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

ΣΥΝΘΕΣΗ ΠΡΟΣΧΕΔΙΩΝ ΕΚΤΕΛΕΣΗΣ ΕΡΩΤΗΜΑΤΩΝ ΔΙΑΡΚΕΙΑΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Όλγας Γκουντούνα

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Αθήνα, Οκτώβριος 2007

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

ΣΥΝΘΕΣΗ ΠΡΟΣΧΕΔΙΩΝ ΕΚΤΕΛΕΣΗΣ ΕΡΩΤΗΜΑΤΩΝ ΔΙΑΡΚΕΙΑΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

της

Όλγας Γκουντούνα

Επιβλέπων : Τιμολέων Σελλής
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 8η Οκτωβρίου 2007.

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

Αθήνα, Οκτώβριος 2007

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

Οργάνωση του τόμου

Η παρούσα εργασία προσεγγίζει το θέμα της σχεδίασης προσχεδίων εκτέλεσης για την διατύπωση ερωτημάτων διαρκείας σε ρεύματα δεδομένων τόσο σε επίπεδο θεωρητικής μελέτης όσο και σε επίπεδο υλοποίησης. Τα κεφάλαια 1-4 καλύπτουν το θεωρητικό κομμάτι της εργασίας ενώ το κεφάλαιο 5 αναφέρεται στην υλοποίηση της εφαρμογής. Συγκεκριμένα:

Στο **1ο κεφάλαιο** γίνεται μια εισαγωγή στις ιδιαιτερότητες των ερωτημάτων που διατυπώνονται σε εφαρμογές ρευμάτων δεδομένων και αναδεικνύεται η ανάγκη ανάπτυξης ειδικών συστημάτων για τη διαχείρισή τους. Γίνεται επίσης μια παρουσίαση των γνωστότερων πρωτοτύπων συστημάτων που αναπτύσσονται για τον σκοπό αυτό.

Στο **2ο κεφάλαιο** παρουσιάζονται οι διάφοροι σχεσιακοί τελεστές που εφαρμόζονται σε ρεύματα δεδομένων. Αναλύονται τα ζητήματα που ανακύπτουν κατά την προσαρμογή των τελεστών αυτών στα ρεύματα και αναφέρονται οι βασικότεροι τρόποι αντιμετώπισης που έχουν προταθεί για την επίλυση των προβλημάτων που προκύπτουν.

Στο **3ο κεφάλαιο** παρουσιάζονται οι κυριότεροι τύποι παραθυρικών δομών που είναι διαθέσιμοι στην υπάρχουσα βιβλιογραφία και κατατάσσονται σε κατηγορίες ανάλογα με την σημασιολογία τους. Ακόμη, επεξηγείται ο τρόπος τροφοδότησης κάθε τελεστή με δεδομένα μέσα από παράθυρα. Τέλος, επιδεικνύονται τρόποι με τους οποίους τα παράθυρα μπορούν να ενσωματωθούν στο συντακτικό γλωσσών ερωταποκρίσεων μορφής SQL.

Στο **4ο κεφάλαιο** γίνεται εκτενής παρουσίαση ενός πρωτότυπου Συστήματος Διαχείρισης Ρευμάτων Δεδομένων, του TelegraphCQ, το οποίο χρησιμοποιήθηκε από την εφαρμογή ως μηχανισμός εκτέλεσης των ερωτημάτων διαρκείας. Παρουσιάζονται τα βασικά δομικά στοιχεία της αρχιτεκτονικής του και επεξηγείται ο τρόπος λειτουργίας του για την επεξεργασία ρευμάτων δεδομένων.

Το **5ο κεφάλαιο** περιλαμβάνει την επεξήγηση του τρόπου λειτουργίας της παρούσας εφαρμογής και την τεκμηρίωση του κώδικα της υλοποίησής της. Παρουσιάζονται οι απαιτήσεις, τα γενικά στοιχεία και οι περιορισμοί λειτουργίας του συστήματος. Ακλούθως, αναλύονται οι μηχανισμοί πρόβλεψης λαθών του χρήστη και διασφάλισης της συντακτικής ορθότητας των σχεδιαζόμενων ερωτημάτων. Το κεφάλαιο κλείνει με την παρουσίαση των βασικότερων τύπων ερωτημάτων διαρκείας που μπορούν να διατυπωθούν κατά την χρήση του συστήματος χρησιμοποιώντας ως παραδείγματα μια εικονική μελέτη μετρήσεων μετεωρολογικών μεγεθών.

Στο **6ο κεφάλαιο** γίνεται ανασκόπηση της εργασίας και προσπάθεια εξαγωγής χρήσιμων συμπερασμάτων. Ειδικά για το πλαίσιο της υλοποίησης παρουσιάζονται οι μελλοντικές δυνατότητες επέκτασης του συστήματος.

Ευχαριστίες

Με την ολοκλήρωση της διπλωματικής μου εργασίας στο Εργαστήριο Συστημάτων Βάσεων Γνώσεων και Δεδομένων του τμήματος Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του ΕΜΠ, νιώθω τυχερή που ασχολήθηκα με την ανάπτυξη μιας σύγχρονης και πρακτικού ενδιαφέροντος εφαρμογής. Στο τέλος αυτής της δύσκολης και παράλληλα ενδιαφέρουσας διαδρομής, αισθάνομαι έντονη την ανάγκη να ευχαριστήσω όλους όσους με βοήθησαν και μου συμπαραστάθηκαν στην προσπάθειά μου.

Ιδιαίτερη υποχρέωση και θερμές ευχαριστίες οφείλω στον καθηγητή μου κ. Τίμο Σελλή τόσο για την ευκαιρία την οποία μου έδωσε να πραγματοποιήσω την εργασία μου πάνω σε ένα ενδιαφέρον, δημιουργικό και πρακτικό θέμα όσο και για την υπεύθυνη επίβλεψή του σε όλη την πορεία της εργασίας μου.

Επίσης, θέλω να ευχαριστήσω όλους τους καθηγητές που μου παρείχαν τα απαραίτητα εφόδια και γνώσεις κατά τη διάρκεια των σπουδών μου στη Σχολή. Ιδιαιτέρως θέλω να ευχαριστήσω τους καθηγητές κ. Ιωάννη Βασιλείου και κ. Νεκτάριο Κοζύρη οι οποίοι με τις ερωτήσεις τους κατά την εξέταση της εργασίας μου έδωσαν ιδέες για περαιτέρω εμπλουτισμό του παρόντος συγγράμματος.

Ένα πολύ μεγάλο ευχαριστώ θέλω να εκφράσω στον κ. Κώστα Πατρούμπα, υποψήφιο διδάκτορα ΕΜΠ, για την άρτια και μεθοδική συνεργασία μας. Το πραγματικό του ενδιαφέρον, η καθοδήγηση, η επιστημονική στήριξη που μου παρείχε και η πολύτιμη συμβολή του τόσο στο παρόν σύγγραμμα όσο και στην ανάπτυξη του λογισμικού της εφαρμογής υπήρξαν καθοριστικές σε όλη τη διάρκεια της εκπόνησης της διπλωματικής μου εργασίας.

Θα ήταν παράλειψη να μην εκφράσω την τεράστια ευγνωμοσύνη που νιώθω για την γιαγιά μου Όλγα για την αστείρευτη φροντίδα και τη συμπαράστασή της. Την ευχαριστώ όχι μόνο για την αγάπη και την εμπιστοσύνη της που με στήριξε και με ενθάρρυνε καθ' όλη τη διάρκεια των σπουδών μου, αλλά και για τη δύναμη και τα εφόδια που μου έδωσε σε όλη τη ζωή μου.

Δε θα μπορούσα ακόμη να μην αναφερθώ στους φίλους και τις φίλες μου και να τους ευχαριστήσω γιατί βρίσκονται πάντα δίπλα μου.

Τέλος, νιώθω την ανάγκη να αφιερώσω την παρούσα εργασία στη μνήμη της μητέρας μου Μαρίας.

ΣΥΝΟΨΗ

Σε πολλές σύγχρονες εφαρμογές, όπως τα χρηματιστηριακά συστήματα, τα δίκτυα αισθητήρων, παρακολούθηση κινούμενων οχημάτων, εποπτεία φυσικών φαινομένων, κ.α. τα δεδομένα προς επεξεργασία παίρνουν τη μορφή συνεχώς μεταβαλλόμενων ρευμάτων δεδομένων. Για την επεξεργασία τέτοιων στοιχείων διατυπώνονται ερωτήματα διαρκείας που παραμένουν ενεργά για μεγάλο χρονικό διάστημα.

Αντικείμενο της διπλωματικής εργασίας είναι ο σχεδιασμός και η υλοποίηση ενός *γραφικού περιβάλλοντος* διατύπωσης και εκτέλεσης πολλαπλών ερωτημάτων διαρκείας με διάφορους τύπους παραθυρικών δομών. Τα ερωτήματα θα περιλαμβάνουν επιλεγμένους σχεσιακούς τελεστές όπως σύνδεση, συνάθροιση και εφαρμογή κατηγορημάτων για φιλτράρισμα (επιλογή, προβολή) των ρευμάτων εισόδου.

Η διατύπωση των ερωτημάτων γίνεται με σχεδιασμό του προσχεδίου εκτέλεσης πάνω σε μια ειδική φόρμα διατύπωσης. Ο χρήστης της εφαρμογής έχει στη διάθεσή του μια εργαλειοθήκη τελεστών, βάσει των οποίων μπορεί να σχεδιάσει διαγραμματικά τη μορφή του ερωτήματος που επιθυμεί. Πρόκειται ουσιαστικά για το λογικό δένδρο εκτέλεσης, όπου οι τελεστές διασυνδέονται μεταξύ τους με ουρές. Οι ιδιότητες κάθε τελεστή μπορούν να καθοριστούν γραφικά ώστα να ανταποκρίνονται στην ακριβή σημασιολογία του ερωτήματος.

Μετά την ολοκλήρωση του προσχεδίου, η εφαρμογή διατρέχει το δένδρο, ελέγχει την συντακτική ορθότητά του και δημιουργεί τον **SQL** κώδικα του ερωτήματος διαρκείας. Η συντακτική διατύπωση προβάλλεται στη φόρμα και ο χρήστης –εφόσον επιθυμεί– μπορεί να το τροποποιήσει πριν το υποβάλλει. Για την εκτέλεση των ερωτημάτων επιλέχθηκε το πρωτότυπο σύστημα διαχείρισης ρευμάτων δεδομένων TelegraphCQ.

Τα αποτελέσματα της επεξεργασίας έχουν συνεχή ροή και εμφανίζονται σε πίνακες του γραφικού περιβάλλοντος, δυναμικά και σε πραγματικό χρόνο. Επίσης, υπάρχουν οι δυνατότητες αναστολής της εκτέλεσης και επανεκκίνησης κάθε ερωτήματος. Τέλος, υπάρχει η δυνατότητα ταυτόχρονης σχεδίασης, διατύπωσης και εκτέλεσης πολλαπλών ερωτημάτων διαρκείας.

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ:

Ερώτημα διαρκείας
Προσχέδιο εκτέλεσης ερωτήματος
Ρεύμα δεδομένων
Παράθυρο
Τελεστής

ABSTRACT

Many modern monitoring applications, such as systems for financial tickers, sensor networks, traffic surveillance, natural phenomena observation etc, produce massive amounts of dynamic *data streams*. To efficiently manage such data elements, a new processing paradigm has been developed, by expressing long-running *continuous queries*.

The subject of this thesis is to design and implement a *Graphical User Interface* for expressing and executing multiple continuous queries that are being applied against rapidly changing data streams. Such queries include several relational operators such as join, aggregation, filtering, along with various window specifications that determine the range of data examined each time.

Queries are expressed by designing their respective execution plan on a special form in the application. Users have the ability to choose any operator from a toolbar and design their query as a tree diagram, which represents the logical query plan where operators are interconnected with queues. The properties of each operator can be defined graphically according to query semantics.

As soon as the query plan is completed, the application parses the execution tree, checks its syntactic correctness and then generates an **SQL-**like command. The SQL syntax is shown to the user, allowing possible modifications before submitting the query. Finally, the query expression is executed in *TelegraphCQ*, a prototype Data Stream Management System.

Query results have a continuous flow and are shown in tabular format within the graphical interface in real-time. Moreover, it is possible to suspend and resume execution of any continuous query. Besides, this interface offers the ability to design, express and execute multiple continuous queries concurrently.

KEY WORDS:

Continuous query
Query plan
Data stream
Window
Operator

Πίνακας Περιεχομένων

Κεφάλαιο 1

Εισαγωγή στην διαχείριση ρευμάτων δεδομένων

1.1 Γενικά

Σε πολλές σύγχρονες εφαρμογές, όπως είναι τα δίκτυα αισθητήρων, τα χρηματιστηριακά συστήματα, η συλλογή στοιχείων για τη δημοτικότητα ιστοσελίδων, η παρακολούθηση κινούμενων αντικειμένων, κ.ά., η πληροφορία προς επεξεργασία παίρνει τη μορφή ταχύτατα μεταβαλλόμενων *ρευμάτων δεδομένων* (*data streams*). 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.

Η εξάπλωση αυτών των εφαρμογών όπου μεγάλοι όγκοι δεδομένων καταφθάνουν με κυμαινόμενο ρυθμό και απαιτούν επεξεργασία σε πραγματικό χρόνο, καθιστά τα κλασικά σχεσιακά Συστήματα Διαχείρισης Βάσεων Δεδομένων μη ικανά να ανταποκριθούν στις νέες συνθήκες καθώς έχουν σχεδιαστεί υπό ένα στατικότερο πρίσμα, όπου:

- τα δεδομένα βρίσκονται αποθηκευμένα στο σύστημα, έτσι ώστε να είναι πάντα προσβάσιμα
- ενδιαφέρει μόνο η παρούσα κατάστασή τους
- δίνουν ιδιαίτερη βαρύτητα στην ακρίβεια των αποτελεσμάτων επεξεργασίας και όχι στο ρυθμό παραγωγής τους.

Η εμφανής ανεπάρκεια των ΣΔΒΔ σε τέτοια ρευστά περιβάλλοντα πληροφορίας επιδιώκεται να καλυφθεί από τα Συστήματα Διαχείρισης Ρευμάτων Δεδομένων (ΣΔΡΔ), τα οποία παρέχουν την απαιτούμενη ευελιξία και προσαρμοστικότητα. Για την επεξεργασία των ρευμάτων δεδομένων διατυπώνονται ερωτήματα διαρκείας. Τα τελευταία εκτελούνται συνεχώς και τα αποτελέσματά τους ανανεώνονται με την άφιξη νέων δεδομένων. Τα ΣΔΡΔ

επιχειρούν να αντιμετωπίσουν κατάλληλα τις έντονες ανάγκες επεξεργασίας, τις διακυμάνσεις στο φόρτο εργασίας, στο ρυθμό άφιξης των δεδομένων αλλά και όποιες μεταβολές προκύψουν από εξωτερικούς παράγοντες (απαιτήσεις χρηστών, ποιότητα μετάδοσης κλπ.).

1.2 Ρεύματα Δεδομένων

Ως **ρεύμα δεδομένων** (*data stream*) μπορεί να θεωρηθεί μια ακολουθία στοιχείων που παράγονται διαρκώς από κάποια πηγή δεδομένων. Τα στοιχεία του ρεύματος διατηρούν την γνώριμη μορφή της σχεσιακής πλειάδας (*tuple*) που συνοδεύεται από χρονόσημο (*timestamp*). Δεν βρίσκονται λοιπόν αποθηκευμένα σε στατικούς πίνακες, αλλά διαρκώς καταφθάνουν στο σύστημα με κυμαινόμενο ρυθμό, οπότε το μέγεθος του ρεύματος μπορεί να είναι απεριόριστο. Το σύστημα δεν ελέγχει τη διάταξη ή τον ρυθμό με τον οποίο καταφθάνουν τα δεδομένα. Κάθε στοιχείο του ρεύματος μετά την επεξεργασία του είτε αποθηκεύεται είτε απορρίπτεται. Επομένως, μελλοντικές προσπελάσεις του είναι είτε ιδιαίτερα δαπανηρές είτε εντελώς αδύνατες.

Η πηγή δεδομένων μπορεί να είναι ένας αισθητήρας μέτρησης φυσικών μεγεθών (πχ. θερμοκρασία), διαδικτυακές συναλλαγές που γίνονται μέσω ροών πακέτων, μια μετοχή της οποίας η τιμή παρακολουθείται συνεχώς, κλπ. Οι εφαρμογές παρακολούθησης στις οποίες εμφανίζεται συνήθως η ανάγκη για δυναμική επεξεργασία ρευμάτων δεδομένων αναπτύχθηκαν βασιζόμενες στην μεγάλη εξάπλωση των δικτύων. Αυτές επεξεργάζονται, αναλύουν και αντιδρούν σχεδόν άμεσα σε ποικίλες μορφές δεδομένων.

Βασικότερα χαρακτηριστικά των ρευμάτων δεδομένων είναι ότι παράγονται συνεχώς, με μεταβλητό ρυθμό, άγνωστο μέγεθος, αβέβαιες συνθήκες μετάδοσης ενώ επιβάλλεται η επεξεργασία τους με την μικρότερη δυνατή καθυστέρηση.

1.3 Ερωτήματα

Τα ***ερωτήματα (queries)*** που τίθενται σε ρεύματα δεδομένων μοιάζουν αρκετά μ' εκείνα που υποβάλλονται στα περιεχόμενα μιας τυπικής βάσης δεδομένων. Είναι το βασικό εργαλείο επεξεργασίας των δεδομένων και παρουσιάζουν κάποια νέα χαρακτηριστικά σε σχέση με αυτά των κλασικών σχεσιακών βάσεων δεδομένων.

Τα ερωτήματα μπορούν να κατηγοριοποιηθούν ανάλογα με την διάρκεια κατά την οποία παραμένουν ενεργά. Γίνεται διάκριση σε *ερωτήματα στιγμιοτύπου (snapshot ή one-time queries)* που εξετάζουν την κατάσταση των δεδομένων του συστήματος μόνο τη χρονική στιγμή της υποβολής τους (pull model) και σε *ερωτήματα διαρκείας (continuous queries)* που εκτελούνται για μεγάλα χρονικά διαστήματα μέχρι να αποφασιστεί η αναστολή ή ο τερματισμός τους. Αυτά μπορούν να επιστρέφουν δεδομένα χωρίς να απαιτείται η επαναλαμβανόμενη υποβολή τους. Τα ερωτήματα διαρκείας αποτελούν το βασικότερο εργαλείο επεξεργασίας των ρευμάτων δεδομένων και σ' αυτά επικεντρώνεται η εργασία αυτή.

Σε αντίθεση με τα κλασικά συστήματα διαχείρισης βάσεων (ΣΔΒΔ) όπου το σύνολο των δεδομένων είναι προκαθορισμένο και η υποβολή ερωτημάτων σε αυτά εκκινεί τους υπολογισμούς, σε ένα σύστημα διαχείρισης ρευμάτων (ΣΔΡΔ) υπάρχει ένα ενεργό σύνολο ερωτημάτων στο οποίο τροφοδοτούνται τα ρεύματα ενώ ο ρυθμός υπολογισμού των απαντήσεων θα πρέπει να συμβαδίζει με το ρυθμό άφιξης των δεδομένων.

Η διαρκής εκτέλεση των ερωτημάτων σε συνδυασμό με την αβεβαιότητα ως προς το μέγεθος των δεδομένων και το ρυθμό άφιξής τους, προϋποθέτουν μια διαφορετική προσέγγιση όσο αναφορά τη σημασιολογία, το είδος των χρησιμοποιούμενων τελεστών

αλλά και το τεχνολογικό υπόβαθρο επεξεργασίας τους. Η διάθεση των πόρων του συστήματος, η κατάστρωση των πλάνων εκτέλεσης *(query plan)*, ο ορισμός νέων δομών όπως των χρονικών παραθύρων, η ανάπτυξη αλγορίθμων επεξεργασίας που πραγματοποιούν την μεγαλύτερη δυνατή συλλογή πληροφορίας με ένα μόνο πέρασμα των δεδομένων, η διατύπωση των ερωτημάτων είναι κάποιοι από τους τομείς όπου απαιτείται αναθεώρηση και σχεδιασμός σε ένα ΣΔΡΔ.

Στην εκτέλεση των ερωτημάτων διαρκείας θα πρέπει να τονιστεί ο ρόλος των *παραθύρων (windows)*. Πρόκειται για δομές που επιτρέπουν την απόσπαση πεπερασμένου πλήθους πλειάδων από το απειράριθμο ρεύμα. Από την μία λοιπόν μετριάζουν τον όγκο των δεδομένων που θα συμμετάσχουν στον υπολογισμό της απάντησης, ενώ από την άλλη δίνουν έμφαση στα πιο πρόσφατα στοιχεία, κάτι χρήσιμο σε πολλές εφαρμογές.

Τα ερωτήματα προσδιορίζονται και βάσει του χρόνου υποβολής τους σε σχέση με τον χρόνο άφιξης των δεδομένων. Έτσι, τα ερωτήματα που είναι γνωστά εκ των προτέρων (πριν ξεκινήσει η εισροή των δεδομένων) χαρακτηρίζονται ως *προκαθορισμένα (predefined)* και συνήθως είναι ερωτήματα διαρκείας. Σε αντιδιαστολή, τα *περιστασιακά ερωτήματα (ad hoc)* υποβάλλονται online σε κάποια επόμενη φάση, αφού τα ρεύματα δεδομένων αρχίσουν να εισέρχονται στο σύστημα.

1.4 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-GUI*).

Τα ερωτήματα διαρκείας εστιάζουν συνήθως στην επεξεργασία της πιο πρόσφατης πληροφορίας, εγείροντας την ανάγκη για χρονική σήμανση των στοιχείων, καθώς και 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.

Οι **τελεστές** (*operators*) των συμβατικών συστημάτων βάσεων δεδομένων παρουσιάζουν αρκετές δυσκολίες κατά την προσαρμογή τους σε ρεύματα δεδομένων. Ορισμένοι τελεστές μπορούν να εφαρμοστούν απευθείας στο ρεύμα δεδομένων, διότι μπορούν να επεξεργαστούν αυτοτελώς κάθε πλειάδα του, ενώ άλλοι χρειάζονται διαρκώς πρόσβαση σε όλα τα δεδομένα εισόδου για να εξάγουν σωστά αποτελέσματα. Όμως, η αποθήκευση όλων των πλειάδων ενός ρεύματος συνήθως δεν είναι εφικτή.

Για την αντιμετώπιση αυτού του προβλήματος η τροφοδότηση τέτοιων τελεστών γίνεται από **παράθυρα** (*windows*). Σκοπός των παραθυρικών δομών είναι να αποσπούν συνεχώς κάποιο πεπερασμένο πλήθος στοιχείων του ρεύματος παρέχοντας διαρκώς πρόσβαση σε περιορισμένο τμήμα του. 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.

Η τροφοδότηση των προβληματικών τελεστών με παράθυρα έχει ως αποτέλεσμα την απεμπλοκή τους, με αντίτιμο την αποδοχή προσεγγίσεων στις επιστρεφόμενες απαντήσεις. 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, 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, με την χρήση παραθύρων 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.

1.5 Ανεπάρκεια παραδοσιακών ΣΔΒΔ

Υπάρχει μια σειρά αιτιών που καθιστούν τα παραδοσιακά Συστήματα διαχείρισης Βάσεων Δεδομένων ανεπαρκή για τη διαχείριση και επεξεργασία των ρευμάτων δεδομένων. Οι σημαντικότερες από αυτές είναι:

- **Μεταβλητότητα και όγκος δεδομένων**. Τα δεδομένα δε βρίσκονται στατικά αποθηκευμένα στο δίσκο απ' όπου μπορούν να ανακληθούν κατά βούληση, αλλά εισέρχονται στο σύστημα με την μορφή ρευμάτων. Συνεπώς το σύστημα επεξεργασίας δεν είναι σε θέση να διευθύνει τον χειρισμό των δεδομένων αλλά καλείται να αντιδράσει όσο το δυνατόν αποτελεσματικότερα στην άφιξή τους. Επίσης, δεν είναι δυνατή η έγκυρη προετοιμασία στατιστικών στοιχείων σχετικών με τα δεδομένα ώστε να διευκολυνθεί η εκτέλεση των ερωτημάτων. Τέλος, σημαντικό ρόλο παίζει και η χρονική διάταξη των στοιχείων των ρευμάτων η οποία οδηγεί σε σημαντικές τροποποιήσεις στη σύνταξη των ερωτημάτων.

- **Ερωτήματα διαρκείας**. Τα ερωτήματα διαρκείας είναι συνεχώς ενεργά και λειτουργούν ως "φίλτρα" στα εισερχόμενα δεδομένα. Στα παραδοσιακά συστήματα ένα νέο ερώτημα απευθύνεται σε ένα σύνολο αποθηκευμένων δεδομένων, ενώ εδώ τα εισερχόμενα δεδομένα απευθύνονται σε ένα σύνολο προϋπαρχόντων ερωτημάτων διαρκείας. Το απεριόριστο μέγεθος των ρευμάτων απαιτεί την τροποποίηση τελεστών όπως των συναθροιστικών (COUNT, MAX, MIN, κλπ), την εισαγωγή μηχανισμών που περιορίζουν το μέγεθος των προς επεξεργασία δεδομένων (παράθυρα) , ενώ επιβάλλει την εισαγωγή της έννοιας της προσέγγισης στην παραγωγή των απαντήσεων. Ακόμη, επειδή τα ερωτήματα μπορεί να παραμείνουν ενεργά για μεγάλα χρονικά διαστήματα, θα πρέπει να ανταποκρίνονται σε αλλαγές του φόρτου εργασίας, του ρυθμού άφιξης και των χαρακτηριστικών των δεδομένων.

- **Κοινή επεξεργασία**. Ο πιθανά μεγάλος αριθμός ενεργών ερωτημάτων διαρκείας σε συνδυασμό με την ανάγκη εξαγωγής αποτελεσμάτων σε πραγματικό χρόνο, οδηγεί στην ανάπτυξη μηχανισμών που να επιτρέπουν την κοινή επεξεργασία δεδομένων. Δηλαδή το κοινό τμήμα επεξεργασίας των ερωτημάτων πραγματοποιείται μόνο μια

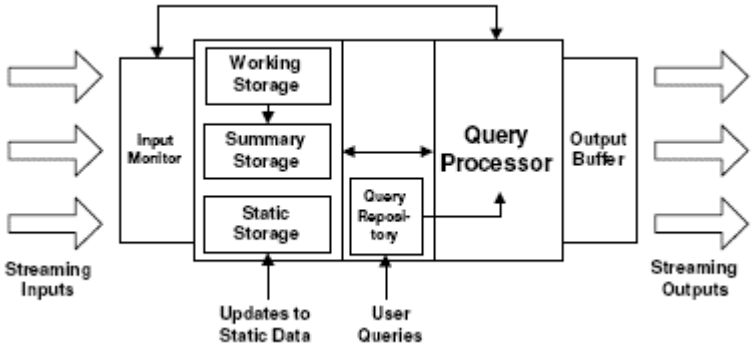
φορά για τα ίδια δεδομένα. Επίσης, θα πρέπει το σύστημα να χειρίζεται αποτελεσματικά την ασύγχρονη υποβολή, αναστολή και τροποποίηση των ερωτημάτων διαρκείας.

- **Απρόβλεπτες συνθήκες άφιξης.** Το περιβάλλον όπου χρησιμοποιούνται συνήθως ρεύματα δεδομένων είναι αρκετά ευάλωτο σε απώλειες δεδομένων που οφείλονται σε σφάλματα συσκευών ή μετάδοσης. Συνεπώς ο ρυθμός άφιξης δεδομένων στο σύστημα δεν είναι προβλέψιμος.

1.6 Επισκόπηση κυριότερων συστημάτων ρευμάτων δεδομένων

Τα συστήματα διαχείρισης ρευμάτων δεδομένων παρουσιάζουν αρκετές ομοιότητες με αυτά που χρησιμοποιούν υλοποιημένες όψεις *(materialized views)*. Οι τελευταίες θυμίζουν αρκετά ερωτήματα διαρκείας τόσο όσο αναφορά την «αυτοσυντήρηση» τους, δηλαδή την αποθήκευση αρκετών δεδομένων ώστε να διατηρείται η όψη ακόμα κι όταν η Βάση Δεδομένων δεν είναι διαθέσιμη όσο και τη «λήξη» δεδομένων, δηλαδή πότε κάποιο δεδομένο μπορεί να θεωρηθεί άχρηστο και να απομακρυνθεί. Οι σημαντικότερες διαφορές ανάμεσα στις όψεις και τα ερωτήματα διαρκείας είναι ότι τα αποτελέσματα των τελευταίων μπορούν να εξάγονται με τη μορφή ρευμάτων δεδομένων. Επίσης, τα ερωτήματα διαρκείας μπορούν να εκτελούνται και μόνο με append-only δεδομένα, δεν ενδιαφέρονται τόσο για την ακρίβεια των αποτελεσμάτων και θα πρέπει να μπορούν να προσαρμόζονται στα μεταβαλλόμενα χαρακτηριστικά των δεδομένων.

Υλοποιημένα Συστήματα Διαχείρισης Ρευμάτων Δεδομένων είναι τα: ***TelegraphCQ, STREAM, Aurora***. Το TelegraphCQ εστιάζει στην ευελιξία και προσαρμοστικότητα εκτέλεσης ερωτημάτων διαρκείας, εισάγοντας τον μηχανισμό *Eddy*. Απευθύνεται σε έντονα και ασταθή περιβάλλοντα πληροφορίας. Στην ανάπτυξη του STREAM ιδιαίτερη βαρύτητα δόθηκε στην αποτελεσματική εκμετάλλευση περιορισμένης μνήμης κατά την επεξεργασία των ερωτημάτων. Εξετάζει τρόπους παραγωγής προσεγγιστικών απαντήσεων και επιχειρεί να προσδιορίσει τις απαιτήσεις σε μνήμη. Τέλος, το Aurora επιδιώκει να παρέχει στους χρήστες τη δυνατότητα να επιλέγουν δυναμικά την ποιότητα υπηρεσιών που επιθυμούν (Quality of Service) για την υλοποίηση των ερωτημάτων και στη συνέχεια να την χρησιμοποιούν για την κατανομή του φόρτου εργασίας.



Σχήμα 1.1: *Γενικό διάγραμμα Αρχιτεκτονικής Συστημάτων Διαχείρισης Ρευμάτων Δεδομένων. (Πηγή: [GO03])*

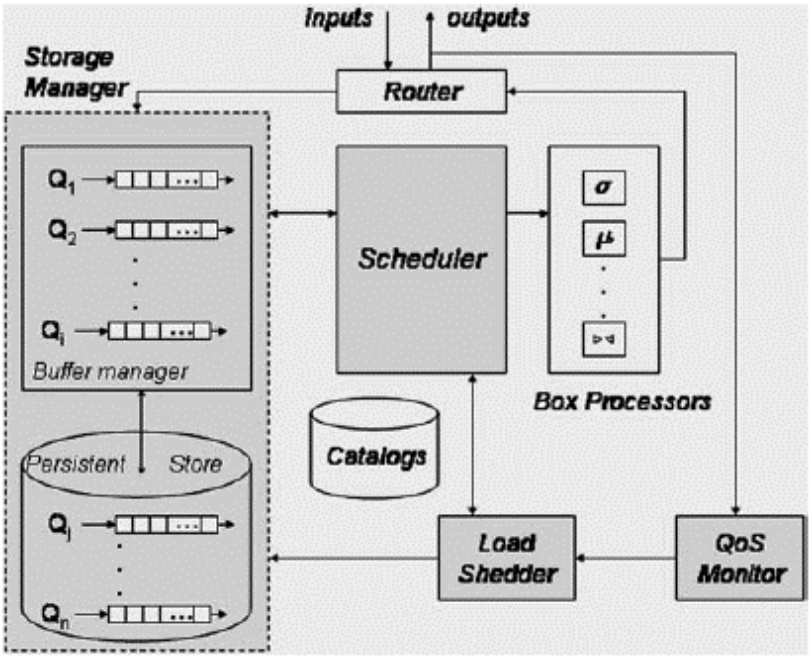
1.6.1 Aurora

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. Προτείνεται μία προσέγγιση βασισμένη σε ένα γραφικό περιβάλλον με «κουτιά» και «βέλη» θυμίζοντας ένα διάγραμμα ροής δεδομένων αναπαριστώντας το προσχέδιο εκτέλεσης κάποιου ερωτήματος διαρκείας **[ACC+03b]**.

Ο *χρονοπρογραμματιστής* (*scheduler*) αποτελεί το συνδετικό κρίκο όλων των τμημάτων της αρχιτεκτονικής του συστήματος. 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.



Σχήμα 1.2: *Αρχιτεκτονική του συστήματος Aurora. (Πηγή: [ACC+03b])*

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* τελεστών.

1.6.2 Gigascope

Πρόκειται για ένα σύστημα διαχείρισης ρευμάτων δεδομένων που αναπτύσσεται από την AT&T σε συνεργασία με το Πανεπιστήμιο Carnegie Mellon **[CJSS03]**. 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.

1.6.3 STREAM (STanford stREam datA Management)

Το STREAM είναι ένα Σύστημα Διαχείρισης Ρευμάτων Δεδομένων γενικού σκοπού, το οποίο αναπτύσσεται από το 2001 στο Πανεπιστήμιο Stanford 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 **[MWA+03]**. Η ανάπτυξη έχει τρεις κυρίως στόχους: (α) Ένα ευέλικτο τρόπο *διεπαφής* (*interface*) προκειμένου να διευκολύνεται η ανάγνωση και η εγγραφή ρευμάτων δεδομένων, ως μέρος μιας ιεραρχικής διαχείρισης του αποθηκευτικού χώρου. (β) Την αποτελεσματική επεξεργασία των ερωτημάτων διαρκείας που διατυπώνονται σε SQL ή με τελεστές της σχεσιακής άλγεβρας, συμπεριλαμβανομένων των *συναθροιστικών* (*aggregation*). (γ) Ένα περιβάλλον API για την υποβολή των ερωτημάτων διαρκείας και τη λήψη των απαντήσεων σ' αυτά.

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).

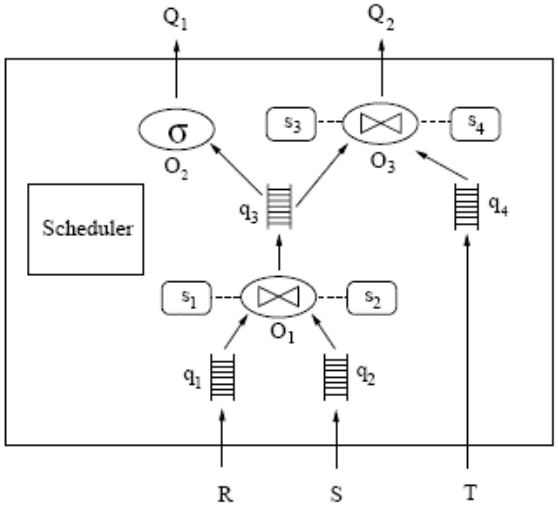
Η εκτέλεση των ερωτημάτων ρυθμίζεται από έναν καθολικό *χρονοπρογραμματιστή* (*global scheduler*). Στην τρέχουσα υλοποίηση του συστήματος, χρησιμοποιείται ένα σχήμα round-robin για να προχωρά απρόσκοπτα η εκτέλεση σε όσους τελεστές είναι έτοιμοι, αν και φυσικά μπορεί να υλοποιηθούν περισσότερο πολύπλοκες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



Σχήμα 1.3: *Προσχέδια εκτέλεσης του συστήματος STREAM για τα ερωτήματα διαρκείας Q_1 και Q_2 πάνω στα ρεύματα δεδομένωνR, S και T. (Πηγή: [MWA+03])*

1.6.4 TelegraphCQ

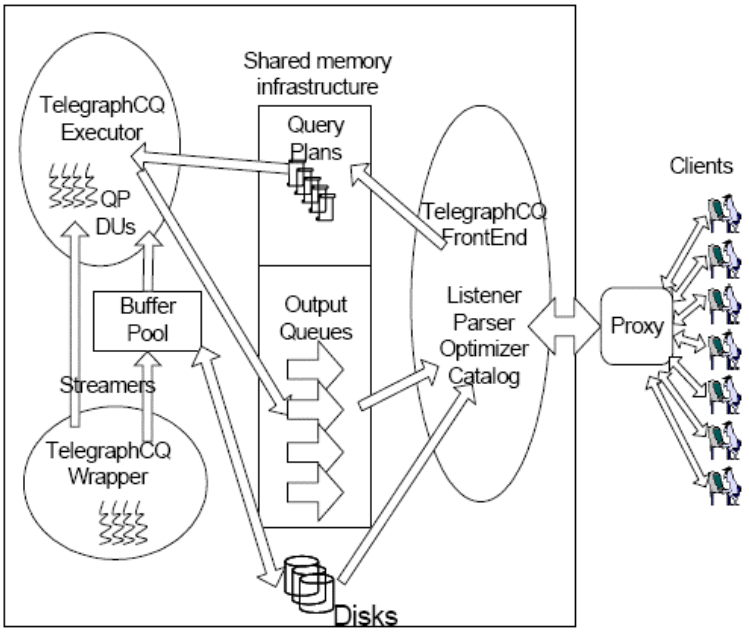
Το σύστημα αυτό αποτελεί μετεξέλιξη του πρωτοτύπου Telegraph που συντονίζεται από το Πανεπιστήμιο Berkeley ήδη από το 2000, με κύριο στόχο την ανάπτυξη μιας αρχιτεκτονικής προσαρμοζόμενης στη ροή των δεδομένων κυρίως σε δικτυακά περιβάλλοντα, με έμφαση στα δίκτυα αισθητήρων **[CCD+03]**. Ωστόσο, η αναγκαιότητα 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 αφορούν το *εξωτερικό μέρος* (*Front End*) της επικοινωνίας με τους χρήστες και τα ερωτήματα που θέτουν στο σύστημα.

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 Loadbalancing 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.



Σχήμα 1.4: *Αρχιτεκτονική του συστήματος TelegraphCQ. (Πηγή:[CCD+03])*

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.

1.7 Αντικείμενο της εργασίας

Στην παρούσα διπλωματική εργασία επιδιώκεται η υλοποίηση ενός *γραφικού περιβάλλοντος* (GUI) διατύπωσης και εκτέλεσης πολλαπλών ερωτημάτων διαρκείας με διάφορους τύπους παραθυρικών δομών. Τα ερωτήματα θα περιλαμβάνουν επιλεγμένους σχεσιακούς τελεστές όπως σύνδεση (join), συνάθροιση (aggregate) και εφαρμογή κατηγορημάτων για φιλτράρισμα των ρευμάτων εισόδου.

Η σύνθεση των προσχεδίων εκτέλεσης ερωτημάτων γίνεται σε μορφή **δένδρων** όπου κόμβοι είναι οι τελεστές του ερωτήματος, ενώ τα *βέλη* που τους συνδέουν δείχνουν τη λογική διασύνδεσή τους. Η φορά του βέλους δηλώνει τροφοδότηση με δεδομένα από το ρεύμα εξόδου του ενός στο ρεύμα εισόδου του επόμενου τελεστή στην ακολουθία.

Στόχος είναι η υποστήριξη όλων των βασικών τύπων τελεστών και παραθύρων για ρεύματα δεδομένων, όπως:

- Προβολή (projection) [π],
- Επιλογή (selection) [σ],
- Σύνδεση (join) [⋈],
- Συνάθροιση (aggregation) [γ],
- Απαλοιφή διπλοτύπων (duplicate elimination) [δ].
- Παράθυρα [ω] διαφόρων τύπων.

Από το γραφικά σχεδιασμένο προσχέδιο εκτέλεσης θα παράγεται ο κώδικας του ερωτήματος διαρκείας σε γλώσσα SQL, προσαρμοσμένη όμως στο συντακτικό του Συστήματος Διαχείρισης Ρευμάτων Δεδομένων που χρησιμοποιείται.

Τα αποτελέσματα του ερωτήματος, δηλαδή οι πλειάδες του ρεύματος εξόδου, θα πρέπει να έχουν μια συνεχή ροή και να εμφανίζονται σε πίνακες δυναμικά και σε πραγματικό χρόνο.

Τέλος, τονίζεται ότι θα υπάρχει η δυνατότητα παράλληλης σχεδίασης, διατύπωσης και εκτέλεσης πολλαπλών ερωτημάτων διαρκείας.

Για την υλοποίηση του περιβάλλοντος χρήστη επιλέχτηκε η γλώσσα προγραμματισμού JAVA ενώ για την εκτέλεση των παραγόμενων ερωτημάτων αξιοποιήθηκε ένας ήδη υπάρχων μηχανισμός επεξεργασίας ρευμάτων δεδομένων (TelegraphCQ).

Κεφάλαιο 2

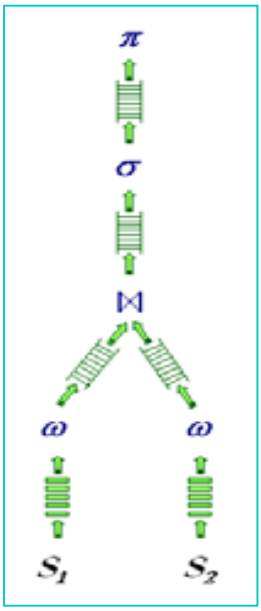
Τελεστές σε ρεύματα δεδομένων

2.1 Εισαγωγή

Για την διατύπωση των ερωτημάτων διαρκείας συνήθως γίνεται χρήση κάποιας γλώσσας παραπλήσιας με την SQL. Εναλλακτικά η διατύπωση αυτή μπορεί να γίνει μέσω κάποιου γραφικού περιβάλλοντος που επικοινωνεί με ένα Σύστημα Διαχείρισης Ρευμάτων Δεδομένων. Ακολουθεί η επιλογή από το σύστημα του είδους και της διάταξης των τελεστών (προσχέδιο εκτέλεσης) και των παραθύρων που θα χρησιμοποιηθούν κατά την εκτέλεση του ερωτήματος.

Οι τελεστές που παίρνουν μέρος στα ερωτήματα διαρκείας προέρχονται από τελεστές της σχεσιακής άλγεβρας, κατάλληλα τροποποιημένους. Αρκετοί τελεστές όμως παρουσιάζουν ιδιαίτερα προβλήματα κατά την προσαρμογή τους στο περιβάλλον επεξεργασίας ρευμάτων δεδομένων. Επιπλέον, οι περιορισμένοι διαθέσιμοι πόροι του συστήματος, οι απαιτήσεις των ερωτημάτων διαρκείας για άμεσες και συνεχείς αποκρίσεις καθώς και η ίδια η φύση των ρευμάτων, χωρίς διάταξη και με πιθανώς άπειρο πλήθος στοιχείων, προκαλούν επιπρόσθετους προβληματισμούς κατά την επιλογή κατάλληλων προσχεδίων για τα ερωτήματα και κατά την εκτέλεσή τους. Προκύπτουν έτσι ζητήματα σχετικά με την υλοποίηση των τελεστών, την διάταξη των στοιχείων από τα εισερχόμενα ρεύματα και την πολιτική χρονοδρομολόγησης των τελεστών.

Στο *σχήμα 2.1* φαίνεται η μορφή ενός απλού προσχεδίου εκτέλεσης για ένα ερώτημα διαρκείας το οποίο διατυπώνεται πάνω σε δυο ρεύματα δεδομένων και στο οποίο συμμετέχουν τελεστές προβολής επιλογής και σύνδεσης. Οι δυο ειδικοί τελεστές που εφαρμόζονται άμεσα πάνω στα ρεύματα και παρεμβάλλονται πριν τις εισόδους του τελεστή σύνδεσης, συμβολίζονται με το λατινικό γράμμα W ονομάζονται *παράθυρα* και η λειτουργία τους θα αναλυθεί σε επόμενες ενότητες.



Σχήμα 2.1: *Προσχέδιο εκτέλεσης ερωτήματος διαρκείας*

2.2 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.

Στην συνέχεια παρουσιάζονται οι κυριότεροι τελεστές και ο επιδιωκόμενος τρόπος λειτουργίας του.

2.2.1 Προβολή

Ο τελεστής προβολής δίνει ως αποτέλεσμα ορισμένα μόνο επιλεγμένα γνωρίσματα από το σύνολο των γνωρισμάτων που περιέχονται στις πλειάδες ενός ρεύματος δεδομένων, ενώ δεν θα πρέπει να παραλείπεται κάποια πλειάδα από το αποτέλεσμα.

Παράδειγμα 2.1: Έστω ότι υπάρχει ένα ρεύμα δεδομένων κάποιας χρηματιστηριακής εφαρμογής με στοιχεία που περιλαμβάνουν πληροφορία για το όνομα κάθε μετοχής, την τρέχουσα τιμή της, το όνομα και την πόλη του χρηματιστηρίου στο οποίο συμμετέχει και τη χρονική στιγμή που μετρήθηκε η συγκεκριμένη τιμή. Αν θέλουμε να κρατήσουμε τα στοιχεία μόνο του ονόματος κάθε μετοχής και της τρέχουσας τιμής της, τότε αυτό το "φιλτράρισμα" πραγματοποιείται με τη χρήση ενός τελεστή προβολής με είσοδο το ρεύμα δεδομένων που αναφέρθηκε και επιλεγόμενα πεδία εκείνα που ενδιαφέρουν, δηλαδή το όνομα και την τιμή κάθε μετοχής.

2.2.2 Επιλογή

Ο τελεστής αυτός δίνει ως αποτέλεσμα ορισμένες μόνο από το σύνολο των πλειάδων ενός ρεύματος δεδομένων, ανάλογα με το αν ικανοποιούν ή όχι μια ορισμένη συνθήκη. Το

κατηγόρημα της συνθήκης της επιλογής αφορά κάποια ή κάποιες από τις τιμές των γνωρισμάτων των πλειάδων του ρεύματος.

Παράδειγμα 2.2: Έστω ότι στο ρεύμα του παραδείγματος 2.1 θέλουμε να κρατήσουμε μόνο τα στοιχεία του ρεύματος για τα οποία οι τιμές των μετοχών είναι πάνω από μια καθορισμένη τιμή, ή ότι θέλουμε να μελετήσουμε μόνο εκείνες που ανήκουν στο χρηματιστήριο Αθηνών, ή ακόμη μπορεί να θέλουμε να επεξεργαστούμε μόνο κάποιες συγκεκριμένες μετοχές. Όλα αυτά τα ερωτήματα μπορούν να υλοποιηθούν με κατάλληλο φιλτράρισμα που γίνεται με τη χρήση του τελεστή επιλογής, ο οποίος θα έχει ως είσοδο αυτό το ρεύμα δεδομένων.

Στην πρώτη περίπτωση το κατηγόρημα της επιλογής θα έχει την μορφή: « τιμή μετοχής > κατώτατο όριο» , όπου ως όριο θέτουμε την κατώτερη επιθυμητή τιμή. Στην δεύτερη περίπτωση το κατηγόρημα θα έχει τη μορφή: «πόλη = "ΑΘΗΝΑ"» ενώ στην τελευταία περίπτωση θα πάρει την μορφή «όνομα μετοχής = *όνομα1* AND όνομα μετοχής = *όνομα2* AND ... »

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.

2.2.3 Απαλοιφή διπλοτύπων

Ο τελεστής αυτός θα πρέπει να επιστρέφει την πρώτη εμφάνιση κάθε πλειάδας., απαλείφοντας κάθε άλλο ακριβές αντίγραφο της πλειάδας αυτής.

Παράδειγμα 2.3: Έστω ότι στο ρεύμα του παραδείγματος 2.1 θέλουμε να κρατήσουμε μόνο τα στοιχεία του ρεύματος για τα οποία παρατηρούμε διαφορές στις τιμές κάθε μετοχής. Με εφαρμογή του τελεστή απαλοιφής διπλοτύπων θα λαμβάνουμε λοιπόν πληροφορία για κάθε μετοχή μόνο όταν πάρει διαφορετική τιμή από αυτές που έχει πάρει κατά το παρελθόν.

2.2.4 Συνάθροιση

Όπως και για τον ανάλογο τελεστή της εκτεταμένης σχεσιακής άλγεβρας, πρέπει να προηγηθεί ομαδοποίηση των πλειάδων που εφαρμόζεται στα στοιχεία εισόδου του. Η ομαδοποίηση θα πρέπει να γίνει σύμφωνα με τις τιμές των στοιχείων στα γνωρίσματα των πλειάδων του ρεύματος, όπως προσδιορίζονται στη λίστα των γνωρισμάτων ομαδοποίησης του τελεστή. Για κάθε ομάδα, δηλαδή συνδυασμό τιμών των γνωρισμάτων ομαδοποίησης, εφαρμόζεται μια από τις συναρτήσεις συνάθροισης: καταμέτρηση (COUNT), υπολογισμός αθροίσματος (SUM), εύρεση ελαχίστου (MIN), εύρεση μεγίστου(MAX) και υπολογισμός μέσου όρου (AVG).

Παράδειγμα 2.4: Αν θεωρήσουμε ξανά το ρεύμα του παραδείγματος 2.1 και θέλουμε να εξάγουμε πληροφορίες οι οποίες αφορούν τον μέσο όρο των τιμών κάθε μετοχής, τότε θα πρέπει οι πλειάδες του ρεύματος να ομαδοποιηθούν με βάση το όνομα μετοχής και στη συνέχεια να εφαρμοστεί στα στοιχεία κάθε ομάδας η συνάρτηση AVG. Η διαδικασία αυτή υλοποιείται με την εφαρμογή ενός τελεστή συνάθροισης στο ρεύμα δεδομένων του παραδείγματος.

2.2.5 Σύνδεση

Ο σύνθετος αυτός τελεστής, εφαρμόζεται μεταξύ δύο ρευμάτων S_1 και S_2 είτε μεταξύ ρεύματος S και σχεσιακού πίνακα R και έχει παρόμοια λειτουργία με τον αντίστοιχο σχεσιακό τελεστή. Στο αποτέλεσμα προβάλλεται ο συνδυασμός των πλειάδων των δύο ρευμάτων τα οποία ικανοποιούν τη *συνθήκη σύνδεσης*. Η συνθήκη αυτή είναι μια ισότητα ή ανισότητα μεταξύ των γνωρισμάτων των δύο ρευμάτων.

Παράδειγμα 2.5: Έστω δύο ρεύματα δεδομένων που αφορούν την ίδια χρηματιστηριακή εφαρμογή του παραδείγματος 2.1, με τη διαφορά ότι αναφέρονται σε διαφορετικά χρηματιστήρια το καθένα. Αν θέλουμε να δούμε ποιες από τις μετοχές του ενός έτυχε να πάρουν ίδια τιμή με μετοχές του άλλου, τότε εφαρμόζεται τελεστής σύνδεσης με εισόδους τα δύο αυτά ρεύματα και συνθήκη σύνδεσης « τιμή μετοχής από το πρώτο ρεύμα = τιμή μετοχής από το δεύτερο ρεύμα».

2.2.6 Συνολοθεωρητικοί τελεστές

Οι γνωστές συνολοθεωρητικές πράξεις εφαρμοζόμενες σε στοιχεία ρευμάτων ορίζουν τους εξής τελεστές:

- Ο τελεστής ένωσης (union), ο οποίος πρέπει να δίνει ως απάντηση, κάθε πλειάδα που περιέχεται στα δυο ρεύματα δεδομένων.
- Ο τελεστής τομής (intersection). Στο σύνολο της απάντησης πρέπει να περιλαμβάνεται κάθε πλειάδα που περιέχεται και στα δύο ρεύματα.
- Ο τελεστής διαφοράς (difference), ο οποίος πρέπει να διατηρεί στην απάντηση κάθε πλειάδα που περιέχεται στο ένα ρεύμα και δεν περιλαμβάνεται στο δεύτερο.

Παράδειγμα 2.6: Αν στα δύο ρεύματα του προηγούμενου παραδείγματος εφαρμόσουμε τελεστή ένωσης, τότε το αποτέλεσμα θα περιλαμβάνει τις διαθέσιμες πληροφορίες για κάθε μετοχή και των δύο χρηματιστηρίων.

Παράδειγμα 2.7: Με την υπόθεση ότι κάποιες μετοχές μπορούν να συμμετάσχουν σε περισσότερα του ενός χρηματιστήρια, τότε αν στα δύο ρεύματα του προηγούμενου παραδείγματος εφαρμόσουμε τελεστή τομής, το αποτέλεσμα θα περιλαμβάνει τις διαθέσιμες πληροφορίες για τις μετοχές που συμμετέχουν παράλληλα και στα δύο χρηματιστήρια. Ενώ αν εφαρμόσουμε τελεστή διαφοράς, τότε το αποτέλεσμα θα περιλαμβάνει τις διαθέσιμες πληροφορίες για τις μετοχές που συμμετέχουν στο πρώτο χρηματιστήριο, αλλά όχι στο δεύτερο.

2.3 Προβλήματα στην εφαρμογή τελεστών σε ρεύματα δεδομένων

Οι ειδικές συνθήκες κάτω από τις οποίες καλούνται να λειτουργήσουν οι διάφοροι τελεστές όταν εφαρμόζονται σε ρεύματα δεδομένων, προκαλούν αρκετούς προβληματισμούς πάνω στον σχεδιασμό και την υλοποίησή τους.

Τα κυριότερα προβλήματα που προκύπτουν συνδέονται με τα εξής:

- Η πιθανή έλλειψη διάταξης ή συγχρονισμού των εισερχόμενων ρευμάτων.
- Η συνεχής άφιξη νέων στοιχείων πληροφορίας, ενώ το σύστημα ήδη επεξεργάζεται άλλα προγενέστερα δεδομένα.

- Το πιθανόν άπειρο πλήθος στοιχείων προς επεξεργασία, σε συνδυασμό με τους περιορισμούς του συστήματος σε διαθέσιμους πόρους μνήμης και επεξεργαστικής ισχύος.
- Οι τυχόν διαγραφές στοιχείων από τα αποτελέσματα, καθώς με την πάροδο του χρόνου και την εξέλιξη των ρευμάτων, καθίστανται παρωχημένα.
- Την απαιτούμενη υποστήριξη διοχέτευσης των αποτελεσμάτων τους.

Τελεστές όπως η προβολή και η επιλογή 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).

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 η ομαδοποίηση, η ταξινόμηση, η διαφορά, κτλ.

Οι τελεστές διατήρησης κατάστασης όπως είναι η σύνδεση, απαλοιφή διπλοτύπων κ.α. 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).

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).

2.4 Τρόποι αντιμετώπισης των προβλημάτων

Στην υπάρχουσα βιβλιογραφία έχουν προταθεί διάφορες μέθοδοι για την αντιμετώπιση όλων αυτών των προβλημάτων. Κοινά χαρακτηριστικά είναι το κέρδος σε χώρο μνήμης και σε χρόνο επεξεργασίας και το επακόλουθο κόστος που αφορά την ακρίβεια των αποτελεσμάτων. Οι βασικότερες από αυτές τις μεθόδους παρουσιάζονται συνοπτικά σε αυτήν την ενότητα.

24.1 Περιλήψεις

Οι *Περιλήψεις ή συνόψεις* δεδομένων (summaries or data synopses) επιχειρούν να προσφέρουν μια συνοπτική αναπαράσταση της εισερχόμενης πληροφορίας, με κόστος την μείωση της ακρίβειας του αποτελέσματος. Μια σύνοψη θα πρέπει να έχει σημαντικά μικρότερο μέγεθος από αυτό των δεδομένων που αναπαριστά, θα πρέπει να μπορεί να υπολογιστεί με μία μόνο ανάγνωση των δεδομένων και να ακολουθεί το ρυθμό άφιξής τους.

Ο επεξεργαστής ερωτημάτων θεωρητικά μπορεί να συνδυάζει στοιχεία από διάφορες περιλήψεις κατά τον υπολογισμό της απάντησης.

Οι περιλήψεις αποτελούν και ένα σημαντικό εργαλείο σε περιπτώσεις όπου δεν υπάρχουν κατάλληλες δομές για τη διαχείριση των δεδομένων που επεξεργάζεται ένα ερώτημα. Τέτοια ερωτήματα είναι αυτά που περιέχουν συνδέσεις και συναθροιστικούς τελεστές, τα οποία θα απαιτούσαν τη συντήρηση όλου του ρεύματος δεδομένων για την παραγωγή της απάντησής τους, ενώ τώρα χρησιμοποιούν την πληροφορία που συγκρατείται στις συνόψεις.

Όσο αναφορά την χρήση συνόψεων, τα κύρια ζητήματα που τίθενται αφορούν το κατά πόσο μπορεί να υπολογιστεί η ακρίβεια των απαντήσεων που χρησιμοποιούν αυτές, η επιλογή των καταλληλότερων τεχνικών περίληψης σε διάφορες συνθήκες άφιξης ρευμάτων αλλά και ο συνολικός καταμερισμός μνήμης ανάμεσα σε διαφορετικές συνόψεις που ανταγωνίζονται γι' αυτή και ο οποίος θα πρέπει να είναι δυναμικός και προσαρμοζόμενος στις αλλαγές που αφορούν είτε τα ερωτήματα είτε τα ίδια τα δεδομένα.

Οι σημαντικότερες τεχνικές περίληψης είναι τα σκίτσα (sketching), η τυχαία δειγματοληψία (random sampling), τα wavelets και τα ιστογράμματα.

2.4.2 Στίξεις

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.

2.4.3 Παράθυρα

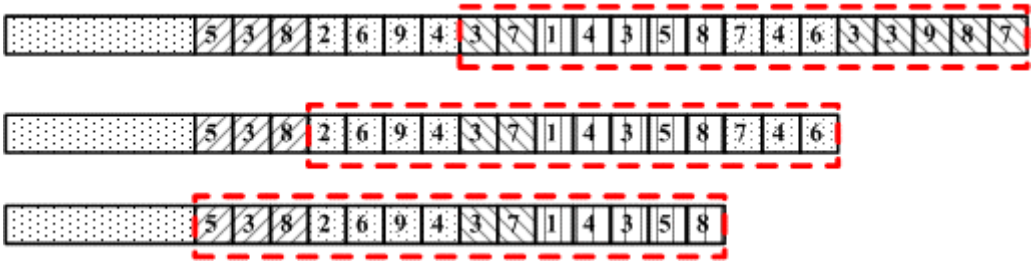
Η συνηθέστερη λύση που προτείνεται για την απεμπλοκή των προβληματικών τελεστών είναι η επιβολή *παραθύρων* στα ρεύματα δεδομένων που καλούνται να επεξεργαστούν. Οι τελεστές πλέον εφαρμόζονται πάνω στα περιεχόμενα των παραθύρων επιτυγχάνοντας τον περιορισμό του πλήθους των στοιχείων που επεξεργάζονται 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). Έτσι, τα παράθυρα απομπλέκουν τους προβληματικούς τελεστές, και καθιστούν δυνατή την επεξεργασία στοιχείων τα οποία ανήκουν σε ρευμάτων δεδομένων από αυτούς.

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.

Στο *σχήμα 2.2* απεικονίζεται ο τρόπος λειτουργίας ενός παραθύρου, το οποίο μετακινείται με την πάροδο του χρόνου και απομονώνει κάθε στιγμή ένα πεπερασμένο υποσύνολο των στοιχείων ενός -πιθανώς άπειρου- ρεύματος δεδομένων.

Εκτενέστερη ανάλυση σχετικά με τον ορισμό των παραθύρων, τους τύπους και τα χαρακτηριστικά των παραθυρικών δομών ακολουθεί στο επόμενο κεφάλαιο.



Σχήμα 2.2: *Παράδειγμα Παραθύρου που εφαρμόζεται πάνω σε ρεύμα δεδομένων.*

Κεφάλαιο 3

Παράθυρα σε ρεύματα δεδομένων

3.1 Εισαγωγή

Όπως αναφέρθηκε σε προηγούμενα, κάποιο σχεσιακοί τελεστές μπορούν να εφαρμόζονται απευθείας πάνω σε ρεύματα δεδομένων, όπως είναι η επιλογή και η προβολή, διότι έχουν την ικανότητα να επεξεργάζονται την κάθε πλειάδα του ρεύματος αυτοτελώς. Άλλοι τελεστές όμως, όπως είναι η σύνδεση και η συνάθροιση, πρέπει να έχουν διαρκώς πρόσβαση σε όλα τα δεδομένα για να εξάγουν σωστά αποτελέσματα. Λόγω της φύσης των ρευμάτων δεδομένων κάτι τέτοιο δεν είναι εφικτό, έτσι οι προβληματικοί τελεστές τροφοδοτούνται μέσα από ειδικές παραθυρικές δομές.

Τα παράθυρα μπορούν να θεωρηθούν ειδικοί 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, ανάλογα με κάποιες ειδικές παραμέτρους οι οποίες καθορίζουν κάθε παράθυρο. Έτσι, ένα παράθυρο περιλαμβάνει ένα διαρκώς μεταβαλλόμενο υποσύνολο των στοιχείων του ρεύματος με το οποίο συσχετίζεται. Το στιγμιότυπο εξόδου του παραθύρου μπορεί να θεωρηθεί σαν μια προσωρινή σχέση, όπως θα οριζόταν σε συμβατικές βάσεις δεδομένων.

Η χρήση των παραθύρων 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:

- *κάτω άκρο* (*lower bound*), η οποία αντιπροσωπεύει το παλαιότερο στοιχείο του ρεύματος που θα συμπεριληφθεί στο παράθυρο.

- *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 από το *χρονικό διάστημα* (*range*) που αυτές καλύπτουν.
- ειδικό *τρόπο μεταβολής* των περιεχομένων του μέσω κατάλληλων μετατοπίσεων στα άκρα του.

Ανάλογα με τον συνδυασμό των τιμών των ιδιοτήτων αυτών, τα παράθυρα χωρίζονται σε διάφορους τύπους. Μια κατηγοριοποίηση των συνηθέστερων παραθύρων που αναφέρονται στην τρέχουσα βιβλιογραφία **[PS06]** παρουσιάζεται αναλυτικά στις επόμενες παραγράφους.

3.2 Ιδιότητες παραθύρων

Κάθε παράθυρο αποσπά πεπερασμένο πλήθος πλειάδων από το ρεύμα δεδομένων στο οποίο εφαρμόζεται, ακολουθώντας τους δικούς του κανόνες. Οι διάφοροι τύποι παραθύρων μπορούν να χωριστούν σε κατηγορίες 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 **[PS06]**.

3.2.1 Μονάδα μέτρησης περιεχομένων

Θεωρώντας γνωστό το ένα από τα δυο άκρα του παραθύρου, το δεύτερο προσδιορίζεται βάσει του μεγέθους του, ώστε τελικά να μπορούν έμμεσα να εντοπιστούν οι πλειάδες που θα συμπεριληφθούν σ' αυτό, μέσω της σχετικής θέσης τους σε σχέση με το γνωστό άκρο. Ως γνωστό άκρο του παραθύρου συνήθως θεωρείται το πέρας του, ερμηνεύοντάς το είτε ως τρέχον χρονόσημο είτε ως την πιο πρόσφατη πλειάδα του ρεύματος.

Η *εμβέλεια* (*scope*) του παραθύρου, δηλώνεται σε:

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

3.2.2 Μετατόπιση άκρων

Εναλλακτικά, το παράθυρο μπορεί να οριστεί ρητώς από τα δύο άκρα του. Συχνά χρησιμοποιούνται τα χρονόσημα των πλειάδων άνω και κάτω άκρου 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, ενώ το άλλο επιτρέπεται να μετακινείται ελεύθερα. Τα *παράθυρα ορόσημου* (*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*). Στα οποία και τα δύο άκρα του παραθύρου μεταβάλλονται ελεύθερα με την πάροδο του χρόνου. Στην συνηθέστερη περίπτωση, η αφετηρία και το πέρας του παραθύρου εξελίσσονται συντονισμένα με τον ίδιο ρυθμό, όπως συμβαίνει λ.χ. στα *κυλιόμενα παράθυρα* (*sliding windows*).

3.2.3 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, με έναν από τους δυο τρόπους:

- Με *μοναδιαίο βήμα (unit step)*, οπότε τα άκρα του παραθύρου προχωρούν 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), του οποίου τα άκρα μετακινούνται παράλληλα, διατηρώντας σταθερό το μέγεθος του παραθύρου.
- Με *άλματα (hops)*, συνήθως μη επικαλυπτόμενα. Χαρακτηριστικό παράδειγμα αποτελούν τα επάλληλα παράθυρα (tumbling windows), τα οποία εξυπηρετούν όταν επιδιώκεται τα περιεχόμενα διαδοχικών στιγμιοτύπων τους να είναι ξένα μεταξύ τους, έτσι ώστε κάθε φορά να εξετάζονται εντελώς διαφορετικά στοιχεία του ρεύματος χωρίς απώλεια πληροφορίας.

Στην περίπτωση των χρονικών κυλιόμενων παραθύρων είναι αναμενόμενο να υπάρχουν επικαλύψεις μεταξύ διαδοχικών στιγμιοτύπων, έτσι όλες οι πλειάδες με χρονόσημο μικρότερο από τη νέα αρχή του παραθύρου διαγράφονται, ώστε να ενταχθούν όσα στοιχεία αναφέρονται στην τρέχουσα χρονική στιγμή. Ο αριθμός των στοιχείων που περιλαμβάνονται σε διαφορετικά στιγμιότυπα του παραθύρου μπορεί να διαφέρει.

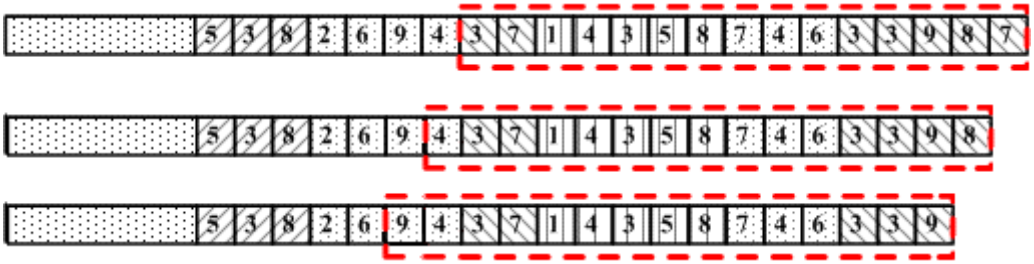
Αντίθετα, αν το παράθυρο εκφράζεται ως αριθμός πλειάδων, τότε κάθε νεοεισερχόμενη πλειάδα συνεπάγεται την απόρριψη της παλαιότερης που υπάρχει στο παράθυρο. Άρα, καθώς το παράθυρο μετατοπίζεται, μεταβολές σημειώνονται μόνο στα άκρα του (διαγραφές στο κατώτερο, εισαγωγές στο ανώτερο), αφήνοντας άθικτα τα ενδιάμεσα στοιχεία.

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.

Ακολουθεί μια κατηγοριοποίηση των παραθύρων και παρουσίαση των βασικότερων τύπων παραθυρικών δομών που απαντώνται στη σύγχρονη βιβλιογραφία [PS05].

3.3 Τύποι φυσικών παράθυρων

Λόγω του ότι τέτοια παράθυρα ορίζονται από έναν προκαθορισμένο αριθμό πλειάδων, συνήθως είναι κυλιόμενα. 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 πιθανώς άπειρου 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. Στις δυο αυτές κατηγορίες που περιγράφονται, χρησιμοποιήθηκε μοναδιαίο βήμα, εφόσον δεν θεωρείται πιθανό να οριστεί μια παράμετρος βήματος πολλαπλών πλειάδων.



Σχήμα 3.1: *Παράδειγμα εφαρμογής παραθύρου πλειάδων πάνω σε ρεύμα δεδομένων.*

3.3.1 Παράθυρα πλειάδων

Τα παράθυρα πλειάδων (Tuple-based windows) σε κάθε στιγμιότυπο, παρέχουν πρόσβαση στις N τελευταίες πλειάδες του ρεύματος S.

Τα παράθυρα πλειάδων έχουν 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.

Ο καθορισμός των περιεχομένων τους γίνεται με βάση δύο συνθήκες. Η μία προσδιορίζει το μέγιστο αριθμητικό πλήθος N των πλειάδων που θα περιέχονται στο παράθυρο W. Η δεύτερη συνθήκη διασφαλίζει ότι το παράθυρο θα περιλαμβάνει N από τις πιο πρόσφατες πλειάδες του ρεύματος. Αντίστοιχες συνθήκες απαντώνται και στις στους υπόλοιπους τύπους παραθύρων, με τις απαραίτητες πάντα τροποποιήσεις.

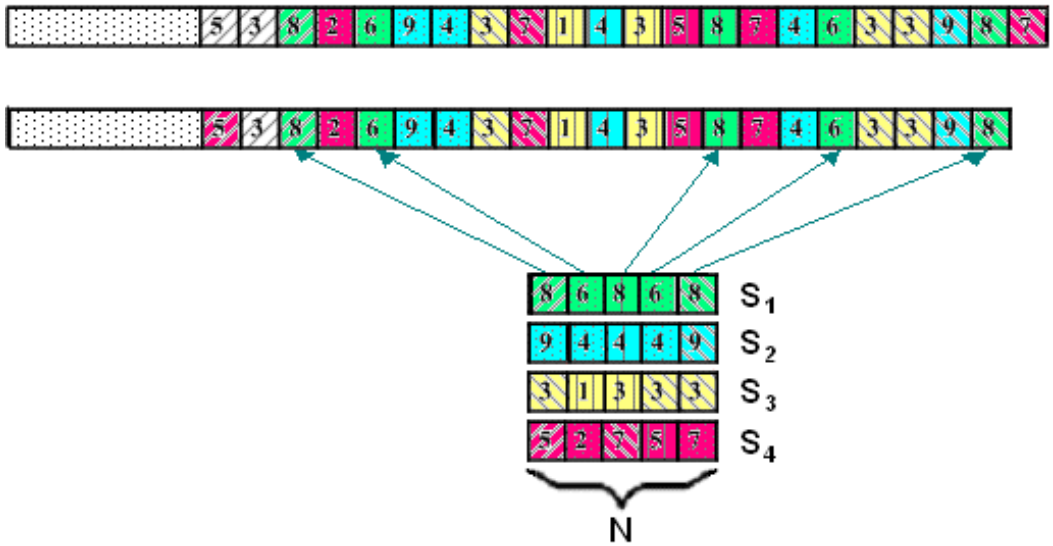
Τα στοιχεία εντός του παραθύρου υπολογίζονται ξεκινώντας από την παρούσα χρονική στιγμή και προχωρώντας προς το παρελθόν, επιλέγοντας συνεχώς τις πιο πρόσφατες N πλειάδες.

Στο *σχήμα 3.1* βλέπουμε την προοδευτική πορεία ενός παραθύρου πλειάδων, διατηρώντας πάντα στην έξοδό του τις N πιο πρόσφατες πλειάδες του στιγμιοτύπου του ρεύματος εισόδου του.

3.3.2 Μεριστικά παράθυρα

Ένα μεριστικό παράθυρο (Partitioning window) 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. Είναι σημαντικό να επισημανθεί ότι το πεδίο χρονοσήμου δεν μπορεί να συμμετάσχει στη λίστα γνωρισμάτων ομαδοποίησης. Μετά την ομαδοποίηση δεν εφαρμόζεται κάποια συναθροιστική συνάρτηση (MAX, AVG, κτλ.) 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.



Σχήμα 3.2 *Παράδειγμα εφαρμογής μεριστικού παραθύρου πάνω σε ρεύμα δεδομένων.*

Τα στοιχεία εντός του παραθύρου υπολογίζονται 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.

Στο *σχήμα 3.2* βλέπουμε την προοδευτική πορεία ενός μεριστικού παραθύρου, διατηρώντας πάντα στην έξοδό του τις N πιο πρόσφατες πλειάδες του στιγμιοτύπου του ρεύματος εισόδου του, ανάλογα με τις τιμές των γνωρισμάτων της λίστας ομαδοποίησης.

3.4 Τύποι λογικών παραθύρων

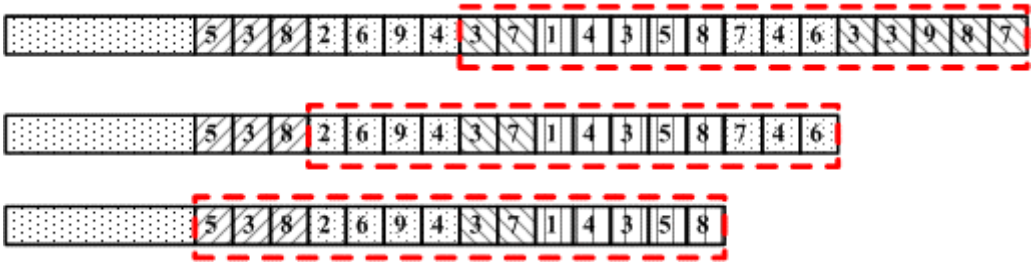
Τα λογικά παράθυρα προϋποθέτουν ρητό προσδιορισμό του χρονικού διαστήματος μεταξύ των άκρων τους. Ο προσδιορισμός των πλειάδων του ρεύματος που εμπίπτουν στο διάστημα αυτό γίνεται ελέγχοντας αν η τιμή του χρονοσήμου τους βρίσκεται εντός του χρονικού αυτού διαστήματος. Αυτή η απαίτηση απλουστεύεται αρκετά αν οριστεί η έννοια της εμβέλειας (scope) κάθε παραθύρου. Για κάθε χρονική στιγμή η συνάρτηση εμβέλειας επιστρέφει απαραιτήτως τα χρονικά όρια (άκρα) του παραθύρου, λαμβάνοντας ως παραμέτρους τις ιδιότητες που ορίζουν τον τύπο του παραθύρου.

Στην κατηγορία των λογικών παραθύρων εντάσσονται τα κυλιόμενα παράθυρα, τα επάλληλα παράθυρα και τα παράθυρα οροσήμου.

3.4.1 Κυλιόμενα παράθυρα

Ο τύπος αυτών των παραθύρων (Sliding windows) 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 με βάση το χρονόσημο των πλειάδων
- το βήμα μπορεί να είναι είτε μοναδιαίο είτε κατ' άλματα, πάντοτε όμως σε χρονικές μονάδες.



Σχήμα 3.3: *Παράδειγμα εφαρμογής κυλιόμενου χρονικού παραθύρου πάνω σε ρεύμα δεδομένων.*

Για τον ορισμό τους χρησιμοποιούνται δυο βασικές 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.

Έτσι τα περιεχόμενα ενός κυλιόμενου παραθύρου είναι οι πλειάδες εκείνες που το χρονόσημό τους είναι εντός των ορίων της έκτασης του παραθύρου, ορίζοντας ως κάτω άκρο του παραθύρου τη χρονική στιγμή της τελευταίας ανανέωσης των περιεχομένων του. Η τελευταία γίνεται κάθε δ χρονικές μονάδες, όπου δ η παράμετρος βήματος του παραθύρου.

Επειδή γενικά ισχύει ότι η παράμετρος βήματος δ είναι μικρότερη από την παράμετρο εύρους ω, 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.

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. Έτσι με την εφαρμογή ενός τέτοιου κυλιόμενου παραθύρου αποσπώνται διαρκώς από το ρεύμα οι πλειάδες των ω τελευταίων χρονικών στιγμών.

Στο *σχήμα 3.3* βλέπουμε την προοδευτική πορεία ενός κυλιόμενου παραθύρου, μετακινώντας παράλληλα τα δυο χρονικά άκρα του, με βήμα 6, διατηρώντας στο στιγμιότυπο της εξόδου του σταθερό χρονικό εύρος ω.

3.4.2 Επάλληλα παράθυρα

Τα επάλληλα παράθυρα (Tumbling windows) μπορούν να θεωρηθούν ως υποπερίπτωση των κυλιόμενων. Στα κυλιόμενα παράθυρα η συνάρτηση εμβέλειας, εφαρμόζεται για τιμές βήματος δ εν γένει μικρότερες του εύρους ω. Στα επάλληλα παράθυρα αντίθετα, οι παράμετροι αυτοί συνήθως είναι ίσες. Η συνάρτηση εμβέλειας των κυλιόμενων χρονικών παραθύρων ισχύει αυτούσια ακόμα και στην περίπτωση όπου το χρονικό βήμα θεωρείται ίσο ή μεγαλύτερο του εύρους. Σε αυτή την περίπτωση, τα περιεχόμενα του ρεύματος θα επιστρέφονται κατά κύματα, χωρίς επαναλαμβανόμενες πλειάδες. Βασικός στόχος των επάλληλων 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 ω του παραθύρου. Με την συνθήκη αυτή διασφαλίζεται ότι όλες οι πλειάδες του ρεύματος θα περιληφθούν κάποια στιγμή στα περιεχόμενα του παραθύρου. Τέτοια επάλληλα παράθυρα αποδεικνύονται ιδιαιτέρως χρήσιμα όταν ζητούνται συναθροιστικά αποτελέσματα σε διαδοχικά, μη επικαλυπτόμενα τμήματα του ρεύματος [ACC+03b].

Σχήμα 3.4: *Παράδειγμα εφαρμογής επάλληλου παραθύρου πάνω σε ρεύμα δεδομένων.*

Επίσης, υπάρχει 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_ω 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 $[\tau_\omega + 1,\ \tau_\omega + \delta - \omega\ ]$. Ένα τέτοιο παράθυρο με παραμέτρους $\delta = 10$ και $\omega = 4$ δευτερόλεπτα, χρησιμοποιείται για να παρέχει κάθε 10 δευτερόλεπτα πρόσβαση στα περιεχόμενα των 4 τελευταίων δευτερολέπτων. Έτσι σε κάθε ανανέωση θα υπάρχει απώλεια των δεδομένων που κατέφθασαν στα 6 δευτερόλεπτα εκείνα που μεσολαβούν και που δεν θα συμπεριληφθούν στο επόμενο στιγμιότυπο.

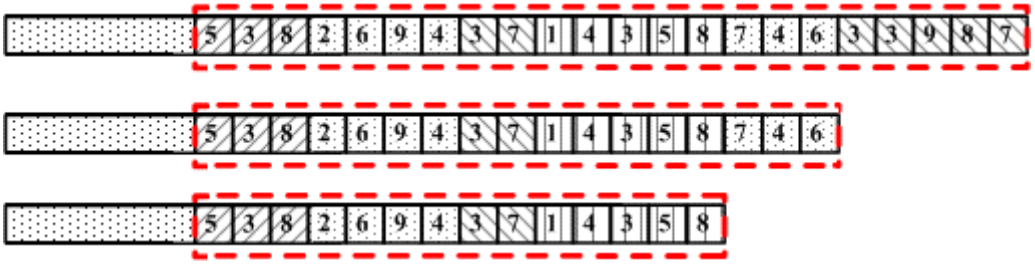
Στο *σχήμα 3.4* βλέπουμε την προοδευτική πορεία ενός επάλληλου παραθύρου, με ίσες παραμέτρους εύρους και βήματος, παρέχοντας στην έξοδό του μη επικαλυπτόμενα διαδοχικά στιγμιότυπα του ρεύματος εισόδου του, χωρίς απώλεια πληροφορίας.

3.4.3 Παράθυρα οροσήμου

Τα παράθυρα οροσήμου (Landmark windows) διατηρούν πάντοτε σταθερό το ένα από τα δύο χρονικά άκρα τους, αφήνοντας το άλλο να ακολουθεί την εξέλιξη του χρόνου. 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:

♦ Η αφετηρία τ_L του παραθύρου θεωρείται γνωστή (*Lower-bounded*), οπότε για κάθε χρονική στιγμή τ που αποτελεί το πέρας του («ανοιχτό άκρο») θα ισχύει $\tau \geq \tau_0$.

Τα περιεχόμενα του παραθύρου προκύπτουν καθώς τα χρονόσημα των πλειάδων του ρεύματος ελέγχονται ως προς την εμβέλεια, θέτοντας ως αφετηρία τη χρονική στιγμή τ_L. Έτσι, στην περίπτωση αυτή το παράθυρο θα συνεχίσει να παρέχει πλειάδες επ' αόριστον, μέχρις ότου το σχετικό ερώτημα διαρκείας καταργηθεί (ή το ρεύμα εξαντληθεί).



Σχήμα 3.5: *Παράδειγμα εφαρμογής παραθύρου οροσήμου πάνω σε ρεύμα δεδομένων.*

♦ Το πέρας τ_u του παραθύρου θεωρείται γνωστό (*Upper-bounded*). Η περίπτωση αυτή είναι λιγότερο συχνή. Καθώς η αφετηρία διατρέχει το χρόνο, αναπόφευκτα το άνω όριο κάποτε θα υπερκαλυφθεί και το παράθυρο θα παγιωθεί.

Προφανώς, δεν έχει έννοια ο προσδιορισμός παραθύρων με πέρατα χρονικές στιγμές στο παρελθόν.

Στην περίπτωση του Upper-bounded παραθύρου, τα άκρα του θα παγιωθούν όταν ο θα λαμβάνονται πλέον πλειάδες του ρεύματος με χρονόσημα μεγαλύτερα από την χρονική στιγμή τ_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 με εκείνη των συμβατικών σχεσιακών πινάκων (για όσο χρόνο τα στοιχεία κρατηθούν στη μνήμη του συστήματος).

Στο *σχήμα 3.5* βλέπουμε την προοδευτική πορεία ενός παραθύρου ορισμού, διατηρώντας πάντα σταθερό το ένα χρονικό άκρο του, παρέχοντας στην έξοδό του ένα διαρκώς αυξανόμενο υποσύνολο πλειάδων του στιγμιοτύπου του ρεύματος εισόδου του.

3.5 Τροφοδότηση τελεστών μέσα από παράθυρα

Με την τροφοδότηση μέσα από παράθυρα των τελεστών ερωτημάτων που διατυπώνονται σε ρεύματα δεδομένων επιτυγχάνεται απεμπλοκή των προβληματικών τελεστών (ανασχετικών, διατήρησης κατάστασης) ώστε να γίνεται εφικτή η χρήση τελεστών της σχεσιακής άλγεβρας σε ρεύματα με το επακόλουθο κόστος στην ακρίβεια των αποτελεσμάτων. Κάθε τελεστής εφαρμόζεται μόνο στα περιεχόμενα του στιγμιοτύπου του παραθύρου που τον τροφοδοτεί με δεδομένα και όχι στο σύνολο των στοιχείων του ρεύματος, έτσι προκύπτουν οι αντίστοιχες παραθυρικές παραλλαγές των τελεστών αυτών, οι οποίες παρουσιάζονται αναλυτικά σε αυτήν την ενότητα **[PS06]**.

3.5.1 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 γνωρισμάτων της προβολής.

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.

3.5.2 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. Ο τελεστής αυτός διατηρεί στην έξοδό του κάθε πλειάδα του παραθύρου η οποία ικανοποιεί την συνθήκη.

Η συνθήκη επιλογής F μπορεί να έχει τις εξής μορφές:

- Attribute = value, οπότε εφαρμόζεται κάποιο κατηγόρημα επιλογής ανάμεσα σε ένα γνώρισμα των πλειάδων και μία τιμή από το πεδίο ορισμού αυτού του γνωρίσματος, ή
- Attribute1 = Attribute2, 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.

3.5.3 Παραθυρική Απαλοιφή διπλοτύπων

Ο τελεστής αυτός εφαρμόζεται υποχρεωτικά πάνω σε ένα παράθυρο W ενός ρεύματος S και επιστρέφει την πιο πρόσφατη εμφάνιση κάθε πλειάδας, απαλείφοντας κάθε άλλο ακριβές αντίγραφο της πλειάδας αυτής που βρίσκεται στην εμβέλεια του παραθύρου. Και αυτός ο τελεστής μπορεί να οδηγήσει σε σημαντική μείωση της πληροφορίας που εξάγεται απ' αυτόν, ανάλογα με το πλήθος των διπλοτύπων κάθε πλειάδας που εμπίπτουν στο εύρος του παραθύρου.

3.5.4 Παραθυρική Σύνδεση

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, ο τελεστής σύνδεσης παραθύρων επιστρέφει τη συνένωση (concatenation) ζευγών πλειάδων που εμφανίζονται στα αντίστοιχα στιγμιότυπα των παραθύρων.

Κάθε νεοεισερχόμενη πλειάδα του ρεύματος S_1 ελέγχεται ως προς τη συνθήκη σύνδεσης 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. Εναλλακτικά, μια δεύτερη πρόταση για την σήμανση των αποτελεσμάτων θα ήταν η απόδοση της χρονικής στιγμής παραγωγής κάθε πλειάδας από τον τελεστή σύνδεσης, 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 E 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.

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.

Η πιο αξιοσημείωτη υλοποίηση παραθυρικής σύνδεσης διοχέτευσης είναι αυτή που προκύπτει από την εφαρμογή κυλιόμενων παραθύρων (sliding-window join), χωριστά για κάθε ρεύμα. 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.

3.5.5 Παραθυρική Συνάθροιση

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.

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.

3.5.6 Παραθυρικοί Συνολοθεωρητικοί τελεστές

Οι γνωστές συνολοθεωρητικές πράξεις όταν εφαρμοστούν σε στοιχεία παραθύρων, υπακούν 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 παράθυροW_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.

3.6 Συντακτικές δομές προσδιορισμού παραθύρων

Στα διάφορα πρωτότυπα συστήματα διαχείρισης ρευμάτων δεδομένων (ΣΔΡΔ) έχουν προταθεί τρόποι σύνταξης των εκφράσεων που επιτρέπουν τον προσδιορισμό των παραπάνω τύπων παραθύρων σε γλώσσες μορφής παραπλήσιας της SQL **[Τζω05], [siteTCQ]**. Κάθε παράθυρο αναφέρεται σε ένα μόνο ρεύμα και δηλώνεται στην πρόταση FROM του ερωτήματος διαρκείας. Έτσι ορίζονται:

3.6.1 Παράθυρα πλειάδων

Στα παράθυρα αυτά εξετάζονται οι πιο 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 αυθαίρετα, κατά μη ντετερμινιστικό τρόπο.

Οι N πιο πρόσφατες πλειάδες ενός ρεύματος δεδομένων S επιλέγονται με την έκφραση: S [ROWS N] στη γλώσσα CQL του συστήματος STREAM.

Στην παρούσα εργασία η έκφραση που χρησιμοποιείται για τον ορισμό αυτών των παραθύρων ακολουθεί την παρόμοια σύνταξη: S [ROWS 'N']

3.6.2 Μεριστικά παράθυρα

Αυτά διαμελίζουν το ρεύμα S σε υπορρεύματα βάσει της λίστας γνωρισμάτων ομαδοποίησης. Για κάθε υπόρρευμα, 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.

Στη γλώσσα CQL η έκφραση των μεριστικών παραθύρων που χρησιμοποιείται είναι: S [PARTITION BY $A_1, A_2, ..., A_k$ ROWS N]

Η σύνταξη των εκφράσεων μεριστικών παραθύρων στην παρούσα εργασία έχει τη μορφή : S [ROWS 'N' PARTITIONED BY 'A'], όπου για λόγους απλότητας της εφαρμογής έγινε χρήση ενός μόνο γνωρίσματος ομαδοποίησης, του A.

3.6.3 Χρονικά παράθυρα

Η CQL με την έκφραση S [RANGE T SLIDE D], ορίζει 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 = ∞, 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.

Στο TelegraphCQ η γλώσσα σύνταξης ερωτημάτων διαρκείας, επίσης παραπλήσια της SQL, ορίζει τα χρονικά παράθυρα κυλιόμενα ή επάλληλα με την έκφραση: S [RANGE BY 'T' SLIDE BY 'D' START AT 'T_O'], όπου τα T και D έχουν την ίδια σημασία όπως στα προηγούμενα, ενώ το T_O παίρνει τιμές *χρονοσήμου* και ορίζει την χρονική στιγμή από την οποία αρχίζει να εφαρμόζεται το παράθυρο πάνω στο ρεύμα. Στην ειδική περίπτωση των επάλληλων παραθύρων η έκφραση αυτή διαμορφώνεται ανάλογα ως εξής: S [RANGE BY 'T' SLIDE BY 'T' START AT 'T_O']. Την σύνταξη αυτή υιοθετεί και η εφαρμογή της παρούσας εργασίας.

3.6.4 Παράθυρα οροσήμου

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 στην γλώσσα CQL είναι 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].

Τα παράθυρα οροσήμου στο συντακτικό των ερωτημάτων της παρούσας εργασίας εκφράζονται με την εξής μορφή: S [START AT 'T_O'] , όπου το T_O παίρνει τιμές *χρονοσήμου* και δηλώνει το σταθερό κάτω άκρο (αφετηρία) του παραθύρου, ενώ οι υπόλοιπες λιγότερο συχνές περιπτώσεις δεν προβλέπονται.

Κεφάλαιο 4

Διαχείριση ρευμάτων δεδομένων με το TelegraphCQ

4.1 Εισαγωγή

Το TelegraphCQ είναι ένα πρωτότυπο Σύστημα Διαχείρισης Ρευμάτων Δεδομένων που αναπτύσσεται στο Πανεπιστήμιο Berkeley **[siteTCQ]**. Επιχειρεί να ανταποκριθεί στις ανάγκες εφαρμογών μεγάλης κλίμακας δικτυακής φύσης που λειτουργούν σε ασταθείς συνθήκες και απαιτούν στενή αλληλεπίδραση με τους χρήστες. Αντιμετωπίζει τα δεδομένα ως ρεύματα των οποίων η επεξεργασία πραγματοποιείται με την εκτέλεση ερωτημάτων διαρκείας.

Το σύστημα αυτό επιλέχθηκε για την διαχείριση των ρευμάτων δεδομένων και την εκτέλεση των ερωτημάτων διαρκείας στην εφαρμογή της παρούσας εργασίας. Οι κύριοι λόγοι που προτιμήθηκε έναντι των άλλων προτεινόμενων Συστημάτων Διαχείρισης Ρευμάτων Δεδομένων είναι:

- Αξιοποιεί τον μηχανισμό της PostgreSQL εμπλουτισμένο με παραθυρικές δομές που καθιστούν ικανή την διαχείριση ρευμάτων δεδομένων.
- Η γλώσσα ερωταποκρίσεων αποτελεί μια τροποποιημένη εκδοχή της SQL, με την οποία υπάρχει εκ των προτέρων εξοικείωση, γεγονός που ευνοεί σημαντικά την διατύπωση των ερωτημάτων με δηλωτικό τρόπο.
- Έχει σχεδιαστεί με έμφαση στην ευελιξία και στην προσαρμοστικότητα σε κυμαινόμενους ρυθμούς δεδομένων.
-

Το TelegraphCQ αποτελεί μέρος ενός ευρύτερου έργου που ξεκίνησε στο Πανεπιστήμιο Berkeley το 2000 **[CCC+02]**. Αρχικά, αναπτύχθηκε το Σύστημα **Telegraph,** στη σχεδίαση του οποίου δόθηκε ιδιαίτερη βαρύτητα στην προσαρμοστικότητα της

επεξεργασίας. Επιθυμείται η δυνατότητα αντιμετώπισης μεταβολών προερχόμενων από απαιτήσεις των χρηστών ή αστάθειες των δικτύων μετάδοσης δεδομένων.

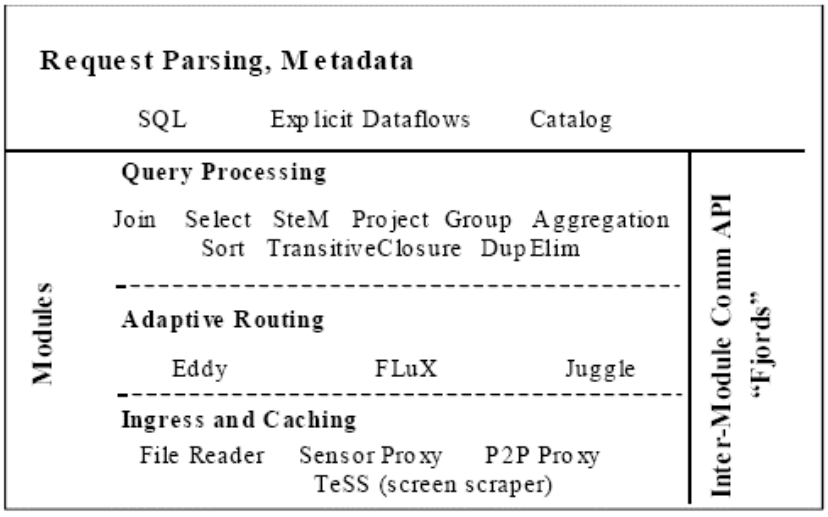
Στη συνέχεια, υλοποιήθηκαν τα Συστήματα **CACQ** και **Psoup** που επεκτείνουν το Telegraph ώστε να υποστηρίζεται η κοινή επεξεργασία μεταξύ ρευμάτων δεδομένων, χωρίς να εγκαταλείπεται ο αρχικός στόχος για ευελιξία και προσαρμοστικότητα. Οι βασικές αδυναμίες τους είναι:

- Περιορισμός στον όγκο των δεδομένων προς επεξεργασία ώστε να χωρούν στην κυρίως μνήμη.
- Δεν προβλέπεται η χρονοδρομολόγηση ή μοίρασμα της επεξεργασίας για ερωτήματα με μικρή επικάλυψη.
- Δεν υποστηρίζουν την έννοια της ποιότητας υπηρεσιών (Quality of Service) ώστε να ανταποκρίνονται ανάλογα στους περιορισμούς των πόρων.
- Δεν προσφέρουν τη δυνατότητα επιλογής λιγότερης ή περισσότερης προσαρμοστικότητας, ώστε να ρυθμίζεται το κόστος στην επεξεργασία που αυτή συνεπάγεται.

Το **TelegraphCQ** είναι μεταγενέστερο και καλείται να ξεπεράσει αυτές τις αδυναμίες.

4.2 Βασικά δομικά στοιχεία του Telegraph

Η επεξεργασία δεδομένων στο Telegraph πραγματοποιείται από ένα σύνολο τελεστών ή οντοτήτων οι οποίες παράγουν και καταναλώνουν πλειάδες. Συνεπώς η επεξεργασία παίρνει την μορφή *ροής δεδομένων* καθώς αυτά ανταλλάσσονται μεταξύ των διαφόρων οντοτήτων σύγχρονα (*push*) ή ασύγχρονα (*pull*). Η διαδικασία αυτή πυροδοτείται είτε μέσω περιστασιακών ερωτημάτων (*ad-hoc queries*) των χρηστών είτε από προκαθορισμένα προσχέδια επεξεργασίας.



Σχήμα 4.1: *Αρχιτεκτονική του Telegraph. (Πηγή: [CCC+02])*

4.2.1 Τύποι οντοτήτων του Telegraph

Το Telegraph αποτελείται από τρεις βασικούς τύπους οντοτήτων (Σχήμα 4.1):

- **Οντότητες εκτέλεσης ερωτημάτων.** Η επεξεργασία ερωτημάτων στο Telegraph επιτυγχάνεται με τη δρομολόγηση πλειάδων μεταξύ των διαφόρων οντοτήτων επεξεργασίας. Οι οντότητες αυτές είναι επέκταση των κλασικών σχεσιακών τελεστών όπως επιλογή (SELECTION), προβολή (PROJECTION), σύνδεση (JOIN) με τη διαφορά ότι είναι διασωληνωμένες (pipelined) και μη ανασχετικές (non-blocking), ενώ σε αυτές περιλαμβάνεται και μία νέα, η SteM (State Module).

- **Οντότητες δυναμικής δρομολόγησης πλειάδων.** Η εκτέλεση ερωτημάτων στο Telegraph δεν βασίζεται σε κάποιο προκαθορισμένο προσχέδιο εκτέλεσης, αλλά πραγματοποιείται από οντότητες που δρομολογούν πλειάδες βάσει των εκάστοτε συνθηκών, με δυνατότητες δυναμικής αναπροσαρμογής ενώ τα ερωτήματα εξακολουθούν να είναι ενεργά. Οι οντότητες αυτές απλώς παράγουν και καταναλώνουν πλειάδες όπως και οι υπόλοιπες, αλλά η λειτουργία τους είναι η δρομολόγηση των δεδομένων. Οι βασικότερες είναι οι *Eddies*, οι οποίοι κατευθύνουν τις πλειάδες στις κατάλληλες οντότητες εκτέλεσης ερωτημάτων, ο *Juggle* που πραγματοποιεί δυναμική αναδιάταξη των δεδομένων, και ο *Flux* που φροντίζει για τη δρομολόγηση πλειάδων μεταξύ μηχανημάτων μιας συστάδας (cluster) υπολογιστών όταν η επεξεργασία γίνεται παράλληλα. Η διασύνδεση των παραπάνω οντοτήτων γίνεται μέσω του *Fjords* API.

- **Οντότητες πρόσβασης και αποθήκευσης** οι οποίες πραγματοποιούν τη διασύνδεση με τις εξωτερικές πηγές δεδομένων. Επιπλέον, μπορούν να αποθηκεύουν προσωρινά τα δεδομένα ώστε να μην επηρεάζεται το σύστημα από τα προβλήματα που παρουσιάζονται κατά την μετάδοση. Ονομάζονται *wrappers* και μπορούν να χρησιμοποιηθούν ως HTML/XML screen scrapers ("Tess"), ως αντιπρόσωποι για την μεταφορά δεδομένων μέσω δικτύων peer-to-peer ("TeleNap") αλλά και για την ανάγνωση τοπικών αρχείων.

4.2.2 Eddies

Ο ρόλος της οντότητας Eddy είναι να δρομολογεί πλειάδες σε ένα σύνολο άλλων οντοτήτων με τις οποίες είναι συνδεδεμένη, ακολουθώντας κάποια τακτική δρομολόγησης. Το σύνολο των συνδεόμενων οντοτήτων μπορεί να μεταβάλλεται, όπως επίσης και το σχέδιο δρομολόγησης, καθώς το Eddy είναι σε θέση να παρακολουθεί τη συμπεριφορά και απόδοση κάθε συνδεόμενης οντότητας. Η επεξεργασία μίας πλειάδας από μια οντότητα μπορεί να οδηγήσει στην επιστροφή της πλειάδας στο Eddy ή και στη δημιουργία νέων πλειάδων. Το Eddy αποβάλλει μια πλειάδα όταν αυτή περάσει από όλες τις συνδεόμενες οντότητες. Αυτή η διαδικασία επιβάλλει την απόδοση ενός πεδίου «κατάστασης» σε κάθε πλειάδα, η οποία υποδεικνύει το σύνολο των οντοτήτων που έχει επισκεφθεί. Το σύνολο των συνδεόμενων οντοτήτων στο Eddy μπορεί να αποτελείται από οποιοδήποτε αριθμό και είδος οντοτήτων, ακόμη και από άλλα Eddies.

4.2.3 SteMs

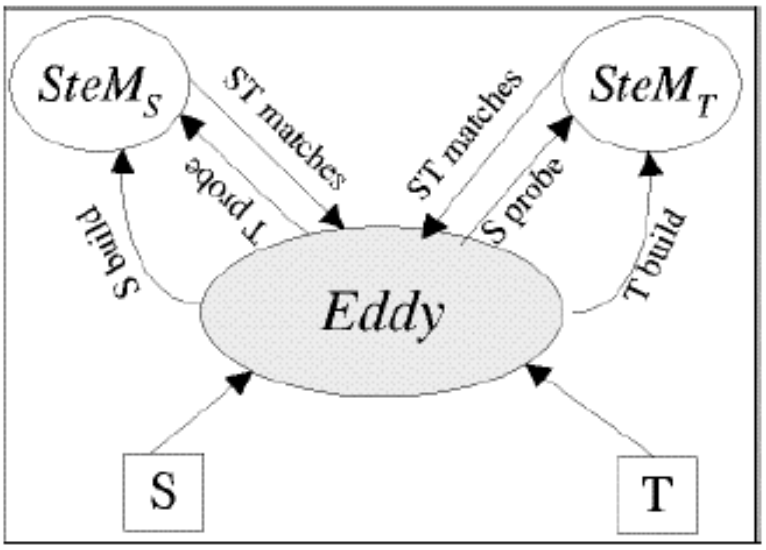
Η οντότητα SteM (State Module) λειτουργεί ως ένας προσωρινός χώρος αποθήκευσης ομογενών πλειάδων, ο οποίος πραγματοποιεί το μισό μέρος της λειτουργίας του παραδοσιακού τελεστή σύνδεσης *(join)*. Υποστηρίζει λειτουργίες εισαγωγής *(insert)*, αναζήτησης *(probe)*, και προαιρετικά διαγραφής *(delete)*. Όταν μία πλειάδα δρομολογείται σε μία οντότητα SteM, είτε εισάγεται σε αυτή αν ικανοποιεί κάποια συνθήκη κατασκευής είτε συνδυάζεται με τις ήδη αποθηκευμένες ώστε να επιστραφούν τα ζευγάρια τους που ικανοποιούν ένα δοσμένο κατηγόρημα. Η χρήση ευρετηρίων στις μονάδες SteM επιταχύνει την παραπάνω διαδικασία. Με τη συνδυασμένη χρήση οντοτήτων Eddies και SteMs μπορεί να επιτευχθεί η υλοποίηση ενός ευπροσάρμοστου τελεστή σύνδεσης μεταξύ ρευμάτων.

Η διασύνδεση οντοτήτων Eddy και SteMs απεικονίζεται στο σχήμα 4.2.

4.2.4 Διασύνδεση μεταξύ των οντοτήτων μέσω Fjords

Η μονάδα Fjords είναι ένα API το οποίο πραγματοποιεί την σύνδεση και την επικοινωνία μεταξύ των διάφορων οντοτήτων για το σχηματισμό ενός προσχεδίου εκτέλεσης.

Το βασικότερο πλεονέκτημα που παρουσιάζει είναι ότι επιτρέπει διάφορους συνδυασμούς ασύγχρονης και σύγχρονης επικοινωνίας μεταξύ των οντοτήτων του προσχεδίου, δίνοντας έτσι τη δυνατότητα εκτέλεσης ερωτημάτων που διατυπώνονται πάνω σε ένα συνδυασμό από στατικά και μεταβαλλόμενα δεδομένα. Το μεγαλύτερο πρόβλημα στη συνεργασία μονάδων επεξεργασίας δεδομένων είναι οι διαφορετικοί ρυθμοί παραγωγής και κατανάλωσης πλειάδων από την καθεμιά. Η λύση σε αυτό δίνεται με την χρησιμοποίηση διαφορετικών ειδών ουρών μεταξύ των οντοτήτων, οι οποίες ρυθμίζουν τον τρόπο διασύνδεσης.



Σχήμα 4.2: *Eddies και Stems. (Πηγή: [CCC+02])*

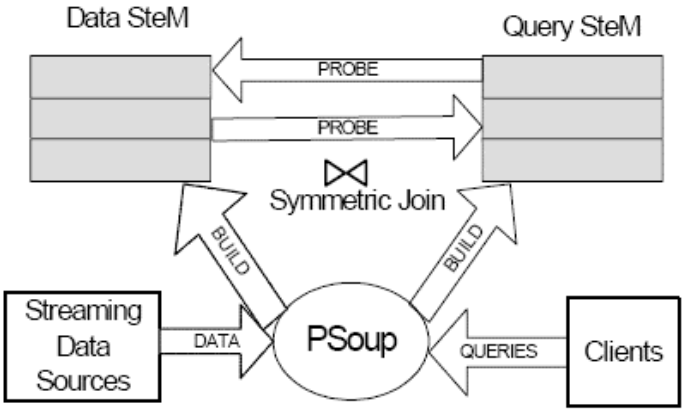
4.2.5 Κλιμάκωση ροής δεδομένων με Flux

Η κλιμάκωση (scalability) αποτελεί κύριο μέλημα των συστημάτων επεξεργασίας ρευμάτων δεδομένων. Η παραδοσιακή προσέγγιση ακολουθεί τον οριζόντιο καταμερισμό των τελεστών επεξεργασίας στα μηχανήματα μιας συστάδας, με τη ροή των δεδομένων να αποτελεί το συνδετικό τους ιστό. Παρά ταύτα, σε ένα δυναμικό περιβάλλον σαν αυτά στα οποία απευθύνεται το Telegraph, οι τελεστές πρέπει να προσαρμόζουν δυναμικά την κατανομή τόσο της εσωτερικής τους κατάστασης όσο και των ενδιάμεσων ρευμάτων δεδομένων, ανάλογα με τις μεταβολές των συνθηκών. Ο επανακαταμερισμός είναι συνήθως μια δύσκολη και απαιτητική διαδικασία, την οποία η μονάδα Flux επιδιώκει να πραγματοποιήσει. Η λειτουργία της παρεμβάλλεται μεταξύ οντοτήτων παραγωγών – καταναλωτών, ρυθμίζοντας τη ροή των δεδομένων και τη διαδικασία δρομολόγησης. Ουσιαστικά επεκτείνει την υπάρχουσα μονάδα *Exchange*, με τη διαφορά ότι επιτρέπει την εξισορρόπηση του καταμερισμού εργασίας (load-balancing) και την ανοχή σε σφάλματα (fault tolerance).

4.2.6 Αρχικά Συστήματα υποστήριξης ερωτημάτων διαρκείας

♦ **CACQ**. Υπήρξε το πρώτο σύστημα βασισμένο στο Telegraph που υποστηρίζει την εκτέλεση ερωτημάτων διαρκείας. Η κύρια καινοτομία που εισήγαγε είναι η τροποποίηση της μονάδας *Eddy* ώστε να μπορεί να εκτελεί πολλαπλά ερωτήματα ταυτόχρονα. Ένα άλλο σημαντικό χαρακτηριστικό του είναι η χρήση ειδικών ευρετηρίων, των *grouped filters* σε συνδυασμό με τις μονάδες *SteMs* για την αύξηση της αποτελεσματικότητας στην επεξεργασία: Όταν ένα νέο ερώτημα εισάγεται στο σύστημα, διασπάται σε κριτήρια που παίρνουν απαντήσεις μόνο δύο τιμών (boolean factors). Κάθε κριτήριο που αφορά μία μεταβλητή μπορεί να αντιστοιχιστεί σε ένα grouped filter ενώ κριτήρια περισσότερων μεταβλητών σε μία SteM.

♦ **PSoup**. Το PSoup επεκτείνει το CACQ προς δύο κατευθύνσεις: 1) Επιτρέπει την πρόσβαση των ερωτημάτων σε ιστορικά δεδομένα, 2) υποστηρίζει τη λειτουργία εκτός σύνδεσης, δηλαδή διαχωρίζει την παραγωγή των αποτελεσμάτων από τη λήψη τους. Η βασική καινοτομία του PSoup είναι η αντιμετώπιση ερωτημάτων και δεδομένων *συμμετρικά*. Η άφιξη νέων δεδομένων πυροδοτεί την εκτέλεση ενεργών ερωτημάτων, ενώ με την υποβολή νέων ερωτημάτων γίνεται πρόσβαση σε ιστορικά δεδομένα. Η εκτέλεση των ερωτημάτων πραγματοποιείται ως σύνδεση μεταξύ αυτών και των δεδομένων όπως φαίνεται στο σχήμα 4.3. Τα αποτελέσματα υλοποιούνται συνεχώς και διατηρούνται σε μια ειδική δομή αποτελεσμάτων (Results Structure) μέχρι να ζητηθούν ρητά.



Σχήμα 4.3: *Psoup. (Πηγή: [CCC+02])*

4.3 Επεξεργασία ρευμάτων δεδομένων με το σύστημα TelegraphCQ

4.3.1 Κύρια χαρακτηριστικά του TelegraphCQ

Το TelegraphCQ σχεδιάστηκε για να συμπληρώσει τα προηγούμενα Συστήματα στους εξής τομείς **[CCC+02]**:

- χρονοδρομολόγηση και διαχείριση πόρων για πολλαπλά ερωτήματα
- υποστήριξη για out-of-core δεδομένα
- διαβάθμιση στην προσαρμοστικότητα
- δυναμική επιλογή ποιότητας υπηρεσιών (QoS)
- παράλληλη και κατανεμημένη επεξεργασία σε συστάδες υπολογιστών.

Η πιο πρόσφατη και βελτιωμένη έκδοση του συστήματος, η *TelegraphCQ v2.0* υποστηρίζει τα επιπλέον χαρακτηριστικά **[siteTCQ]**:

- μεγαλύτερη σταθερότητα
- καλύτερες επιδόσεις
- νέο συντακτικό για τα παράθυρα, της μορφής:
  [RANGE BY' ' SLIDE BY' ' START AT' ']
- μια επιπρόσθετη συναθροιστική συνάρτηση: wtime(*) η οποία επιστρέφει το τελευταίο χρονόσημο του τρέχοντος χρονικού παραθύρου στην έξοδο
- διατύπωση ένθετων υποερωτημάτων (nested queries) με τη χρήση προτάσεων WITH συντακτικού παρόμοιου με της SQL:1999.

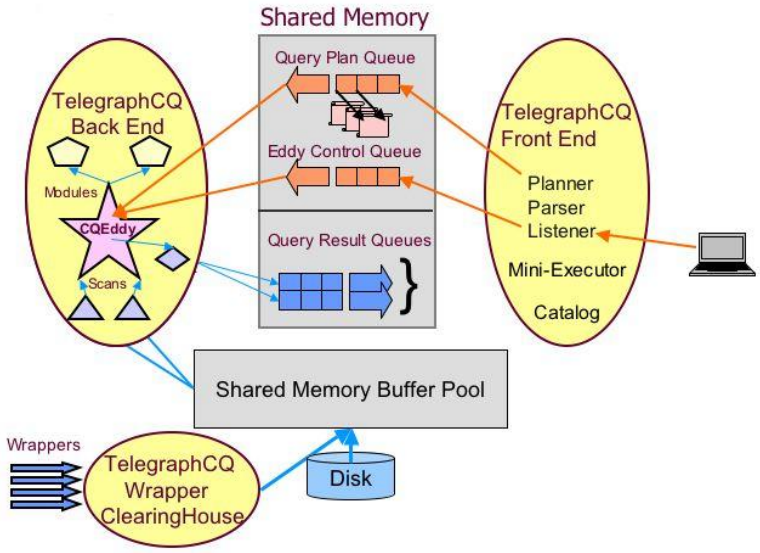
♦ δημιουργία περιλήψεων για τις πλειάδες που απορρίπτονται λόγω φόρτου πληροφορίας.

4.3.2 Σχεδιασμός του συστήματος

Η υλοποίηση του TelegraphCQ πραγματοποιήθηκε με χρήση C/C++. Ο κυριότερος λόγος που προτιμήθηκαν έναντι της Java, είναι ότι επέτρεψαν την ανάπτυξη του συστήματος πάνω στον προϋπάρχοντα κώδικα της PostgreSQL. Αν και το TelegraphCQ διαφέρει ουσιωδώς από τους παραδοσιακούς μηχανισμούς εκτέλεσης ερωτημάτων, θεωρήθηκε σκόπιμη η επαναχρησιμοποίηση έτοιμων μονάδων επεξεργασίας όπου αυτό ήταν δυνατό.

Το μοντέλο επεξεργασίας της PostgreSQL δημιουργεί μια διεργασία για κάθε σύνδεση μέσω του *Postmaster*. Αυτή περιλαμβάνει έναν *Listener*, ο οποίος ανταποκρίνεται στις αιτήσεις του χρήστη για να επιστρέφει αποτελέσματα σε αυτόν και έναν *Executor* ερωτημάτων, που εκτελεί τα ερωτήματα που υποβάλει ο χρήστης, αφού γίνει προηγουμένως η κατάλληλη προεργασία (parsing, optimization) και τοποθετηθούν στο προσχέδιο εκτέλεσης. Οι δομές δεδομένων που είναι κοινές για τις διεργασίες τοποθετούνται στην κοινή μνήμη.

Τα μέρη της PostgreSQL τα οποία χρησιμοποιούνται με ελάχιστες αλλαγές στο TelegraphCQ είναι τα : Postmaster, Listener, System Catalog, Query Parser, Optimizer. Οι κυριότερες καινοτομίες του αφορούν την υποστήριξη ρευμάτων δεδομένων, ερωτημάτων διαρκείας, την κοινή επεξεργασία και την προσαρμοστικότητα. Ένα καίριο σχεδιαστικό ζήτημα ήταν η χρήση νημάτων εκτέλεσης. Η απόδοση νέας διεργασίας εξυπηρέτησης σε κάθε νέα σύνδεση όπως πραγματοποιείται στην PostgreSQL δεν είναι ασφαλής (thread-safe), ενώ η χρήση πολλαπλών νημάτων θεωρείται βασικό χαρακτηριστικό του TelegraphCQ. Τελικά αποφασίστηκε η υιοθέτηση του υπάρχοντος μοντέλου διεργασιών στα τμήματα που επαναχρησιμοποιούνται, ενώ το μοντέλο πολλαπλών νημάτων (multi-threading) ακολουθείται μόνο σε νέα τμήματα.



Σχήμα 4.4: *Αρχιτεκτονική του TelegraphCQ v2.0 (Πηγή: [siteTCQ])*

Όπως φαίνεται στο σχήμα 4.4 οι βασικές διεργασίες είναι τρεις, αυτές που βρίσκονται σε ελλειπτικά πλαίσια. Η δεξιότερη καλείται "FrontEnd", δημιουργείται σε κάθε νέα σύνδεση. Δέχεται τις αιτήσεις για πολλαπλά ερωτήματα διαρκείας από τον χρήστη και τα εντάσσει σε ένα προσαρμοζόμενο σχέδιο εκτέλεσης. Το τελευταίο τοποθετείται σε μια ουρά πλάνων εκτέλεσης (QPQueue) που βρίσκεται στην κοινή μνήμη και απ΄ όπου θα ληφθούν για να εκτελεστούν από τον Executor. Η διεργασία "wrapper" αναλαμβάνει τις λειτουργίες μεταφοράς δεδομένων στο σύστημα από τις πηγές τους.

4.3.3 Η μονάδα εκτέλεσης ερωτημάτων στο TelegraphCQ

Ένα πρόβλημα που ανακύπτει είναι η χρήση νημάτων στην εκτέλεση ερωτημάτων, ούτως ώστε να επιτυγχάνεται προσαρμοστικότητα με μικρό κόστος. Θεωρητικά, θα μπορούσε να χρησιμοποιηθεί μια διεργασία για όλα τα ερωτήματα, με την οντότητα Eddy να λειτουργεί ως ένας γενικότερος χρονοδρομολογητής. Κάτι τέτοιο δεν θα ήταν αποδοτικό, αφού η οντότητα Eddy δεν έχει σχεδιαστεί γι΄ αυτό το σκοπό. Από την άλλη, η χρήση διαφορετικού νήματος για την εκτέλεση κάθε ερωτήματος είναι ιδιαίτερα αντιοικονομική, αν και η ύπαρξη πολλαπλών νημάτων είναι επιθυμητή για την εκμετάλλευση των δυνατοτήτων της παράλληλης επεξεργασίας.

Η λύση που ακολουθείται είναι η δημιουργία *«Αντικειμένων εκτέλεσης» (Execution Objects)*, σε κάθε ένα από τα οποία αποδίδεται ένα ξεχωριστό νήμα. Τα τελευταία αποτελούνται από έναν χρονοδρομολογητή, ουρές συμβάντων και από ένα *σύνολο «Μονάδων διεκπεραίωσης» (Dispatch units)* , οι οποίες αντιπροσωπεύουν οντότητες εργασίας που πραγματοποιούνται στο σύστημα και που μπορούν να εκτελεστούν βάσει ενός κοινού πλάνου χρονοδρομολόγησης. Η επεξεργασία των μονάδων διεκπεραίωσης μπορεί να πάρει να είναι μία από τις παρακάτω διαδικασίες:

1. Ένα παραδοσιακό προσχέδιο εκτέλεσης της PostgreSQL με τον κλασικό επεξεργαστή ερωτημάτων.

2. Ένα προσχέδιο εκτέλεσης με έναν Eddy και τελεστές Fjords.

3. Ένα προσχέδιο εκτέλεσης σαν το παραπάνω με τη διαφορά ότι θα αναφέρεται σε ένα ερώτημα διαρκείας με κοινή επεξεργασία.

Βασικός παράγοντας επιτυχίας της προσέγγισης αυτής είναι ο διαχωρισμός των ερωτημάτων σε ομάδες ανάλογα με τα ρεύματα και τους πίνακες δεδομένων πάνω στους οποίους εκτελούνται, ώστε να γίνεται όσο το δυνατόν αποτελεσματικότερα η κοινή επεξεργασία τους.

4.3.4 Πρόσβαση στα δεδομένα

Με την χρήση του μηχανισμού των wrappers, η πρόσβαση στα νεοαφιχθέντα δεδομένα θυμίζει αρκετά αυτή σε παρελθοντικά ή ακόμη και σε στατικά δεδομένα. Καθώς η λήψη δεδομένων μπορεί να προκαλέσει αναστολή των υπόλοιπων λειτουργιών του συστήματος, υλοποιείται σε μια ξεχωριστή διεργασία. Οι πηγές δεδομένων μπορούν να πυροδοτούνται είτε από το σύστημα όπως στις παραδοσιακές βάσεις δεδομένων (pull sources), είτε από μόνες τους (push sources). Ο wrapper μπορεί να επιλέξει τη σύνδεσή του

με αυτές, ενώ σε αντίθετη περίπτωση η πηγή θα πρέπει να συνδεθεί σε μια γνωστή θύρα απ' όπου «ακούει» ο wrapper. Οι *streamers* είναι ένα εργαλείο της διεργασίας των wrappers που δίνουν στα εισερχόμενα ρεύματα δεδομένων την μορφή πλειάδων, ώστε να είναι επεξεργάσιμα από τον εκτελεστή ερωτημάτων.

4.3.5 Συντακτικό του TelegraphCQ

Το συντακτικό του TelegraphCQ για διατύπωση ερωτημάτων διαρκείας σε ρεύματα δεδομένων ακολουθεί την γενική μορφή της SQL:1999, με την προσθήκη στοιχείων που αφορούν τη διαχείριση ρευμάτων δεδομένων. Συγκεκριμένα στην πρόταση FROM δίνεται μια λίστα από τα εισερχόμενα στο σύστημα ρεύματα δεδομένων που χρησιμοποιούνται στο ερώτημα. Κάθε ρεύμα υποχρεωτικά συνοδεύεται από ένα χρονικό παράθυρο.

Το TelegraphCQ υποστηρίζει την εκτέλεση ερωτημάτων διαρκείας σε οποιονδήποτε συνδυασμό ρευμάτων δεδομένων και στατικών πινάκων. Καθώς τα ρεύματα ενδέχεται να έχουν απεριόριστο μέγεθος, λειτουργίες όπως η σύνδεση (join) είναι αδύνατο να πραγματοποιηθούν αποτελεσματικά. Η εισαγωγή παραθύρων δίνει μια λύση σε αυτό το πρόβλημα, καθώς απομονώνει τμήματα των ρευμάτων δεδομένων ώστε να συμμετέχει ένα ορισμένο μέρος τους στον υπολογισμό των απαντήσεων κάθε φορά. Οι μηχανισμοί παραθύρων εφαρμόζονται στα δεδομένα που έχουν ήδη φτάσει αλλά και σε αυτά που αναμένονται μελλοντικά. Επίσης παρέχει ευέλικτους μηχανισμούς για τη λήψη των αποτελεσμάτων, υποστηρίζοντας τόσο την άμεση εξαγωγή τους στον χρήστη (push) του CACQ, όσο και την παρουσίασή τους κατά βούληση (pull) του PSoup.

Η γενική μορφή της δηλωτικής διατύπωσης ερωτήματος διαρκείας για το TelegraphCQ είναι η εξής **[siteTCQ]**:

```
SELECT      <select_list>
FROM        <Stream_ list>
WHERE       <predicates>
GROUP BY    <group_by_expressions>
HAVING      <predicate>;     //προαιρετικά
```

Κάθε ρεύμα της λίστας Stream_ list ακολουθείται από τη δήλωση του παραθύρου του. Στο TelegraphCQ, το παράθυρο που εφαρμόζεται πάνω σε ένα ρεύμα δεδομένων, ορίζει το σύνολο των πλειάδων που θα συμμετέχουν στον υπολογισμό της απάντησης του ερωτήματος. Με την εκτέλεση του ερωτήματος στα δεδομένα του παραθύρου προκύπτει ένα σύνολο αποτελεσμάτων. Η συνολική απάντηση του παρουσιάζεται στον χρήστη σαν μια ακολουθία αποτελεσμάτων, με κάθε ένα να συσχετίζεται με μια συγκεκριμένη χρονική στιγμή. Ο χρόνος μπορεί να θεωρηθεί ίδιος με το φυσικό (physical), ή να οριστεί σαν μια ακολουθία αριθμημένων πλειάδων (logical).

Το TelegraphCQ υποστηρίζει μόνο **Κυλιόμενα χρονικά παράθυρα (sliding timebased windows),** των οποίων το παλιότερο και το νεότερο άκρο μετακινούνται προς τα εμπρός με την άφιξη νέων δεδομένων.

Τα χαρακτηριστικά του παραθύρου που πρέπει να καθορίσει ο χρήστης είναι:

- Το *εύρος* του παραθύρου (*range*)

- Το *χρονόσημο* έναρξης του παραθύρου (πχ. '2007-10-1 20:22:00 EET').

- Το *βήμα* μεταξύ των διαδοχικών παραθύρων *(slide)*. Στη γενική περίπτωση του κυλιόμενου παραθύρου, το βήμα είναι μικρότερο από το εύρος του, σε περίπτωση που είναι ίσο με το εύρος έχουμε την υποπερίπτωση του επάλληλου παραθύρου (tumbling window) με μη επικαλυπτόμενα στιγμιότυπα στην έξοδό του, ενώ αν είναι μεγαλύτερο από το εύρος του παραθύρου ενδέχεται να παραληφθούν δεδομένα κατά την εκτέλεση του ερωτήματος.

Η σύνταξη των παραθύρων στο TelegraphCQ v2.0 είναι της μορφής:

```
[RANGE BY'x seconds'
 SLIDE BY'y seconds'
 START AT'<timestamp>']
```

Η *σύνδεση (join)* δεν υποστηρίζεται με συγκεκριμένες δεσμευμένες λέξεις, αλλά με ενσωμάτωση του κατηγορήματός της στην πρόταση WHERE.

Για παράδειγμα το ερώτημα

```
SELECT     R.i, R.j, count(*)
FROM  R [RANGE BY 't1 seconds' SLIDE BY 't2 seconds'],
      S [RANGE BY 't3 seconds' SLIDE BY 't4 seconds']
WHERE R.k = S.k
```

υλοποιεί τη σύνδεση μεταξύ των ρευμάτων R και S (έχοντας περάσει μέσα από κατάλληλα χρονικά παράθυρα) πάνω στο κοινό τους πεδίο k (R join S on R.k=S.k).

Επίσης, στην πιο πρόσφατη έκδοση του συστήματος υποστηρίζεται και η χρήση ένθετων υποερωτημάτων (*nested subqueries*) τα οποία υλοποιούνται με WITH clauses **[siteTCQ]**. Η σύνταξη που ακολουθείται για τη διατύπωση ένθετων υποερωτημάτων έχει τη γενική μορφή:

```
WITH NewStream_1 AS (
      SELECT <select_list>
      FROM <stream_list>
      ...)
SELECT <select_list>
FROM NewStream_1 [<window>]
...
```

Κεφάλαιο 5

Αρχιτεκτονική εφαρμογής

5.1 Εισαγωγή

Σκοπός της παρούσας εργασίας είναι η υλοποίηση ενός *γραφικού περιβάλλοντος χρήστη* (*GUI*) για εύκολη και ορθή διατύπωση πολλαπλών ερωτημάτων διαρκείας πάνω σε ρεύματα δεδομένων, καθώς και online προβολή των αποτελεσμάτων τους σε πίνακες. Τα ερωτήματα θα περιλαμβάνουν διάφορους τύπους παραθυρικών δομών και επιλεγμένους σχεσιακούς τελεστές όπως σύνδεση, συνάθροιση και εφαρμογή κατηγορημάτων για φιλτράρισμα των ρευμάτων εξόδου.

Η σύνθεση των προσχεδίων εκτέλεσης ερωτημάτων γίνεται σε μορφή *δένδρων* όπου κόμβοι είναι οι τελεστές του ερωτήματος, ενώ τα *βέλη* που τους συνδέουν δείχνουν τη λογική διασύνδεσή τους.

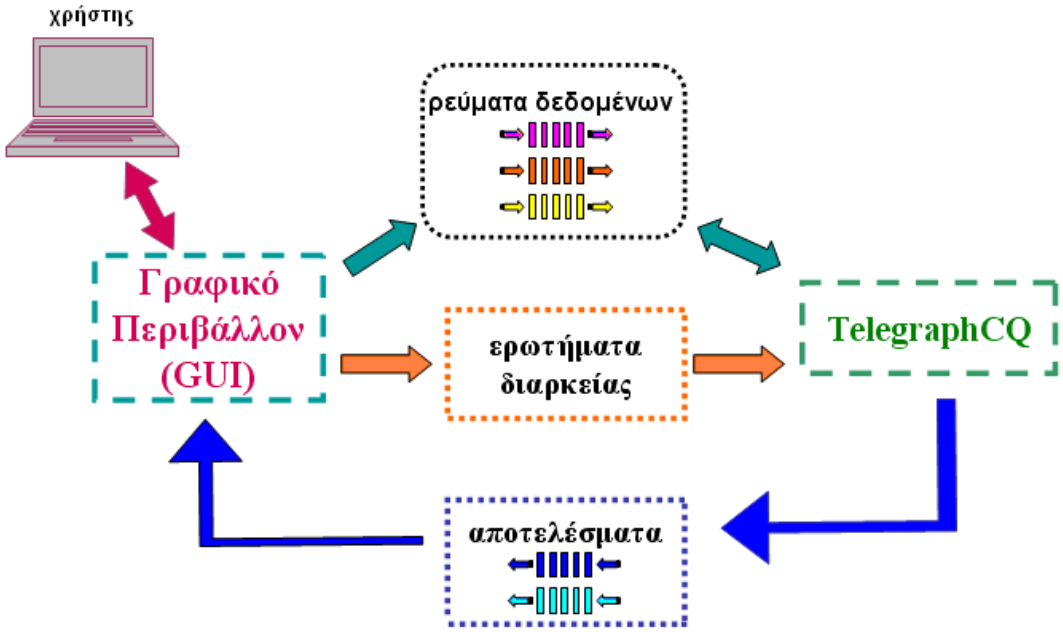
Από το γραφικά σχεδιασμένο προσχέδιο εκτέλεσης θα παράγεται ο κώδικας του ερωτήματος διαρκείας σε γλώσσα SQL, προσαρμοσμένη όμως στο συντακτικό του Συστήματος Διαχείρισης Ρευμάτων Δεδομένων που χρησιμοποιείται.

Τα αποτελέσματα του ερωτήματος, δηλαδή οι πλειάδες του ρεύματος εξόδου, θα έχουν μια συνεχή ροή και θα εμφανίζονται σε πίνακες δυναμικά και σε πραγματικό χρόνο.

Η δημιουργία των ρευμάτων εισόδου καθώς και η εκτέλεση των ερωτημάτων γίνεται μέσω υπάρχοντος μηχανισμού επεξεργασίας ρευμάτων.

Η εφαρμογή αποτελείται από δύο βασικά δομικά στοιχεία:

- **Το Σύστημα Διαχείρισης Ρευμάτων Δεδομένων**. Το TelegraphCQ που έχει επιλεγεί, διαχειρίζεται τα εισερχόμενα ρεύματα δεδομένων και εκτελεί τα ερωτήματα διαρκείας που ορίζονται πάνω σε αυτά, δίνοντας ως έξοδο παράγωγα ρεύματα δεδομένων.
- **Το Γραφικό περιβάλλον διεπαφής(GUI)**. Αποτελεί το πλαίσιο σχεδίασης των ερωτημάτων και το μέσο επικοινωνίας του χρήστη με το Σύστημα Διαχείρισης Ρευμάτων Δεδομένων.



Σχήμα 5.1: *Αρχιτεκτονική του συστήματος.*

Το γραφικό περιβάλλον υλοποιήθηκε σε γλώσσα προγραμματισμού JAVA, ώστε να επιτευχθεί ανεξαρτησία πλατφόρμας εκτέλεσης του GUI.

Στο *σχήμα 5.1* φαίνεται το διάγραμμα της αρχιτεκτονικής της εφαρμογής. Ο χρήστης αλληλεπιδρά με το γραφικό περιβάλλον για να δημιουργήσει τα νέα ρεύματα δεδομένων και για να σχεδιάσει και να υποβάλλει ερωτήματα διαρκείας στο ΣΔΡΔ.

Η εντολή για το σχηματισμό νέων ρευμάτων γίνεται μέσα από ειδικές φόρμες του περιβάλλοντος και αποστέλλεται στο TelegraphCQ, το οποίο προσομοιώνει ρεύματα δεδομένων χρησιμοποιώντας την πληροφορία που βρίσκεται σε πλειάδες δεδομένων από ειδικά διαμορφωμένα ASCII αρχεία. Ο ρυθμός άφιξης των πλειάδων καθορίζεται επίσης από τον χρήστη μέσα από το γραφικό περιβάλλον.

Η διατύπωση των ερωτημάτων διαρκείας γίνεται με γραφικό τρόπο μέσα από ειδικές φόρμες σχεδίασης των προσχεδίων εκτέλεσης των ερωτημάτων. Κάθε προσχέδιο σχηματίζεται σε μορφή δένδρου με τη βοήθεια ειδικών κόμβων που υλοποιούν τους τελεστές (επιλογή, προβολή, σύνδεση, κτλ.) του ερωτήματος. Οι τελεστές αυτοί είναι διαθέσιμοι στον χρήστη πάνω σε μια ειδική εργαλειοθήκη τελεστών, από την οποία με χρήση drag-and-drop σύρονται και τοποθετούνται κατάλληλα πάνω στη φόρμα σχεδίασης του ερωτήματος. Ακολούθως, γίνεται η διασύνδεση των τελεστών του δένδρου με βέλη, τα οποία δείχνουν τη ροή της πληροφορίας μεταξύ των κόμβων του δένδρου. Κάθε τελεστής παραμετροποιείται μέσα από ειδικές φόρμες του γραφικού περιβάλλοντος, για τη ρύθμιση των ιδιοτήτων του. Η παραμετροποίηση αυτή γίνεται ως επί το πλείστον εύκολα και γρήγορα από τον χρήστη με επιλογή τιμών από έτοιμα σύνολα επιλογών (λ.χ. πεδία πινάκων, συναθροιστικές συναρτήσεις, κτλ.).

Μετά την ολοκλήρωση του προσχεδίου, η εφαρμογή διατρέχει το δένδρο, ελέγχει την συντακτική ορθότητά του και δημιουργεί την εντολή SQL του ερωτήματος διαρκείας. Η σύνταξη της εντολής προβάλλεται στη φόρμα και ο χρήστης –εφόσον επιθυμεί– μπορεί να την τροποποιήσει πριν την υποβάλλει.

Στην εφαρμογή το TelegraphCQ εκτελεί όσα ερωτήματα του υποβάλλονται μέσα από το γραφικό περιβάλλον και επιστρέφει τα αποτελέσματά τους σταδιακά (incrementally) καθώς αυτά παράγονται. Το ρεύμα εξόδου κάθε ερωτήματος προβάλλεται στην οθόνη από τον πίνακα των αποτελεσμάτων της φόρμας του ερωτήματος.

Πολλαπλά ερωτήματα θα μπορούν να σχεδιάζονται γραφικά να διατυπώνονται με δηλωτικό τρόπο (SQL) και να εκτελούνται ταυτόχρονα.

Τέλος, ο χρήστης έχει τη δυνατότητα αναστολής εκτέλεσης (suspend) και επανεκκίνησης (resume) κάθε ερωτήματος.

5.2 Υλοποίηση συστήματος

Η γλώσσα προγραμματισμού Java αποτέλεσε το εργαλείο ανάπτυξης του γραφικού περιβάλλοντος διεπαφής χρήστη. Συνεπώς, οι προγραμματιστικές οντότητες της εφαρμογής είναι ***κλάσεις(classes)*** της γλώσσας Java.

- **mainform** (κεντρικό παράθυρο της εφαρμογής)
- **Connect** (σύνδεση με JDBC)
- **QueryForm** (φόρμα σύνταξης ερωτημάτων διαρκείας και προβολής αποτελεσμάτων)
- **Operator** (τελεστές ερωτημάτων διαρκείας)
- **SQLcodeGenerator** (δημιουργία του κώδικα του ερωτήματος σε SQL από το δένδρο προσχεδίου εκτέλεσης)
- **CursorThread** (εκτέλεση ερωτήματος διαρκείας)

Η κλάση του κεντρικού παραθύρου (mainform) είναι αυτή που υλοποιεί το μεγαλύτερο μέρος του γραφικού περιβάλλοντος διεπαφής και ο υποδοχέας όλων των υπολοίπων κλάσεων.

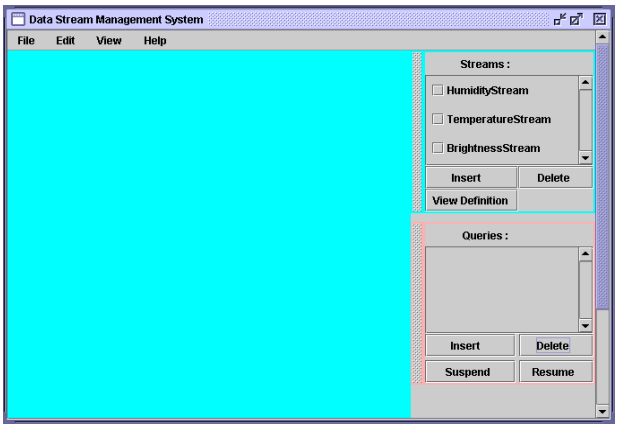
Η Connect φροντίζει για τη διασύνδεση του γραφικού περιβάλλοντος με το Σύστημα Διαχείρισης Ρευμάτων Δεδομένων (DSMS) TelegraphCQ.

Σε κάθε φόρμα σχεδίασης ερωτημάτων διαρκείας (QueryForm) γίνεται γραφικά ο σχηματισμός του δένδρου εκτέλεσης του ερωτήματος διαρκείας και η εμφάνιση του παραγόμενου κώδικα σε SQL καθώς και η προβολή των αποτελεσμάτων σε ειδικούς πίνακες.

Οι κλάσεις των τελεστών (Operator) υλοποιούν καθέναν από τους τελεστές: προβολής, επιλογής, σύνδεσης, απαλοιφής διπλοτύπων, ομαδοποίησης καθώς και τους βασικότερους τύπους παραθυρικών δομών (παράθυρα πλειάδων, χρονικά, μεριστικά και οροσήμου) για τη δημιουργία του δένδρου εκτέλεσης με γραφικό τρόπο.

Η κλάση SQLcodeGenerator διατρέχει το δένδρο εκτέλεσης κάθε ερωτήματος και παράγει μια συμβολοσειρά (String) που περιέχει τον κώδικα του ερωτήματος σε γλώσσα SQL προσαρμοσμένη στο συντακτικό του TelegraphCQ.

Τέλος, οι κλάσεις εκτέλεσης ερωτημάτων (CursorThread) είναι οι κλάσεις που εκτελούν την υποβολή των ερωτημάτων διαρκείας στο TelegraphCQ και την μετέπειτα αποστολή των αποτελεσμάτων τους στο γραφικό περιβάλλον διασύνδεσης, στους πίνακες αποτελεσμάτων της QueryForm.



Σχήμα 5.2: *Το κύριο παράθυρο της εφαρμογής (mainform).*

Επίσης, μεταξύ των στιγμιοτύπων των παραπάνω οντοτήτων υπάρχουν επιμέρους αλληλοσυσχετίσεις και συνεργασίες καθώς οι περισσότερες επιτελούν βασικές λειτουργίες, απαραίτητες σε διάφορες φάσεις κατά την περίοδο εκτέλεσης της εφαρμογής. Στη συνέχεια αναπτύσσεται λεπτομερώς ο ρόλος και τα βασικά χαρακτηριστικά καθεμιάς από τις βασικές προγραμματιστικές οντότητες.

5.2.1 Κλάση Mainform

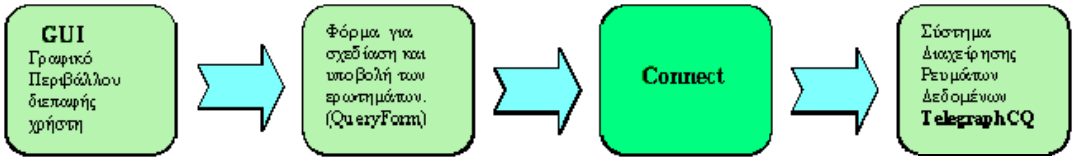
Η κλάση αυτή είναι ο συνδετικός κρίκος όλων των συστατικών του γραφικού περιβάλλοντος. Περιέχει το βασικό μενού με όλες τις λειτουργίες του συστήματος (main menu), τις εργαλειοθήκες διαχείρισης ρευμάτων δεδομένων (stream toolbar) και ερωτημάτων διαρκείας (query toolbar) και έχει το ρόλο του υποδοχέα για όλα τα άλλα γραφικά στοιχεία της εφαρμογής.

Οι φόρμες δημιουργίας ρευμάτων δεδομένων, οι φόρμες διατύπωσης ερωτημάτων διαρκείας, ο επιλογέας αρχείων (για τα ASCII αρχεία τροφοδοσίας ρευμάτων) καθώς και τα μηνύματα ειδοποίησης (Alert Box Messages) αποτελούν όλα εσωτερικά παράθυρα του κεντρικού πάνελ (desktop panel) του παραθύρου της κλάσης mainform.

Στο *Σχήμα 5.2* φαίνεται το παράθυρο της εφαρμογής χωρίς ανοικτές εσωτερικές φόρμες. Διακρίνονται το βασικό μενού στο επάνω μέρος και οι εργαλειοθήκες ρευμάτων και ερωτημάτων στα δεξιά της κεντρικής φόρμας.

5.2.2 Κλάση Connect

Βασική πτυχή της εφαρμογής είναι η διατύπωση και υποβολή ερωτημάτων διαρκείας. Το σύστημα διαχείρισης ρευμάτων δέχεται ερωτήματα διαρκείας μέσα από συνδέσεις στον ειδικό εξυπηρετητή της postgreSQL, τον postmaster. Μία σύνδεση μπορεί να υποστηρίξει μόνο ένα ενεργό ερώτημα κάθε φορά. Σε μια εφαρμογή λοιπόν που επιδιώκεται η ταυτόχρονη εκτέλεση πολλαπλών ερωτημάτων, είναι απαραίτητη η δημιουργία πολλών συνδέσεων με το TelegraphCQ, μία για κάθε ερώτημα. Αυτή την ανάγκη καλύπτει η κλάση Connect: *κάθε στιγμιότυπό της επιστρέφει μια σύνδεση με το Σύστημα Διαχείρισης Ρευμάτων Δεδομένων (Σχήμα 5.3).*



Σχήμα 5.3: *Λειτουργία της κλάσης Connect: Σύνδεση γραφικού περιβάλλοντος διεπαφής με το Σύστημα Διαχείρισης Ρευμάτων Δεδομένων.*

Το άλλο βασικό χαρακτηριστικό της, είναι ότι σε σχέση με τον χρήστη πραγματοποιείται στο «παρασκήνιο». Κάθε φορά που ο χρήστης υποβάλλει κάποιο ερώτημα δημιουργείται ένα στιγμιότυπο της Connect και επομένως μία νέα σύνδεση με τον postmaster. Η παραπάνω διαδικασία γίνεται μέσα από κώδικα JAVA, χωρίς την ανάγκη για περαιτέρω ανάμιξη του χρήστη. Το σημαντικό σε αυτήν την ιδιότητα, είναι ότι παρακάμπτει μια «άκομψη» και χρονοβόρα διαδικασία που θα έπρεπε να ακολουθήσει κανείς μέχρι τώρα:

1. να ανοίξει ένα παράθυρο εντολών

2. να μπει σαν χρήστης στην postgreSQL

3. στη συνέχεια να κάνει login ως χρήστης στον ειδικό postmaster του TelegraphCQ.

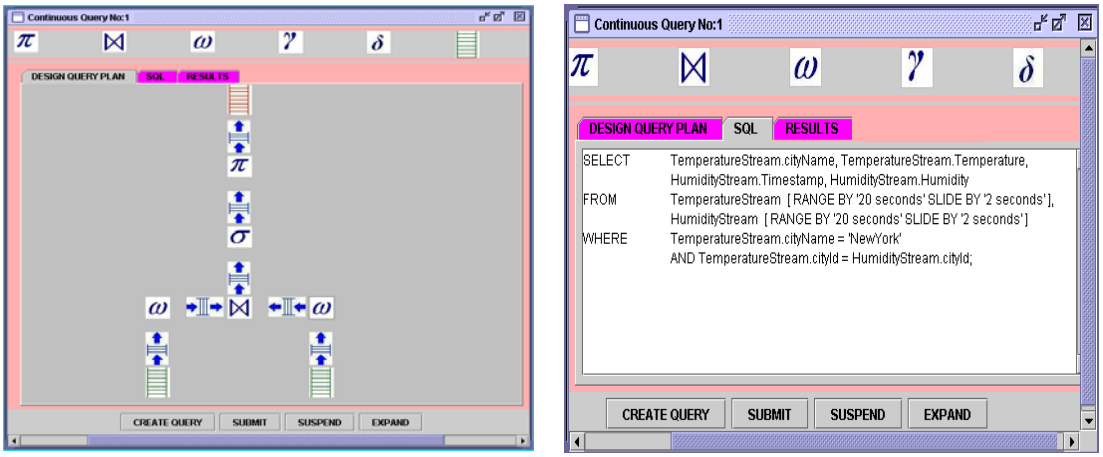
Η κλάση Connect, χρησιμοποιεί έναν οδηγό μεταγλώττισης (org.postgres.sql driver) που παρέχει η Postgres , ώστε η εκτέλεση των παραπάνω να γίνεται «αθέατα» μέσα από ένα πρόγραμμα σε JAVA. Με τη δημιουργία ενός στιγμιοτύπου της Connect είναι διαθέσιμο ένα αντικείμενο τύπου σύνδεσης (connection). Το στιγμιότυπο σύνδεσης χρησιμοποιείται για τη δημιουργία ενός στιγμιοτύπου της κλάσης statement. Η statement παρέχει όλες τις συναρτήσεις που χρειάζονται για την υποβολή SQL εντολών στη βάση. Οι σημαντικότερες μέθοδοι της statement αφορούν την εκτέλεση ερωτήματος, τη λήψη των αποτελεσμάτων του όπως επίσης και τις βασικές πράξεις σε μία βάση (insert, update, delete). Η συνδυασμένη χρήση των Connect και statement αντικαθιστούν την χρήση παραθύρων εντολών και επιτρέπουν στον χρήστη να έρχεται σε επαφή μόνο με το γραφικό περιβάλλον της εφαρμογής για να ενεργοποιήσει ένα ερώτημα διαρκείας. Συνεπώς, η Connect είναι απαραίτητη για την αυτονομία και την ευελιξία της εφαρμογής.

5.2.3 Κλάση QueryForm

Η κλάση αυτή προσφέρει στον χρήστη τα κατάλληλα γραφικά εργαλεία σύνταξης του ερωτήματος και εποπτείας των αποτελεσμάτων του.

Τα στιγμιότυπα της κλάσης **QueryForm** αποτελούν ένα σημαντικό μέρος του γραφικού περιβάλλοντος διεπαφής παρέχοντας τα εξής εργαλεία:

- Φόρμα σχεδίασης του προσχεδίου εκτέλεσης του ερωτήματος.
- Εργαλειοθήκη τελεστών και παραθύρων με τα οποία γίνεται η σχεδίαση με drag-and-drop.
- Περιοχή κειμένου όπου προβάλλεται ο παραγόμενος κώδικας του ερωτήματος σε γλώσσα SQL.
- Πίνακα προβολής των αποτελεσμάτων, με δυναμική ανανέωση.



Σχήμα 5.4: *Φόρμα διατύπωσης ερωτημάτων διαρκείας (QueryForm).*

Κατά την εκτέλεση, ανοίγει ένα νέο παράθυρο εντός της κυρίως εφαρμογής, το οποίο διαθέτει τρεις καρτέλες (tabs), από τις οποίες μπορεί να επιλέξει ο χρήστης να παρακολουθεί μόνο την μία. Η φόρμα περιέχει επίσης ένα πίνακα ελέγχου (control panel) με επιλογές: δημιουργίας του κώδικα σε SQL, υποβολής και παύσης της εκτέλεσης του ερωτήματος.

Στην πρώτη καρτέλα βρίσκεται μια φόρμα όπου ο χρήστης συντάσσει το ερώτημα που τον ενδιαφέρει. Στη δεύτερη καρτέλα τυπώνεται ο παραγόμενος κώδικας σε SQL. Στην τρίτη καρτέλα τοποθετείται ένας πίνακας ο οποίος προβάλλει τα τρέχοντα αποτελέσματα του ερωτήματος με τα πιο πρόσφατα να τοποθετούνται στην κορυφή. Ο πίνακας αποτελεσμάτων δημιουργείται μόλις υποβληθεί το ερώτημα.

Ο χρήστης μπορεί να εξετάζει είτε τη φόρμα σχεδιασμού του ερωτήματος είτε την περιοχή κειμένου του σε SQL είτε τον πίνακα αποτελεσμάτων του.

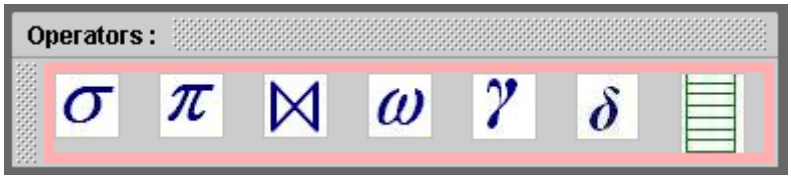
Η κεντρική κλάση μπορεί να δημιουργήσει πολλαπλά στιγμιότυπα της QueryForm με βασικό περιορισμό τις επιπτώσεις της ταυτόχρονης εκτέλεσης τους στους πόρους του συστήματος. Ενδεχομένως υπάρχει τότε μείωση απόδοσης της εφαρμογής, αν ως μέτρο των επιδόσεων θεωρηθεί ο χρόνος απόκρισης του γραφικού περιβάλλοντος στις ενέργειες του χρήστη και στο ρυθμό εισαγωγής των δεδομένων.

Η QueryForm παρέχει ακόμη τις δυνατότητες:

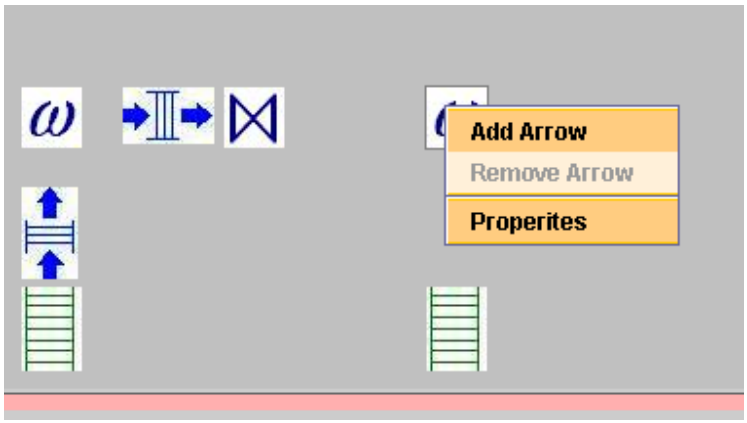
- Παύση κι επανεκκίνηση του ερωτήματος που περιέχει (Suspend, Resume).
- Τροποποίηση της διατύπωσής του ερωτήματος στην περιοχή κειμένου (editable text area) του κώδικα σε γλώσσα SQL πριν την εκτέλεσή του.
- Αύξηση του μεγέθους της φόρμας σύνταξης ερωτημάτων για μεγαλύτερα και πολυπλοκώτερα δέντρα προσχεδίων εκτέλεσης (Expand).

5.2.4 Κλάση Operator

Η κλάση αυτή υλοποιεί τους τελεστές που αποτελούν τα δομικά στοιχεία κάθε δένδρου εκτέλεσης ερωτημάτων διαρκείας. Κάθε τελεστής έχει ένα συγκεκριμένο τύπο (επιλογή, προβολή, σύνδεση, ομαδοποίηση, απαλοιφή διπλοτύπων, παράθυρο) και το αντίστοιχο εικονίδιο που τον αντιπροσωπεύει.



Σχήμα 5.5: *Εργαλειοθήκη τελεστών (Operators) για τη σύνθεση (με drag-and-drop) προσχεδίων εκτέλεσης ερωτημάτων διαρκείας.*

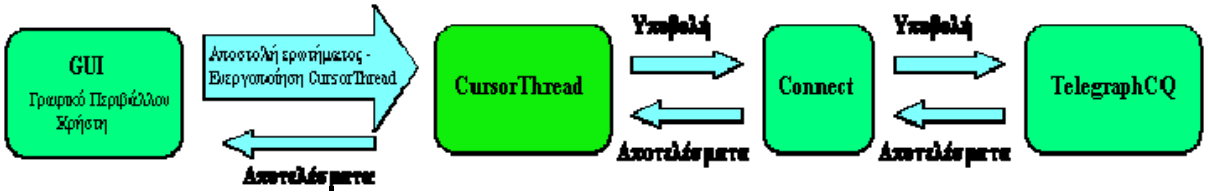


Σχήμα 5.6: *Άνοιγμα Popup Menu ενός τελεστή παραθύρου πάνω στο πάνελ σχεδίασης.*

Επίσης τα ρεύματα δεδομένων εισόδου υλοποιούνται ως τελεστές στο προσχέδιο εκτέλεσης. Ο χρήστης επιλέγει από την εργαλειοθήκη τελεστών αυτούς που χρειάζεται για να συνθέσει το δένδρο και με drag-and-drop τους τοποθετεί κατάλληλα πάνω στο πάνελ της πρώτης καρτέλας της φόρμας του ερωτήματος. Στο *Σχήμα 5.5* φαίνεται η εργαλειοθήκη τελεστών με όλους τους προαναφερθέντες τελεστές που υλοποιήθηκαν στην εφαρμογή.

Η κλάση Operator περικλείει άλλες δύο κλάσεις. Η πρώτη υλοποιεί ένα Popup menu που ανοίγει με δεξί κλικ πάνω στο εικονίδιο κάθε τελεστή. Η δεύτερη υλοποιεί την φόρμα παραμετροποίησης όπου ρυθμίζονται οι τιμές των ιδιοτήτων κάθε τελεστή (*π.χ. εύρος παραθύρου, κριτήριο επιλογής του τελεστή selection κτλ.*) και ανοίγει με την επιλογή "Properties" από το Popup menu. Το Popup menu επίσης έχει τις επιλογές σύνδεσης τελεστή με άλλον μέσω βέλους ("Add Arrow"), και αφαίρεση βέλους ("Remove Arrow"), όπως φαίνεται και στο *Σχήμα 5.6*.

Τα βέλη υλοποιούνται και αυτά ως τελεστές χωρίς κάποια λειτουργία (dummy operators) που απλά δείχνουν την ροή των πλειάδων πληροφορίας από την έξοδο ενός τελεστή στην είσοδο κάποιου άλλου, στο προσχέδιο εκτέλεσης. Το εικονίδιο που χρησιμοποιείται για το συμβολισμό των βελών είναι αυτό της ουράς. Στο *Σχήμα 6.5* διακρίνονται δυο βέλη: ένα από ένα ρεύμα εισόδου προς ένα παράθυρο και ένα άλλο από το παράθυρο προς ένα τελεστή σύνδεσης.



Σχήμα 5.7: *Λειτουργία της κλάσης CursorThread.*

5.2.5 Κλάση SQLcodeGenerator

Η κλάση αυτή διατρέχει το δένδρο εκτέλεσης κόμβο-κόμβο, διαβάζει όλες τις απαραίτητες πληροφορίες (είδος τελεστή, τιμές των ιδιοτήτων του) και δημιουργεί τον κώδικα του ερωτήματος σε γλώσσα SQL προσαρμοσμένη στο συντακτικό του TelegraphCQ για ρεύματα δεδομένων. Το αποτέλεσμα αποθηκεύεται σε μια μεταβλητή τύπου String και επιστρέφεται στην QueryForm για να προβληθεί στην αντίστοιχη καρτέλα κειμένου.

Η SQLcodeGenerator καλείται όταν ο χρήστης τελειώσει με τη σχεδίαση του προσχεδίου εκτέλεσης και επιλέξει "CREATE QUERY" από το control panel της φόρμας του ερωτήματος.

5.2.6 Κλάση CursorThread

Η CursorThread είναι η κλάση που φροντίζει για την επικοινωνία του γραφικού περιβάλλοντος διεπαφής με το Σύστημα Διαχείρισης Ρευμάτων TelegraphCQ.

Όπως φαίνεται και στο *Σχήμα 5.7* ο ρόλος της είναι:

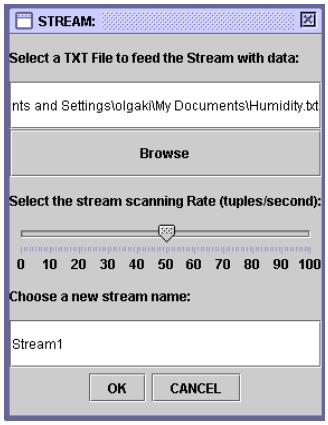
- η υποβολή κάθε ερωτήματος στο TelegraphCQ
- η λήψη των αποτελεσμάτων του και αποστολή τους στους πίνακες προβολής τους.

Αφού συνδεθεί με το TelegraphCQ μέσω της Connect και αποκτήσει ένα στιγμιότυπο της κλάσης statement, είναι σε θέση να υποβάλλει το ερώτημα. Στη συνέχεια, χρησιμοποιεί το στιγμιότυπο της statement και χρησιμοποιεί ένα δρομέα (cursor) ως δείκτη στα αποτελέσματα του ερωτήματος.

Η κλάση υλοποιείται ως νήμα (thread), ώστε να μην «μπλοκάρει» την εφαρμογή με την εκτέλεσή της. Κάθε φορά που το νήμα της ενεργοποιείται μετά την υποβολή του ερωτήματος, φέρνει (fetch) την αμέσως επόμενη πλειάδα από το σύνολο των αποτελεσμάτων και τη στέλνει στην κλάση *QueryForm* για να προβληθεί. Στη συνέχεια το νήμα της CursorThread πέφτει σε σύντομη περίοδο αδράνειας.

5.3 Λειτουργία συστήματος

Στο *Σχήμα 5.2* φαίνεται το κυρίως παράθυρο (main frame) της εφαρμογής. Διακρίνονται οι δύο βασικές εργαλειοθήκες ρευμάτων και ερωτημάτων διαρκείας (Streams toolbar, Queries toolbar).



Σχήμα 5.8: *Φόρμα δημιουργίας ρεύματος.*

5.3.1 Εισαγωγή νέου ρεύματος εισόδου

Η δημιουργία των ρευμάτων εισόδου (input streams) και η τροφοδότησή τους με δεδομένα γίνεται από ειδικά διαμορφωμένα ASCII αρχεία. Η εισαγωγή νέου ρεύματος γίνεται από την εργαλειοθήκη ρευμάτων (streams toolbar) επιλέγοντας "Insert".

Εμφανίζεται έτσι η φόρμα όπου προσδιορίζονται από το χρήστη: το ASCII αρχείο που θα τροφοδοτεί με δεδομένα το νέο ρεύμα, ο ρυθμός (σε πλειάδες ανά δευτερόλεπτο) και το όνομα του ρεύματος. Στο *Σχήμα 5.8* φαίνεται ένα παράδειγμα φόρμας δημιουργίας νέου ρεύματος εισόδου.

Εν συνεχεία η εφαρμογή δίνει στο TelegraphCQ την εντολή δημιουργίας ρεύματος:

```
CREATE STREAM <stream_name> (
              <attribute1> <type1>,
              <attribute2> <type2>,
                 . . .
              <timestamp_attribute> timestamp TIMESTAMPCOLUMN
) TYPE UNARCHIVED;
```

καθώς και την εντολή πρόσθεσης ενός ειδικού βοηθητικού προγράμματος (wrapper) για την εισοχή των δεδομένων στο σύστημα:

```
ALTER STREAM <stream_name>
ADD WRAPPER <wrapper_ name>;
```

Οι εντολές αυτές εμφανίζονται και σε ένα πλαίσιο μηνύματος, για να ενημερωθεί ο χρήστης για την μορφή των πλειάδων του ρεύματος.

Σημειώνεται ότι τα ονόματα των πεδίων (attributes) των πλειάδων του ρεύματος καθώς και οι τύποι των τιμών τους (integer, float , double, char, varchar, timestamp) διαβάζονται από την 1^{H} γραμμή (Header Line) του ASCII αρχείου που επιλέχθηκε, και η οποία θα πρέπει να είναι της μορφής:
`<attribute_name1> <type1>; <attribute_name2> <type2>; ...`

Η τροφοδότηση του ρεύματος με δεδομένα γίνεται με την εντολή:
`cat <ASCII_filename> | ./feed.pl localhost 5533 <wrapper_name>, <stream_name>`

Με την ολοκλήρωση αυτής της διαδικασίας, το όνομα του νέου ρεύματος προστίθεται στην εργαλειοθήκη ρευμάτων (streams toolbar). Από εκεί υπάρχουν οι δυνατότητες ανασκόπησης του ορισμού του ρεύματος (επιλέγοντας "View Definition") και διαγραφής του (επιλέγοντας "Delete"). Στην πρώτη περίπτωση επανεμφανίζεται το μήνυμα ορισμού νέου ρεύματος. Στη δεύτερη, η εφαρμογή δίνει στο TelegraphCQ την εντολή διαγραφής ρεύματος:

```
DROP STREAM <stream_name>;
```

5.3.2 Σχεδίαση νέου Προσχεδίου Ερωτήματος Διαρκείας

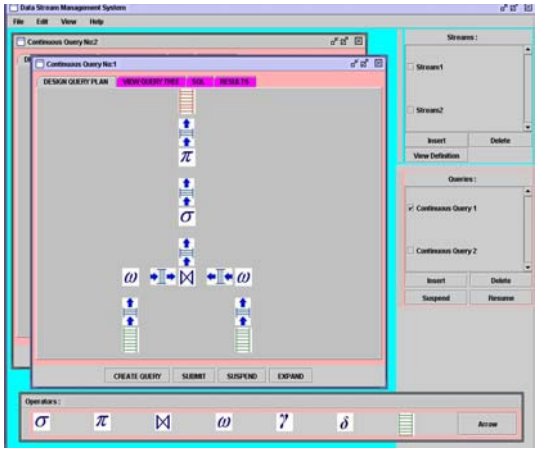
Το άνοιγμα νέας φόρμας γραφικής σχεδίασης προσχεδίων εκτέλεσης (Continuous Query Plans) γίνεται από την εργαλειοθήκη ερωτημάτων (Query toolbar) επιλέγοντας "Insert".

Στο *Σχήμα 5.9* φαίνεται ένα παράδειγμα φόρμας σχεδίασης και δημιουργίας νέου ερωτήματος διαρκείας.

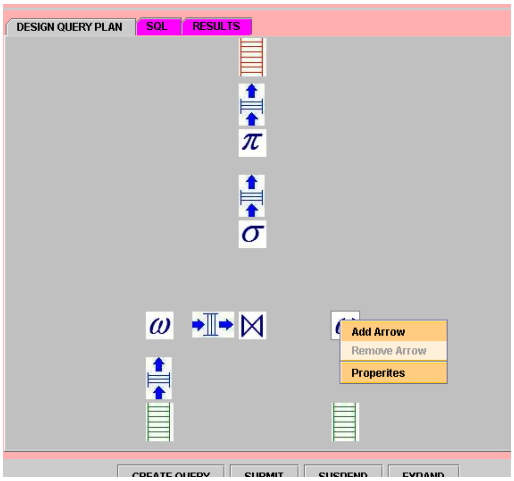
Για το προσχέδιο υποστηρίζονται όλοι των βασικοί τύποι τελεστών και παραθύρων:

- Προβολή (projection) [π],
- Επιλογή (selection) [σ],
- Σύνδεση (join) [⋈],
- Συνάθροιση (aggregation) [γ],
- Απαλοιφή διπλοτύπων (duplicate elimination) [δ].
- Παράθυρα [ω] (πλειάδων, χρονικά, οροσήμου και μεριστικά).

Ο χρήστης τοποθετεί κάθε επιθυμητό τελεστή από την εργαλειοθήκη με χρήση Drag-and-drop πάνω στο πάνελ σχεδίασης. Οι τελεστές συνδέονται μεταξύ τους με βέλη. Για την δημιουργία ενός βέλους μεταξύ δυο τελεστών ο χρήστης θα κάνει click στον πρώτο τελεστή (αρχή βέλους) και δεξί click στον δεύτερο τελεστή (πέρας βέλους). Στο Popup Menu που θα εμφανιστεί επιλέγει "Add Arrow" και η λογική ουρά που αντιπροσωπεύει το βέλος εμφανίζεται στην οθόνη όπως φαίνεται στο *Σχήμα 5.10*.



Σχήμα 5.9: *Παράδειγμα φόρμας σχεδίασης δένδρου προσχεδίου εκτέλεσης ερωτήματος διαρκείας στο γραφικό περιβάλλον.*



Σχήμα 5.10: *Παράδειγμα εισαγωγής βέλους στο προσχέδιο εκτέλεσης ερωτήματος.*

Σχηματίζεται έτσι το λογικό προσχέδιο εκτέλεσης (logical query plan). Οι κόμβοι του δένδρου είναι οι τελεστές, ενώ τα βέλη αντιπροσωπεύουν τη φορά ροής πληροφορίας μεταξύ των τελεστών (λογικές ουρές από την έξοδο ενός τελεστή στην είσοδο ενός άλλου).

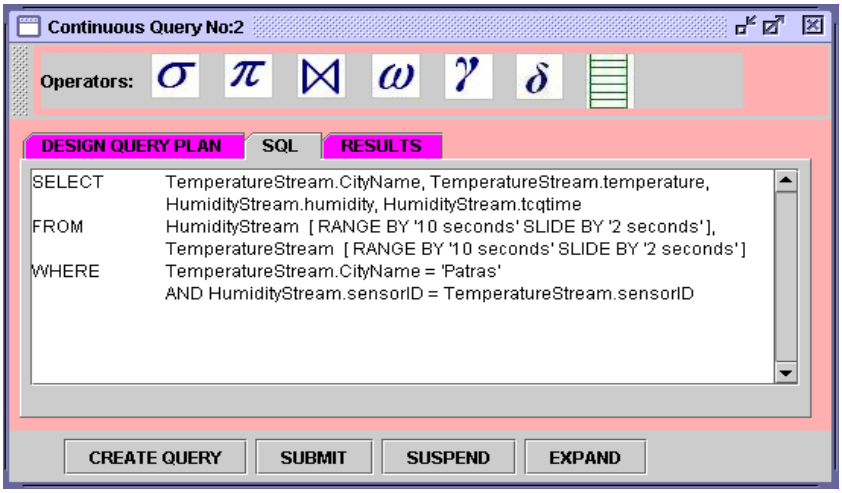
Οι ιδιότητες κάθε τελεστή θα ρυθμίζονται κατάλληλα μέσα από ειδικές φόρμες παραμετροποίησης (Properties forms). Οι φόρμες αυτές ανοίγουν με δεξί click στον κάθε τελεστή και επιλογή "Properties" από το Popup menu.

5.3.3 Παραγωγή του κώδικα SQL του ερωτήματος

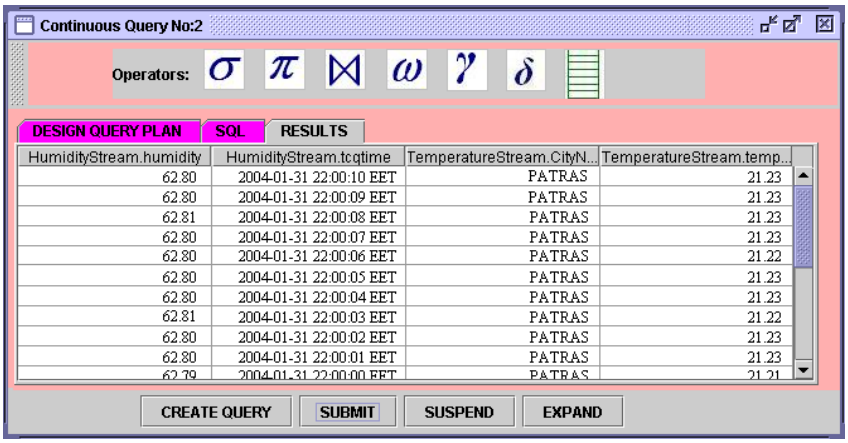
Αφού ολοκληρωθεί η σχεδίαση του προσχεδίου, επιλέγεται "CREATE QUERY" από το πάνελ ελέγχου της φόρμας του ερωτήματος.

Η εφαρμογή διατρέχει το δένδρο κόμβο-κόμβο (parsing) και δημιουργεί τον κώδικα του ερωτήματος διαρκείας σε γλώσσα μορφής SQL προσαρμοσμένη στο συντακτικό του TelegraphCQ για χειρισμό ρευμάτων δεδομένων. Η συντακτική ανάλυση (parsing) γίνεται από αντικείμενο της κλάσης SqlCodeGenerator().

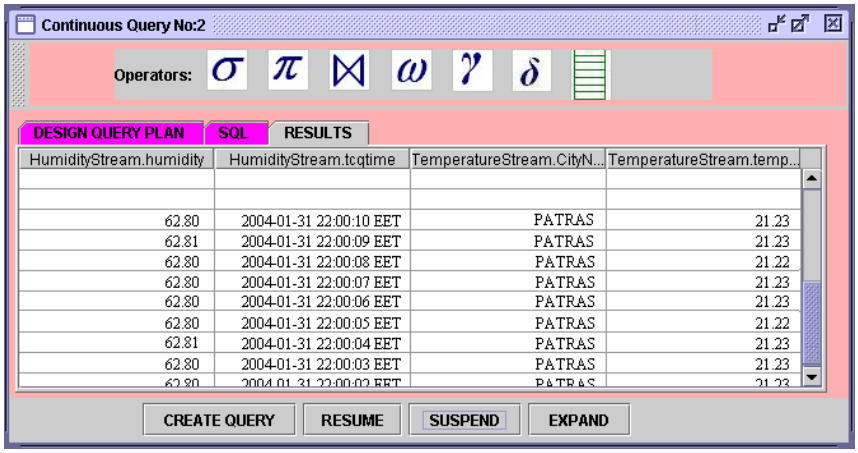
Ο παραγόμενος κώδικας εμφανίζεται στη φόρμα του ερωτήματος σε μια περιοχή κειμένου, όπως φαίνεται στο *Σχήμα 5.11*. Επίσης, δίνεται η δυνατότητα παρέμβασης και μεταβολής (editing) του κειμένου από το χρήστη.



Σχήμα 5.11: *Παραγόμενος SQL κώδικας από το παράδειγμα του ερωτήματος του σχήματος 5.9.*



Σχήμα 5.12 α: *Παραγόμενα αποτελέσματα του παραδείγματος του σχήματος 5.9.*



Σχήμα 5.12 β: *Αναστολή εκτέλεσης στο παράδειγμα του σχήματος 5.12(α).*

5.3.4 Υποβολή (Submit) και Εκτέλεση του ερωτήματος διαρκείας

Μετά τη δημιουργία του κώδικα σε SQL και τις όποιες τροποποιήσεις από τον χρήστη, επιλέγεται "SUBMIT" από το πάνελ ελέγχου της φόρμας του ερωτήματος.

Η εφαρμογή υποβάλλει το ερώτημα στο Σύστημα Διαχείρισης Ρευμάτων Δεδομένων (TelegraphCQ), μέσω αντικειμένου της κλάσης CursorThread(). Δημιουργείται λοιπόν ένας δρομέας (cursor) για το ερώτημα, το υποβάλλει και στη συνέχεια λειτουργεί σαν νήμα που ενεργοποιείται περιοδικά, λαμβάνει τα αποτελέσματα (FETCH) του ερωτήματος από το TelegraphCQ και τα επιστρέφει στην φόρμα του ερωτήματος για να προβληθούν στον πίνακα αποτελεσμάτων.

Τα αποτελέσματα εμφανίζονται δυναμικά και σταδιακά (incrementally) καθώς αυτά παράγονται στους πίνακες της αντίστοιχης φόρμας ερωτήματος, όπως φαίνεται στο *Σχήμα 5.12(α)*.

5.3.5 Αναστολή εκτέλεσης (Suspend) – Επανεκκίνηση (Resume)

Κατά τη διάρκεια εκτέλεσης του ερωτήματος και προβολής των αποτελεσμάτων, υπάρχει δυνατότητα αναστολής της εκτέλεσης, επιλέγοντας "SUSPEND" από το πάνελ ελέγχου της φόρμας του ερωτήματος.

Το πλήκτρο "SUBMIT" μετονομάζεται σε "RESUME". Στο *Σχήμα 5.12(β)* φαίνεται η αναστολή της εκτέλεσης ενός ερωτήματος διαρκείας. Επιλέγοντας "RESUME" από το πάνελ ελέγχου της φόρμας του ερωτήματος γίνεται επανεκκίνηση της διαδικασίας εκτέλεσης του ερωτήματος και συνέχιση της προβολής των αποτελεσμάτων.

5.4 Περιορισμοί συστήματος

Κατά την σχεδίαση του γραφικού περιβάλλον της εφαρμογής έγινε η προσπάθεια να είναι όσο το δυνατόν πιο φιλικό προς το χρήστη, ενώ παράλληλα να παρέχει αρκετές δυνατότητες δημιουργίας ποικίλων ερωτημάτων που να μπορούν να εκτελούνται παράλληλα.

Εντούτοις, η χρήση του TelegraphCQ ως βασικού μηχανισμού εκτέλεσης των ερωτημάτων ενέχει κάποιους εγγενείς περιορισμούς **[siteTCQ]**:

- Η μόνη παραθυρική δομή που υποστηρίζεται είναι τα *κυλιόμενα χρονικά παράθυρα (Sliding time-based windows).*

- Η σύνδεση μεταξύ ρευμάτων γίνεται μόνο με ισότητα μεταξύ πεδίων (όχι με συναρτήσεις).

- Δεν υποστηρίζεται σύνδεση ενός ρεύματος με τον εαυτό του (*self join*).

- Τα κριτήρια επιλογής (selection) πρέπει έχουν την απλή μορφή:
  <attribute> OP <constant>.

- Δεν μπορεί να χρησιμοποιηθεί διάζευξη (OR) ή άρνηση (NOT) στα κριτήρια επιλογής.

- Δεν υποστηρίζεται απαλοιφή διπλοτύπων.

Ωστόσο, σαν προσπάθεια πρόβλεψης μελλοντικών εκδόσεων του TelegraphCQ όπου ενδεχομένως να απαλειφθούν οι παραπάνω περιορισμοί καθώς και για λόγους πληρότητας της εφαρμογής, το γραφικό περιβάλλον έχει κάποιες από τις παραπάνω επιλογές. Έτσι, υποστηρίζονται όλοι οι γνωστοί τύποι παραθύρων με τις παραμέτρους του καθενός, η απαλοιφή διπλοτύπων, τα self joins, όπως και κριτήρια επιλογής της μορφής:
`<attribute1> OP <attribute2>`, όπου: OP είναι ένα από τα '=', '<', '>', '<=' και '>='.

5.5 Μηχανισμοί ελέγχου λαθών

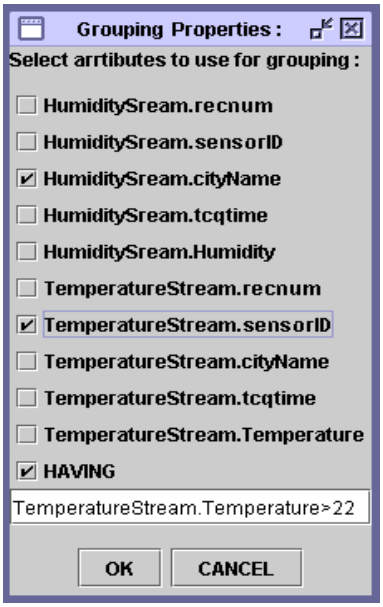
Στην παρούσα εργασία έγινε προσπάθεια δημιουργίας ενός εργαλείου γενικού σκοπού που θα είναι φιλικό προς το χρήστη και θα τον καθοδηγεί σε ορθή διατύπωση ερωτημάτων σε SQL κώδικα. Έτσι δόθηκε ιδιαίτερη έμφαση τόσο στην πρόβλεψη όσο και στην ανίχνευση των πιθανών λαθών.

5.5.1 Πρόβλεψη

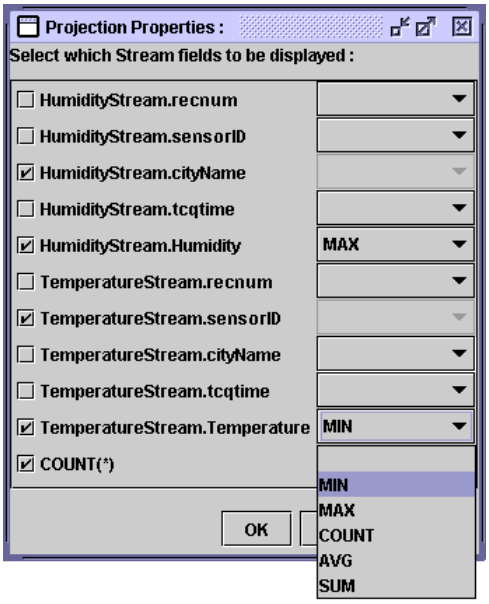
Κατά τη διάρκεια της σχεδίασης του προσχεδίου εκτέλεσης και της ρύθμισης των ιδιοτήτων κάθε τελεστή στις φόρμες παραμετροποίησης, εφαρμόστηκαν κάποιοι κανόνες διασφάλισης της ορθότητας στη σύνταξη ερωτημάτων από τον χρήστη.

♦ Τα γνωρίσματα φιλτραρίσματος ενός τελεστή προβολής (projection), τα γνωρίσματα ομαδοποίησης (grouping attributes) ενός τελεστή ομαδοποίησης (group by) επιλέγονται από ένα σύνολο *Check Boxes* με τα ονόματα των γνωρισμάτων που υπάρχουν στις πλειάδες του ρεύματος δεδομένων που εισέρχεται στην είσοδο του αντίστοιχου τελεστή.
Στα *Σχήματα 5.13* και *5.14* απεικονίζονται δυο παραδείγματα φορμών παραμετροποίησης για τελεστές ομαδοποίησης και προβολής αντίστοιχα.

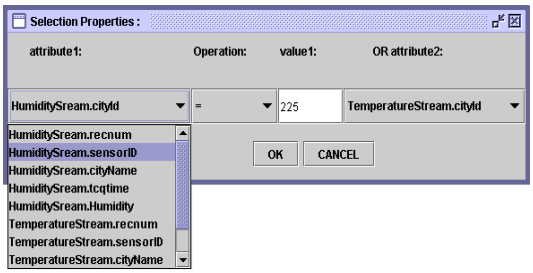
♦ Τα γνωρίσματα που συμμετέχουν σε κριτήριο επιλογής της μορφής: <attribute> OP <constant> ή <attribute1> OP <attribute2>, τα γνωρίσματα με τα οποία γίνεται η σύνδεση σε ένα τελεστή σύνδεσης, καθώς επίσης τα γνωρίσματα (partitioning attributes) ενός μεριστικού παραθύρου επιλέγονται από ειδικά *Combo Boxes* με τα ονόματα των γνωρισμάτων που υπάρχουν στις πλειάδες του ρεύματος εισόδου του αντίστοιχου τελεστή.



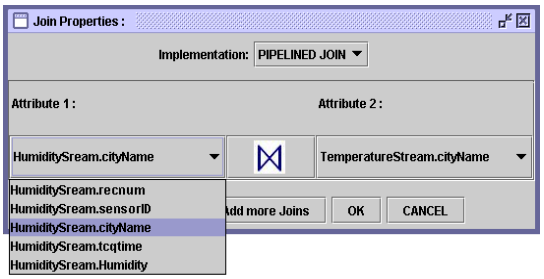
Σχήμα 5.13: *Φόρμα Παραμετροποίησης τελεστή Ομαδοποίησης (group by).*



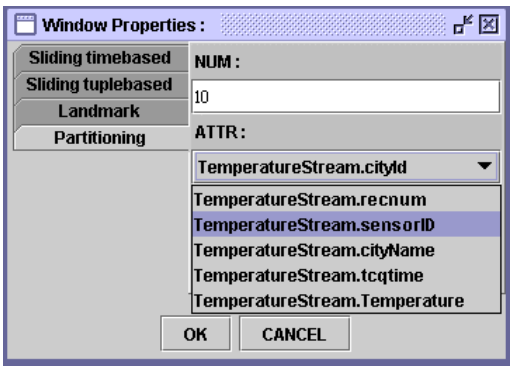
Σχήμα 5.14: *Φόρμα Παραμετροποίησης τελεστή Προβολής (projection).*



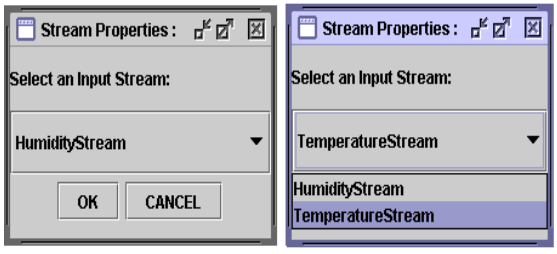
Σχήμα 5.15: *Φόρμα Παραμετροποίησης τελεστή Επιλογής (selection).*



Σχήμα 5.16: *Φόρμα Παραμετροποίησης τελεστή Σύνδεσης (join).*



Σχήμα 5.17: *Φόρμα Παραμετροποίησης τελεστή Μεριστικού Παραθύρου (partitioning window).*



Σχήμα 5.18: *Φόρμες παραμετροποίησης τελεστών ρευμάτων δεδομένων εισόδου.*

Στα *Σχήματα 5.15, 5.16* και *5.17* απεικονίζονται τρεις φόρμες παραμετροποίησης για τελεστές επιλογής, σύνδεσης και μεριστικού παραθύρου αντίστοιχα.

♦ Οι συναθροιστικές συναρτήσεις (aggregate functions) μπορούν να εφαρμοστούν στα προβαλλόμενα γνωρίσματα των πλειάδων του ρεύματος εξόδου είναι οι MIN, MAX, SUM, AVG, COUNT. Αυτές επιλέγονται από ειδικά *Combo Boxes* στη φόρμα παραμετροποίησης του τελεστή προβολής (projection), όπως φαίνεται στο *Σχήμα 5.14*.

♦ Όταν ο τελεστής προβολής (projection) προηγείται άλλων τελεστών (κατά τη φορά ροής πληροφορίας) στο προσχέδιο εκτέλεσης του ερωτήματος, τότε το φιλτράρισμα των γνωρισμάτων στις πλειάδες του ρεύματος εξόδου της προβολής επηρεάζει τα σύνολα των *Check Boxes* και *Combo Boxes* στις φόρμες παραμετροποίησης των τελεστών που έπονται.

♦ Όταν ο τελεστής ομαδοποίησης (group by) προηγείται του τελεστή προβολής (projection) κατά τη φορά ροής πληροφορίας στο προσχέδιο εκτέλεσης του ερωτήματος, τότε στα γνωρίσματα ομαδοποίησης του τελεστή ομαδοποίησης δεν επιτρέπεται να εφαρμοστούν συναθροιστικές συναρτήσεις (aggregate functions) στον τελεστή της προβολής. Έτσι, στη φόρμα παραμετροποίησης του τελεστή προβολής όσα γνωρίσματα συμμετέχουν στην ομαδοποίηση είναι απενεργοποιημένα.

Αν συγκρίνουμε τις φόρμες των Σχημάτων 5.13 και 5.14 παρατηρούμε ότι για τα δυο επιλεγμένα γνωρίσματα ομαδοποίησης (`HumidityStream.cityName` και `TemperatureStream.sensorID`) του τελεστή ομαδοποίησης (σχήμα 5.13) είναι απενεργοποιημένα τα Combo Boxes συναθροιστικών συναρτήσεων στη φόρμα τελεστή προβολής (σχήμα 5.14).

♦ Όταν ο τελεστής ομαδοποίησης (group by) προηγείται του τελεστή επιλογής (selection) κατά τη φορά ροής πληροφορίας στο προσχέδιο εκτέλεσης του ερωτήματος, τότε τα γνωρίσματα που θα χρησιμοποιηθούν στο κριτήριο του τελεστή της επιλογής πρέπει να είναι κάποια από τα γνωρίσματα ομαδοποίησης του τελεστή ομαδοποίησης. Αυτό συμβαίνει διότι υπό αυτήν την προϋπόθεση ισχύει η αντιμεταθετική ιδιότητα:

$\sigma_{\Theta}({}_{A}\gamma_{F}(\text{Stream})) = {}_{A}\gamma_{F}(\sigma_{\Theta}(\text{Stream})) \Leftrightarrow$ Το Θ χρησιμοποιεί γνωρίσματα μόνο από το Α.

Έτσι υπάρχει η δυνατότητα αντιμετάθεσης των τελεστών επιλογής και ομαδοποίησης, όπως φαίνεται στο *Σχήμα 5.19 (β)* που οδηγεί σε ένα απλό ερώτημα της μορφής:

```
SELECT   <select_list>
FROM     <Stream_ list>
WHERE    <predicate>
GROUP BY<group_by_expressions>
HAVING   <predicate>;     //προαιρετικά
```

Διαφορετικά (*σχήμα 5.19 (α)*)θα έπρεπε να προκύψει ένα ένθετο υποερώτημα (nested query) της μορφής:

```
WITH NewStream1 AS
           (
           SELECT     *
           FROM       <Stream_ list>
           WHERE      <join_predicate>
```

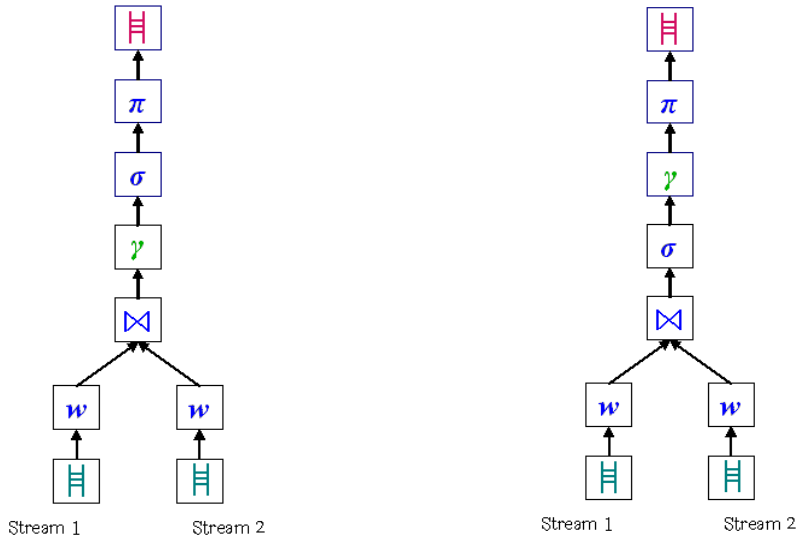
```
                GROUP BY    <group_by_expressions>
                HAVING      <predicate>
                )
SELECT          <select_list>
FROM            NewStream1
WHERE           <selection_predicate>;
```

Μια παρόμοια τακτική σε πολυπλοκότερα δέντρα προσχεδίων εκτέλεσης θα δυσχέραινε τη μορφή του κώδικα SQL.

♦ Κατά τη δημιουργία ενός νέου ρεύματος εισόδου, η επιλογή "Browse" της φόρμας ανοίγει έναν επιλογέα αρχείων (file chooser) με κατάλληλο φίλτρο ώστε να είναι ορατά μόνο ASCII αρχεία (επέκτασης .txt). Ο ρυθμός πλειάδων ανά δευτερόλεπτο του ρεύματος επιλέγεται γραφικά από slider ώστε να είναι θετικός ακέραιος αριθμός μέσα σε ένα πεπερασμένο φάσμα τιμών (0 έως 100). Επίσης, σε περίπτωση που ο χρήστης αμελήσει να θέσει ένα νέο όνομα στο αντίστοιχο πεδίο κειμένου, εμφανίζεται προειδοποιητικό μήνυμα στην οθόνη (alert message).

♦ Στη φόρμα παραμετροποίησης ρεύματος εισόδου (σε τελεστή input Stream operator πάνω στο πάνελ σχεδίασης) εμφανίζεται *Combo Box* το οποίο περιέχει τα ονόματα όλων των ρευμάτων που έχουν δημιουργηθεί από το χρήστη και έχουν προστεθεί στην εργαλειοθήκη ρευμάτων (streams toolbar).

Στο *Σχήμα 5.18* φαίνονται δύο φόρμες παραμετροποίησης ρευμάτων εισόδου με δυο ρεύματα δεδομένων να έχουν δημιουργηθεί και εισαχθεί στην εφαρμογή.



Σχήμα 5.19: (α) *Δένδρο εκτέλεσης με τελεστή ομαδοποίησης να προηγείται του τελεστή επιλογής κατά τη φορά ροής πληροφορίας.*
(β) *Δένδρο προσχεδίου εκτέλεσης με τελεστή επιλογής να προηγείται του τελεστή ομαδοποίησης κατά τη φορά ροής πληροφορίας.*

Τα (α) και (β) είναι ισοδύναμα όταν και μόνον όταν τα γνωρίσματα που συμμετέχουν στο κριτήριο του τελεστή επιλογής ανήκουν στο σύνολο των γνωρισμάτων ομαδοποίησης του τελεστή ομαδοποίησης.

5.5.2 Έλεγχος κατά τη διάσχιση του δένδρου εκτέλεσης

Με την ολοκλήρωση της γραφικής σχεδίασης του ερωτήματος από τον χρήστη και τον ορισμό των ιδιοτήτων των τελεστών που το αποτελούν, η εφαρμογή θα πρέπει να διατρέξει το δένδρο εκτέλεσης του ερωτήματος κόμβο-κόμβο (*parsing*) κρατώντας όσες πληροφορίες χρειάζονται για τη διατύπωση του ερωτήματος. Αν κατά τη διάσχιση αυτή η εφαρμογή εντοπίσει συντακτικά λάθη, προβάλλει στην οθόνη του γραφικού περιβάλλοντος κατάλληλα προειδοποιητικά μηνύματα και σταματά τη διαδικασία, δίνοντας στο χρήστη την ευκαιρία να επανασχεδιάσει το δένδρο με ορθό συντακτικά τρόπο πριν υποβάλλει ξανά το ερώτημα. Τα μηνύματα που απευθύνονται στο χρήστη δίνουν πληροφορία για το είδος του σφάλματος και τον τελεστή στον οποίο έχει συμβεί και είναι τριών βασικών κατηγοριών:

♦ Προειδοποιητικά μηνύματα σε περίπτωση ανίχνευσης εσφαλμένων συνδέσεων μεταξύ τελεστών, λ.χ. ένας τελεστής σύνδεσης που έχει μόνο μία ή περισσότερες από δύο ουρές δεδομένων εισόδου.

♦ Προειδοποιητικά μηνύματα σε περίπτωση που οι είσοδοι ενός τελεστή σύνδεσης δεν διασυνδέονται με παράθυρα.

♦ Προειδοποιητικά μηνύματα σε περίπτωση που τα ρεύματα εισόδου συνδέονται απευθείας σε κάποιο τελεστή, χωρίς να έχουν οδηγηθεί πρώτα σε κάποιο παράθυρο.

5.5.2 Έλεγχος κατά τη δημιουργία του κώδικα SQL

Αφού διασχίσει το δένδρο εκτέλεσης ενός ερωτήματος, η εφαρμογή έχει διαβάσει όλες τις απαραίτητες πληροφορίες για τη διατύπωσή του με δηλωτικό τρόπο σε κώδικα μορφής *SQL* προσαρμοσμένο στο συντακτικό του ΣΔΡΔ *TelegraphCQ*. Η εφαρμογή καλείται να προνοήσει για τυχόν παραλείψεις του χρήστη που θα οδηγούσαν σε λάθη κατά την εκτέλεση (runtime errors) και να παράξει τον κώδικα με κατάλληλες τροποποιήσεις:

♦ Αν σε τελεστή προβολής (projection) έχουν εφαρμοστεί συναθροιστικές συναρτήσεις σε κάποια από τα γνωρίσματα που έχουν επιλεγεί, ενώ σε άλλα δεν έχουν εφαρμοστεί συναρτήσεις, τότε τα τελευταία πρέπει υποχρεωτικά να γίνουν γνωρίσματα ομαδοποίησης (grouping attributes). Αν ο χρήστης αμελήσει να προσθέσει τελεστή ομαδοποίησης, η κλάση δημιουργίας του κώδικα σε SQL προσθέτει αυτόματα μια πρόταση ομαδοποίησης (grouping clause) στον κώδικα του ερωτήματος:

```
GROUP BY   <grouping_attributes>
```

Παράδειγμα 5.1:

```
SELECT     Str1.attr1, Str1.attr2, MAX(Str1.attr3) AS max1,
           SUM(Str1.attr4) AS sum2
FROM  Str1 [SLIDE BY '20 seconds'
            RANGE BY '2 seconds'
            START AT '2007-09-10 18:50:20']
WHERE      Str1.attr2 >= 22
GROUP BY   Str1.attr1, Str1.attr2;
```

Όπως φαίνεται στο παράδειγμα 5.1, στα γνωρίσματα attr1 και attr2 του ρεύματος Str2 δεν έχει εφαρμοστεί καμία συναθροιστική συνάρτηση. Αντίθετα, στο γνώρισμα attr3 έχει εφαρμοστεί η συνάρτηση μέγιστης τιμής (MAX) και στο γνώρισμα attr4 η συνάρτηση αθροίσματος (SUM). Έτσι, τα attr1 και attr2 θα πρέπει να είναι γνωρίσματα ομαδοποίησης. Αν ο χρήστης της εφαρμογής δεν προβλέψει να εισάγει τελεστή ομαδοποίησης στον οποίο να επιλέξει τα attr1 και attr2, τότε η κλάση SQLCodeGenerator() θα προσθέσει αυτόματα την τελευταία γραμμή της εντολής του παραδείγματος 5.1.

5.6 Παραδείγματα εκτέλεσης ερωτημάτων σε μετρήσεις αισθητήρων

Έστω ότι έχουν τοποθετηθεί αισθητήρες μέτρησης σε διάφορες πόλεις της Ελλάδας. Οι μετρήσεις που λαμβάνουμε έχουν τη μορφή της σχεσιακής πλειάδας με ένα σύνολο πεδίων που δίνουν πληροφορίες για τον αισθητήρα που έκανε την κάθε μέτρηση, την τιμή της μέτρησης, την γεωγραφική περιοχή και τον χρόνο καταγραφής. Υπάρχουν δύο κατηγορίες αισθητήρων: Οι αισθητήρες της πρώτης κατηγορίας μετρούν τιμές υγρασίας και τροφοδοτούν με δεδομένα τις πλειάδες του ρεύματος HumidityStream, ενώ αυτοί της δεύτερης μετρούν τιμές θερμοκρασίας και τροφοδοτούν με δεδομένα τις πλειάδες του ρεύματος TemperatureStream.

- Το *HumidityStream* έχει πλειάδες της μορφής :

recnum integer; sensorID integer; CityName varchar(30); t integer; humidity float; tcqtime timestamp

- Το *TemperatureStream* έχει πλειάδες της μορφής :

recnum integer; sensorID integer; CityName varchar(30); t integer; temperature float; tcqtime timestamp

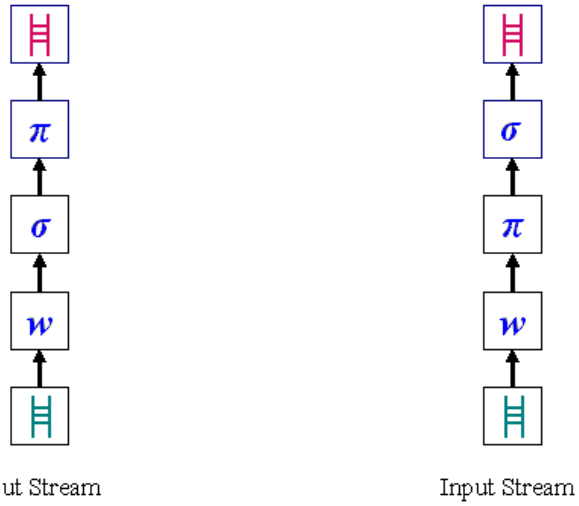
Αναλυτικότερα η πληροφορία κάθε πλειάδας περιλαμβάνει:

- Το πεδίο recnum περιέχει τον αριθμό της κάθε πλειάδας (αύξων αριθμός) του ρεύματος.
- Το πεδίο sensorID περιέχει τον κωδικό αριθμό (αναγνωριστικό) του αισθητήρα που πραγματοποίησε τη μέτρηση.
- Το πεδίο CityName περιέχει το όνομα της πόλης στην οποία βρίσκεται ο αισθητήρας.
- Το πεδίο t περιέχει μία ακέραια τιμή που δηλώνει τον απόλυτο χρόνο της μέτρησης.
- Τα πεδία humidity και temperature περιέχουν τις μετρούμενες τιμές υγρασίας και θερμοκρασίας αντίστοιχα.
- Το πεδίο tcqtime περιέχει την ώρα που καταγράφηκε η μέτρηση (χρονόσημο). Έχει τη μορφή : 'χχχχ-μμ-ηη ωω:λλ:δδ' (όπου χ: έτος, μ: μήνας, η: ημέρα, ω: ώρα, λ: λεπτά, δ: δευτερόλεπτα)

Στα επόμενα παραδείγματα γίνεται μια προσπάθεια επεξήγησης των βασικότερων τύπων προσχεδίων εκτέλεσης ερωτημάτων διαρκείας και της μορφής που θα έχει ο παραγόμενος από την εφαρμογή κώδικας σε SQL σε κάθε περίπτωση.

5.6.1 Απλά ερωτήματα μόνο με επιλογή και προβολή

Ισχύει η αντιμεταθετική ιδιότητα : π (σ (Stream)) = σ (π (Stream)).
Οι παραθυρικές δομές θα μπορούσαν να παραλειφθούν χωρίς να υπάρξει πρόβλημα στην εκτέλεση του ερωτήματος, εφόσον η επιλογή και η προβολή είναι μη ανασχετικοί τελεστές.



Σχήμα 5.20: *Δύο ισοδύναμες μορφές απλού προσχεδίου εκτέλεσης ερωτήματος διαρκείας με τελεστές επιλογής και προβολής.*

Η γενική μορφή του παραγόμενου κώδικα σε SQL των ερωτημάτων διαρκείας αυτής της μορφής είναι:

```
SELECT      <select_list>
FROM        <Stream_name> [Window]
WHERE       <Selection_predicate>
GROUP BY    <group_by_expressions>;     //προαιρετικά
```

Παράδειγμα 5.2:
" Δείξε την τρέχουσα τιμή Υγρασίας στην περιοχή της Πάτρας. "

```
SELECT      HumidityStream.sensorID, HumidityStream.humidity,
            HumidityStream.tcqtime
FROM        HumidityStream [RANGE BY '1 second'
                           SLIDE BY '1 second'
                           START AT '2007-10-11 00:00:00 EET']
WHERE       HumidityStream.CityName = 'Patras';
```

Επιλέγουμε να προβάλλουμε τον κωδικό αριθμό του αισθητήρα, την τιμή μέτρησης Υγρασίας για το πιο πρόσφατο χρονόσημο, ξεκινώντας από τις 00:00 π.μ. της 11/09/2007 και ανανεώνοντας την απόκριση κάθε δευτερόλεπτο, δημιουργώντας έτσι ένα επάλληλο χρονικό παράθυρο εφαρμοζόμενο πάνω στο ρεύμα εισόδου. Παίρνουμε έτσι μόνο την τρέχουσα τιμή υγρασίας για κάθε δευτερόλεπτο, ενώ δεν υπάρχει απώλεια πληροφορίας.

Παράδειγμα 5.3:
" Δείξε τις περιοχές στις οποίες επικρατούν συνθήκες ξηρασίας για τα τελευταία 2 λεπτά."

```
SELECT      HumidityStream.CityName, MAX(HumidityStream.humidity)
            AS MAX1, HumidityStream.t, COUNT(*) AS cnt
FROM        HumidityStream [RANGE BY '120 seconds'
                           SLIDE BY '120 seconds'
```

```
                              START AT '2007-09-11 00:00:00 EET']
WHERE       HumidityStream.humidity < 40.00
GROUP BY    HumidityStream.CityName, HumidityStream.t;
```

Σε αυτό το παράδειγμα προβάλλουμε το όνομα της πόλης, την μέγιστη τιμή μετρούμενης υγρασίας, τον χρόνο μέτρησης και τον αριθμό των πλειάδων που έλαβαν μέρος στο στιγμιότυπο του παραθύρου, για τα τελευταία 120 δευτερόλεπτα, ανανεώνοντας την απόκριση κάθε 120 δευτερόλεπτα (επάλληλο χρονικό παράθυρο). Οι συνθήκες ξηρασίας σημαίνουν υγρασία κάτω από 40.

Οι πλειάδες του αποτελέσματος θα πρέπει να εμφανίζονται ομαδοποιημένες κατά το όνομα της πόλης και τον χρόνο μέτρησης, λόγω της ύπαρξης συναθροιστικών συναρτήσεων στα υπόλοιπα προβαλλόμενα πεδία. Έτσι, εφόσον ο χρήστης δεν έχει προνοήσει να χρησιμοποιήσει τελεστή ομαδοποίησης, η εφαρμογή θα προσθέσει αυτόματα την εντολή GROUP BY της τελευταίας σειράς.

Παράδειγμα 5.4:
" Δείξε την μετρούμενη θερμοκρασία, για οποιαδήποτε πόλη, σε συνθήκες παγετού. "

```
SELECT      TemperatureStream.CityName,
            TemperatureStream.tcqtime,
            TemperatureStream.temperature
FROM        TemperatureStream
WHERE       TemperatureStream.temperature <= 0.00;
```

Επιλέγουμε να προβάλλουμε το όνομα της πόλης, την τιμή μέτρησης θερμοκρασίας και την ακριβή ώρα μέτρησης (χρονόσημο). Η συνθήκη επιλογής για τις τιμές θερμοκρασίας να είναι μικρότερες ή ίσες από 0°C, εξασφαλίζει τις συνθήκες παγετού που αναφέρει ο χρήστης στο ερώτημα.

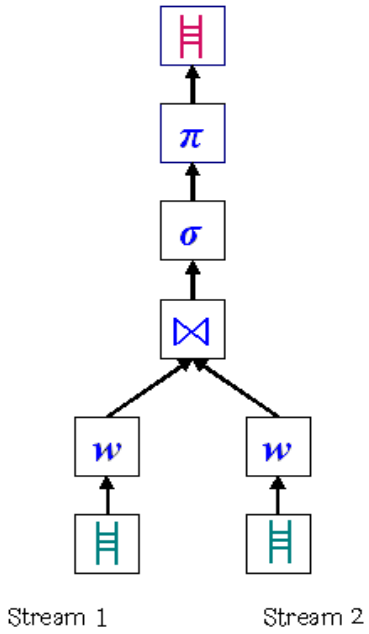
Στο παράδειγμα αυτό δεν εφαρμόζεται παράθυρο διότι οι τελεστές επιλογής και προβολής έχουν τη δυνατότητα να επεξεργάζονται κάθε πλειάδα του ρεύματος αυτοτελώς χωρίς να χρειάζονται φιλτράρισμα από παράθυρο στην είσοδό τους.

5.6.2 Απλά ερωτήματα με σύνδεση

Τα παράθυρα στις εισόδους του τελεστή σύνδεσης είναι απαραίτητα.

Η γενική μορφή του παραγόμενου κώδικα σε SQL των ερωτημάτων διαρκείας αυτής της μορφής είναι:

```
SELECT      <Select_list>
FROM        <Stream1_name> [window1],
            <Stream2_name> [window2]
WHERE       <Selection_predicate>
            AND <Join_predicate>
GROUP BY    <Group_by_expressions>;     //προαιρετικά
```



Σχήμα 5.21: *Μορφή απλού προσχεδίου εκτέλεσης ερωτήματος διαρκείας με σύνδεση (διακλάδωση στο δένδρο).*

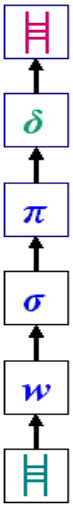
Παράδειγμα 5.5:
" Δείξε τις μετρούμενες τιμές υγρασίας και θερμοκρασίας για τα τελευταία 2 λεπτά, για κάθε πόλη. "

```
SELECT      HumidityStream.CityName, HumidityStream.humidity,
            TemperatureStream.temperature, TemperatureStream.t
FROM        HumidityStream [RANGE BY '120 seconds'
                           SLIDE BY '20 seconds'
                           START AT '2007-10-11 00:00:00 EET']
            TemperatureStream [RANGE BY '120 seconds'
                             SLIDE BY '20 seconds'
                             START AT '2007-10-11 00:00:00 EET']
WHERE       HumidityStream.CityName =TemperatureStream.CityName;
```

Από το σύνολο των πλειάδων των δύο ρευμάτων για τις οποίες η υγρασία και η θερμοκρασία μετρήθηκαν στην ίδια πόλη, επιλέγουμε να προβάλλουμε το όνομα της πόλης, τις μετρούμενες τιμές υγρασίας και θερμοκρασίας και τον χρόνο μέτρησης για τα τελευταία 120 δευτερόλεπτα, ανανεώνοντας την απόκριση κάθε 20 δευτερόλεπτα, δημιουργώντας ένα κυλιόμενο χρονικό παράθυρο. Η συχνή ανανέωση των αποτελεσμάτων σε σχέση με το εύρος του παραθύρου γίνεται για λόγους ακρίβειας.

5.6.3 Απλά ερωτήματα με απαλοιφή διπλοτύπων

Τα παράθυρα είναι απαραίτητα και σε αυτήν την περίπτωση. Η γενική μορφή του παραγόμενου κώδικα σε SQL των ερωτημάτων διαρκείας αυτής της μορφής είναι:



Σχήμα 5.22: *Μορφή απλού προσχεδίου εκτέλεσης ερωτήματος διαρκείας με τελεστή απαλοιφής διπλοτύπων.*

```
SELECT DISTINCT  <select_list>
FROM             <Stream_name> [window]
WHERE            <Selection_predicate>
GROUP BY         <group_by_expressions>;    //προαιρετικά
```

Παράδειγμα 5.6:
" Δείξε όλες τις διακριτές τιμές υγρασίας που έχουν μετρηθεί το τελευταίο λεπτό στην περιοχή της Αθήνας. "

```
SELECT DISTINCT  HumidityStream.sensorID, HumidityStream.humidity
FROM       HumidityStream [RANGE BY '60 seconds'
                           SLIDE BY '10 seconds'
                           START AT '2007-11-11 09:00:00 EET']
WHERE      HumidityStream.CityName = 'Athens';
```

Χωρίς να εμφανίζονται στο αποτέλεσμα δύο ή περισσότερες φορές οι πανομοιότυπες πλειάδες, επιλέγουμε να προβάλλουμε τον κωδικό αριθμό του αισθητήρα και την μετρούμενη τιμή Υγρασίας για τα τελευταία 60 δευτερόλεπτα, ανανεώνοντας την απόκριση κάθε 10 δευτερόλεπτα (κυλιόμενο χρονικό παράθυρο).

Το παράδειγμα αυτό βοηθά στην μελέτη της διακύμανσης της υγρασίας, χωρίς το φόρτο πλεονάζουσας επαναλαμβανόμενης πληροφορίας.

Παράδειγμα 5.7:
" Δείξε όλες τις διακριτές μέγιστες τιμές Θερμοκρασίας που μετρήθηκαν τα τελευταία 30 δευτερόλεπτα, σε συνθήκες καύσωνα. "

```
SELECT DISTINCT  TemperatureStream.sensorID,
           TemperatureStream.temperature
FROM       TemperatureStream [RANGE BY '30 seconds'
                              SLIDE BY '30 seconds'
```

```
                                    START AT '2007-12-1 12:00:00 EET']
WHERE       TemperatureStream.temperature >= 40.00;
```

Επιλέγουμε να προβάλλουμε τον κωδικό αριθμό του αισθητήρα και την μετρούμενη τιμή Θερμοκρασίας για τα τελευταία 30 δευτερόλεπτα, ανανεώνοντας την απόκριση κάθε 30 δευτερόλεπτα (επάλληλο χρονικό παράθυρο), χωρίς να εμφανίζονται στο αποτέλεσμα δύο ή περισσότερες φορές οι πανομοιότυπες πλειάδες.

Όπως και στο προηγούμενο παράδειγμα, διευκολύνεται η μελέτη των διαφορετικών τιμών της θερμοκρασίας η διακύμανσή της, οι αποκλίσεις μεταξύ των τιμών της, χωρίς απαραίτητα να μας ενδιαφέρει η διάρκεια την οποία είχαν οι τιμές αυτές. Η συνθήκη ότι η θερμοκρασία θα έχει τιμές πάνω από 40°C, εξασφαλίζει τις συνθήκες καύσωνα που ζητήθηκαν από τον χρήστη.

5.6.4 Απλά ερωτήματα με ομαδοποίηση

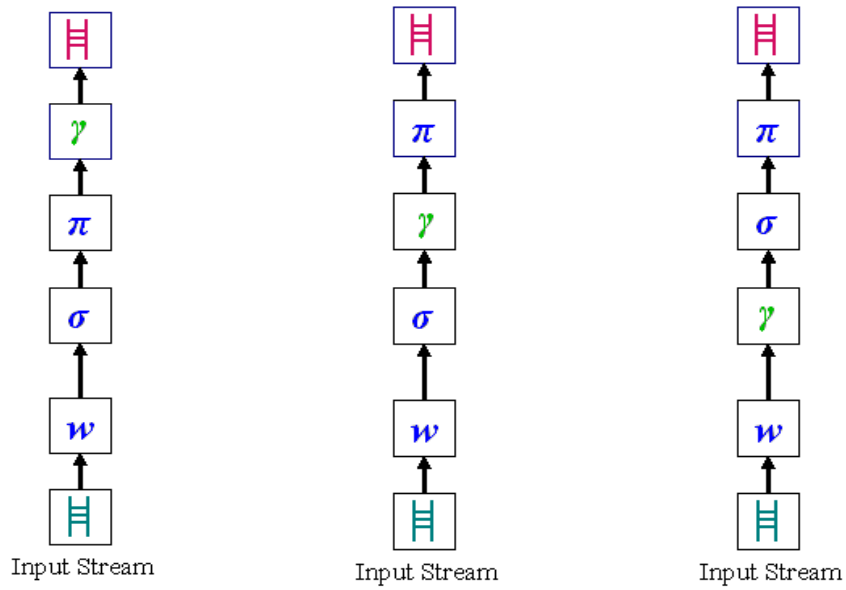
Όπως έχει αναφερθεί σε προηγούμενη ενότητα τα τρία δένδρα εκτέλεσης του *Σχήματος 5.19* είναι ισοδύναμα όταν :

$\sigma_{\Theta}({}_{A}\gamma_{F}(\text{Stream})) = {}_{A}\gamma_{F}(\sigma_{\Theta}(\text{Stream})) \Leftrightarrow$ Το Θ χρησιμοποιεί γνωρίσματα μόνο απ' το A.

δηλαδή όταν ισχύει η αντιμεταθετική ιδιότητα των τελεστών ομαδοποίησης και επιλογής.

Η γενική μορφή του παραγόμενου κώδικα σε SQL των ερωτημάτων διαρκείας αυτής της μορφής είναι :

```
SELECT      <select_list>
FROM        <Stream_name> [window]
WHERE       <Selection_predicate>
GROUP BY    <group_by_expressions>
HAVING      <predicate>;      //προαιρετικά
```



Σχήμα 5.23: *Ισοδύναμες μορφές προσχεδίων εκτέλεσης ερωτήματος διαρκείας με τελεστή ομαδοποίησης.*

Παράδειγμα 5.8:
" Δείξε τη μέγιστη τιμή υγρασίας που μετρήθηκε τα τελευταία 20 λεπτά από κάθε αισθητήρα, για τις περιοχές με υψηλές τιμές υγρασίας (πάνω από 55,00) "

```
SELECT      HumidityStream.sensorID, MAX(HumidityStream.humidity)
            AS max1
FROM        HumidityStream [RANGE BY '20 minutes'
                            SLIDE BY '5 minutes']
WHERE       HumidityStream.humidity >= 55.00
GROUP BY    HumidityStream.sensorID;
```

Τα πεδία που ενδιαφέρουν σε αυτό το ερώτημα είναι ο κωδικός αριθμός του αισθητήρα και η μετρούμενη τιμή υγρασίας. Κάθε στιγμιότυπο αποτελέσματος αφορά τα τελευταία 20 λεπτά, ανανεώνοντας την απόκριση κάθε 5 λεπτά (κυλιόμενο χρονικό παράθυρο).

Οι πλειάδες του αποτελέσματος θα πρέπει να ομαδοποιηθούν με βάση τον αισθητήρα που πραγματοποίησε τη μέτρηση, ώστε η συναθροιστική συνάρτηση MAX να εφαρμοστεί για κάθε αισθητήρα χωριστά.

Παράδειγμα 5.9:
" Δείξε τη μέση τιμή θερμοκρασίας που μετρήθηκε την τελευταία μισή ώρα για κάθε πόλη με καλές καιρικές συνθήκες θερμοκρασίας. "

```
SELECT      TemperatureStream.CityName,
            MAX(TemperatureStream.tcqtime) AS max1,
            AVG(TemperatureStream.temperature) AS avg2, COUNT(*)
            AS cnt
FROM        TemperatureStream [RANGE BY '30 minutes'
                               SLIDE BY '10 minutes'
                              START AT '2007-08-22 1:00:00 EET']
GROUP BY    TemperatureStream.CityName
HAVING      AVG(TemperatureStream.temperature) <= 25.00
            AND AVG(TemperatureStream.temperature) >= 20.00;
```

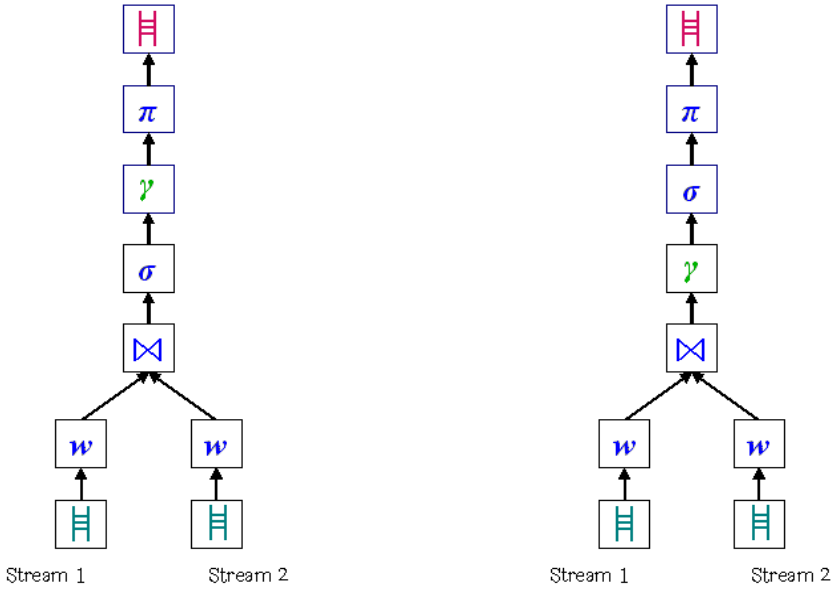
Επιλέγουμε να προβάλλουμε το όνομα της πόλης, τη μέση τιμή της μετρούμενης Υγρασίας, τον μέγιστο χρόνο μέτρησης (πιο πρόσφατο χρονόσημο) και τον αριθμό των πλειάδων που έλαβαν μέρος στον υπολογισμό, για τα τελευταία 30 λεπτά, ανανεώνοντας την απόκριση κάθε 10 λεπτά (κυλιόμενο χρονικό παράθυρο). Οι καλές καιρικές συνθήκες μετερεολογικά αποδίδονται σε μέσες τιμές θερμοκρασιών οι οποίες κυμαίνονται μεταξύ 20 και 25 βαθμών Κελσίου.

Όπως προηγουμένως, θα πρέπει να προηγηθεί ομαδοποίηση των πλειάδων με βάση το όνομα της πόλης όπου πραγματοποιήθηκε κάθε μέτρηση, ώστε ο μέσος όρος να υπολογιστεί για κάθε ομάδα ξεχωριστά.

5.6.5 Ερωτήματα με ομαδοποίηση και σύνδεση

Διακρίνονται σε δύο περιπτώσεις ανάλογα με τη θέση του τελεστή ομαδοποίησης στο προσχέδιο εκτέλεσης του ερωτήματος.

5.6.5.1 Τελεστής ομαδοποίησης στον κύριο κλάδο του δένδρου



Σχήμα 5.24: *Ισοδύναμες μορφές προσχεδίων εκτέλεσης ερωτήματος διαρκείας με τελεστές ομαδοποίησης και σύνδεσης (διακλάδωση στο δένδρο).*

Όπως και στην προηγούμενη περίπτωση τα δυο δέντρα εκτέλεσης των σχημάτων (α) και (β) είναι ισοδύναμα όταν ισχύει η αντιμεταθετική ιδιότητα των τελεστών ομαδοποίησης και επιλογής. Η γενική μορφή του παραγόμενου κώδικα σε SQL των ερωτημάτων διαρκείας αυτής της μορφής είναι:

```
SELECT      <select_list>
FROM        <Stream1_name> [window1],
            <Stream2_name> [window2]
WHERE       <selection_predicate> AND <join_predicate>
GROUP BY    <group_by_expressions>
HAVING      <predicate>;      //προαιρετικά
```

Παράδειγμα 5.10:

" Δείξε όλες τις περιοχές στις οποίες επικρατούν κανονικές συνθήκες (ΚΣ) θερμοκρασίας και υγρασίας κατά μέσο όρο, για τα τελευταία 20 λεπτά. "

```
SELECT      HumidityStream.CityName,
            AVG(TemperatureStream.temperature) AS avg1,
            AVG(HumidityStream.humidity) AS avg2
FROM        HumidityStream [RANGE BY '20 minutes'
                           SLIDE BY '2 minutes'
                           START AT '2007-10-28 00:00:00 EET'],
            TemperatureStream [RANGE BY '20 minutes'
                             SLIDE BY '2 minutes'
                             START AT '2007-10-28 00:00:00 EET']
WHERE       HumidityStream.t = TemperatureStream.t
GROUP BY    HumidityStream.CityName
HAVING      AVG(TemperatureStream.temperature) BETWEEN 22.00 AND 25.00
AND         AVG(HumidityStream.humidity) BETWEEN 40.00 AND 45.00;
```

Εδώ ενδιαφέρουν οι μέσες τιμές θερμοκρασίας και υγρασίας για τα τελευταία 20 λεπτά. Ανανεώνοντας την απόκριση κάθε 2 λεπτά (κυλιόμενο χρονικό παράθυρο) επιτυγχάνεται μεγαλύτερη ακρίβεια των αποτελεσμάτων.

Τα αποτελέσματα αφορούν στοιχεία των ρευμάτων που μετρήθηκαν κατά τον ίδιο χρόνο και με μέση τιμή θερμοκρασίας να είναι μεταξύ 22°C και 25°C ενώ η μέση τιμή υγρασίας κυμαίνεται μεταξύ 40,00 και 45,00. Οι πλειάδες του αποτελέσματος επιθυμούμε να προβάλλονται ομαδοποιημένες με βάση το όνομα της πόλης όπου έγινε η μέτρηση.

Παράδειγμα 5.11:

" Δείξε τις μέγιστες τιμές υγρασίας και θερμοκρασίας που μετρήθηκαν στον ίδιο χρόνο, σε κάθε πόλη, και τον Μετεωρολογικό Δείκτη τύπου 1 για τα τελευταία 2 λεπτά. "

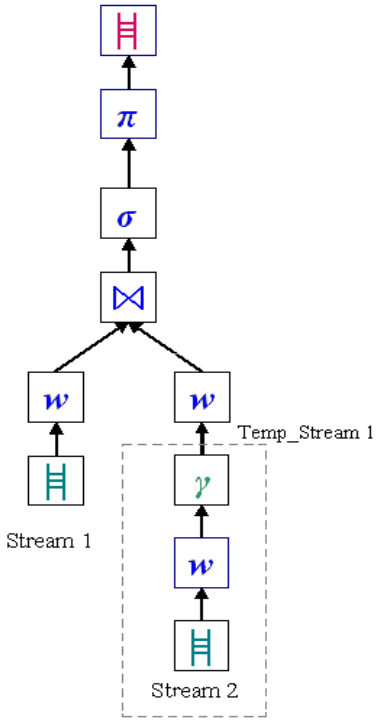
```
SELECT      TemperatureStream.CityName,
            MAX(HumidityStream.humidity) AS max1,
            MAX(TemperatureStream.temperature) AS max2,
            (max1/max2) AS index1, MAX(HumidityStream.tcqtime) AS
            max3
FROM        HumidityStream [RANGE BY '120 seconds'
                           SLIDE BY '60 seconds'],
            TemperatureStream [RANGE BY '120 seconds'
                              SLIDE BY '60 seconds']
WHERE       HumidityStream.tcqtime = TemperatureStream.tcqtime
AND         HumidityStream.CityName = TemperatureStream.CityName
GROUP BY    TemperatureStream.CityName;
```

Ο Μετεωρολογικός Δείκτης που ζητείται ορίζεται ως ο λόγος των μέγιστων τιμών θερμοκρασίας και υγρασίας. Για τον υπολογισμό αυτών των μέγιστων τιμών, χρησιμοποιούμε κυλιόμενο παράθυρο εύρους 120 δευτερολέπτων, ανανεώνοντας την απόκριση κάθε 60 δευτερόλεπτα, ελέγχοντας ότι θερμοκρασία και υγρασία μετρήθηκαν στον ίδιο ακριβή χρόνο και στην ίδια πόλη (κατηγόρημα του τελεστή σύνδεσης). Ως χρονόσημο του αποτελέσματος χρησιμοποιούμε το μέγιστο χρονόσημο, δηλαδή τον χρόνο που μετρήθηκε η πιο πρόσφατη ένδειξη που συμμετέχει στον τρέχοντα υπολογισμό. Οι πλειάδες του παραθύρου θα πρέπει να ομαδοποιηθούν με βάση το όνομα της πόλης όπου μετρήθηκαν και στη συνέχεια να υπολογιστούν οι ζητούμενες μέγιστες τιμές.

5.6.5.2 Τελεστής ομαδοποίησης σε υποκλάδο του δένδρου (ένθετο υποερώτημα)

Αποτελεί ειδική περίπτωση της κατηγορίας των ένθετων υποερωτημάτων (nested subqueries). Η γενική μορφή του παραγόμενου κώδικα σε γλώσσα SQL των ερωτημάτων διαρκείας αυτής της μορφής είναι:

```
WITH <NEW_Stream_name> AS (
SELECT      *
FROM        <Stream2_name> [window2],
GROUP BY    <group_by_expressions>
HAVING      <predicate>     //προαιρετικά
)
SELECT      <select_list>
FROM        <Stream1_name> [window1],
            <NEW_Stream_name> [window3]
WHERE       <selection_predicate> AND <join_predicate>;
```



Σχήμα 5.25: *Μορφή προσχεδίου εκτέλεσης ερωτήματος διαρκείας με τελεστή ομαδοποίησης σε κλάδο εισόδου του τελεστή σύνδεσης (ένθετο υποερώτημα).*

Παράδειγμα 5.12:
" Δείξε την υγρασία και την μέγιστη απόκλιση θερμοκρασιών που μετρήθηκαν σε κάθε πόλη, για την τελευταία 1 ώρα. "

```
WITH Temp_Stream1 AS
(
SELECT      TemperatureStream.CityName,
            MAX(TemperatureStream.temperature) AS max1,
            MIN(TemperatureStream.temperature) AS min1,
FROM        TemperatureStream [RANGE BY '30 minutes'
                               SLIDE BY '10 minutes']
GROUP BY    TemperatureStream.CityName
)
SELECT      Temp_Stream1.CityName, HumidityStream.humidity,
            (Temp_Stream1.max1 - Temp_Stream1.min1) AS div1
FROM        HumidityStream [RANGE BY '60 minutes'
                            SLIDE BY '20 minutes'],
            Temp_Stream1 [RANGE BY '60 minutes'
                          SLIDE BY '20 minutes']
WHERE       HumidityStream.CityName = Temp_Stream1.CityName;
```

Για τον υπολογισμό της απόκλισης των θερμοκρασιών χρησιμοποιούμε τη διαφορά της μέγιστης και της ελάχιστης τιμής θερμοκρασίας που παρατηρούνται σε κάθε πόλη σε ένα παράθυρο μισής ώρας, ανανεώνοντας την απόκριση κάθε 10 λεπτά.

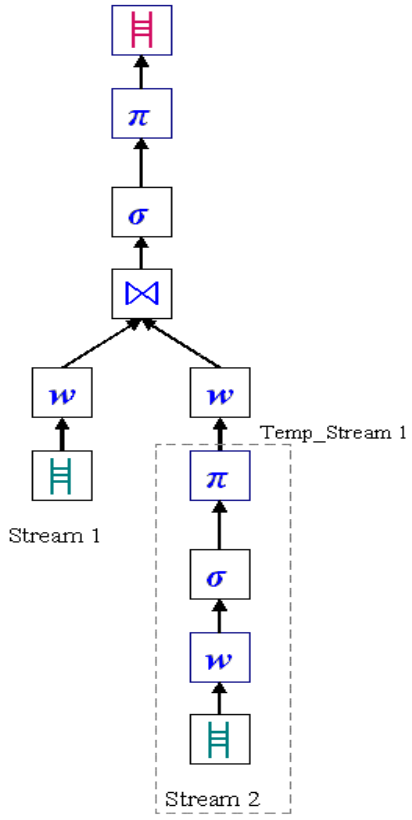
Τα τελικά αποτελέσματα υπολογίζονται για τις πλειάδες που μετρήθηκαν την τελευταία 1 ώρα, ενώ ανανεώνονται κάθε 20 λεπτά.

5.6.6 Ερωτήματα με ένθετα υποερωτήματα (nested subqueries)

5.6.6.1 Σε έναν μόνο υποκλάδο της σύνδεσης:

Τα ένθετα υποερωτήματα αυτής της κατηγορίας απεικονίζονται όπως φαίνεται στο σχήμα 5.26 και διατυπώνονται σε κώδικα γλώσσας SQL με τη χρήση μίας WITH clause και έχουν τη μορφή:

```
WITH <NEW_Stream_name> AS
(
SELECT      <select_list1>
FROM        <Stream2_name> [window2],
WHERE       <selection_predicate>
GROUP BY    <group_by_expressions>      //προαιρετικά
)
SELECT      <select_list2>
FROM        <Stream1_name> [window1],
            <NEW_Stream_name> [window3]
WHERE       <Selection_predicate> AND <Join_predicate>
GROUP BY    <group_by_expressions>;     //προαιρετικά
```



Σχήμα 5.26: *Απλή μορφή προσχεδίου εκτέλεσης ερωτήματος διαρκείας με ένα ένθετο υποερώτημα.*

Παράδειγμα 5.13:
" Δείξε την τρέχουσα τιμή υγρασίας και την μέση τιμή θερμοκρασίας για κάθε πόλη. "

```
WITH Temp_Stream_2 AS
(
SELECT     TemperatureStream.CityName,
           MAX(TemperatureStream.tcqtime) AS max1,
           AVG(TemperatureStream.temperature) AS avg2
FROM       TemperatureStream [RANGE BY '60 seconds'
                              SLIDE BY '20 seconds']
GROUP BY   TemperatureStream.CityName
)
SELECT     Temp_Stream_2.max1, Temp_Stream_2.avg2,
           HumidityStream.humidity, HumidityStream.tcqtime
FROM       HumidityStream [RANGE BY '1 second'
                           SLIDE BY '1 second'],
           Temp_Stream_2 [RANGE BY '1 second'
                         SLIDE BY '1 second']
WHERE      HumidityStream.CityName = Temp_Stream_2.CityName;
```

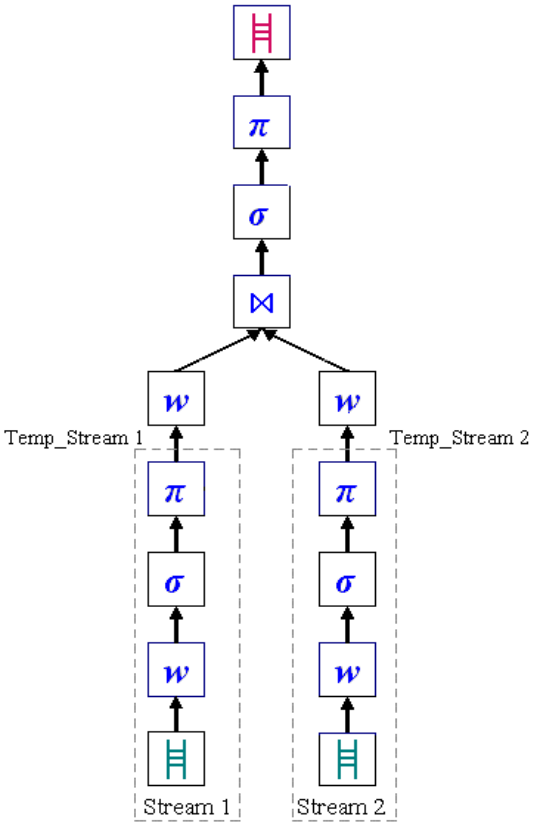
Στο παράδειγμα αυτό η μέση τιμή θερμοκρασίας υπολογίζεται μέσα σε ένα κυλιόμενο χρονικό παράθυρο εύρους μίας ώρας και βήματος 20 λεπτών.

Το γεγονός ότι το ρεύμα TemperatureStream περνά μέσα από τελεστές επιλογής και προβολής και η μέση τιμή της θερμοκρασίας υπολογίζεται πριν γίνει η σύνδεση των δυο ρευμάτων μειώνει αρκετά τον όγκο της πληροφορίας και το χρόνο επεξεργασίας στον κύριο κλάδο του ερωτήματος.

5.6.6.2 Και στους δυο υποκλάδους της σύνδεσης:

Η γενική μορφή της διατύπωσης σε γλώσσα SQL των ερωτημάτων διαρκείας αυτής της μορφής με τη χρήση δύο WITH clauses είναι:

```
WITH <NEW_Stream3_name> AS
(
SELECT     <select_list>
FROM       <Stream1_name> [window1],
WHERE      <selection1_predicate>
GROUP BY   <group_by_expressions>       //προαιρετικά
)
<NEW_Stream4_name> AS
(
SELECT     <select_list>
FROM       <Stream2_name> [window2],
WHERE      <selection2_predicate>
GROUP BY   <group_by_expressions>       //προαιρετικά
)
SELECT     <select_list>
FROM       <NEW_Stream3_name> [window3],
           <NEW_Stream4_name> [window4]
WHERE      <selection3_predicate> AND <join_predicate>
GROUP BY   <group_by_expressions>;     //προαιρετικά
```



Σχήμα 5.27: *Μορφή προσχεδίου εκτέλεσης ερωτήματος διαρκείας με δύο ένθετα υποερωτήματα.*

Παράδειγμα 5.14:

" Χρησιμοποιώντας τις μέγιστες αποκλίσεις τιμών υγρασίας και θερμοκρασίας που μετρήθηκαν σε κάθε πόλη, για τα τελευταία 20 λεπτά και υπολόγισε τον Μετεωρολογικό Δείκτη τύπου 2."

```
WITH Temp_Stream_1 AS
(
SELECT      HumidityStream.CityName, MIN(HumidityStream.humidity)
            AS min1, MAX(HumidityStream.humidity) AS max1
FROM        HumidityStream [RANGE BY '20 minutes'
                           SLIDE BY '5 minutes']
)
WITH Temp_Stream_2 AS
(
SELECT      TemperatureStream.CityName,
            MIN(TemperatureStream.temperature) AS min2,
            MAX(TemperatureStream.temperature) AS max2
FROM        TemperatureStream [RANGE BY '20 minutes'
                              SLIDE BY '5 minutes']
)
SELECT      Temp_Stream_1.CityName,
            ((TEMP_Str_1.min1 - TEMP_Str_1.max1)/(TEMP_Str_2.min2
            - TEMP_Str_2.max2)) AS index2
```

```
FROM        Temp_Stream_1 [RANGE BY '5 minutes'
                           SLIDE BY '5 minutes'],
            Temp_Stream_2 [RANGE BY '5 minutes'
                           SLIDE BY '5 minutes']
WHERE       Temp_Stream_1.CityName = Temp_Stream_2.CityName;
```

Ως Μετεωρολογικό Δείκτη τύπου 2 ορίζουμε τον λόγο της απόκλισης τιμών θερμοκρασίας προς την αντίστοιχη απόκλιση τιμών της υγρασίας.

Για τον υπολογισμό της μέγιστης και της ελάχιστης τιμής υγρασίας και θερμοκρασίας χρησιμοποιούμε τις μετρήσεις που πραγματοποιήθηκαν κατά τα τελευταία 20 λεπτά, ανανεώνοντας τα δεδομένα εισόδου κάθε 5 λεπτά (κυλιόμενο χρονικό παράθυρο).

Το γεγονός ότι τα ρεύματα TemperatureStream και HumidityStream περνούν μέσα από τελεστές επιλογής και προβολής πριν γίνει η μεταξύ τους σύνδεση, ενώ επίσης οι μέγιστες και ελάχιστες ζητούμενες τιμές υπολογίζονται στους επιμέρους κλάδους του προσχεδίου εκτέλεσης μειώνει σημαντικά τον όγκο της πληροφορίας προς επεξεργασία στον κύριο κλάδο του ερωτήματος.

Παράδειγμα 5.15:

" Δείξε τις πόλεις στις οποίες επικρατούν άσχημες καιρικές συνθήκες, δηλαδή υγρασία πάνω από 70 και θερμοκρασίες χαμηλότερες των 18°C, και υπολόγισε τον Μετεωρολογικό Δείκτη τύπου 3."

```
WITH Temp_Stream_1 AS
(
SELECT      AVG(HumidityStream.humidity) AS avg1,
            HumidityStream.CityName,
            MAX(HumidityStream.tcqtime) AS max1
FROM        HumidityStream [RANGE BY '40 minutes'
                            SLIDE BY '20 minutes']
GROUP BY    HumidityStream.CityName
HAVING      AVG(HumidityStream.humidity) > 70.00
)
WITH Temp_Stream_2 AS
(
SELECT      AVG(TemperatureStream.temperature) AS avg2,
            TemperatureStream.CityName,
            MAX(TemperatureStream.tcqtime) as max2
FROM        TemperatureStream [RANGE BY '40 minutes'
                               SLIDE BY '20 minutes']
GROUP BY    TemperatureStream.CityName
HAVING      AVG(TemperatureStream.temperature) < 18.00
)
SELECT      Temp_Stream_1.avg1, Temp_Stream_2.avg2,
            (Temp_Stream_1.avg1/Temp_Stream_2.avg2) AS index3
            Temp_Stream_1.CityName, Temp_Stream_2.max2
FROM        Temp_Stream_1 [RANGE BY '20 minutes'
                           SLIDE BY '20 minutes'],
            Temp_Stream_2 [RANGE BY '20 minutes'
                           SLIDE BY '20 minutes']
WHERE       Temp_Stream_1.CityName = Temp_Stream_2.CityName;
```

Ορίζουμε ως Μετεωρολογικό Δείκτη τύπου 3 τον λόγο της μέσης τιμής θερμοκρασίας προς την αντίστοιχη μέση τιμή της υγρασίας. Ο υπολογισμός των μέσων όρων γίνεται στα ένθετα υποερωτήματα, ενώ ο ζητούμενος δείκτης υπολογίζεται μετά τη σύνδεση των δύο ρευμάτων.

Όπως και στο προηγούμενο παράδειγμα, το γεγονός ότι τα ρεύματα `TemperatureStream` και `HumidityStream` περνούν μέσα από τελεστές επιλογής και προβολής και η μέση τιμή της θερμοκρασίας και της υγρασίας υπολογίζονται πριν γίνει η σύνδεση των δυο ρευμάτων μειώνει πολύ σημαντικά τον όγκο της πληροφορίας και το χρόνο επεξεργασίας στον κύριο κλάδο του ερωτήματος.

Κεφάλαιο 6

Επίλογος

6.1 Συμπεράσματα - Ανασκόπηση

Στην παρούσα διπλωματική εργασία επιχειρήθηκε να σχεδιαστεί και να υλοποιηθεί ένα φιλικό προς τον χρήστη και γενικού σκοπού εργαλείο διαχείρισης ρευμάτων δεδομένων.

Η εφαρμογή αυτή δίνει τη δυνατότητα εύκολης και συντακτικά ορθής διατύπωσης διαφόρων ερωτημάτων διαρκείας, χρησιμοποιώντας τους βασικότερους τύπους τελεστών και παραθύρων. Όπως αναλύθηκε στο κεφάλαιο 5, τα διαφορετικά είδη δενδρικών δομών που αναπαριστούν προσχέδια εκτέλεσης και τα οποία μπορούν να διατυπωθούν με χρήση της εφαρμογής ποικίλλουν σημαντικά.

Ειδικά με την υποστήριξη ένθετων υποερωτημάτων ο χρήστης μπορεί να συντάξει ερωτήματα με μεγάλο βαθμό πολυπλοκότητας, στα οποία περιλαμβάνεται μεγάλος αριθμός τελεστών. Αυτό έχει ως αποτέλεσμα την ανάλογη αύξηση του πλήθους των παραμέτρων του ερωτήματος (εύρος και βήμα παραθύρων, κατηγορήματα επιλογών και συνδέσεων, κ.ά.) οι οποίες πρέπει να καθοριστούν.

Όσο αυξάνεται η πολυπλοκότητα των ερωτημάτων και πληθαίνουν οι τελεστές που λαμβάνουν μέρος σε αυτά, τόσο πιο έντονα διαφαίνεται η χρησιμότητα της παρούσας εργασίας. Σε τέτοια ερωτήματα το μέγεθος της SQL εντολής αυξάνεται, με αποτέλεσμα η πιθανότητα λάθους σε προσπάθεια απευθείας διατύπωσης να μεγαλώνει. Αντίθετα, η εφαρμογή εξασφαλίζει την συντακτική ορθότητα της εντολής με ελέγχους σε όλα τα στάδια σχεδίασης του προσχεδίου εκτέλεσης και καθορισμού των παραμέτρων του ερωτήματος. Επιπλέον οι ευκολίες που παρέχει το γραφικό περιβάλλον για την δημιουργία των προσχεδίων και την διαχείριση ρευμάτων κάνουν ακόμα πιο προσφιλή προς τον χρήστη την διατύπωση ερωτημάτων διαρκείας μέσω της εφαρμογής, εφόσον δεν απαιτείται η γνώση κάποιας γλώσσας ερωταποκρίσεων με δηλωτική διατύπωση όπως η SQL.

Η σημασία της δυνατότητας αναστολής και επανεκκίνησης της εκτέλεσης των ερωτημάτων γίνεται επίσης εμφανής σε περιπτώσεις σύνθετων και πολύπλοκων ερωτημάτων. Αν ο χρήστης επιθυμεί να διατυπώσει παραπλήσια ερωτήματα, μπορεί να χρησιμοποιήσει το ήδη υπάρχον προσχέδιο, να τροποποιήσει κατάλληλα τις παραμέτρους και στη συνέχεια να το υποβάλει ξανά στο σύστημα. Με αυτόν τον τρόπο αποφεύγεται η διαδικασία ανασύνθεσης του προσχεδίου εκτέλεσης, ενώ οι όποιες αλλαγές γίνονται εύκολα και γρήγορα.

Τέλος, η δυνατότητα ταυτόχρονης γραφικής σχεδίασης, διατύπωσης με δηλωτικό τρόπο (SQL) και εκτέλεσης πολλαπλών ερωτημάτων διαρκείας δίνει στην παρούσα εφαρμογή μεγάλη ευελιξία. Διαφορετικά ερωτήματα, τα οποία ενδεχομένως χρησιμεύουν για την εξαγωγή συμπερασμάτων για τις ανάγκες μιας μελέτης, υποβάλλονται και μπορούν να παραμένουν ενεργά για μεγάλο χρονικο διάστημα προβάλλοντας τα αποτελέσματά τους σε πραγματικό χρόνο.

6.2 Μελλοντικές επεκτάσεις

Στο μέλλον μπορούν να γίνουν διάφορες επεκτάσεις αυτής της εργασίας. Ενδεικτικά σημειώνονται οι εξής:

- Διασύνδεση της εφαρμογής και με άλλα Συστήματα Διαχείρισης Ρευμάτων Δεδομένων όπως είναι τα Aurora, STREAM κτλ. Η παραλλαγή αυτή στην ουσία προϋποθέτει μικρές αλλαγές στην κλάση που δημιουργεί τον SQL κώδικα κάθε ερωτήματος, έτσι ώστε οι παραγόμενες εντολές να είναι σύμφωνες με το συντακτικό του ΣΔΡΔ που θα έχει επιλεγεί.
- Δυνατότητα δημιουργίας και απεικόνισης δένδρου εκτέλεσης ερωτήματος απευθείας από δοσμένη εντολή γλώσσας SQL. Ουσιαστικά πρόκειται για την αντίστροφη διαδικασία από αυτήν που υλοποιεί η εφαρμογή. Θα πρέπει να υλοποιηθεί ένας διαφορετικός parser ο οποίος θα διατρέχει το κείμενο της εντολής SQL και θα παράγει πάνω στο γραφικό περιβάλλον την απεικόνιση των τελεστών που συμμετέχουν και την μεταξύ τους διασύνδεση.
- Απευθείας εκτέλεση των ερωτημάτων διαρκείας από το προσχέδιο εκτέλεσης χωρίς την διαμεσολάβηση της μετάφρασης σε SQL-like κώδικα και της διασύνδεσης με κάποιο ήδη υπάρχον σύστημα. Οι τελεστές που συμμετέχουν στα ερωτήματα θα μπορούσαν να είναι υλοποιημένοι σε κάποια γλώσσα προγραμματισμού (π.χ. κλάσεις της C++) και να καλούνται απευθείας από την εφαρμογή, η οποία θα πρέπει να φροντίζει για τον συγχρονισμό τους και την κατάλληλη τροφοδότηση με δεδομένα από την έξοδο ενός στην είσοδο του επόμενου τελεστή του προσχεδίου εκτέλεσης.
- Βελτιστοποίηση του δένδρου που απεικονίζει το προσχέδιο εκτέλεσης των ερωτημάτων. Η δυνατότητα αυτή παρέχεται από τα περισσότερα συστήματα διαχείρισης ρευμάτων δεδομένων. Πρόκειται ουσιαστικά για μια αναδιάταξη της αλληλουχίας των τελεστών του ερωτήματος ώστε να εξάγονται τα ίδια αποτελέσματα αλλά να έχουμε κέρδος στην απόδοση και στον χρόνο εκτέλεσης. Οι τελεστές φιλτραρίσματος εφαρμόζονται όσο το δυνατόν νωρίτερα στα δεδομένα εισόδου για να μειωθεί ο όγκος των στοιχείων προς επεξεργασία στους επόμενους τελεστές στην ακολουθία του προσχεδίου.

Βιβλιογραφικές Αναφορές

[ACC+03b] 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.

[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.

[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.

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

[CL03] R. Cadenhead and L. Lemay. Sams Teach Yourself Java 2 in 21 days, *M. Γκιούρδας*, 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.

[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.

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

[HA00] J. M. Hellerstein and R. Avnur. Eddies: Continuously Adaptive Query Processing. In *Proceedings of the 19th ACM SIGMOD International Conference on Management of Data*, pp. 261–272, Dallas, Texas, May 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.

[MF02] Samuel R. Madden and Michael J. Franklin. Fjording the Stream: An Architecture for Queries over Streaming Sensor Data. In *Proceedings of the 18th International Conference on Data Engineering*, pp. 555-566, San Jose, California, February 2002.

[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.

[PS06] K. Patroumpas and T. Sellis. Window Specification over Data Streams. In *Proceedings of International Conference on Semantics of a Networked World (ICSNW'06)*, pp. 445-464, Munich, Germany, March 2006.

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

[SKS04] A. Silberschatz, H.F. Korth and S. Sudarshan. Database System Concepts, *M. Γκιούρδας*, 2004.

[SCZ05] M. Stonebraker, U. Cetintemel, and S. Zdonik. The 8 Requirements of Real-Time Stream Processing. *ACM SIGMOD Record*, 34(4):42-47, December 2005.

[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.

[WA91] A. Wilschut and P. Apers. Dataflow Query Execution in a Parallel Main-Memory Environment. In *Proceedings of the 1st International Conference on Parallel and Distributed Information Systems (PDIS 1991)*, pp. 68-77, Miami, Florida, December 1991.

[Πατ03] Κ. Πατρούμπας. Συστήματα Ρευμάτων Δεδομένων για Κινούμενα Αντικείμενα. *Μεταπτυχιακή διπλωματική εργασία στο Δ.Π.Μ.Σ. Τεωπληροφορική', Ε.Μ.Π.*, Ιούνιος 2003.

[Τζω05] Η. Τζωρτζακάκης. Προσδιορισμός Παραθύρων σε Ρεύματα Δεδομένων. *Διπλωματική εργασία στο Εργαστήριο Συστημάτων Βάσεων Γνώσεων & Δεδομένων, Ε.Μ.Π.*, Οκτώβριος 2005.

Διαδικτυακοί τόποι :

[siteJAVA] Ι στοσελίδα της Java και tutorials:
http://java.sun.com/j2se/1.4.2/
http://java.sun.com/docs/books/tutorial/uiswing/index.html
http://java.sun.com/docs/books/tutorial/uiswing/dnd/index.html

[siteTCQ] Ιστοσελίδα του συστήματος TelegraphCQ:
http://telegraph.cs.berkeley.edu/telegraphcq/v2.0/index.html
http://telegraph.cs.berkeley.edu/telegraphcq/v2.0/Windows.html
http://telegraph.cs.berkeley.edu/telegraphcq/v2.0/Punctuation.html

[siteSTREAMSQL] Ιστοσελίδα για την γλώσσα StreamSQL:
http://streambase.com/developers/docs/latest/streamsql/index.html
http://streambase.com/developers/docs/latest/streamsql/select.html
http://streambase.com/developers/docs/latest/streamsql/createwindow.html

Ορολογία

ανασταλτικός τελεστής	blocking operator
απαλοιφή διπλοτύπων	duplicate elimination
αποβολή φόρτου	load shedding
γλώσσα ερωταποκρίσεων	query language
γνώρισμα	attribute
εμβέλεια	scope
ένθετο υποερώτημα	nested subquery
ένωση	union
επιλογή	selection
ερώτημα διαρκείας	continuous query
ερώτημα στιγμιοτύπου	one-time, snapshot query
κυματίδιο	wavelet
παραθυρικός τελεστής	windowed operator
παράθυρο επάλληλο	tumbling window
παράθυρο κυλιόμενο	sliding window
παράθυρο μεριστικό	partitioned window
παράθυρο πλειάδων	tuple-based window
παράθυρο ορισήμου	landmark window
περίληψη	summary
προβολή	projection
προσέγγιση	approximation
προσχέδιο εκτέλεσης ερωτήματος	query execution plan
ρεύμα δεδομένων	data stream
σκίτσο	sketch
στίξη	punctuation
συνάθροιση	aggregation
σύνδεση	join
συνένωση	concatenation
σύνοψη	synopsis
συγχώνευση	merge
σχέση	relation
τελεστής	operator
τομή	intersection
χρονόσημο	timestamp

Σύνθεση προσχεδίων εκτέλεσης ερωτημάτων διαρκείας

Όλγα Γκουντούνα
el01140@mail.ntua.gr

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

1. Εισαγωγή

Σε πολλές σύγχρονες εφαρμογές (λ.χ. σε δίκτυα αισθητήρων, χρηματιστηριακά συστήματα, δημοτικότητα ιστοσελίδων κ.ά.) συλλέγονται στοιχεία με τη μορφή ταχύτατα μεταβαλλόμενων *ρευμάτων δεδομένων* (*data streams*). Επίσης, για την επεξεργασία τέτοιων στοιχείων διατυπώνονται *ερωτήματα διαρκείας* (*continuous queries*) που παραμένουν ενεργά για μεγάλο χρονικό διάστημα. Για να είναι εφικτή η ανανέωση των αποτελεσμάτων σε πραγματικό χρόνο, εισάγονται *παράθυρα* (*windows*) που απομονώνουν ένα πεπερασμένο υποσύνολο των στοιχείων του πιθανόν ανεξάντλητου ρεύματος. Οι παραθυρικές αυτές δομές αποτελούν κοινή συνιστώσα στις μέχρι τώρα προσεγγίσεις που αφορούν επεξεργασία ρευμάτων δεδομένων.

Στην παρούσα διπλωματική εργασία επιδιώκεται η υλοποίηση ενός *γραφικού περιβάλλοντος* (GUI) διατύπωσης και εκτέλεσης πολλαπλών ερωτημάτων διαρκείας με διάφορους τύπους παραθυρικών δομών. Τα ερωτήματα θα περιλαμβάνουν επιλεγμένους σχεσιακούς τελεστές όπως σύνδεση (join), συνάθροιση (aggregate) και εφαρμογή κατηγορημάτων για φιλτράρισμα των ρευμάτων εισόδου. Για την υλοποίηση του περιβάλλοντος χρήστη επιλέχτηκε η γλώσσα προγραμματισμού JAVA ενώ για την εκτέλεση των παραγόμενων ερωτημάτων αξιοποιήθηκε ένας ήδη υπάρχων μηχανισμός επεξεργασίας ρευμάτων δεδομένων (TelegraphCQ).

2. Επεξεργασία Ρευμάτων

Ως **ρεύμα δεδομένων** (*data stream*) μπορεί να θεωρηθεί μια ακολουθία στοιχείων που παράγονται διαρκώς από κάποια πηγή δεδομένων. Τα στοιχεία του ρεύματος διατηρούν την γνώριμη μορφή της σχεσιακής πλειάδας (*tuple*) που συνοδεύεται από *χρονόσημο* (*timestamp*). Δεν βρίσκονται λοιπόν αποθηκευμένα σε στατικούς πίνακες, αλλά διαρκώς καταφθάνουν στο σύστημα με κυμαινόμενο ρυθμό, οπότε το μέγεθος του ρεύματος μπορεί να είναι απεριόριστο. Το σύστημα δεν ελέγχει τη διάταξη ή τον ρυθμό με τον οποίο καταφθάνουν τα δεδομένα. Κάθε στοιχείο του ρεύματος μετά την επεξεργασία του είτε αποθηκεύεται είτε απορρίπτεται. Επομένως, μελλοντικές προσπελάσεις του είναι είτε ιδιαίτερα δαπανηρές είτε εντελώς αδύνατες.

Οι **τελεστές** (*operators*) των συμβατικών συστημάτων βάσεων δεδομένων παρουσιάζουν αρκετές δυσκολίες κατά την προσαρμογή τους σε ρεύματα δεδομένων. Ορισμένοι τελεστές (*προβολή και επιλογή*) μπορούν να εφαρμοστούν απευθείας στο ρεύμα δεδομένων, διότι μπορούν να επεξεργαστούν αυτοτελώς κάθε πλειάδα του. Άλλοι τελεστές (*συνάθροισης, σύνδεσης, απαλοιφής διπλοτύπων* κτλ.) χρειάζονται διαρκώς πρόσβαση σε όλα τα δεδομένα εισόδου για να εξάγουν σωστά αποτελέσματα. Όμως, η αποθήκευση όλων των πλειάδων ενός ρεύματος συνήθως δεν είναι εφικτή.

Για την αντιμετώπιση αυτού του προβλήματος η τροφοδότηση τέτοιων τελεστών γίνεται από **παράθυρα** (*windows*). Σκοπός των παραθυρικών δομών είναι να αποσπούν συνεχώς κάποιο πεπερασμένο πλήθος στοιχείων του ρεύματος παρέχοντας διαρκώς πρόσβαση σε περιορισμένο τμήμα του. Συνήθεις τύποι παραθύρων είναι οι εξής:

- **Παράθυρα Πλειάδων** (*tuple-based windows*). Διατηρούν στην έξοδό τους τις N τελευταίες πλειάδες από το στιγμιότυπο του ρεύματος εισόδου.
- **Χρονικά Κυλιόμενα Παράθυρα** (*sliding time-based windows*). Διατηρούν στην έξοδό τους όσες από τις πιο πρόσφατες πλειάδες έχουν χρονόσημο που εμπίπτει στην εμβέλεια του παραθύρου.
- **Χρονικά Επάλληλα Παράθυρα** (*tumbling windows*). Ειδική περίπτωση των

κυλιόμενων. Στην έξοδο προκύπτουν μη επικαλυπτόμενα διαδοχικά στιγμιότυπα του ρεύματος εισόδου χωρίς απώλεια πληροφορίας.

♦ **Παράθυρα Οροσήμου** (*landmark windows*). Διατηρούν σταθερό το ένα χρονικό τους άκρο, δίνοντας στο ρεύμα εξόδου τους ένα διαρκώς αυξανόμενο αριθμό πλειάδων.

♦ **Μεριστικά Παράθυρα** (*partitioning windows*). Διατηρούν στην έξοδό τους τις *N* πιο πρόσφατες πλειάδες για συγκεκριμένο συνδυασμό τιμών από μια λίστα *γνωρισμάτων* (*partitioning attributes*).

3. Προδιαγραφές εφαρμογής

Η εφαρμογή αποτελείται από δύο βασικά δομικά στοιχεία:

- **Το Σύστημα Διαχείρισης Ρευμάτων Δεδομένων (TelegraphCQ).** Διαχειρίζεται τα εισερχόμενα ρεύματα δεδομένων και εκτελεί τα ερωτήματα διαρκείας που ορίζονται πάνω σε αυτά, δίνοντας ως έξοδο παράγωγα ρεύματα δεδομένων.
- **Το Γραφικό περιβάλλον διεπαφής.** Αποτελεί το πλαίσιο σχεδίασης των ερωτημάτων και το μέσο επικοινωνίας του χρήστη με το Σύστημα Διαχείρισης Ρευμάτων Δεδομένων.

Η δημιουργία των **ρευμάτων** και η τροφοδότησή τους με δεδομένα γίνεται από ειδικά διαμορφωμένα ASCII αρχεία. Η επιλογή των αρχείων αυτών και του ρυθμού μετάδοσης των πλειάδων επιλέγεται από τον χρήστη.

Η σύνθεση των προσχεδίων εκτέλεσης ερωτημάτων γίνεται σε μορφή **δένδρων** όπου κόμβοι είναι οι τελεστές του ερωτήματος, ενώ τα βέλη που τους συνδέουν δείχνουν τη λογική διασύνδεσή τους. Η φορά του βέλους δηλώνει τροφοδότηση με δεδομένα από το ρεύμα εξόδου του ενός στο ρεύμα εισόδου του επόμενου τελεστή στην ακολουθία.

Δεν επιχειρείται καμία βελτιστοποίηση του δέντρου, απλώς διευκολύνεται η διατύπωση των ερωτημάτων από το χρήστη.

Στόχος είναι η υποστήριξη όλων των βασικών τύπων τελεστών και παραθύρων:

- Προβολή (projection) [π],
- Επιλογή (selection) [σ],
- Σύνδεση (join) [$\bowtie$],
- Συνάθροιση (aggregation) [γ],
- Απαλοιφή διπλοτύπων (duplicate elimination) [δ].
- Παράθυρα [ω] (πλειάδων, χρονικά, οροσήμου και μεριστικά).

Από το γραφικά σχεδιασμένο προσχέδιο εκτέλεσης θα παράγεται ο κώδικας του ερωτήματος διαρκείας σε γλώσσα **SQL**, προσαρμοσμένη όμως στο συντακτικό του Συστήματος Διαχείρισης Ρευμάτων Δεδομένων που χρησιμοποιείται.

Τα **αποτελέσματα** του ερωτήματος, δηλαδή οι πλειάδες του ρεύματος εξόδου, θα πρέπει να έχουν μια συνεχή ροή και να εμφανίζονται σε πίνακες δυναμικά και σε πραγματικό χρόνο.

Τέλος, τονίζεται ότι θα υπάρχει η δυνατότητα παράλληλης σχεδίασης, διατύπωσης και εκτέλεσης **πολλαπλών ερωτημάτων** διαρκείας.

4. Σύστημα TelegraphCQ.

Το TelegraphCQ αποτελεί ένα πρωτότυπο Σύστημα Διαχείρισης Ρευμάτων Δεδομένων που αναπτύσσεται στο πανεπιστήμιο Berkeley (πιο πρόσφατη έκδοση η 2.0). Εκτός από την αναμενόμενη υποστήριξη επεξεργασίας ρευμάτων, προτιμήθηκε για τους εξής λόγους:

- Αξιοποιεί τον μηχανισμό της PostgreSQL εμπλουτισμένο με παραθυρικές δομές που καθιστούν ικανή την διαχείριση ρευμάτων δεδομένων.
- Χρησιμοποιεί ως γλώσσα ερωταποκρίσεων την SQL, γεγονός που ευνοεί σημαντικά την διατύπωση των ερωτημάτων με δηλωτικό τρόπο.
- Έχει σχεδιαστεί με έμφαση στην προσαρμοστικότητα σε κυμαινόμενους ρυθμούς δεδομένων.

Χρησιμοποιήθηκε η έκδοση **TelegraphCQ v2.0** όπου η σύνταξη σε SQL των ερωτημάτων διαρκείας πάνω σε ρεύματα δεδομένων ακολουθεί τη γενική μορφή:

SELECT <*select_list*>
FROM <*Stream*> [<*window* >],
…,
<*Stream*> [<*window*>]
WHERE <*predicate*>
GROUP BY <*group_by_expressions*>
HAVING <*predicate*>;

Τα παράθυρα που υλοποιεί το

TelegraphCQ είναι μόνο χρονικά παράθυρα κυλιόμενα ή επάλληλα ανάλογα με τις τιμές των παραμέτρων έκτασης (range) και βήματος (slide). Η σύνταξη των παραθύρων ακολουθεί τη γενική μορφή:

[RANGE BY '*x* **seconds'**
SLIDE BY '*y* **seconds'**
START AT '*z* **seconds']**

5. Αρχιτεκτονική Εφαρμογής

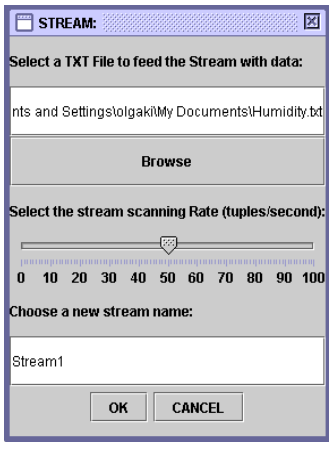
Το περιβάλλον διεπαφής χρήστη (user interface) υλοποιήθηκε σε γλώσσα προγραμματισμού JAVA ώστε να επιτευχθεί ανεξαρτησία πλατφόρμας εκτέλεσης της εφαρμογής.

5.1 Τρόπος λειτουργίας

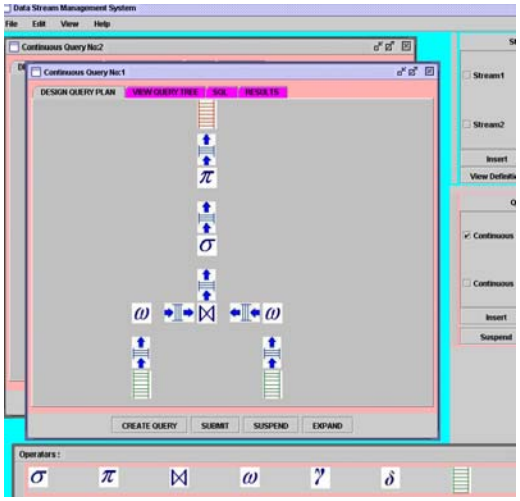
Η εισδοχή των δεδομένων γίνεται από ειδικά βοηθητικά προγράμματα (wrappers) που δέχονται ως είσοδο ειδικά διαμορφωμένα ASCII αρχεία και στέλνουν τα περιεχόμενά τους στο TelegraphCQ με τη μορφή ρεύματος.

Η επιλογή του ASCII αρχείου και του ρυθμού κάθε ρεύματος επιλέγεται από τον χρήστη μέσα από ειδική φόρμα.

Στο *Σχήμα 1* βλέπουμε τη μορφή μιας τέτοιας φόρμας όπου ο χρήστης επιλέγει ένα .txt αρχείο και ένα ρυθμό (πλειάδες ανά δευτερόλεπτο) και εισάγει ένα όνομα για το νέο ρεύμα δεδομένων που θα δημιουργηθεί.



Σχήμα 1: *Φόρμα δημιουργίας νέου ρεύματος δεδομένων εισόδου.*



Σχήμα 2: *Φόρμα σχεδίασης προσχεδίων εκτέλεσης ερωτημάτων διαρκείας και εργαλειοθήκη τελεστών (toolbar).*

Ο χρήστης της εφαρμογής έχει στη διάθεσή του μια εργαλειοθήκη τελεστών (operators' toolbar), όπου η δημιουργία κάθε δέντρου προσχεδίου εκτέλεσης γίνεται με drag-and-drop πάνω σε ένα πάνελ μιας φόρμας διατύπωσης ερωτήματος διαρκείας. Στη συνέχεια ενώνει τους τελεστές με βέλη ώστε να φαίνεται η λογική τους διασύνδεση στο δέντρο του ερωτήματος.

Στο *Σχήμα 2* φαίνεται ένα παράδειγμα λειτουργίας της εφαρμογής κατά τη φάση της σχεδίασης ενός δένδρου προσχεδίου εκτέλεσης ενός σχετικά απλού ερωτήματος διαρκείας.

Ακολούθως, ο χρήστης μπορεί να καθορίσει τις ιδιότητες του κάθε τελεστή (λ.χ. επιλεγμένα πεδία της προβολής, κατηγόρημα της επιλογής, εύρος του παραθύρου, κτλ.) μέσω φορμών παραμετροποίησης.

Σε αυτές τις φόρμες γίνεται ένα πρώτο επίπεδο ελέγχου της ορθότητας του ερωτήματος, εφόσον δεν επιτρέπεται στο χρήστη να εισάγει οτιδήποτε θελήσει. Πολλές τιμές δίνονται από Combo-Boxes ή Check-Boxes προκαθορισμένων επιλογών, ενώ αν κριθεί σκόπιμο μπορεί να απενεργοποιηθούν κάποιες επιλογές για αποφυγή λαθών.

Επίσης, επιτρέπονται ένθετα υποερωτήματα (*nested subqueries*) τα οποία υλοποιούνται με **WITH** clauses:

WITH NewStream_1 **AS (**

SELECT <*select_list*>

FROM <stream_list>

...**)**

SELECT <*select_list*>

FROM NewStream_1 [<window>]

...

Μετά την ολοκλήρωση του προσχεδίου, η εφαρμογή διατρέχει το δένδρο, ελέγχει την συντακτική ορθότητά του και δημιουργεί τον **SQL** κώδικα του ερωτήματος διαρκείας. Το αποτέλεσμα προβάλλεται στη φόρμα και ο χρήστης –εφόσον επιθυμεί– μπορεί να το τροποποιήσει πριν το υποβάλλει.

Στην εφαρμογή το TelegraphCQ εκτελεί όσα ερωτήματα του υποβάλλονται μέσα από το γραφικό περιβάλλον και επιστρέφει τα αποτελέσματά τους σταδιακά (incrementally) καθώς αυτά παράγονται. Το ρεύμα εξόδου του κάθε ερωτήματος προβάλλεται στην οθόνη από τον πίνακα των αποτελεσμάτων.

Ο χρήστης έχει τη δυνατότητα αναστολής εκτέλεσης (suspend) και επανεκκίνησης (resume) κάθε ερωτήματος.

5.2 Περιορισμοί συστήματος

Κατά την σχεδίαση του γραφικού περιβάλλον της εφαρμογής έγινε η προσπάθεια να είναι όσο το δυνατόν πιο φιλικό προς το χρήστη, ενώ παράλληλα να παρέχει κατά το δυνατόν περισσότερες δυνατότητες δημιουργίας ποικίλων ερωτημάτων που να μπορούν να εκτελούνται παράλληλα.

Εντούτοις, η χρήση του TelegraphCQ ως βασικού μηχανισμού εκτέλεσης των ερωτημάτων ενέχει κάποιους εγγενείς περιορισμούς:

- Η μόνη παραθυρική δομή που υποστηρίζεται είναι τα *κυλιόμενα χρονικά παράθυρα.*
- Η σύνδεση μεταξύ ρευμάτων γίνεται μόνο με ισότητα μεταξύ πεδίων (όχι με συναρτήσεις).
- Δεν υποστηρίζεται σύνδεση ενός ρεύματος με τον εαυτό του (*self join*).
- Τα κριτήρια επιλογής (selection) πρέπει έχουν την απλή μορφή: <attribute> OP <constant>.
- Δεν μπορεί να χρησιμοποιηθεί διάζευξη (OR) ή άρνηση (NOT) στα κριτήρια επιλογής.
- Δεν υποστηρίζεται απαλοιφή διπλοτύπων.

Ωστόσο, σαν προσπάθεια πρόβλεψης μελλοντικών εκδόσεων του TelegraphCQ όπου ενδεχομένως να απαλειφθούν οι παραπάνω περιορισμοί καθώς και για λόγους πληρότητας της εφαρμογής, το γραφικό περιβάλλον έχει κάποιες από τις παραπάνω επιλογές. Έτσι, υποστηρίζονται όλοι οι γνωστοί τύποι παραθύρων με τις παραμέτρους του καθενός, η απαλοιφή διπλοτύπων, τα self joins, όπως και κριτήρια επιλογής της μορφής: <attribute1> OP <attribute2>.

6. Συμπεράσματα

Στην παρούσα διπλωματική εργασία επιχειρήθηκε να σχεδιαστεί και να υλοποιηθεί ένα φιλικό προς το χρήστη και γενικού σκοπού εργαλείο διαχείρισης ρευμάτων δεδομένων, το οποίο θα δίνει τη δυνατότητα συντακτικά ορθής διατύπωσης ερωτημάτων διαρκείας. Πολλαπλά ερωτήματα μπορούν να σχεδιάζονται γραφικά, να διατυπώνονται με δηλωτικό τρόπο (SQL) και να εκτελούνται παράλληλα.

Στο μέλλον μπορούν να γίνουν διάφορες επεκτάσεις αυτής της εργασίας. Ενδεικτικά σημειώνονται οι εξής:

- Διασύνδεση της εφαρμογής και με άλλα Συστήματα Διαχείρισης Ρευμάτων Δεδομένων, όπως Aurora, STREAM κτλ.
- Δυνατότητα δημιουργίας και απεικόνισης δένδρου εκτέλεσης ερωτήματος απευθείας από δοσμένη εντολή γλώσσας SQL.
- Απευθείας εκτέλεση των ερωτημάτων διαρκείας από το προσχέδιο εκτέλεσης με χρήση τελεστών υλοποιημένων σε κάποια γλώσσα προγραμματισμού (π.χ. κλάσεις της C++), χωρίς την διαμεσολάβηση της μετάφρασης σε SQL-like κώδικα και της διασύνδεσης με κάποιο ήδη υπάρχον σύστημα.
- Βελτιστοποίηση του δέντρου που απεικονίζει το προσχέδιο εκτέλεσης των ερωτημάτων.