλάσης `UnaryOperator` επιστρέφοντας ένα λογικό αντίγραφο στην `inputQueue` του τελεστή.
- μία συμβολοσειρά με όνομα `operation` με την οποία θα γίνει επιλογή της επιθυμητής σύγκρισης, ανάμεσα από τις τιμές {`=`, `<>`, `>`, `>=`, `<`, `<=`}. Αν το όρισμα `operation` δεν αντιστοιχεί σε κάποια από αυτές τις τιμές, τότε δεν επιλέγεται καμία λειτουργία του τελεστή και απλώς δεν διέρχεται καμία πλειάδα στο ρεύμα εξόδου.
- μία **συμβολοσειρά** με όνομα `value1`, η οποία θα περιέχει την τιμή με την οποία επιθυμείται σύγκριση κάποιου γνωρίσματος των πλειάδων του εισερχόμενου ρεύματος. Αν επιθυμείται σύγκριση μεταξύ δύο γνωρισμάτων τότε αυτή η παράμετρος μπορεί να έχει οποιαδήποτε τιμή αφού στην συνέχεια θα αγνοηθεί. Τονίζεται ότι ακόμα και αν η τιμή που θα συγκριθεί είναι π.χ. ακέραιος το όρισμα αυτό πρέπει περαστεί ως **συμβολοσειρά**.
- έναν ακέραιο με όνομα `index1` που δηλώνει το *γνώρισμα* των πλειάδων εισόδου με το οποίο θα γίνεται η σύγκριση είτε με την παράμετρο `value1` είτε με κάποιο άλλο *γνώρισμα* της ίδιας πλειάδας. Αυτό αποσαφηνίζεται με την τιμή της επόμενης παραμέτρου. Επιτρεπτές τιμές είναι οι ακέραιοι από [0 .. (πλήθος ιδιοτήτων ρεύματος εισόδου -1)].
- έναν ακέραιο με όνομα `index2` που δηλώνει το *γνώρισμα* των πλειάδων εισόδου το οποίο συγκρίνεται με το *γνώρισμα* της ίδιας πλειάδας που ορίζει ο ακέραιος `index1`. Η παράμετρος αυτή παίρνει εξ' ορισμού τιμή ίση με -1 εκτός και αν δηλωθεί κάτι διαφορετικό, ενώ έχει το ίδιο εύρος τιμών με την παράμετρο `index1`.

Ο *κατασκευαστής* δημιουργεί μία κενή φυσική ουρά εξόδου ιδίου σχήματος με την ουρά εισόδου. Και στον τελεστή αυτό η μέθοδος `operate()` σε κάθε κύκλο εργασίας της καταναλώνει όλες τις νέες πλειάδες που προστέθηκαν στην ουρά εισόδου καλώντας για καθεμία από αυτές την μέθοδο `checkNode()`. Τέλος για την μέθοδο `checkNode()` διακρίνονται οι παρακάτω περιπτώσεις ανάλογα με το είδος του κόμβου που δέχεται σαν όρισμα:

- Αν πρόκειται για κόμβο στίξεως τότε καλείται η μέθοδος `insertPunctuation()` της `outputQueue` με όρισμα την τιμή χρονοσήμου του κόμβου εισόδου. Εισάγεται λοιπόν ένας κόμβος στίξης στην ουρά εξόδου και η `checkNode()` τερματίζει.
- Αν πρόκειται για κόμβο δεδομένων, τότε ανάλογα με τον τύπο του γνωρίσματος που ορίζει η παράμετρος `index1` αποθηκεύεται η τιμή του ως προσωρινό πεδίο ακεραίου ή δεκαδικού ή πίνακα χαρακτήρων. Ακολούθως εξετάζεται η τιμή της παραμέτρου `index2`. Αν η τιμή αυτή είναι -1 τότε η ζητούμενη σύγκριση διενεργείται μεταξύ του *γνωρίσματος* `index1` και της τιμής `value1`. Αν ωστόσο ο `index2` περιέχει τιμή έγκυρη τότε πρέπει να γίνει σύγκριση μεταξύ των γνωρισμάτων `index1` και `index2`. Σε κάθε περίπτωση και ανάλογα με τον τύπο που καθορίζεται από τον `index1` (ίσως και τον `index2`), οι συγκρινόμενες τιμές αποθηκεύονται προσωρινά σε δύο δεκαδικούς ή σε δύο πίνακες χαρακτήρων. Ανάλογα με την επιλεγμένη λειτουργία του τελεστή γίνεται σύγκριση των δύο αυτών πεδίων και αποφασίζεται αν πρέπει ή όχι να περάσει η πλειάδα αυτή στην ουρά εξόδου. Αν χρειαστεί προσθήκη στην φυσική ουρά εξόδου, τότε αυτή πραγματοποιείται με την μέθοδο `insertNodeBytes ()` της `outputQueue` και ορίσματα τα bytes και το χρονόσημο του κόμβου που εξετάστηκε.

Αξίζει να τονιστεί ότι με την προαγωγή ενός ακεραίου (*int*) σε δεκαδικό (*double*) που πραγματοποιείται πριν την απόφαση προσθήκης του κόμβου στην έξοδο, υποστηρίζονται σωστές

συγκρίσεις μεταξύ πεδίων τόσο του ιδίου τύπου όσο και συγκρίσεις μεταξύ ακεραίων και δεκαδικών. Δεν υποστηρίζονται συγκρίσεις μεταξύ ενός γνωρίσματος που δηλώθηκε με τύπο int ή double και άλλου γνωρίσματος που δηλώθηκε με τύπο charn ή char.

5.5.3 *Παραθυρική σύνδεση διοχέτευσης* (*Windowed pipelined join*)

Για την υλοποίηση του τελεστή σύνδεσης διοχέτευσης δημιουργήθηκε η κλάση `PipelinedJoin`. Η σύνδεση διενεργείται βάσει ενός γνωρίσματος από κάθε εισερχόμενο ρεύμα. Ο τελεστής αυτός καλείται να επεξεργάζεται μόνο τα νέα δεδομένα από κάθε ουρά εισόδου του, προσπαθώντας να τα συνδέσει με πλειάδες που περιέχονται στο νέο στιγμιότυπο της άλλης ουράς εισόδου. Ο *κατασκευαστής* αυτής της κλάσης δέχεται τις εξής παραμέτρους:

- δύο δείκτες για τις ουρές εισόδου του, τους οποίους περνάει στον κατασκευαστή της υπερ-κλάσης `BinaryOperator` ο οποίος επιστρέφει λογικά αντίγραφά τους στις `inputQueue1` και `inputQueue2` του τελεστή. Για τις ουρές αυτές ρυθμίζονται οι δείκτες head, tail και τα υπόλοιπα πεδία των αντιγράφων έτσι ώστε να αποτυπώνουν τα πιο πρόσφατα στιγμιότυπα των ουρών εισόδου του τελεστή. Μέσα στην υπέρ-κλάση αρχικοποιούνται και οι δείκτες `workingNode1` και `workingNode2` ώστε να δείχνουν στην αρχή των αντίστοιχων ουρών. Αυτοί οι δείκτες χρησιμοποιούνται από τον τελεστή, για να σημαδεύει σε κάθε ουρά τον κόμβο στον οποίο φτάνει μέχρι στιγμής η επεξεργασία.
- έναν ακέραιο με όνομα `jAttr1` που δηλώνει το *γνώρισμα* της ουράς εισόδου `inputQueue1` που θα χρησιμοποιηθεί για την λειτουργία της σύνδεσης.
- έναν ακέραιο με όνομα `jAttr2` που δηλώνει το *γνώρισμα* της ουράς εισόδου `inputQueue2` που θα χρησιμοποιηθεί για την λειτουργία της σύνδεσης.

Για την δημιουργία ουράς εξόδου κατάλληλου σχήματος δημιουργείται μία συμβολοσειρά αντίστοιχη με την πρώτη γραμμή κάθε ASCII αρχείου εισόδου για την προσομοίωση κάποιου ρεύματος δεδομένων (εικόνα 5.2). Για τον σκοπό αυτό χρησιμοποιείται η πληροφορία σχήματος των δύο ουρών εισόδου με αποτέλεσμα μία συμβολοσειρά που περιέχει το σχήμα όλων των γνωρισμάτων της πρώτης ουράς, ακολουθούμενο από το σχήμα όλων των ιδιοτήτων της δεύτερης ουράς. Στην συνέχεια καλείται ο αντίστοιχος *κατασκευαστής* με όρισμα την συμβολοσειρά αυτή και τον χαρακτήρα που χρησιμοποιήθηκε για *διαχωρισμό των γνωρισμάτων* (*delimiter character*), με αποτέλεσμα την δημιουργία κενής φυσικής ουράς `outputQueue` του επιθυμητού σχήματος.

Για πρακτικούς λόγους, η υλοποίηση του τελεστή σύνδεσης διοχέτευσης που παρέχεται εφαρμόζεται μόνο πάνω στις εξόδους παραθύρων πλειάδων. Με βάση αυτόν τον περιορισμό περιγράφονται στην συνέχεια οι μέθοδοι `operate()` και `checkNode()` του τελεστή.

Η μέθοδος `operate()` ακολουθεί τα εξής βήματα:

- Αρχικά ενημερώνει τα στιγμιότυπα των ουρών εισόδου και στην συνέχεια ανιχνεύει με την βοήθεια των πεδίων `workingNode1` και `workingNode2` τυχόν προσθήκη νέου κόμβου στις ουρές. Με βάση την παραδοχή ότι η είσοδος προέρχεται από παράθυρα πλειάδων, θεωρείται δεδομένο ότι αν υπήρξε μεταβολή σε κάποια από τις ουρές εισόδου τότε αυτή θα ήταν προσθήκη ενός μόνο κόμβου (με την αντίστοιχη βέβαια διαγραφή από την αρχή της ουράς). Αυτός ακριβώς ο κόμβος από κάθε ουρά είναι που πρέπει να περαστεί στην `checkNode()` για να επιχειρηθεί η σύνδεσή του με το πλέον ενημερωμένο στιγμιότυπο της άλλης ουράς.
- Έτσι αν υπήρξε αλλαγή στην `inputQueue1` ο δείκτης `workingNode1` τίθεται στον νέο κόμβο, η μεταβλητή `whichInput_1Or2` τίθεται σε 1 σηματοδοτώντας άφιξη κόμβου από την `inputQueue1` και καλείται η `checkNode()` με όρισμα τον `workingNode1`. Αντίστοιχη διαδικασία ακολουθείται και για τυχόν αλλαγή στην `inputQueue2`.
- Στο τελευταίο βήμα της `operate()` και αφού έχουν πραγματοποιηθεί οι τυχόν αλλαγές στην έξοδο του τελεστή αποφασίζεται αν πρέπει να προστεθεί σε αυτήν κόμβος στίξης. Η απόφαση αυτή λαμβάνεται με χρήση της πληροφορίας που κρατά ο τελεστής για το χρονόσημο τόσο του τελευταίου κόμβου στίξης που εξέπεμψε (`lastOutputPunctuation`) όσο και του τελευταίου κόμβου στίξης που έλαβε από κάθε ουρά εισόδου (`lastPunctuation1` και

lastPunctuation2). Η ορθή συντήρηση της πληροφορίας αυτής για τις εισόδους ανατίθεται στην checkNode() ενώ η operate() συντηρεί την σωστή τιμή για την έξοδο.

Η checkNode() αρχικά εξετάζει αν ο κόμβος που δέχτηκε είναι κόμβος στίξης. Αν αυτό ισχύει, τότε ανάλογα με την τιμή της μεταβλητής whichInput_1Or2 ενημερώνει είτε την τιμή lastPunctuation1 είτε την τιμή lastPunctuation2 και ακολούθως τερματίζει. Αν δέχτηκε ως όρισμα έναν κόμβο δεδομένων, εξετάζει κάθε κόμβο από το πλέον ενημερωμένο στιγμιότυπο της άλλης ουράς εισόδου για ισότητα ως προς το αντίστοιχο γνώρισμα σύνδεσης (jAttr1 ή jAttr2). Αν υπάρχει ισότητα προστίθεται στην ουρά εξόδου του τελεστή η συνένωση των δύο κόμβων αντιγράφοντας τα bytes. Το χρονόσημο του νέου κόμβου προκύπτει ως το **μικρότερο** από τα χρονόσημα των πιο πρόσφατων στοιχείων κάθε ουράς. Αυτός ο τρόπος ορισμού του χρονοσήμου έχει στόχο την διασφάλιση της χρονικής διάταξης της εξόδου αλλά και της ορθής σήμανσης των τοποθετούμενων κόμβων στίξης. Εξετάζεται και εδώ η περίπτωση ισότητας ενός ακεραίου και ενός δεκαδικού με την προαγωγή των ακεραίων σε δεκαδικούς πριν από την σύγκριση.

Ο βασικός λόγος για τον οποίο οι είσοδοι αυτού του τελεστή περιορίστηκαν να συνδέονται σε παράθυρα πλειάδων, έχει να κάνει με το γεγονός ότι γνωρίζουμε εκ των προτέρων την επίπτωση στην ουρά εξόδου που επιφέρει η επεξεργασία μίας πλειάδας από την είσοδο. Με κάθε εισερχόμενη πλειάδα προστίθεται ακριβώς ένα νέο στοιχείο στο τέλος της ουράς εξόδου, ενώ πιθανόν να αφαιρείται το πολύ ένας κόμβος από την αρχή αυτής. Αντίθετα, τα κυλιόμενα ή τα επάλληλα παράθυρα δεν προσφέρουν αυτήν την γνώση. Η είσοδος μίας νέας πλειάδας μπορεί να προκαλέσει απρόβλεπτη μεταβολή στο πλήθος των στοιχείων της ουράς εξόδου ανάλογα με τις εκάστοτε παραμέτρους. Για τα παράθυρα οροσήμου που υλοποιήθηκαν, δεν έχει νόημα η τροφοδότηση ενός τελεστή σύνδεσης με τέτοια παράθυρα, αφού δεν περιορίζουν σημαντικά το πλήθος των στοιχείων.

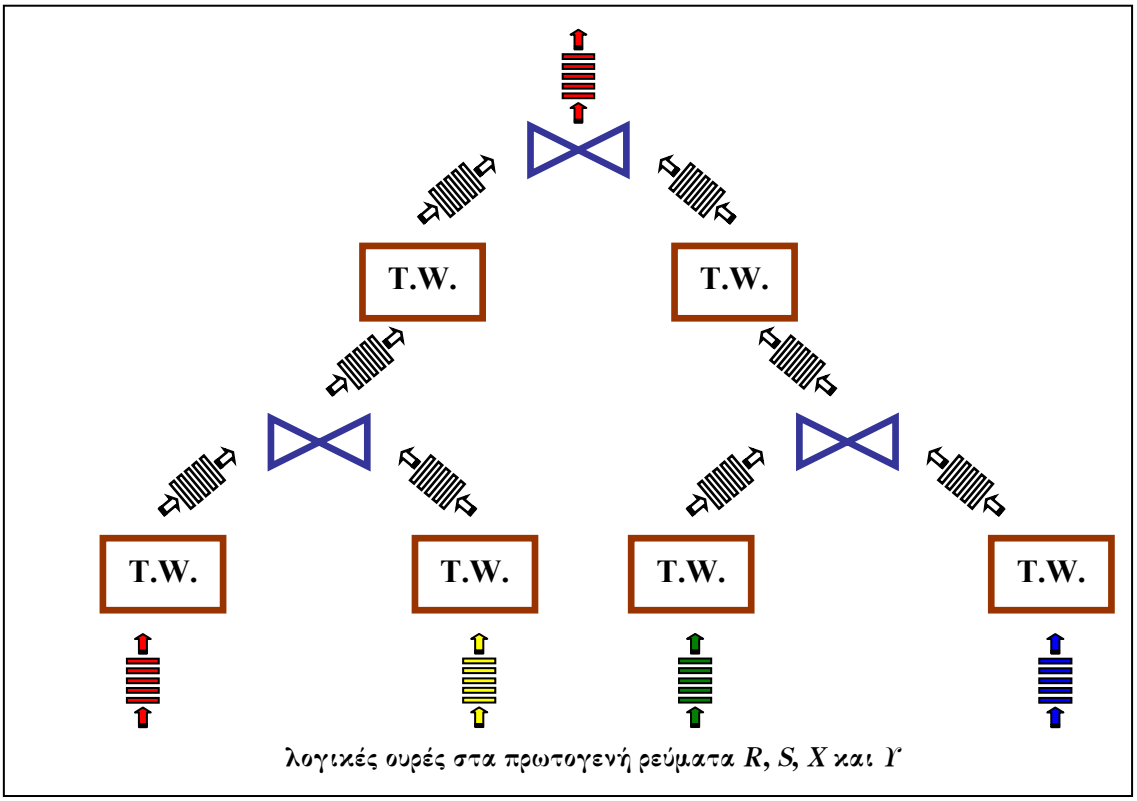
Το πρόβλημα λοιπόν που δημιουργείται με τα κυλιόμενα παράθυρα, είναι ότι δεν μπορούν να προσδιοριστούν τα ζευγάρια κόμβων μεταξύ των οποίων θα επιχειρηθεί η σύνδεση. Ακόμη, σε κάθε νέα κλήση της operate του τελεστή, υπάρχει άγνοια για την θέση των workingNode1 και workingNode2 συγκριτικά με το νέο στιγμιότυπο της εισόδου. Αν κάποιος από αυτούς τους δείκτες δείχνει σε κόμβο προγενέστερο της αρχής του νέου στιγμιοτύπου της αντίστοιχης ουράς εισόδου, τότε θα πρέπει να κινηθεί μέχρι την αρχή του νέου στιγμιοτύπου αγνοώντας τους ενδιάμεσους κόμβους. Στην περίπτωση όπου τα παράθυρα δημιουργούν επικάλυψη, τότε οι δύο αυτοί δείκτες θα δείχνουν σε κάποιο ενδιάμεσο κόμβο του νέου στιγμιοτύπου της αντίστοιχης ουράς εισόδου. Και πάλι όμως, δεν μπορεί να υπολογιστεί με σαφήνεια η σχετική τους θέση ως προς το τέλος της εκάστοτε ουράς, με αποτέλεσμα να μην μπορούν να προσδιοριστούν τα νέα ζεύγη κόμβων για τα οποία πρέπει να ελεγχθεί η συνθήκη σύνδεσης. Η εφαρμογή κυλιόμενων ή επάλληλων παραθύρων στις εισόδους του τελεστή σύνδεσης αποτελεί αντικείμενο μελλοντικής εργασίας και παρουσιάζει αρκετές ιδιαιτερότητες. Ωστόσο ο τελεστής παραθυρικής σύνδεσης διοχέτευσης στην παρούσα υλοποίηση του μπορεί να χρησιμοποιηθεί ως δομικό στοιχείο για την κατασκευή τελεστών *σύνδεσης πολλαπλών ρευμάτων* (*multiway join*) – σχήμα 5.6.

5.6 Σχηματισμός ερωτημάτων – κλάση Query

Στην κλάση αυτή δημιουργούνται τα φυσικά προσχέδια των επιθυμητών ερωτημάτων, διασυνδέοντας κατάλληλα τους διάφορους τελεστές και τις πηγές των δεδομένων. Για λόγους απλότητας σχηματίστηκαν ορισμένα χαρακτηριστικά ερωτήματα διαρκείας, τα οποία μπορούν να επιλεχθούν για εκτέλεση.

Ο *κατασκευαστής* αυτής της κλάσης στην παρούσα μορφή της δέχεται τα εξής ορίσματα:

- Τον ακέραιο selectQuery για την επιλογή ενός εκ των προκατασκευασμένων ερωτημάτων που παρέχονται και την αρχικοποίηση της μεταβλητής which κάθε αντικειμένου αυτής της κλάσης.
- Δύο παραμέτρους για το τις φυσικές ουρές των διαθέσιμων ρευμάτων δεδομένων. Στην εφαρμογή που έγινε θεωρήσαμε μόνο δύο διαφορετικά ρεύματα. Πρόσθετες παράμετροι είναι δυνατόν να εισαχθούν αν χρειαστεί προσομοίωση περισσότερων ρευμάτων.

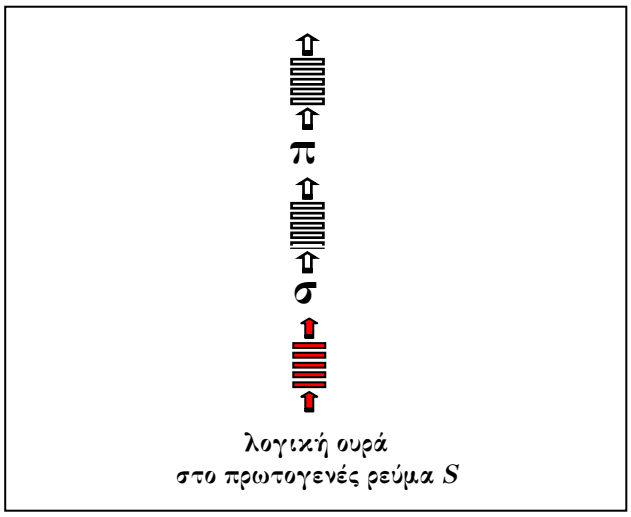


Σχήμα 5.6: *Σύνθεση τελεστή σύνδεσης πολλαπλών ρευμάτων, βάσει της διαθέσιμης υλοποίησης του τελεστή σύνδεσης διοχέτευσης*

Στον κατασκευαστή αυτής της κλάσης αρχικά δημιουργούνται λογικά αντίγραφα ουρών πάνω στις φυσικές ουρές που περάστηκαν ως παράμετροι. Στην συνέχεια γίνεται επιλογή ερωτήματος μέσω μίας `switch` εντολής πάνω στην `which` μεταβλητή. Σε καθεμία από τις διαθέσιμες επιλογές δημιουργείται το φυσικό προσχέδιο εκτέλεσης κάθε ερωτήματος συνδέοντας κατάλληλα τις ουρές εισόδου και εξόδου των τελεστών και των παραθύρων. Υπενθυμίζεται ότι δεν αποτέλεσε τμήμα του αρχικού σχεδιασμού η δυνατότητα κατασκευής προσχεδίων εκτέλεσης βασισμένα σε διατύπωση ερωτημάτων SQL μορφής.

Κάθε αντικείμενο της κλάσης, διαθέτει μία μέθοδο `runQuery` η οποία χρησιμοποιώντας την τιμή της μεταβλητής `which` μπαίνει σε έναν ατέρμονα κυκλικό βρόχο. Σε κάθε κύκλο του βρόχου αρχικά γίνεται εισαγωγή μίας πλειάδας (αν υπάρχει διαθέσιμη από τα πρωτογενή ρεύματα) για κάθε μία από τις λογικές ουρές που αναφέρονται σε αυτά. Στην συνέχεια καλείται με συγκεκριμένη σειρά η μέθοδος operate των τελεστών και των παραθύρων που σχημάτισαν το ερώτημα. Να σημειωθεί ότι κάθε ερώτημα καταναλώνει τα διαθέσιμα δεδομένα των πρωτογενών ρευμάτων με τον δικό του ρυθμό γεμίζοντας τις λογικές ουρές που διαθέτει πάνω σε αυτά.

Στην συνέχεια παρουσιάζονται τα προσχέδια εκτέλεσης των πρότυπων ερωτημάτων διαρκείας που δημιουργήθηκαν και μπορούν να επιλεχθούν από τον χρήστη. Από τα ερωτήματα αυτά, είναι εμφανής η απουσία των κυλιόμενων και των επάλληλων παραθύρων. Αυτό συμβαίνει γιατί ο μοναδικός τελεστής που υλοποιήθηκε και απαιτεί την χρήση των παραθύρων είναι ο τελεστής σύνδεσης διοχέτευσης. Ωστόσο, για τους λόγους που αναφέρθηκαν στην ενότητα 5.5.3, δεν υποστηρίζεται μέχρι στιγμής η τροφοδότηση του από αυτούς τους τύπους παραθύρων. Οι διαθέσιμες υλοποιήσεις αυτών των παραθύρων δοκιμάστηκαν με την απλή εφαρμογή τους πάνω σε ένα ρεύμα δεδομένων και την διαρκή παρακολούθηση της εξόδου τους. Η συμπεριφορά τους ήταν η αναμενόμενη, αποδεικνύοντας την ορθότητα της υλοποίησης τους. Επισημαίνεται ότι τα παραδείγματα θα ήταν περισσότερο ρεαλιστικά αν υπήρχε διαθέσιμη κάποια υλοποίηση συναθροιστικού τελεστή.



Σχήμα 5.7: *Προσχέδιο εκτέλεσης ερωτήματος επιλογής – προβολής*

5.6.1 Δοκιμαστικά δεδομένα

Για την διατύπωση των ερωτημάτων χρησιμοποιούνται τα ακόλουθα δεδομένα:

- Ένα ρεύμα δεδομένων *S* που αντιπροσωπεύει τις μετρήσεις από αισθητήρες θερμοκρασίας τοποθετημένους στις μεγαλύτερες πόλεις του Ελλαδικού χώρου. Το σχήμα αυτού του ρεύματος ορίζεται ως: < τ, sensorId, city, temperature > όπου τ το χρονόσημο που αναπαριστά την ώρα που πραγματοποιήθηκε η κάθε μέτρηση, sensorId ένας αύξων ακέραιος που προσδιορίζει μοναδικά κάθε αισθητήρα, city το όνομα της πόλης και temperature η μέτρηση του αισθητήρα.
- Κατ' αντίστοιχο τρόπο ορίζεται ένα ρεύμα δεδομένων *R* για τις μετρήσεις αισθητήρων υγρασίας σε κάθε πόλη. Το σχήμα αυτή τη φορά ορίζεται ως < τ, sensorId, city, humidity, > όπου το γνώρισμα humidity αντιπροσωπεύει την μετρούμενη υγρασία.

Για λόγους απλότητας και έλλειψης συναθροιστικών τελεστών θα θεωρηθεί ότι σε κάθε πόλη υπάρχει εγκατεστημένος ένας μόνο αισθητήρας θερμοκρασίας και ένας μόνο αισθητήρας υγρασίας.

5.6.2 Πρότυπο ερώτημα επιλογής – προβολής

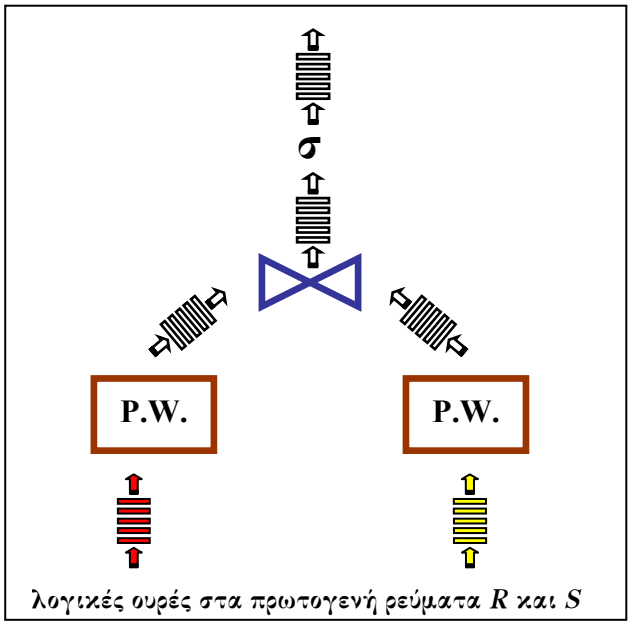
Ζητείται η διαρκής παρακολούθηση της θερμοκρασίας στην Αθήνα.

Το επιλεγμένο προσχέδιο εκτέλεσης απεικονίζεται στο σχήμα 5.7, ενώ χρησιμοποιώντας την σύνταξη της «ψευδό–SQL» που παρουσιάστηκε στην ενότητα 3.6 το ερώτημα αυτό θα διατυπωνόταν ως εξής:

```
SELECT τ,temperature
FROM   S
WHERE  city = ΑΘΗΝΑ
```

Το απλό αυτό ερώτημα διαρκείας αντιστοιχεί στις περιπτώσεις όπου δεν χρειάζεται η εφαρμογή κάποιου παραθύρου πάνω στα εισερχόμενα ρεύματα. Αυτό συμβαίνει επειδή δεν χρησιμοποιούνται αναστολτικοί τελεστές ή τελεστές διατήρησης κατάστασης οι οποίοι απαιτούν παράθυρα για την τροφοδότηση τους. Αντίθετα χρησιμοποιούνται μόνο τελεστές προβολής και επιλογής οι οποίοι μπορούν πολύ εύκολα να επεξεργάζονται ξεχωριστά κάθε πλειάδα που φτάνει σε αυτούς.

Σημειώνεται, ότι η απουσία ορισμού κάποιου παραθύρου πάνω στο ρεύμα *S* εκλαμβάνεται από το σύστημα ως *S* [ROWS UNBOUNDED] ή ισοδύναμα *S* [RANGE UNBOUNDED].



Σχήμα 5.8: *Προσχέδιο εκτέλεσης ερωτήματος απλής σύνδεσης*

5.6.3 Πρότυπο ερώτημα απλής σύνδεσης.

Ζητείται η διαρκής παρακολούθηση των μετρήσεων θερμοκρασίας και υγρασίας της Αθήνας με βάση τις 20 τελευταίες μετρήσεις από την πόλη αυτή.

Το ερώτημα αυτό σύμφωνα με την «ψευδό-SQL» της ενότητας 3.6 διατυπώνεται ως εξής:

```
SELECT τ,temperature, humidity
FROM   S [PARTITION BY city ROWS 20] JOIN
       R [PARTITION BY city ROWS 20] ON S.city = R.city
WHERE  city = ΑΘΗΝΑ
```

Από τα εναλλακτικά προσχέδια εκτέλεσης αυτού του ερωτήματος επιλέχθηκε το μάλλον πιο σύνθετο του σχήματος 5.8 για να επιδειχθεί ο τρόπος χρήσης του μεριστικού παραθύρου. Κάθε μεριστικό παράθυρο P.W. αποσπά τις τελευταίες 20 μετρήσεις από κάθε πόλη και με αυτές τροφοδοτεί τον τελεστή σύνδεσης. Εκεί επιχειρείται σύνδεση πάνω στο γνώρισμα της πόλης παράγοντας το ζητούμενο αποτέλεσμα για κάθε πόλη. Στην συνέχεια με έναν τελεστή επιλογής απομονώνονται τα αποτελέσματα της πόλης ενδιαφέροντος.

Αν και το επιλεγμένο προσχέδιο δεν φαίνεται να αποτελεί την βέλτιστη επιλογή θα πρέπει κανείς να γνωρίζει και τα υπόλοιπα ενεργά ερωτήματα για να αποφανθεί με βεβαιότητα. Έτσι αν περιλαμβάνεται πλήθος αντίστοιχων ερωτημάτων για διαφορετικές όμως πόλεις, το συγκεκριμένο προσχέδιο μάλλον αποτελεί την βέλτιστη επιλογή. Με αυτό το προσχέδιο μπορεί να επιτευχθεί η απάντηση όλων των ερωτημάτων, χρησιμοποιώντας έναν μόνο τελεστή σύνδεσης κοινό για όλα τα ερωτήματα και έναν ξεχωριστό τελεστή επιλογής για καθένα από αυτά.

5.7 Λειτουργία συστήματος

Στην ενότητα αυτή αναφέρονται κάποια γενικά στοιχεία της συνολικής λειτουργίας του συστήματος που βοηθούν στην κατανόηση του τρόπου με τον οποίο συνδέονται οι προαναφερθείσες δομές:

- Αρχικά δημιουργούνται αντικείμενα Scanner των οποίων η μέθοδος **`scan()`** θα χρησιμοποιηθεί για προσομοίωση ρευμάτων δεδομένων σε ποικίλους ρυθμούς εισόδου.
- Τα ρεύματα αποθηκεύονται σε φυσικές ουρές προσομοιώνοντας έτσι τον χώρο ενδιάμεσης αποθήκευσης των ρευμάτων (σχήμα 5.1)
- Κατασκευάζονται τα ερωτήματα διαρκείας δημιουργώντας αντικείμενα της κλάσης Query.
- Δημιουργείται ξεχωριστή διεργασία για κάθε συνάρτηση **`scan()`** και **`runQuery()`** και το λειτουργικό σύστημα αναλαμβάνει να αποφασίσει για την χρονοδρομολόγηση τους.
- Το σύστημα στην παρούσα έκδοση του ακολουθεί την τεχνική της διοχέτευσης, αφού κάθε ερώτημα υπολογίζεται κατ' αυτόν τον τρόπο. Διαπιστώνεται αν υπάρχει διαθέσιμο στοιχείο από κάθε ουρά εισόδου και ακολούθως παρέχεται σε κάθε παράθυρο ή τελεστή η δυνατότητα ανανέωσης των δεδομένων εξόδου του. Κάθε τελεστής λειτουργεί μόνο πάνω στα νέα δεδομένα που προστέθηκαν στις εισόδους του ως απόρροια των νέων στοιχείων των ουρών που προωθούνται σε κάθε ερώτημα.
- Υπάρχει συνεχής ροή κόμβων στίξεων στο σύστημα σαν να επρόκειτο για κανονικές πλειάδες. Οι κόμβοι στίξεως εισήχθησαν με σκοπό να σηματοδοτηθεί η δυνατότητα διαγραφής στοιχείων από τις φυσικές ουρές κάθε ερωτήματος (έξοδοι τελεστών ή μεριστικού παραθύρου) όταν στην ρίζα του προσχεδίου εκτέλεσης φτάνει ένας τέτοιος κόμβος στίξεως. Τότε θα υπάρχει δυνατότητα από όλες τις φυσικές ουρές να διαγραφούν τα στοιχεία με το χρονόσημο αυτό.
- Για τα πρωτογενή δεδομένα θα είναι εφικτό να διαγραφούν τα περιεχόμενα των ουρών με χρονόσημο μικρότερο ή ίσο από την ελάχιστη τιμή του χρονοσήμου μεταξύ των κόμβων στίξεων που έχουν φτάσει στην έξοδο των ερωτημάτων.

Η ορθότητα λειτουργίας της υπάρχουσας έκδοσης δοκιμάστηκε πάνω σε διαθέσιμα σύνολα δεδομένων, μελετώντας σε κάθε βήμα τα περιεχόμενα της εξόδου των τελεστών που συνθέτουν τα ερωτήματα. Κατά τις δοκιμές εκτέλεσης των πρότυπων ερωτημάτων για ποικίλους ρυθμούς εισόδου των ρευμάτων, παρατηρήθηκαν φαινόμενα προσεγγιστικών απαντήσεων τα οποία αποδίδονται σε δύο κυρίως λόγους:

i) αδυναμία γρήγορης κατανάλωσης των εισερχόμενων πλειάδων με συνέπεια την εξάντληση της διαθέσιμης μνήμης και
ii) έλλειψη συγχρονισμού κατά την σύνδεση ρευμάτων με διαφορετικό ρυθμό. Το πρόβλημα αυτό συνήθως επιφέρει λανθασμένες επιλογές στα ζεύγη για τα οποία επιχειρείται η σύνδεση, οδηγώντας σε εσφαλμένα αποτελέσματα.

Κεφάλαιο 6

Επίλογος

6.1 Συμπεράσματα – Ανασκόπηση

Επιδίωξη της παρούσας διπλωματικής εργασία ήταν η ανάδειξη της σημασίας των παραθύρων στις εφαρμογές ρευμάτων δεδομένων, φανερώνοντας πόσο επιτακτική είναι η χρήση τους σε ερωτήματα διαρκείας που χρησιμοποιούν βασικούς τελεστές της σχεσιακής άλγεβρας. Το επίπεδο προσέγγισης που πιθανόν να επιφέρουν στις επιστρεφόμενες απαντήσεις είναι καλά προσδιορισμένο από την σημασιολογία των παραθύρων και άμεσα αντιληπτό από τους χρήστες. Πρέπει ωστόσο να σημειωθεί ότι στην πλειοψηφία των εφαρμογών ρευμάτων δεδομένων η επιδιωκόμενη επεξεργασία επικεντρώνεται στην πιο πρόσφατη πληροφορία. Η εφαρμογή των παραθύρων σε αυτήν την περίπτωση έρχεται περισσότερο να εκπληρώσει αυτήν την απαίτηση παρά να επηρεάσει την ακρίβεια των απαντήσεων.

Στα πλαίσια της θεωρητικής πτυχής της εργασίας επιχειρήθηκε η θεμελίωση, η κατηγοριοποίηση και ο αλγεβρικός προσδιορισμός της σημασιολογίας των παραθύρων. Αναλύθηκαν με ομοιογενή τρόπο όλοι οι τύποι παραθύρων που προτείνονται στην υπάρχουσα βιβλιογραφία. Τα αποτελέσματα αυτής της προσπάθειας αξιοποιήθηκαν κατά το στάδιο της υλοποίησης, όπου εφαρμόζοντας την προτεινόμενη άλγεβρα υλοποιήθηκαν οι κυριότερες παραθυρικές δομές. Αναλύθηκαν επίσης διεξοδικά τα προβλήματα που παρουσιάζουν αρκετοί από τους σχεσιακούς τελεστές των κλασικών συστημάτων βάσεων δεδομένων κατά την προσαρμογή τους στο μοντέλο των ρευμάτων δεδομένων. Στους τελεστές αυτούς εστιάζονται κατά κύριο λόγο τα προβλήματα αυτών των εφαρμογών και τα παράθυρα λειτουργούν ως μέσο για την απεμπλοκή τους.

Κατά το στάδιο της υλοποίησης έγινε σημαντική προσπάθεια για να αποφευχθεί η επανάληψη πληροφορίας μέσα στο σύστημα. Για τον σκοπό αυτό έγινε η διάκριση μεταξύ φυσικών και λογικών ουρών επιτυγχάνοντας την χρήση των πρωτογενών δεδομένων από πολλαπλά ερωτήματα. Η απλούστερη λογική θα ήταν να τηρούνται για κάθε τελεστή και παράθυρο τα δικά του αντίγραφα πλειάδων για είσοδο και για έξοδο. Αυτό ωστόσο θα οδηγούσε σε άσκοπη δέσμευση σημαντικής ποσότητας κύριας μνήμης, στην οποία είναι επιθυμητό να περιοριστεί το σύστημα για να μπορέσει να ανταποκριθεί σε υψηλούς ρυθμούς δεδομένων για εφαρμογές με απαιτήσεις online απαντήσεων.

Όσον αφορά την κατασκευή παραθύρων και τελεστών έγιναν λειτουργικές προτάσεις υλοποίησης. Στα παράθυρα εφαρμόστηκαν με επιτυχία οι αλγεβρικές σχέσεις που διατυπώθηκαν, με τα μεριστικά παράθυρα να παρουσιάζουν έναν επιπρόσθετο βαθμό δυσκολίας για τις πλειάδες που εκπίπτουν κάποια στιγμή από την εμβέλεια τους. Για τον λόγο αυτό δημιουργήθηκε μία ιδιαίτερη δομή για τα παράθυρα αυτά επιτρέποντας την προσομοίωση της λειτουργίας τους.

Στην υλοποίηση των τελεστών υπήρξε ιδιαίτερη μέριμνα ώστε να υποστηριχθούν οι κυριότεροι τύποι δεδομένων (ακέραιοι, δεκαδικοί, χαρακτήρες, συμβολοσειρές). Ο τελεστής που παρουσίασε τα περισσότερα προβλήματα ήταν η *παραθυρική σύνδεση διοχέτευσης* (*windowed pipelined join*). Στην παρούσα μορφή υλοποίησης λειτουργεί κατά τον αναμενόμενο τρόπο μόνο όταν στις εισόδους της σύνδεσης εφαρμόζονται παράθυρα πλειάδων, μεριστικά ή παράθυρα οροσήμου. Το πρόβλημα εστιάζεται στην ανίχνευση των νέων πλειάδων που προστέθηκαν στις ουρές εισόδου, τις οποίες καλείται να επεξεργαστεί ο συγκεκριμένος τελεστής σε κάθε κύκλο εργασίας του. Στα κυλιόμενα και στα επάλληλα παράθυρα η έλευση νέας πλειάδας στην είσοδο μπορεί να προκαλέσει μερική, ολική ή μηδαμινή αλλαγή στα δεδομένα της εξόδου, ενώ στα υπόλοιπα παράθυρα η έλευση νέας πλειάδας στην είσοδο δίνει ακριβώς μία νέα πλειάδα στην έξοδο.

Κατά τις δοκιμές εκτέλεσης πρότυπων ερωτημάτων για ποικίλους ρυθμούς των ρευμάτων εισόδου, παρατηρήθηκαν φαινόμενα προσεγγιστικών απαντήσεων. Γίνεται φανερό, ότι πρέπει να αντιμετωπιστούν τα προβλήματα διάταξης και συγχρονισμού των εισόδων στους δυαδικούς τελεστές. Σημειώνεται ότι τα προβλήματα αυτά δεν υφίστανται στα συμβατικά συστήματα βάσεων δεδομένων.

Επιγραμματικά λοιπόν, η παρούσα υλοποίηση μπορεί να χαρακτηριστεί ως ο σκελετός ενός *Συστήματος Διαχείρισης Ρευμάτων Δεδομένων* με διαθέσιμα τα βασικά του στοιχεία. Αποτελεί την βάση για περαιτέρω επέκταση όπως αναλύεται διεξοδικότερα στην επόμενη ενότητα.

6.2 Μελλοντικές επεκτάσεις

Παρακάτω αναφέρονται συνοπτικά μερικές από τις δυνατότητες εξέλιξης του συστήματος:

- Υλοποίηση περισσότερων τελεστών και νέων τύπων παραθύρων ακολουθώντας την λογική των ήδη υφιστάμενων δομών. Ως επόμενα παράθυρα θα μπορούσαν να υλοποιηθούν χωρικά παράθυρα τα οποία θα απαντούν σε ερωτήματα της μορφής «Δώσε στην έξοδο όλα τα οχήματα με συντεταγμένες εντός κάποιας περιοχής ενδιαφέροντος». Ως πρόσθετοι τελεστές θα μπορούσαν αρχικά να υλοποιηθούν είτε συναθροιστικοί τελεστές είτε διαφορετικές εκδοχές της σύνδεσης. Ειδικά για την υλοποίηση *συναθροιστικών τελεστών* (*aggregate operators*) αναμένεται να χρησιμοποιηθούν στοιχεία της παρούσας υλοποίησης αναφορικά με τα μεριστικά παράθυρα.
- Υποστήριξη τροφοδότησης του τελεστή σύνδεσης διοχέτευσης με κυλιόμενα ή επάλληλα παράθυρα. Πρώιμες σκέψεις προς αυτήν την κατεύθυνση υποδεικνύουν ότι ίσως οι ουρές να πρέπει να διατηρούν πληροφορία για το αν υπάγονται σε παράθυρο ή τελεστή και μάλιστα σε ποιον. Έτσι ο νέος τελεστής σύνδεσης διοχέτευσης θα ενεργεί διαφορετικά ανάλογα με το είδος του παραθύρου.
- Αποδοτικότερη χρονοδρομολόγηση διεργασιών σε επίπεδο τελεστή. Ιδιαίτερη προσοχή πρέπει να δοθεί στο ενδεχόμενο οι είσοδοι των τελεστών να λαμβάνουν πολλαπλά στιγμιότυπα της εξόδου των τελεστών που βρίσκονται κάτω από αυτούς. Αν επιτραπεί τέτοιο ενδεχόμενο θα πρέπει να γίνει αντιληπτό ότι εγκαταλείπεται η ιδέα υπολογισμού των ερωτημάτων σε λειτουργία διοχέτευσης. Το πρόβλημα που δημιουργείται π.χ. για τον τελεστή σύνδεσης, είναι ότι δεν θα μπορούν να προσδιοριστούν τα δεδομένα πάνω στα οποία θα πρέπει να δράσει από κάθε είσοδο. Ακόμη και αν αυτό το πρόβλημα επιλυθεί προκύπτει το πρόβλημα της συσχέτισης των σωστών στιγμιοτύπων από κάθε ουρά σε κάθε κύκλο λειτουργίας του τελεστή.
- Αντιμετώπιση προβλημάτων διαφορετικού ρυθμού στις εισόδους δυαδικών τελεστών όποτε απαιτείται επεξεργασία αντίστοιχων χρονικά δεδομένων από κάθε ρεύμα. Σκέψεις για αναμονή της μίας ουράς μέχρι η άλλη να συμβαδίσει θα πρέπει να μεριμνούν επίσης για το πρόβλημα συσχέτισης των σωστών στιγμιοτύπων από τις δύο εισόδους.
- Αποκατάσταση χρονικής διάταξης στα εισερχόμενα ρεύματα. Η πιο απλή λύση θα ήταν η απόρριψη κάθε πλειάδας με χρονόσημο μικρότερο από το μεγαλύτερο χρονόσημο που παρουσιάστηκε σε αυτό το ρεύμα. Ωστόσο πιο αποδοτική μοιάζει η λύση ενός buffer στην είσοδο

που θα δέχεται πλειάδες από το ρεύμα με οποιαδήποτε διάταξη. Μόλις αυτός γεμίσει δίνει τα δεδομένα του για αποθήκευση σε φυσικές ουρές, κάνοντάς τα διαθέσιμα στα ερωτήματα. Πλέον, δέχεται μόνο στοιχεία με χρονόσημο μεγαλύτερο ή ίσο από το μέγιστο που ήδη περιέχεται στις φυσικές ουρές. Το μέγεθος του buffer πρέπει να ρυθμίζεται κατάλληλα ενώ για μέγεθος ίσο με μία πλειάδα προκύπτει εκφυλισμός στην απλοϊκή λύση απόρριψης πλειάδων.

- Ανάπτυξη κατάλληλου *γραφικής διεπαφής χρήστη* (*Graphical User Interface – GUI*) για την άμεση σύνθεση φυσικών προσχεδίων εκτέλεσης. Τα παράθυρα και οι τελεστές θα αποτελέσουν τα δομικά στοιχεία τα οποία θα χρησιμοποιηθούν για τον σχηματισμό των ερωτημάτων. Θα πρέπει επίσης να παρέχεται δυνατότητα παρακολούθησης της εισόδου και εξόδου κάθε τελεστή ή παραθύρου που μετέχει σε κάθε προσχέδιο.

Βιβλιογραφικές αναφορές

[ABB+03] A. Arasu, B. Babcock, S. Babu, M. Datar, K. Ito, R. Motwani, I. Nshizawa, U. Srivastava, D. Thomas, R. Varma, and J. Widom. STREAM: The Stanford Stream Data Manager. *In Bulletin of the IEEE on Data Engineering, 26 (1): 19-26, March 2003.*

[ABW03] A. Arasu, S. Babu, and J. Widom. An Abstract Semantics and Concrete Language for Continuous Queries over Streams and Relations. *In Proceedings of the 9th International Conference on Data Base Programming Languages (DBPL'03), Berlin, Germany, September 2003.*

[ACC+03] D.J. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. Aurora: a New Model and Architecture for Data Stream Management. *VLDB Journal, 12(2):120-139, August 2003.*

[AH00] R. Avnur and J. Hellerstein. Eddies: Continuously Adaptive Query Processing. *In Proceedings of the 19th ACM SIGMOD International Conference on Management of Data, pp. 261–272, Dallas, Texas, USA, May 2000.*

[AN04] A. Ayad and J. Naughton. Static Optimization of Conjunctive Queries with Sliding Windows over Data Streams. In *Proceedings of the 23rd ACM SIGMOD International Conference on Management of Data*, pp. 419-430, Paris, France, June 2004.

[AW04] A. Arasu and J. Widom. Resource Sharing in Continuous Sliding-Window Aggregates. *In Proceedings of the 30th International Conference on Very Large Data Bases (VLDB), Toronto, Canada, September 2004.*

[BBD+02] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and Issues in Data Stream Systems. *In Proceedings of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'02), pp.1-16, Madison, Wisconsin, May 2002.*

[BBDM03] B. Babcock, S. Babu, M. Datar, and R. Motwani. Chain: Operator Scheduling for Memory Minimization in Data Stream Systems. *In Proceedings of the 22nd ACM SIGMOD International Conference on Management of Data, pp. 253-264, San Diego, California, June 2003.*

[BDM04] B. Babcock, M. Datar, and R. Motwani. Load Shedding for Aggregation Queries over Data Streams. *In Proceedings of the 20th IEEE International Conference on Data Engineering (ICDE), Boston, Massachusetts, March 2004.*

[BW01] S. Babu and J.Widom. Continuous Queries over Data Streams. *ACM SIGMOD Record, 30 (3): 109-120, September 2001.*

[CCR+03] D. Carney, U. Cetintemel, A. Rasin, S. Zdonik, M. Cherniack, and M. Stonebraker. Operator Scheduling in a Data Stream Manager. *In Proceedings of the 29th International Conference on Very Large Data Bases (VLDB'03), Berlin, Germany, September 2003.*

[CCD+03] S. Chandrasekaran, O. Cooper, A. Deshpande, M.J. Franklin, J.M. Hellerstein, W. Hong, S. Krishnamurthy, S.R. Madden, V. Raman, F. Reiss, and M.A. Shah. TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. *In Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR), Asilomar, California, January 2003.*

[CF03] S. Chandrasekaran, and M.J. Franklin. PSoup: a system for streaming queries over streaming data. *VLDB Journal, 12(2):140-156, August 2003.*

[CJSS03] C. Cranor, T. Johnson, O. Spataschek, and V. Shkapenyuk. Gigascope: A Stream Database for Network Applications. *In Proceedings of the 22nd ACM SIGMOD International Conference on Management of Data, pp. 647-651, San Diego, California, USA June 2003.*

[Dei05] Deitel *C++ How To Program*, 5th Edition, Pearson Education Inc., 2005.

[GHM+05] T. Ghanem, M. Hammad, M. Mokbel, W.G. Aref, and A.K. Elmagarmid. Query Processing Using Negative Tuples in Stream Query Engines. *Technical Report TR#04-040, Purdue University, Department of Computer Sciences, April 2005.*

[GMUW02] H. Garcia-Molina, J. D. Ullman, and J. Widom. *Database Systems: the Complete Book*, Prentice Hall, 2002.

[GO03] L. Golab, and M. Tamer Ozsu. *Issues in Data Stream Management. ACM SIGMOD Record, 32(2):5-14, June 2003.*

[GO05] L. Golab and M.T. Ozsu. Update-Pattern-Aware Modeling and Processing of Continuous Queries. *In Proceedings of the 24th ACM SIGMOD International Conference on Management of Data, pp. 658-669, Baltimore, Maryland, USA June 2005.*

[HFAE03] M. A. Hammad, M. J. Franklin, W. G. Aref, and A. K. Elmagarmid. Scheduling for Shared Window Joins over Data Streams. *In Proceedings of the 29th International Conference on Very Large Data Bases, pp. 297-308, Berlin, Germany, September 2003.*

[HFC+00] J. M. Hellerstein, M. J. Franklin, S. Chandrasekaran, A. Deshpande, K. Hildrum, S. Madden, V. Raman, and M. Shah. Adaptive query processing: Technology in evolution. *IEEE Data Engineering Bulletin, 23(2): 7-18, 2000.*

[JMSS05] T. Johnson, S. Muthukrishnan, V. Shkapenyuk, and O. Spatscheck. A Heartbeat Mechanism and its Application in Gigascope. *In Proceedings of the 31st International Conference on Very Large Data Bases (VLDB), pp. 1079-1088, Trondheim, Norway, September 2005.*

[LMP+05] J. Li, D. Maier, K. Tufte, V. Papadimos and P.A. Tucker. Semantics and Evaluation Techniques for Window Aggregates in Data Streams. *In Proceedings of the 24th ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, June 2005.*

[LMT+05] J. Li, D. Maier, K. Tufte, V. Papadimos and P.A. Tucker. No Pane, No Gain: Efficient Evaluation of Sliding-Window Aggregates over Data Streams. *ACM SIGMOD Record, 34(1):39-44, March 2005.*

[MHSR02] S. Madden, J. Hellerstein, M. Shah, and V. Raman. Continuously adaptive continuous queries over streams. *In Proceedings of the 21st ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, June 2002.*

[MLT+05] D. Maier, J. Li, P. Tucker, K. Tufte, and V. Papadimos. Semantics of Data Streams and Operators (Invited Presentation). *In Proceedings of the 10th International Conference on Database Theory (ICDT 2005), pp. 37-52, Edinburgh, Scotland, January 2005.*

[MWA+03] R. Motwani , J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein and R.Varma. Query Processing, Approximation, and Resource Management in a Data Stream Management System. *In Proceedings of the 2003 Conference on Innovative Data Systems Research (CIDR), January 2003.*

[Pat03] Κ. Πατρούμπας. *Συστήματα ρευμάτων δεδομένων για κινούμενα αντικείμενα.* Μεταπτυχιακή διπλωματική εργασία στο Δ.Π.Μ.Σ. "Γεωπληροφορική", Ε.Μ.Π., Ιούνιος 2003.

[PS05] K. Patroumpas and T. Sellis. Window Specification over Data Streams. *Unpublished Manuscript, National Technical University of Athens, School of Electrical and Computer Engineering, May 2005.*

[SHCF03] M.A. Shah, J.M. Hellerstein, S. Chandrasekaran, and M.J.Franklin. Flux: An Adaptive Partitioning Operator for Continuous QuerySystems. *In Proceedings of the First Biennial Conference on InnovativeData Systems Research (CIDR 2003), Asilomar, California, January 2003.*

[SH98] M. Sullivan and A. Heybey. Tribeca: A System for Managing Large Databases of Network Traffic. *In Proceedings of the USENIX Annual Technical Conference, New Orleans, Louisiana, USA, June 1998.*

[SKS04] A. Silberschatz, H. Korth, and S. Sudarshan. *Database system concepts*, 4th edition, McGraw-Hill, 2004.

[SW04] U. Srivastava and J. Widom. Flexible Time Management in Data Stream Systems. *In Proceedings of the 23rd ACM Symposium on Principles of Database Systems (PODS 2004), Paris, France, June 2004.*

[TGNO92] D. B. Terry, D. Goldberg, D. Nichols, and B. M. Oki. Continuous Queries over Append-only Databases. *In Proceedings of the 11th ACM SIGMOD International Conference on Management of Data, pp. 321-330, San Diego, California, June 1992.*

[TMSF02] P. Tucker, D. Maier, T. Sheard, and L. Fegaras. Enhancing Relational operators for Querying over Punctuated Data Streams. *In Proceedings of the 28th International Conference On Very Large Data Bases, Hong Kong, China, August 2002.*

[UF01] T. Urhan and M.J. Franklin. Dynamic Pipeline Scheduling for Improving Interactive Query Performance. *In Proceedings of 27th International Conference on Very Large Data Basesv (VLDB'01), pp. 501-510, Roma, Italy, September 2001.*

Δικτυακοί τόποι (Internet Sites)

[siteAur] Aurora : http://www.cs.brown.edu/research/aurora/

[siteStr] Stream : http://www-db.stanford.edu/stream/

[siteTCQ] TelegraphCQ : http://telegraph.cs.berkeley.edu/

Ορολογία

αναδιατύπωση ερωτήματος	query rewriting
ανασταλτικός τελεστής	blocking operator
απαλοιφή διπλοτύπων	duplicate elimination
απεικόνιση	mapping
αποβολή φόρτου	load shedding
γλώσσα ερωταποκρίσεων	query language
γνώρισμα	attribute
διαφορά	difference
δοσοληψία	transaction
δυναμοσύνολο	powerset
εμβέλεια	scope
ένθετο υποερώτημα	nested subquery
ένωση	union
επιλογή	selection
ερώτημα διαρκείας	continuous query
ερώτημα διαστήματος τιμών	range query
ερώτημα μονότονο	monotonic query
ερώτημα περιστασιακό	ad-hoc query
ερώτημα στιγμιοτύπου	one-time, snapshot query
ερώτημα συζευκτικό	conjunctive query
κυματίδιο	wavelet
παράθυρο επάλληλο	tumbling window
παράθυρο ζώνης	band window
παράθυρο κυλιόμενο	sliding window
παράθυρο μεριστικό	partitioned window
παράθυρο πλειάδων	tuple-based window
παράθυρο ορορήμου	landmark window
περίληψη	summary
πολλαπλή σύνδεση	multi-way join
πολυσύνολο	multiset, bag
προβολή	projection
προοδευτικός υπολογισμός	incremental computation
προσέγγιση	approximation
προσχέδιο εκτέλεσης ερωτήματος	query execution plan
ρεύμα δεδομένων	data stream
ρυθμοαπόδοση	throughput
σκίτσο	sketch
στίξη	punctuation
συνάθροιση	aggregation
σύνδεση	join
συνένωση	concatenation
σύνοψη	synopsis
συγχώνευση	merge
σχέση	relation
σωρευτικό ρεύμα δεδομένων	append-only data stream
τελεστής	operator
ταξινόμηση	sorting
τομή	intersection
χρονόσημο	timestamp

Προσδιορισμός παραθύρων σε ρεύματα δεδομένων

Ηλίας Τζωρτζακάκης
el99684@mail.ntua.gr

Εκτενής περίληψη

Διπλωματική εργασία στο Εργαστήριο Συστημάτων Βάσεων Γνώσεων και Δεδομένων
Επιβλέπων: Καθηγητής ***Τ. Σελλής***

1 Εισαγωγή

Πλήθος σύγχρονων εφαρμογών εγείρουν την ανάγκη διαφορετικού τρόπου διαχείρισης της πληροφορίας που συλλέγεται σε σχέση με τα συμβατικά συστήματα βάσεων δεδομένων. Οι εφαρμογές αυτές εστιάζουν στις πιο πρόσφατες πληροφορίες που εισέρχονται στο σύστημα. Τυπικά παραδείγματα τέτοιων εφαρμογών αποτελούν δίκτυα αισθητήρων, παρακολούθηση κινούμενων αντικειμένων, εφαρμογές χρηματιστηρίου και εποπτεία κίνησης στο διαδίκτυο. Κοινά χαρακτηριστικά αυτών των εφαρμογών αποτελούν:

- η διαχείριση δυναμικά εισερχόμενης πληροφορίας με τη μορφή **ρευμάτων δεδομένων** (*data stream*), με έμφαση στην πιο πρόσφατη πληροφορία,
- η διατύπωση **ερωτημάτων διαρκείας** (*continuous queries*), τα οποία απαιτούν online επεξεργασία και πρέπει να επιστρέφουν απαντήσεις σε πραγματικό χρόνο.

Οι παραθυρικές δομές αποτελούν κοινή συνιστώσα στις μέχρι τώρα προσεγγίσεις που αφορούν επεξεργασία ρευμάτων δεδομένων. Με την βοήθεια των **παραθύρων** (*windows*) καθίσταται εφικτή η έγκαιρη απόκριση στα σχετικά ερωτήματα διαρκείας, επιλύοντας αρκετά από τα εγγενή ζητήματα επεξεργασίας που τα χαρακτηρίζουν.

Η διατύπωση και η επεξεργασία παραθύρων βρίσκονται στο επίκεντρο αυτής της διπλωματικής εργασίας. Στα πλαίσια της εργασίας διερευνήθηκαν τα εξής θέματα:

- αλγεβρική θεμελίωση των παραθυρικών δομών με έμφαση στη σημασιολογία τους,
- υλοποίηση των κυριοτέρων τύπων παραθύρων,
- υλοποίηση επιλεγμένων σχεσιακών τελεστών,
- συνδυασμός παραθύρων με τελεστές για τον σχηματισμό ερωτημάτων διαρκείας,
- έλεγχος ορθής λειτουργίας του συστήματος και αξιολόγηση των επιδόσεων με διαθέσιμα σύνολα δεδομένων.

2 Επεξεργασία ρευμάτων δεδομένων

Ο όρος *ρεύμα δεδομένων* δηλώνει πληροφορία η οποία φτάνει δυναμικά στο σύστημα την ώρα που αυτό επεξεργάζεται προγενέστερα στοιχεία. Το σύστημα δεν έχει άμεσο έλεγχο ούτε στον ρυθμό άφιξης ούτε στην διάταξη της αφικνούμενης πληροφορίας, το μέγεθος της οποίας είναι δυνητικά άπειρο. Τέλος κάθε στοιχείο έχει τη μορφή σχεσιακής *πλειάδας* (*tuple*) και έρχεται με ένα *χρονόσημο* (*timestamp*) που δηλώνει τον χρόνο παραγωγής του.

Ο όρος *ερωτήματα διαρκείας* δηλώνει ερωτήματα τα οποία παραμένουν ενεργά μέχρι να τερματιστούν από τον χρήστη. Αυτά τα ερωτήματα επεξεργάζονται πληροφορία που φτάνει στο σύστημα με την μορφή ρεύματος δεδομένων από τη χρονική στιγμή υποβολής τους και μετά. Δεν αποκλείεται τα ερωτήματα να αναφέρονται και σε στατικές *σχέσεις* (*relations*), αλλά η επεξεργασία ενός τουλάχιστον ρεύματος δεδομένων δίνει νόημα στην ύπαρξη αυτών των ερωτημάτων. Τα συμβατικά *ερωτήματα στιγμιοτύπου* (*one-time* ή *snapshot queries*) που κυριαρχούν στα κλασικά συστήματα βάσεων δεδομένων εφαρμόζονται μόνο πάνω σε στατικές σχέσεις, γι' αυτό και αδυνατούν να ανταποκριθούν στις απαιτήσεις των εφαρμογών ρευμάτων δεδομένων.

Το μοντέλο επεξεργασίας επίσης αλλάζει: αντί ο χρήστης να ζητάει και να «τραβάει» αποτελέσματα από τα υπάρχοντα δεδομένα (*pull model*), στο νέο μοντέλο τα δεδομένα «σπρώχνουν» απαντήσεις στον χρήστη και στα ερωτήματα διαρκείας που ο ίδιος υπέβαλε (*push model*).

Ακόμη χρειάζεται αναθεώρηση των τελεστών που χρησιμοποιούνται στα συστήματα βάσεων δεδομένων. Μερικοί τελεστές χρειάζονται πολύ μικρές προσαρμογές για να επεξεργαστούν ρεύματα δεδομένων. Ωστόσο άλλοι σημαντικοί τελεστές απαιτούν

σημαντικές αλλαγές στον τρόπο λειτουργίας τους, κυρίως επειδή προϋποθέτουν διαθέσιμα όλα τα δεδομένα εισόδου τους. Πρακτικά όμως, επειδή δεν είναι εφικτό να διατηρείται το σύνολο των στοιχείων του ρεύματος, τέτοιοι προβληματικοί τελεστές οδηγούνται ενίοτε σε προσεγγιστικές απαντήσεις.

Ωστόσο, αυτή η έλλειψη ακρίβειας δεν πρέπει να θεωρείται δεδομένη. Σε αρκετές περιπτώσεις, το ίδιο το ερώτημα εστιάζει στην πιο πρόσφατη πληροφορία και όχι σε ολόκληρο το ρεύμα δεδομένων, καθιστώντας ανούσια την διατήρηση των παλαιότερων στοιχείων του ρεύματος καθώς αυτό συνεχώς εξελίσσεται. Οι παραθυρικές δομές έρχονται να επιτελέσουν αυτήν ακριβώς την λειτουργία, παρέχοντας διαρκώς πρόσβαση στα πιο πρόσφατα στοιχεία του ρεύματος.

3 Παραθυρικές Δομές

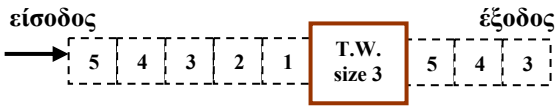
Σκοπός των παραθυρικών δομών είναι να αποσπούν συνεχώς κάποιο πεπερασμένο πλήθος στοιχείων του ρεύματος παρέχοντας διαρκώς πρόσβαση σε περιορισμένο τμήμα του. Ο βασικός ρόλος τους είναι η τροφοδότηση προβληματικών τελεστών κατά την προσαρμογή τους στο μοντέλο εξυπηρέτησης ρευμάτων δεδομένων με στόχο την απεμπλοκή των ερωτημάτων διαρκείας. Μπορούν να οριστούν διάφοροι τύποι παραθύρων με κριτήρια όπως:

- η στασιμότητα ή μη των ορίων τους (αρχή και πέρας των στοιχείων που περικλείουν),
- ο τρόπος προσδιορισμού των περιεχομένων τους (πλήθος στοιχείων ή χρονόσημα),
- ο ρυθμός ανανέωσής τους.

Έτσι τα παράθυρα μπορούν να έχουν: σταθερά ή μεταβλητά άκρα, περιεχόμενα βάσει χρονοσήμου ή πλειάδας, καθώς και βήμα προόδου μοναδιαίο ή κατ' άλματα. Η έξοδος κάθε παραθύρου αποτελεί επίσης ένα ρεύμα δεδομένων το οποίο τροφοδοτείται τελικά με όλες τις πλειάδες που φτάνουν στο ρεύμα εισόδου του. Οι μόνες πλειάδες που δεν θα διέλθουν από το παράθυρο είναι όσες έχουν χρονόσημο προγενέστερο της εκκίνησης της λειτουργίας του παραθύρου. Παρακάτω ακολουθεί μια σύντομη περιγραφή για καθεμιά από τις παραθυρικές δομές που μελετήθηκαν.

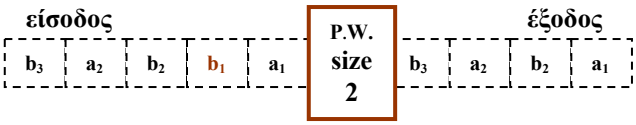
3.1 Παράθυρα πλειάδων (tuple-based windows)

Τα παράθυρα αυτού του τύπου διατηρούν συνεχώς στο ρεύμα εξόδου τους τις N τελευταίες πλειάδες από το στιγμιότυπο του ρεύματος που παρέχεται στην είσοδο. Στο παρακάτω σχήμα φαίνεται ένα παράδειγμα, στο οποίο οι αριθμοί υποδηλώνουν την σειρά άφιξης των στοιχείων στον τελεστή.



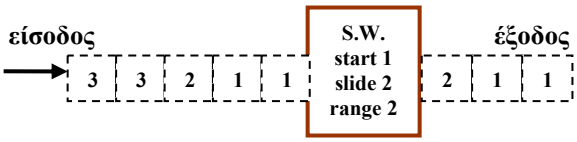
3.2 Μεριστικά παράθυρα (partitioned windows)

Τέτοια παράθυρα προϋποθέτουν τη δήλωση μιας *λίστας γνωρισμάτων* (*attributes*) ομαδοποίησης των στοιχείων. Για κάθε διαφορετικό συνδυασμό τιμών αυτών των γνωρισμάτων, τα μεριστικά παράθυρα διατηρούν στην έξοδό τους τις N πιο πρόσφατες πλειάδες. Στο επόμενο σχήμα, απεικονίζεται ένα μεριστικό παράθυρο, όπου το μοναδικό γνώρισμα ομαδοποίησης δηλώνεται με τον απεικονιζόμενο χαρακτήρα. Ο δείκτης φανερώνει την σειρά με την οποία εμφανίστηκε ο χαρακτήρας αυτός στο ρεύμα εισόδου.



3.3 Κυλιόμενα παράθυρα (sliding windows)

Τα παράθυρα αυτού του τύπου διατηρούν συνεχώς στην έξοδό τους όσες από τις πιο πρόσφατες πλειάδες έχουν χρονόσημο που εμπίπτει στην εμβέλεια του παραθύρου. Η *εμβέλεια* του παραθύρου καθορίζεται από τις παραμέτρους έκτασης, βήματος και χρονοσήμου εκκίνησης. Η έξοδος λ.χ. μπορεί κατ' αυτόν τον τρόπο να αντιπροσωπεύει όσα στοιχεία του ρεύματος ήρθαν τις τελευταίες T χρονικές στιγμές. Στο παρακάτω σχήμα φαίνεται ένα τέτοιο απλό παράδειγμα, όπου οι αριθμοί τώρα υποδηλώνουν το χρονόσημο που συνοδεύει κάθε πλειάδα.

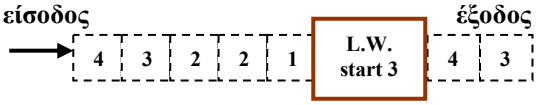


3.4 Επάλληλα παράθυρα (tumbling windows)

Τα παράθυρα αυτά αποτελούν μία ειδική περίπτωση των κυλιόμενων παραθύρων με την διαφορά ότι οι παράμετροι βήματος και έκτασης του κυλιόμενου παραθύρου θεωρούνται πάντοτε ίσες. Αποτέλεσμα αυτής της συνθήκης είναι μη επικαλυπτόμενα διαδοχικά στιγμιότυπα του ρεύματος εξόδου χωρίς απώλεια πληροφορίας.

3.5 Παράθυρα οροσήμου (landmark windows)

Τα παράθυρα αυτού του τύπου διατηρούν πάντοτε σταθερό το ένα από τα δύο χρονικά άκρα τους, οριοθετώντας το ενδιαφέρον του χρήστη για τα αντίστοιχα δεδομένα. Στο παρακάτω σχήμα δίνεται ένα παράδειγμα σταθερού κάτω άκρου με τους αριθμούς να υποδηλώνουν και πάλι το χρονόσημο κάθε πλειάδας.



3.6 Πάγια παράθυρα ζώνης (fixed band windows)

Τέτοιας μορφής παράθυρα διατηρούν πάντοτε σταθερά και τα δύο χρονικά άκρα τους δηλώνοντας το ενδιαφέρον του χρήστη για μία συγκεκριμένη χρονική περίοδο. Στο ρεύμα εξόδου περιέχονται όλες οι πλειάδες του ρεύματος εισόδου με χρονόσημο εντός των ορίων του παραθύρου.

4 Τελεστές σε ρεύματα δεδομένων

Οι τελεστές (*operators*) των συμβατικών συστημάτων βάσεων δεδομένων παρουσιάζουν αρκετές δυσκολίες κατά την προσαρμογή τους σε ρεύματα δεδομένων. Εξαιρέσεις αποτελούν ορισμένοι τελεστές όπως *προβολή* (*projection*) και *επιλογή* (*selection*), οι οποίοι μπορούν να εφαρμοστούν απευθείας στο ρεύμα δεδομένων. Αυτό οφείλεται στο γεγονός ότι μπορούν να επεξεργαστούν κάθε πλειάδα παράγοντας εγκαίρως σωστά αποτελέσματα ανεξάρτητα από τις υπόλοιπες πλειάδες του ρεύματος.

Αρκετοί άλλοι τελεστές στερούνται αυτής της ιδιότητας μιας και χρειάζονται διαρκώς πρόσβαση σε όλα τα δεδομένα εισόδου για να εξάγουν σωστά αποτελέσματα. Πρόκειται για τους τελεστές *συνάθροισης* (SUM, COUNT, AVG), *σύνδεσης* (*join*), *ταξινόμησης* (*sort*), *απαλοιφής διπλοτύπων* (*duplicate elimination*), *τομής* (*intersection*) και *διαφοράς* (*difference*). Όμως ο πιθανόν υψηλός ρυθμός άφιξης των δεδομένων του ρεύματος και το δυνητικά άπειρο πλήθος τους καθιστούν απαγορευτική την διατήρηση όλων των απαιτούμενων πλειάδων. Αντίθετα η απαίτηση online επεξεργασίας και έγκαιρων απαντήσεων υπαγορεύει τον κατά το δυνατό περιορισμό στην κύρια μνήμη.

Στην τρέχουσα βιβλιογραφία, για την αντιμετώπιση του προβλήματος προτείνεται η τροφοδότηση τέτοιων τελεστών από παράθυρα, ώστε σε κάθε χρονική στιγμή να περιορίζεται το πλήθος των στοιχείων εισόδου. Αυτή ακριβώς η λογική ακολουθήθηκε κατά το στάδιο υλοποίησης της παρούσας εργασίας.

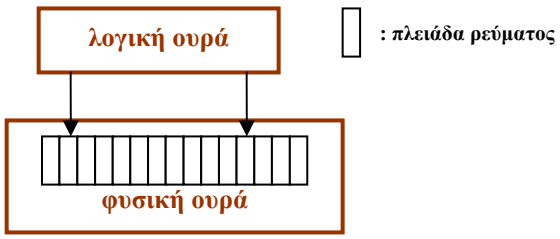
5 Γενικά στοιχεία υλοποίησης

Η υλοποίηση των παραθύρων και των τελεστών έγινε σε standard C++ και δοκιμάστηκε με επιτυχία σε περιβάλλον RedHat Linux 9.0. Αποφεύχθηκε η χρήση έτοιμων πακέτων ή κλάσεων λογισμικού με εξαίρεση την υλοποίηση των *διεργασιών* (*threads*) που παρέχει το Linux σύμφωνα με το πρότυπο POSIX.

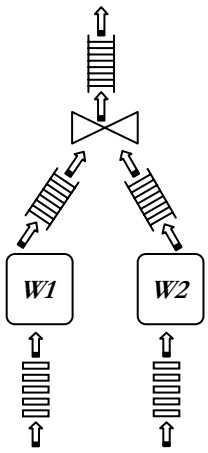
Βασικές υποθέσεις εργασίας που έγιναν ήταν οι εξής:

i. υπάρχει χρονική διάταξη στα εισερχόμενα στοιχεία κάθε ρεύματος,
ii. το χρονόσημο κάθε πλειάδας είναι ένας ακέραιος και
iii. γενικά υπάρχει αρκετή διαθέσιμη κύρια μνήμη για την εξυπηρέτηση των ερωτημάτων.

Τα ρεύματα των δεδομένων προσομοιώνονται διαβάζοντας από πρότυπα αρχεία εισόδου με ελεγχόμενο ρυθμό. Η πρώτη γραμμή του αρχείου δηλώνει το σχήμα των πλειάδων του ρεύματος, μαζί με τους αντίστοιχους τύπους δεδομένων (υποστηρίζονται integer, double, char, και charn). Τα στοιχεία του ρεύματος τηρούνται σε δομές *ουρών* (*queue*): υπάρχουν *φυσικές* ουρές που περιέχουν πραγματικές πλειάδες δεδομένων και *λογικές* ουρές που περιέχουν αναφορές σε φυσικές χωρίς να επαναλαμβάνουν τις περιεχόμενες πλειάδες. Η διάκριση αυτή φαίνεται πιο παραστατικά στο επόμενο σχήμα.



Γενικά υπάρχουν αντικείμενα παραθύρων και τελεστών τα οποία διατηρούν λογικές ή φυσικές ουρές εισόδου και εξόδου. Τα ερωτήματα διαρκείας σχηματίζονται διασυνδέοντας τις ουρές εισόδου και εξόδου των παραθύρων και των τελεστών. Έτσι, ο χρήστης έχει τη δυνατότητα να προσδιορίσει το προσχέδιο εκτέλεσης του ερωτήματος (*query execution plan*), όπως φαίνεται στο παρακάτω παράδειγμα.



Ρεύματα δεδομένων εισόδου

5.1 Υλοποίηση παραθύρων

Υλοποιήθηκαν παράθυρα πλειάδων, μεριστικά παράθυρα με ένα γνώρισμα ομαδοποίησης, κυλιόμενα παράθυρα (άρα και η ειδική περίπτωση των επάλληλων) καθώς και παράθυρα οροσήμου με σταθερό κάτω άκρο. Όλα τα παράθυρα διατηρούν λογικές ουρές εισόδου και εξόδου αφού δεν μεταβάλλουν ούτε την διάταξη ούτε το σχήμα των στοιχείων της φυσικής ουράς στην οποία αναφέρονται. Μοναδική εξαίρεση αποτελεί το μεριστικό παράθυρο το οποίο διατηρεί φυσική ουρά εξόδου εφόσον η έλευση κάθε νέας πλειάδας μπορεί να προκαλέσει διαγραφή σε τυχαίο σημείο της ουράς εξόδου αλλάζοντας την διάταξή της.

Όλα τα παράθυρα επεξεργάζονται σε κάθε κύκλο εργασίας τους το πολύ μία πλειάδα από την ουρά εισόδου τους. Έτσι στην ουρά εξόδου παρέχεται εκείνο το στιγμιότυπο του ρεύματος που καθορίζεται από την σημασιολογία του και τα μέχρι τότε επεξεργασμένα στοιχεία της ουράς εισόδου.

5.2 Υλοποίηση τελεστών

Οι τελεστές που υλοποιήθηκαν είναι: επιλογή, προβολή και μία εκδοχή του τελεστή σύνδεσης (*pipeline join*). Όλες οι ουρές εισόδου είναι λογικές σε αντίθεση με τις ουρές εξόδου οι οποίες είναι φυσικές αφού διαφέρουν από την είσοδο ως προς τα περιεχόμενα, την διάταξη ή το σχήμα των πλειάδων.

Οι τελεστές επιλογής και προβολής επεξεργάζονται σε κάθε κύκλο εργασίας τους το σύνολο των πλειάδων που προστέθηκαν στην ουρά εισόδου τους παρέχοντας έγκαιρα και σωστά αποτελέσματα. Ο υλοποιημένος τελεστής σύνδεσης έχει δύο λογικές ουρές εισόδου οι οποίες συνδέονται αποκλειστικά σε παράθυρα πλειάδων. Τον περιορισμό αυτό επέβαλαν οι εξής απαιτήσεις:

- η σύνδεση πρέπει να επεξεργάζεται μόνο τα νέα στοιχεία των ουρών εισόδου,
- όλες οι πλειάδες πρέπει να σταλούν προς επεξεργασία χωρίς απώλειες και
- να είναι πάντοτε σαφές σε ποιο στιγμιότυπο του ρεύματος αναφέρεται κάθε ουρά εισόδου.

Τα παράθυρα πλειάδων ικανοποιούν αυτές τις απαιτήσεις, δεδομένου ότι σε κάθε κύκλο εργασίας τους επεξεργάζονται το πολύ μία πλειάδα από την είσοδό τους και προωθούν στην έξοδο επίσης μία πλειάδα το πολύ. Αντίθετα, άλλοι τύποι παραθύρων εμφανίζουν επιπλοκές. Λ.χ. τα κυλιόμενα παράθυρα σε κάθε κύκλο εργασίας τους επεξεργάζονται επίσης μία πλειάδα, αλλά είναι άγνωστος ο βαθμός μεταβολής της ουράς εξόδου λόγω του άγνωστου πλήθους των στοιχείων που εμπίπτουν στην εμβέλειά του.

5.3 Λειτουργία συστήματος

Κατά το στάδιο της εκτέλεσης προτιμήθηκε *χρονοδρομολόγηση* (*scheduling*) σε επίπεδο ερωτήματος. Έτσι, κάθε ερώτημα αρχικοποιείται με τον κατάλληλο συνδυασμό παραθύρων και τελεστών. Ακολούθως δημιουργείται μία διεργασία η οποία δίνει κυκλικά χρόνο επεξεργασίας σε όλους τους τελεστές του ερωτήματος, ενώ παράλληλα ανιχνεύει την έλευση νέων στοιχείων στις ουρές εισόδου. Πέρα από τις διεργασίες των ερωτημάτων υπάρχουν και οι διεργασίες που διαβάζουν με ελεγχόμενο ρυθμό τα στοιχεία κάθε ρεύματος από το αντίστοιχο αρχείο και τα τοποθετούν σε φυσικές ουρές, οι οποίες τροφοδοτούν λογικά αντίγραφα για κάθε ερώτημα.

6 Γενικά συμπεράσματα – Μελλοντική εργασία

Η παρούσα εργασία ανέδειξε την σπουδαιότητα των παραθύρων στις εφαρμογές ρευμάτων δεδομένων επιβεβαιώνοντας ότι παρέχουν ικανοποιητικές λύσεις στα προβλήματα που ανακύπτουν σε τέτοιες εφαρμογές. Η συγκεκριμένη υλοποίηση προσφέρει κομβικά συστατικά μέρη ενός απλουστευμένου συστήματος διαχείρισης ρευμάτων δεδομένων. Η μελέτη στηρίχθηκε σε ανάλογες προσεγγίσεις από διάφορες ερευνητικές ομάδες που αναπτύσσουν πρότυπα συστήματα προσανατολισμένα στην διαχείριση ρευμάτων δεδομένων (Aurora, Gigascope, STREAM, TelegraphCQ).

Μια σημαντική καινοτομία της παρούσας υλοποίησης είναι ο περιορισμός στο ελάχιστο των πλειάδων που υπάρχουν μέσα στο σύστημα, εισάγοντας λογικές ουρές όπου ήταν επιτρεπτό. Συνήθως προτιμάται κάθε τελεστής και κάθε παράθυρο να τηρεί τις δικές του πλειάδες εισόδου και εξόδου, υλοποιώντας φυσικές ουρές εισόδου και εξόδου.

Κατά κύριο λόγο, εκτιμάται ότι το αποτέλεσμα αυτής της εργασίας αποτελεί τον βασικό σκελετό για περαιτέρω επεκτάσεις, όπως:

- υλοποίηση πρόσθετων τελεστών ακολουθώντας την κοινή λογική και δομή των υπαρχόντων,
- αντιμετώπιση προβλημάτων έλλειψης διάταξης στα εισερχόμενα στοιχεία,
- αποδοτικότερη χρονοδρομολόγηση διεργασιών σε επίπεδο τελεστή,
- σύνθεση ερωτημάτων μέσω κατάλληλου GUI με χρήση των υπαρχόντων και νέων τελεστών,
- βελτιστοποίηση πολλαπλών ερωτημάτων.