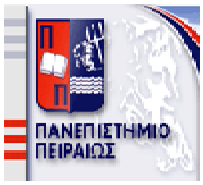


ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

ΤΜΗΜΑ ΔΙΔΑΚΤΙΚΗΣ ΤΗΣ ΤΕΧΝΟΛΟΓΙΑΣ
ΚΑΙ ΨΗΦΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Μεταπτυχιακό πρόγραμμα σπουδών
Κατεύθυνση: Δικτυοκεντρικά Συστήματα

«Ρεύματα δεδομένων:
τρέχουσα κατάσταση και προοπτικές»

Μεταπτυχιακή Διατριβή
του
Ανδρέα Μυλωνά
ΑΜ: ΜΕ/0580

ΠΕΡΙΕΧΟΜΕΝΑ

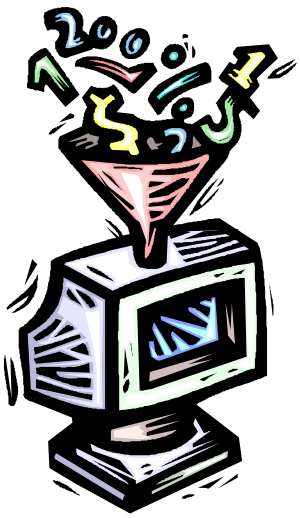
ΠΙΝΑΚΑΣ ΔΙΑΓΡΑΜΜΑΤΩΝ ΚΑΙ ΕΙΚΟΝΩΝ

ΛΙΣΤΑ ΠΙΝΑΚΩΝ

1. Εισαγωγή

1.1. Εισαγωγή στα ΣΔΡΔ

Τα παραδοσιακά Συστήματα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ) - (Relational Database Management Systems - RDBMS), είναι σχεδιασμένα για να αποθηκεύουν και να διαχειρίζονται επιχειρησιακά δεδομένα. Στα συστήματα αυτά, τεράστιοι όγκοι δεδομένων αποθηκεύονται σε κατάλληλα διαμορφωμένα συστήματα υποδομών, τέτοιων που να υποστηρίζουν την άντληση, την επεξεργασία και μετασχηματισμό των δεδομένων καθώς και την αποθήκευση τους σε μορφές τέτοιες που να εξυπηρετούν συγκεκριμένες επιχειρησιακές ανάγκες. Οι χρήστες δημιουργούν ερωτήματα (queries) τα οποία και εφαρμόζουν / εκτελούν πάνω σε αυτά τα δεδομένα. Το μοντέλο αυτό διαχείρισης της πληροφορίας αναφέρεται ως Human-Active / DBMS-Passive (HADP).



Οι σημερινές ανάγκες διαχείρισης της πληροφορίας έχουν δημιουργήσει ένα πλήθος σύγχρονων εφαρμογών οι οποίες καθιστούν επιτακτική την διαφοροποιημένη - σε σχέση με τα συμβατικά ΣΔΒΔ - προσέγγιση του τρόπου διαχείρισης των δεδομένων. Οι εφαρμογές αυτές εστιάζουν στις πιο πρόσφατες πληροφορίες που εισέρχονται στο σύστημα. Παραδείγματα τέτοιων εφαρμογών είναι:

1. Εφαρμογές ελέγχου και παρακολούθησης δικτύων τηλεπικοινωνιών όπου είναι επιθυμητή η παραγωγή σε πραγματικό χρόνο, ειδοποιήσεων για αυξημένο φόρτο, ή άλλα προβλήματα.
2. Εφαρμογές ελέγχου για τον εντοπισμό απατών (fraud detection systems) στο χώρο των τηλεπικοινωνιών και των τραπεζικών συναλλαγών όπου τα δεδομένα (οι κλήσεις ή οι συναλλαγές αντίστοιχα), ελέγχονται σε πραγματικό χρόνο έτσι ώστε να είναι δυνατή η απαγόρευση της συγκεκριμένης ενέργειας σε περίπτωση που το προφίλ του πελάτη δε συμφωνεί με το προφίλ της συγκεκριμένης ενέργειας.
3. Στρατιωτικές εφαρμογές στις οποίες γίνεται συλλογή δεδομένων από αισθητήρες που βρίσκονται στις στολές που φορούν οι στρατιώτες και που αφορούν ζωτικές λειτουργίες του ανθρώπινου σώματος. (Πίεση, θερμοκρασία, εγκεφαλική λειτουργία κτλ)
4. Εφαρμογές όπου συλλέγουν δεδομένα από αισθητήρες που βρίσκονται σε μετεωρολογικούς σταθμούς για την καταγραφή κλιματολογικών δεδομένων
5. Εφαρμογές όπου συλλέγουν δεδομένα από αισθητήρες καταγραφής ατμοσφαιρικής ρύπανσης, κυκλοφορίας οχημάτων κτλ.
6. Εφαρμογές παρακολούθησης και εντοπισμού στόλων οχημάτων. Τέτοιες εφαρμογές δίνουν τη δυνατότητα - σε συνεργασία με συστήματα GPS, GIS, RFID κτλ – παρακολούθησης ταχύτητας, και δρομολογίων αυτοκινήτων, ταξί, φορτηγών κτλ τόσο σε τοπικό, όσο –το κυριότερο- και σε σε παγκόσμιο επίπεδο.
7. Εφαρμογές ανάλυσης και επεξεργασίας χρηματοοικονομικών τιμών και δεικτών.

8. Εφαρμογές παρακολούθησης διακίνησης δεμάτων από τη στιγμή της παραγγελίας μέχρι την παραλαβή από τον παραλήπτη, δίνοντας τη δυνατότητα ενημέρωσης σε πραγματικό χρόνο για την κατάσταση της παραγγελίας του πελάτη.
9. Εφαρμογές επεξεργασίας δεδομένων πρόσβασης και αλληλεπίδρασης χρηστών σε συγκεκριμένους ιστότοπους. Τα στοιχεία αυτά μπορεί να χρησιμοποιούνται για την εξαγωγή στατιστικών στοιχείων, διαμόρφωσης προσωπικών σελίδων κτλ.

Κοινά χαρακτηριστικά αυτών των εφαρμογών αποτελούν:

- η εισροή πληροφορίας με τη μορφή δυναμικών ρευμάτων δεδομένων (data streams) τα οποία παράγονται από αυτόματα συστήματα και διαδικασίες χωρίς την παρέμβαση κάποιου χρήστη,
- η διαχείριση της εισερχόμενης πληροφορίας με έμφαση στην πιο πρόσφατη πληροφορία,
- η ανάγκη διατύπωσης ερωτημάτων διαρκείας (continuous queries), τα οποία απαιτούν online επεξεργασία και πρέπει να επιστρέφουν απαντήσεις σε πραγματικό χρόνο.
- η ανάγκη χειρισμού υψηλών όγκων δεδομένων με έντονες διακυμάνσεις στο ρυθμό άφιξης, σε πραγματικό χρόνο

Έτσι λοιπόν, το αρχικό μοντέλο human-active/DBMS-passive (HADP) που αφορά στα παραδοσιακά ΣΔΒΔ, καθίσταται ανεπαρκές για τις εφαρμογές που αναφέραμε μιας και αφορά στην άντληση δεδομένων αφού αυτά έχουν αποθηκευτεί και επεξεργαστεί σε κάποιο αποθηκευτικό μέσο.

Αντί αυτού, χρησιμοποιείται ένα νέο μοντέλο, το DBMS-active/human-passive (DAHP), για να καλύψει τις νέες ανάγκες διαχείρισης της δυναμικής πληροφορίας σε πραγματικό χρόνο. Το μοντέλο αυτό απαντά στην ανάγκη επεξεργασίας των δεδομένων σχεδόν αποκλειστικά στη φυσική μνήμη του συστήματος αφού προσφέρει υψηλές ταχύτητες. Πέραν αυτού, οι συνήθως υψηλοί όγκοι δεδομένων των εφαρμογών όπως και άλλες ιδιαιτερότητες και απαιτήσεις, καλύπτονται από το μοντέλο DAHP, το οποίο είναι ιδανικό για το χειρισμό ρευμάτων δεδομένων με διακυμάνσεις στο ρυθμό άφιξης των δεδομένων σε πραγματικό χρόνο. Το DAHP υλοποιείται στον πολλά υποσχόμενο τομέα των Συστημάτων Διαχείρισης Ρευμάτων Δεδομένων (ΣΔΡΔ) – (Data Stream Management Systems – DSMS).

1.2. Σκοπός και στόχοι της εργασίας

1.2.1. Σκοπός

Ο σκοπός της παρούσης Μεταπτυχιακής Διατριβής είναι να παρουσιάσει την τρέχουσα κατάσταση των σύγχρονων συστημάτων διαχείρισης ροών δεδομένων τόσο μέσω των ερευνητικών όσο και των εμπορικών υλοποιήσεων τέτοιων συστημάτων και να αναδείξει τις προοπτικές τους με βάση τα πεδία έρευνας σε όλους τους σχετικούς τομείς.

1.2.2. Στόχοι

Οι στόχοι της εργασίας είναι οι εξής:

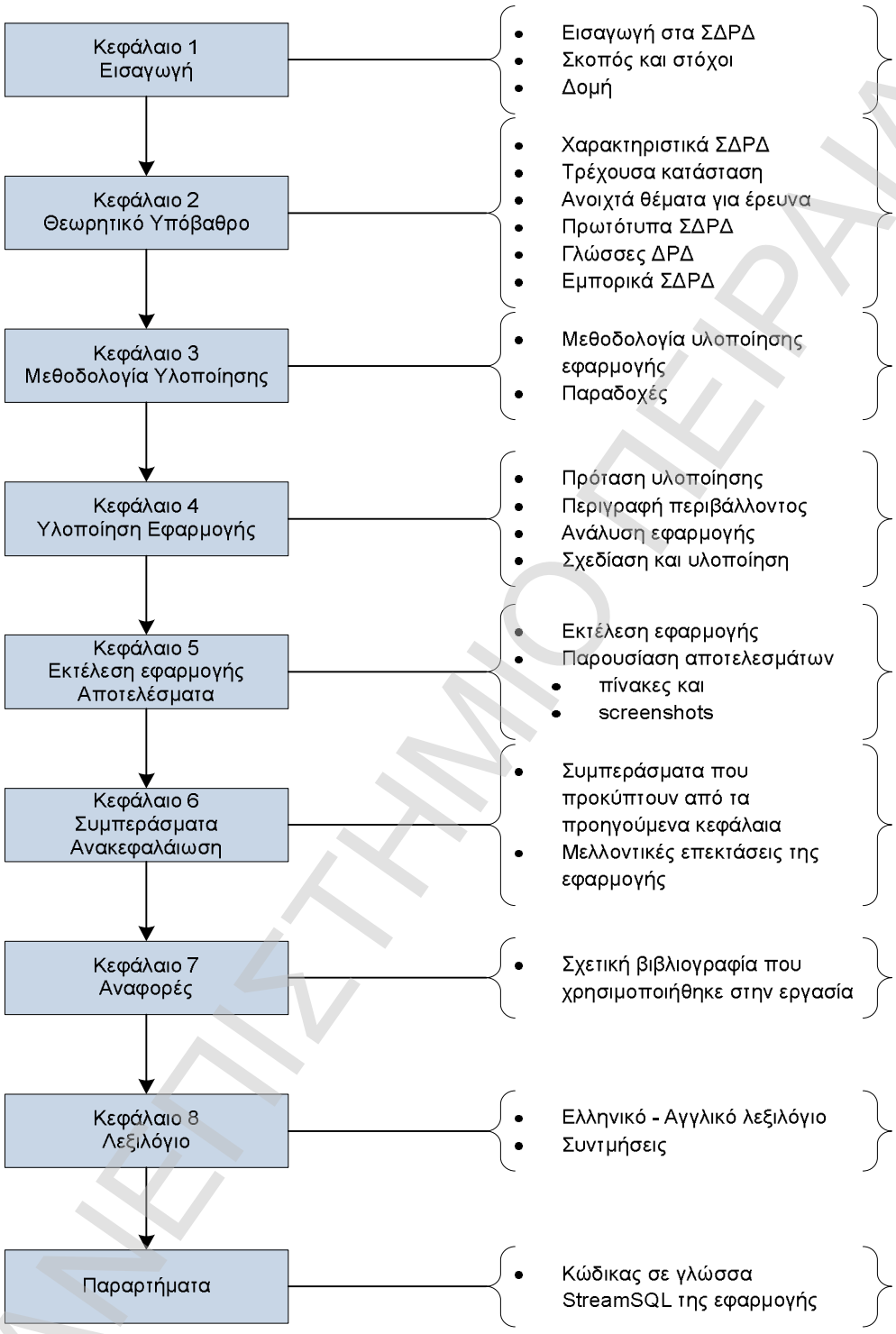
1. Η ανασκόπηση της βιβλιογραφίας που αφορά στα ερευνητικά / πειραματικά συστήματα διαχείρισης ροών δεδομένων και η καταγραφή των κυριοτέρων χαρακτηριστικών αυτών των συστημάτων καθώς και των ανοιχτών πεδίων έρευνας.
2. Η συνοπτική παρουσίαση σύγχρονων εμπορικών συστημάτων διαχείρισης ροών δεδομένων και η ανάδειξη του τρόπου με τον οποίο κληρονόμησαν / υλοποίησαν τα συστήματα αυτά, χαρακτηριστικά από τα υπάρχοντα ερευνητικά συστήματα.
3. Η υλοποίηση μιας εφαρμογής διαχείρισης ροών δεδομένων με σκοπό την επίδειξη χαρακτηριστικών και δυνατοτήτων λειτουργίας ενός σύγχρονου, εμπορικού συστήματος διαχείρισης ροών δεδομένων.

1.3. **Δομή της εργασίας**

Ο τόμος αυτός οργανώνεται ως εξής:

- Το πρώτο κεφάλαιο αναφέρεται στο σκοπό και στους στόχους της εργασίας ενώ παράλληλα αποτελεί μια εισαγωγή στο αντικείμενο των ροών δεδομένων. Επίσης, παρουσιάζει τη δομή της εργασίας.
- Το δεύτερο κεφάλαιο αποτελεί μια έρευνα και παρουσίαση των στοιχείων εκείνων της υπάρχουσας βιβλιογραφίας που αναφέρονται στα δομικά στοιχεία και τη λειτουργικότητα των Συστημάτων Διαχείρισης Ρευμάτων Δεδομένων. Παρουσιάζει τα βασικά χαρακτηριστικά των κυριοτέρων εξ αυτών και όπου αυτό είναι εφικτό, επιχειρείται και σύγκριση με τα «κλασικά» Σχεσιακά Συστήματα Βάσεων Δεδομένων, με σκοπό να καταγραφούν οι διαφορές αλλά και οι όποιες ομοιότητες μεταξύ τους. Επιπλέον επιχειρείται μια καταγραφή θεμάτων και προβλημάτων τα οποία αποτελούν ή θα μπορούσαν να αποτελέσουν βάση για περαιτέρω έρευνα.
- Το τρίτο κεφάλαιο παρουσιάζει τη μεθοδολογία υλοποίησης. Αναφέρεται στις παραδοχές που έγιναν κατά την υλοποίηση και αναλύει τον τρόπο με τον οποίο σχεδιάσθηκαν και υλοποιήθηκαν οι προσομοιωτές δεδομένων εισόδου.
- Το τέταρτο κεφάλαιο αναφέρεται στην υλοποίηση της εφαρμογής η οποία έχει σκοπό να επιδείξει τις δυνατότητες που προσφέρει ένα σύγχρονο εμπορικό σύστημα διαχείρισης ροών δεδομένων. Το σύστημα που επιλέχθηκε για την επίδειξη είναι το StreamBase το οποίο είναι βασισμένο πάνω σε πρότυπα ανοιχτής αρχιτεκτονικής και ανοιχτού κώδικα (open source) όπως PostgresQL και Eclipse και αποτελεί μια πολλά υποσχόμενη υλοποίηση ΣΔΡΔ. Το κεφάλαιο αυτό, καταρχήν παρουσιάζει την πρόταση υλοποίησης και εν συνεχεία, τα βασικά στοιχεία της αρχιτεκτονικής και της πλατφόρμας ανάπτυξης εφαρμογών του StreamBase, προχωρά με την ανάλυση και τις υποθέσεις εργασίας της εφαρμογής και εν συνεχεία παρουσιάζει την πορεία υλοποίησης μέσα από το γραφικό περιβάλλον StreamBase Studio. Τέλος, παρουσιάζει τον τρόπο τροφοδοσίας, εκτέλεσης και παραγωγής αποτελεσμάτων της υλοποιημένης εφαρμογής.
- Το πέμπτο κεφάλαιο καταγράφει και παρουσιάζει τα εμπειρικά δεδομένα και αποτελέσματα της εκτέλεσης της εφαρμογής.
- Το έκτο κεφάλαιο ανακεφαλαιώνει τα βασικότερα σημεία της εργασίας και καταγράφει τα σημαντικότερα συμπεράσματα που προκύπτουν.
- Το έβδομο κεφάλαιο αποτελεί μια λίστα με τις βιβλιογραφικές αναφορές στις οποίες στηρίχθηκε η συγγραφή του τόμου.
- Ακολουθεί η παράθεση του λεξιλογίου με τους ελληνικούς και τους αντίστοιχους αγγλικούς όρους, όπως αυτοί χρησιμοποιήθηκαν στο κείμενο, καθώς και ο πίνακας ελληνικών και αγγλικών συντμήσεων.
- Ο τόμος τελειώνει με το παράρτημα στο οποίο περιέχονται οι εντολές διαχείρισης ρευμάτων δεδομένων στα πλαίσια της υλοποίησης εφαρμογής

Στο σχήμα που ακολουθεί, απεικονίζεται σχηματικά η δομή της εργασίας.

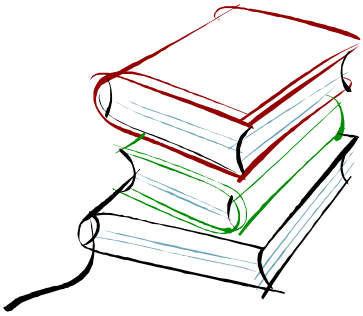


Εικόνα 1-1: Δομή της εργασίας

2. Θεωρητικό υπόβαθρο

2.1. Εισαγωγή

Το μοντέλο επεξεργασίας δεδομένων και ειδικότερα τα μοντέλα ορισμού και εκτέλεσης ερωτημάτων (query execution models) των παραδοσιακών συστημάτων διαχείρισης βάσεων δεδομένων αδυνατούν να ανταποκριθούν στις απαιτήσεις των εφαρμογών ρευμάτων δεδομένων. Ας δούμε όμως τι σημαίνει και που μπορεί να οφείλεται αυτό:



Χαρακτηριστικό των παραδοσιακών ΣΔΒΔ είναι ότι τα ερωτήματα διατυπώνονται μια φορά (ερωτήματα στιγμιότυπου, one-time queries) σε υπάρχοντα δεδομένα, που σημαίνει ότι αυτά βρίσκονται σε κάποιο αποθηκευτικό μέσο και μάλιστα ότι αποτιμούνται με βάση συγκεκριμένο στιγμιότυπο (snapshot) των δεδομένων. Αυτό σημαίνει ότι αν κατά τη διάρκεια εκτέλεσης του ερωτήματος τα δεδομένα τροποποιηθούν, το αποτέλεσμα του ερωτήματος δε θα αλλάξει αλλά θα αντανακλά την εικόνα που τα δεδομένα είχαν κάποια συγκεκριμένη χρονική στιγμή (point in time evaluation). Τα αποτελέσματα των ερωτημάτων αυτών πρέπει να αποθηκευτούν για περαιτέρω επεξεργασία. Το μοντέλο αυτό διαχείρισης δεδομένων ονομάζεται «pull model» μιας και οι χρήστες «τραβάνε» στοιχεία από τα αποθηκευμένα δεδομένα.

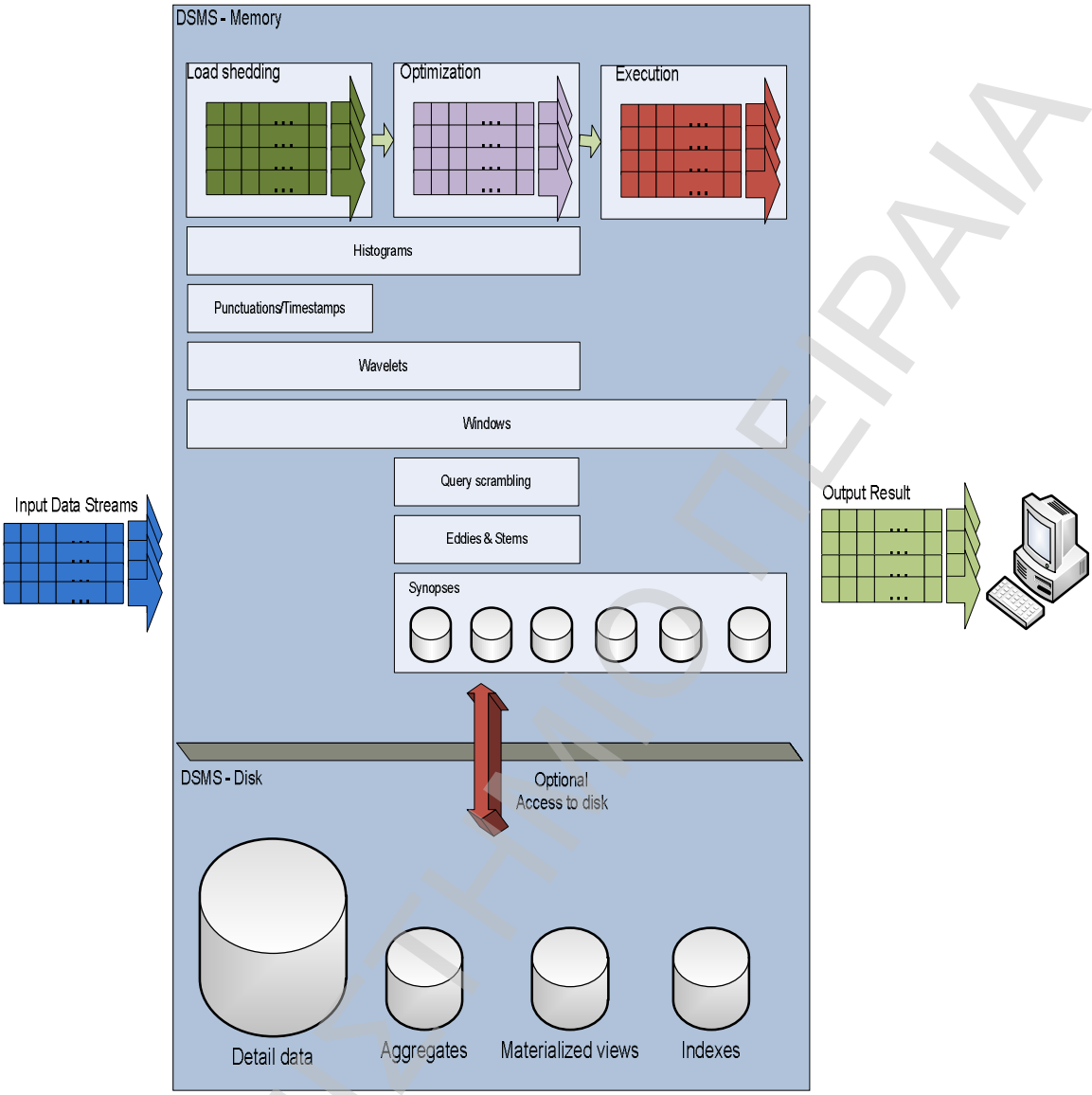
Σε αντίθεση με τα παραπάνω, τα ερωτήματα διαρκείας (continues queries) που υποστηρίζονται από τα συστήματα διαχείρισης ροής δεδομένων (ΣΔΡΔ) αφορούν, ως επί το πλείστον, δυναμικά δεδομένα που μπορεί να μην είναι καν αποθηκευμένα σε κάποιο μέσο αλλά να προέρχονται από πηγές με την ευρεία έννοια, όπως δρομολογητές δικτύου, μετεωρολογικοί αισθητήρες κτλ. Τα ερωτήματα διαρκείας μπορούν να είναι είτε προκαθορισμένα (predefined queries) , είτε περιστασιακά (ad-hoc queries). Τα προκαθορισμένα ερωτήματα ορίζονται και εκτελούνται πριν την άφιξη των δεδομένων στο ΣΔΡΔ ενώ τα τα δυναμικά ερωτήματα μπορούν να ορίζονται και εκτελούνται πολλές φορές κατά τη διάρκεια εκπομπής της ροής. Στο νέο μοντέλο, τα δεδομένα «σπρώχνουν» (push model) απαντήσεις στον χρήστη με βάση τα ερωτήματα διαρκείας που ο ίδιος υπέβαλε. Τα ερωτήματα διαρκείας μπορεί ουσιαστικά να μην έχουν χρονικό τέλος, από τη στιγμή που μια πηγή μπορεί να «εκπέμπει» δεδομένα επ' αορίστω. Μάλιστα, τα δεδομένα αυτά μπορεί να αλλάζουν ρυθμό, όγκο κάθε χρονική στιγμή μιας και ο χρήστης του συστήματος δεν έχει κανέναν έλεγχο πάνω σε αυτά.

Είναι δεδομένο, ότι για χαμηλούς ρυθμούς άφιξης στοιχείων υπό την μορφή ρεύματος, τα κλασικά ΣΔΒΔ θα αντιμετώπιζαν το πρόβλημα με την συντήρηση υλοποιημένων όψεων (materialized views) και επεξεργασία σε περιόδους χαμηλού φόρτου εργασίας. Ο χαμηλός ρυθμός άφιξης αποτελεί σημαντικό αλλά απαραίτητο περιορισμό μιας και η συντήρηση των ενημερωμένων όψεων είναι χρονοβόρα.

Αρκετές όμως εφαρμογές παρουσιάζουν απαιτήσεις άμεσης απόκρισης και επεξεργασίας σε πραγματικό χρόνο. Σε τέτοιες εφαρμογές η λύση των ενημερωμένων όψεων δε θα έδινε λύση αλλά μάλλον, θα δημιουργούσε επιπλέον προβλήματα, ειδικά σε περιόδους φόρτου εργασίας. Ένα άλλο μειονέκτημα των υλοποιημένων όψεων είναι ότι απαιτούν την

ύπαρξη των δεδομένων σε αποθηκευτικό μέσο, κάτι που αυξάνει το χρόνο απόκρισης αλλά και τους όγκους δεδομένων.

Το γεγονός αυτό αλλάζει τις προδιαγραφόμενες απαιτήσεις ακρίβειας των απαντήσεων. Οι απαντήσεις δεν θα πρέπει να εξαρτώνται από το σύνολο των στοιχείων του ρεύματος, λόγω του κόστους αποθήκευσης και επεξεργασίας που δημιουργείται, έστω και αν αυτό προκαλέσει αρνητικές επιπτώσεις στην ακρίβεια των αποτελεσμάτων. Η εξάρτηση από κάποιο είδος συνόψεων (synopses) των παλαιοτέρων στοιχείων κρίνεται πολλές φορές αναγκαία, με στόχο την βελτίωση της προτεινόμενης προσέγγισης. Επιπρόσθετα, η απαίτηση online επεξεργασίας προσανατολίζει τον σχεδιασμό του συστήματος στον κατά το δυνατό περιορισμό του στην κύρια μνήμη, κάτι το οποίο δεν αποτελεί ούτε στόχο ούτε έμφυτο χαρακτηριστικό των κλασικών ΣΔΒΔ. Το σχήμα 2-1 παρουσιάζει την αρχιτεκτονική υψηλού επιπέδου ενός ΣΔΡΔ. Οι λειτουργίες και οι δομές διακρίνονται στις περιοχές μνήμης και πρόσβασης σε δίσκο. Διακρίνονται επίσης και οι διάφορες λειτουργίες και μηχανισμοί ανάλογα με το στάδιο επεξεργασίας. Π.χ. βλέπουμε ότι κατά την είσοδο των ρευμάτων στο σύστημα, εφαρμόζεται η λειτουργία της μείωσης του όγκου τους (load shedding), στη διάρκεια της οποίας χρησιμοποιούνται τα ιστογράμματα και τα χρονόσημα. Ακόμη, βλέπουμε ότι η πρόσβαση στο δίσκο είναι προαιρετική. Το σχήμα παρουσιάζει συνοπτικά τα βασικότερα σημεία στα οποία θα αναφερθούμε στα επόμενα κεφάλαια.



Εικόνα 2-1: Διάγραμμα αρχιτεκτονικής ΣΔΡΔ υψηλού επιπέδου

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

Οι διαφοροποιήσεις των ΣΔΡΔ και των ΣΔΒΔ δεν τελειώνουν εδώ. Απεναντίας, ο κατάλογος είναι μακρύς μιας και ουσιαστικά ο σκοπός δημιουργίας των ΣΔΡΔ αποκλίνει από εκείνο των παραδοσιακών ΣΔΒΔ. Ο παρακάτω πίνακας παρουσιάζει συνοπτικά τις κυριότερες διαφορές των δύο αυτών κατηγοριών συστημάτων.

Συστήματα Διαχείρισης Βάσεων Δεδομένων	Συστήματα Διαχείρισης Ροών Δεδομένων
Δεδομένα που διέπονται από κανόνες και συσχετίσεις μεταξύ τους	Τυχαία ρεύματα δεδομένων
Ερωτήματα που ορίζονται μια φορά και εφαρμόζονται στη συνέχεια	Ερωτήματα διαρκείας. Αυτά μπορεί να είναι είτε προκαθορισμένα (predefined) είτε περιστασιακά (ad-hoc) και μπορεί να εφαρμόζονται είτε πριν είτε κατά τη διάρκεια άφιξης των ρευμάτων δεδομένων
Δεδομένα τυχαίας προσπέλασης. Το σύστημα μπορεί να τροφοδοτείται και να χειρίζεται δεδομένα διαφορετικών πηγών	Σειριακή πρόσβαση δεδομένων. Τα δεδομένα καταφτάνουν σειριακά από συγκεκριμένες πηγές-πομπούς
Απεριόριστες χωρητικότητες δίσκων	Τα δεδομένα οριοθετούνται από το μέγεθος της κύριας μνήμης του συστήματος
Συνήθως η χρονική σήμανση των δεδομένων δε θεωρείται κρίσιμη	Η ιστορικότητα / χρόνος άφιξης θεωρούνται κρίσιμα
Τα ιστορικά δεδομένα πολλές φορές είναι εξίσου σημαντικά με τα νεότερα δεδομένα	Μεγαλύτερο βάρος δίνεται στα πιο πρόσφατα δεδομένα
Τα δεδομένα βρίσκονται ήδη αποθηκευμένα (Στατικά)	Τα δεδομένα αλλάζουν δυναμικά κάθε χρονική στιγμή (Δυναμικά)
Σχετικά χαμηλό ποσοστό αλλαγών στα δεδομένα	Ο ρυθμός άφιξης μπορεί να είναι τα τάξης των αρκετών Gigabytes ανά μονάδα χρόνου. Τα δεδομένα αυτά μπορεί να αλλάζουν συνεχώς τιμές συγκριμένων δεικτών, ποσοτήτων κτλ
Δεν απαιτείται η προσφορά υπηρεσιών σε πραγματικό χρόνο	Οι εφαρμογές έχουν απαιτήσεις πραγματικού χρόνου
Τα αποτελέσματα των ερωτημάτων θεωρούνται ακριβή με βάση τα πηγαία δεδομένα	Τα δεδομένα μπορεί να μην είναι σαφή ή ολοκληρωμένα, να περιέχουν «θόρυβο», ή να έχουν ήδη επικαλυφθεί από άλλα νεότερα
Το πλάνο εκτέλεσης καθορίζεται από στοιχεία όπως τη φυσική σχεδίαση της βάσης δεδομένων, τον βελτιστοποιητή (optimizer) κτλ.	Το πλάνο εκτέλεσης πρέπει να λαμβάνει υπ' όψιν του την απρόβλεπτη και μεταβλητή άφιξη δεδομένων αλλά και χαρακτηριστικών, έτσι ώστε να διαμορφώνει και να παράγει πλάνα εκτέλεσης, ικανά για να αντιμετωπίσουν τη μεταβλητότητα των δεδομένων.

Πίνακας 2-1: Σύγκριση ΣΔΡΔ και ΣΔΒΔ

Για να καλυφθούν οι νέες απαιτήσεις υπάρχουν δύο προσεγγίσεις:

- Η πρώτη, είναι ο σχεδιασμός ΣΔΡΔ με προσπάθεια προσαρμογής των κλασικών ΣΔΒΔ στις νέες απαιτήσεις. Αυτό προϋποθέτει αλλαγές τόσο σε δομή όσο και σε χαρακτηριστικά των συστημάτων αυτών, κάτι που πολλές φορές είναι δύσκολο.
- Η δεύτερη, είναι ο σχεδιασμός από την αρχή, νέων συστημάτων, ικανών να ανταποκριθούν στις ιδιαίτερες απαιτήσεις των εφαρμογών ρευμάτων δεδομένων.

Στις επόμενες παραγράφους περιγράφονται τα δομικά στοιχεία και χαρακτηριστικά των ΣΔΡΔ καθώς και τα διάφορα θέματα που προκύπτουν από τις απαιτήσεις των εφαρμογών και τον τρόπο αντιμετώπισης που προτείνουν τα διάφορα ΣΔΡΔ.

2.2. **Διαχείριση μνήμης**

Τα ρεύματα δεδομένων υποθετικά αλλά και πρακτικά μπορούν να είναι συνεχή και απεριόριστα, που σημαίνει ότι ο όγκος της πληροφορίας που ένα ΣΔΡΔ θα πρέπει να διαχειριστεί, θα είναι και αυτός απεριόριστος. Αυτό σημαίνει με τη σειρά του ότι και οι απαιτήσεις σε μνήμη και αποθηκευτικό χώρο θα πρέπει να είναι απεριόριστες. Κάτι τέτοιο είναι προφανές ότι δεν μπορεί να συμβεί. Έτσι, κατά καιρούς είχαν προταθεί αλγόριθμοι οι οποίοι εφαρμόζονταν σε σχεσιακές βάσεις δεδομένων και οι οποίοι μπορούσαν να διαχειριστούν δεδομένα τόσο στη μνήμη όσο και αποθηκευμένα σε σκληρούς δίσκους. Το πρόβλημα με αυτούς αλγορίθμους ήταν ότι δεν μπορούσαν να υποστηρίξουν εγγενώς ρεύματα δεδομένων ή ότι οι χρόνοι απόκρισης ήταν απαγορευτικοί για εφαρμογές πραγματικού χρόνου.

Το μοντέλο διαχείρισης δεδομένων που παρουσιάζεται στα ρεύματα δεδομένων βρίσκει εφαρμογή σε προβλήματα όπου οι απαντήσεις στα ερωτήματα πρέπει να δίνονται σε πραγματικό χρόνο και όπου υψηλοί όγκοι δεδομένων χρειάζεται να επεξεργαστούν ανά μονάδα χρόνου. Τα ΣΔΡΔ δέχονται νέα δεδομένα εν όσω παλαιότερα δεδομένα επεξεργάζονται. Για αυτό, ο χρόνος επεξεργασίας για κάθε «πακέτο» δεδομένων θα πρέπει να είναι μικρός αλλιώς ο αλγόριθμος δεν θα μπορεί να αντεπεξέλθει στον ρυθμό άφιξης δεδομένων του ρεύματος.

Τα πεδία έρευνας στον συγκεκριμένο τομέα προσανατολίζονται σε αλγορίθμους οι οποίοι δεν απαιτούν προσπέλαση αποθηκευτικών χώρων, αλλά χρησιμοποιούν αποκλειστικά, ή στο μεγαλύτερο δυνατό μέρος, μόνο τη μνήμη του συστήματος. Το πρόβλημα εύρεσης του μεγέθους μνήμης η οποία απαιτείται για να υπολογιστεί ένα ερώτημα διάρκειας είναι αρκετά δύσκολο και αυτή τη στιγμή η έρευνα δείχνει ότι εάν δεν ξέρουμε τον όγκο των δεδομένων, είναι αδύνατον να θέσουμε ένα όριο στις απαιτήσεις μνήμης, ειδικά σε ερωτήματα τα οποία είναι σύνθετα. Έτσι πρέπει να προχωρήσουμε σε συμβιβαστικές λύσεις οι οποίες παράγουν προσεγγιστικές απαντήσεις. Οι λύσεις αυτές όμως, θα πρέπει να πληρούν κάποιες ελάχιστες προϋποθέσεις ποιότητας των προσφερομένων υπηρεσιών.

2.3. **Προσεγγιστική απάντηση σε ερωτήματα**

Όπως αναφέραμε προηγουμένως όταν υπάρχει περιορισμός στη μνήμη, δεν είναι πάντα δυνατή η παραγωγή αποτελεσμάτων με ακρίβεια. Ένα ερώτημα διαρκείας πάντως πολλές φορές είναι αποδεκτό να επιστρέφει απαντήσεις με αρκετά καλή ακρίβεια και όχι με απόλυτη ακρίβεια, ειδικά όταν αυτό συνεισφέρει στην αποδοτικότητα του συστήματος.

Ο τομέας έρευνας προσεγγιστικών αλγορίθμων για την παραγωγή απαντήσεων με όσο το δυνατόν μεγαλύτερη ακρίβεια και μικρότερες επιπτώσεις στην ποιότητα ων αποτελεσμάτων του συστήματος, έχει παράγει μέχρι στιγμής τεχνικές που σκοπό έχουν, καταρχήν τη μείωση των προς επεξεργασία δεδομένων με συγκεκριμένες τεχνικές και εν συνεχεία, τη δημιουργία και χρήση συνόψεων. Για τη μείωση των δεδομένων εφαρμόζονται οι παρακάτω τεχνικές:

- φιλτροποίηση ακριβείας (Precise filtering)
- συγχώνευση δεδομένων (Data merging)
- απόρριψη δεδομένων (Data Dropping)

ενώ για τη δημιουργία και χρήση συνόψεων χρησιμοποιούνται τεχνικές όπως:

- σκίτσα (sketches),
- δειγματοληψία (sampling),
- ιστογράμματα (histograms)
- κυματίδια (wavelets),

Όλες οι παραπάνω τεχνικές αναλύονται σε επόμενα κεφάλαια.

2.4. **Μείωση όγκου δεδομένων**

Στις επόμενες παραγράφους αναλύονται οι τεχνικές φιλτροποίησης ακριβείας (Precise filtering), συγχώνευσης δεδομένων (Data merging) και απόρριψης δεδομένων (Data Dropping). Παρουσιάζονται τα προτερήματα και μειονεκτήματα της κάθε προσέγγισης αλλά και οι τάσεις έρευνας στον τομέα αυτό.

2.4.1. **Φιλτροποίηση ακριβείας** (Precise filtering)

Η φιλτροποίηση ακριβείας έχει ως σκοπό την απόρριψη κάποιων εκ των δεδομένων του ρεύματος, ειδικά κοντά στην πηγή των δεδομένων, με την προσδοκία τη μείωση του όγκου των δεδομένων χωρίς όμως αλλοίωση των τελικών απαντήσεων. Τα φίλτρα ακριβείας γενικά πρέπει να είναι απλά όπως ο τελεστής της επιλογής (select), ή της προβολής (projection) και να μπορούν να εφαρμοστούν σε κάθε πλειάδα του ρεύματος, έτσι ώστε να έχουν μικρές απαιτήσεις σε μνήμη και επεξεργαστική ισχύ. Ειδικότερα, θα πρέπει να μην εισάγουν την παραμικρή καθυστέρηση σε εφαρμογές πραγματικού χρόνου. Η εφαρμογή των φίλτρων μπορεί να γίνει είτε στην πηγή των δεδομένων (π.χ. αισθητήρες ή δρομολογητές), μέσα στο ΣΔΡΔ, ή κατά τη διάρκεια που η ροή ταξιδεύει στο δίκτυο (π.χ. ενδιάμεσοι δρομολογητές και συστήματα που μπορεί ένα κομμάτι της εφαρμογής να είναι εγκατεστημένο, ειδικά για την εφαρμογή τέτοιων φίλτρων).

Για την απόρριψη των δεδομένων χρησιμοποιούνται συγκεκριμένοι κανόνες οι οποίοι είναι ορισμένοι με σαφήνεια και αποτελούν μέρος του συστήματος. Για παράδειγμα, σε ένα σύστημα δημοπρασιών πραγματικού χρόνου, ένα φίλτρο ακριβείας που θα μπορούσε να εφαρμοστεί θα ήταν η απόρριψη όλων των μικρότερων ή ίσων προσφορών από τη μεγαλύτερη προσφορά που έχει γίνει μέχρι στιγμής. Αν το φίλτρο εφαρμοζόταν στην πηγή του ρεύματος, τότε η μείωση θα ήταν μικρότερη μιας και κάθε πηγή γνωρίζει μόνο τα δεδομένα που η ίδια έχει εκπέμψει μέχρι στιγμής, άρα θα μπορούσε να αποκλείσει δεδομένα σε σύγκριση μόνο με τα δικά της. Αν όμως η φιλτροποίηση γινόταν μέσα στο ΣΔΡΔ, τότε η μείωση θα εφαρμοζόταν στα δεδομένα όλων των πηγών συνολικά.

Αυτό βέβαια συνεπάγεται την αύξηση σε απαιτήσεις των πόρων του συστήματος για την εφαρμογή του φίλτρου. Η βέλτιστη λύση θα ήταν κάπου ενδιάμεσα, όπου κάθε πηγή εφαρμόζει το φίλτρο, πιθανά υπάρχουν και ενδιάμεσα στο δίκτυο στάδια εφαρμογής και το ΣΔΡΔ κάνει την τελική εφαρμογή στα εναπομείναντα/συγκεντρωτικά δεδομένα των ρευμάτων. Ένα σύστημα το οποίο κάνει χρήση τέτοιου είδους φίλτρων είναι το Gigascope το οποίο όπως θα δούμε στη συνέχεια, έχει υλοποιηθεί με σκοπό την παρακολούθηση και έλεγχο δικτύων δεδομένων.

2.4.2. **Συγχώνευση δεδομένων** (Data merging)

Η τεχνική αυτή προσπαθεί να συρρικνώσει τα δεδομένα του ρεύματος με τέτοιο τρόπο ώστε το αρχικό ερώτημα και πάλι να μπορεί να απαντηθεί με αξιοπιστία. Για παράδειγμα μια ροή που θα ελέγχει τους πέντε πιο ενεργούς κόμβους ενός δικτύου θα μπορούσε να χρησιμοποιήσει την τεχνική αυτή, δημιουργώντας και συντηρώντας μια είδους συνάθροιση «aggregation» δεδομένων σε κάθε κόμβο ή σε κάθε δρομολογητή του δικτύου, η οποία θα μετρούσε τα πακέτα που έχουν αποσταλεί ανά IP διεύθυνση του κάθε κόμβου.

Αυτό βεβαίως θα εξυπηρετούσε τις ανάγκες του εν λόγω ερωτήματος, αλλά από την άλλη πλευρά θα απαγόρευε ερωτήματα σε χαμηλότερο επίπεδο (IP, packet), άρα δεν θα

μπορούσαν να λειτουργήσουν εφαρμογές όπως ανίχνευση επιθέσεων, η οποία προϋποθέτει την εξέταση του κάθε πακέτου χωριστά, για ύποπτα δεδομένα. Βέβαια, η παραπάνω τεχνική είναι ευρέως διαδεδομένη σε εμπορικά συστήματα για την παραγωγή στατιστικών στοιχείων λειτουργίας του δικτύου και ενσωματωμένη σε δρομολογητές όπως αυτούς της Cisco (Netflow system).

2.4.3. **Απόρριψη δεδομένων** (Data Dropping / Load shedding)

Η τεχνική της απόρριψης δεδομένων εφαρμόζεται σε περιπτώσεις εφαρμογών ΣΔΡΔ που απαιτούν υψηλούς όγκους και ρυθμούς τροφοδοσίας δεδομένων, είτε απορρίπτοντας πλειάδες από το ρεύμα εισόδου, είτε μειώνοντας την επεξεργασία που απαιτείται για τις ήδη επιλεγμένες πλειάδες. Στην αρχική της μορφή η τεχνική αυτή έχει τη μορφή μιας άναρχης, τυχαίας επιλογής πλειάδων οι οποίες απορρίπτονται από το σύστημα χωρίς να έχει προηγηθεί κάποια επεξεργασία, π.χ. τα δεδομένα απορρίπτονται όταν έχουμε υπερχείλιση κάποιου προσωρινού χώρου μνήμης (buffer). Η τεχνική αυτή ονομάζεται τυφλή απόρριψη (blind dropping).

Πιο εξελιγμένα σχήματα και τεχνικές εισάγουν κριτήρια τα οποία διακρίνουν ποιες από τις πλειάδες θα πρέπει να απορριφθούν. Οι τεχνικές αυτές είναι βασισμένες για παράδειγμα, στις επιπτώσεις που θα είχε η εφαρμογή τους στην ακρίβεια της απάντησης του συγκεκριμένου ερωτήματος. Άλλες τεχνικές βασίζονται στην προσπάθεια για να εξαχθεί ένα αντιπροσωπευτικό δείγμα από το ρεύμα δεδομένων, απορρίπτοντας τις λιγότερο κατάλληλες πλειάδες.

Είπαμε προηγουμένως, ότι στην απλή μορφή της η τεχνική της απόρριψης δεδομένων γίνεται χωρίς να προηγηθεί κάποιου είδους ανάλυση στα δεδομένα. Άλλες υλοποιήσεις της τεχνικής αυτής, λαμβάνουν υπόψη τους διαθέσιμους πόρους του συστήματος και απορρίπτουν δεδομένα όταν για παράδειγμα η επεξεργαστική ισχύς ή η διαθέσιμη μνήμη πέσουν κάτω από κάποια κρίσιμα όρια. Κάποιες πλειάδες τότε απορρίπτονται τυχαία έως ότου ο ρυθμός επεξεργασίας του ρεύματος δεδομένων επανέλθει σε φυσιολογικά επίπεδα. Είναι φανερό ότι η τεχνική αυτή μπορεί να επιφέρει αλλοιώσεις των δεδομένων που εισάγονται στο σύστημα άρα και αλλοιώσεις στις απαντήσεις στα διάφορα ερωτήματα. Βελτιωμένες εκδοχές αυτής της τεχνικής προσπαθούν να απορρίπτουν πλειάδες βασισμένες σε απλούς κανόνες όπως για παράδειγμα ότι θα γίνεται απόρριψη μιας ανά είκοσι πλειάδες, που όμως και πάλι δεν εγγυάται ένα σταθερό ρυθμό εισόδου δεδομένων στο σύστημα, ή την απόρριψη όλων των πλειάδων οι οποίες ξεπερνούν το ανώτατο όριο ρυθμού επεξεργασίας, π.χ. το σύστημα θα επεξεργάζεται 30 πλειάδες ανά λεπτό, άρα όλες οι υπόλοιπες θα απορρίπτονται.

Η απόρριψη μπορεί να γίνει είτε στην αρχή του ρεύματος είτε σε ενδιάμεσα σημεία, είτε κατά τη διάρκεια εφαρμογής του πλάνου εκτέλεσης του συστήματος. Το ΣΔΡΔ Aurora χρησιμοποιεί έναν ειδικό τελεστή απόρριψης (drop operator) ο οποίος μπορεί να παρεμβληθεί σε ένα η περισσότερα σημεία μέσα στο πλάνο εκτέλεσης. Ο τελεστής μπορεί να απορρίπτει πλειάδες είτε τυχαία, είτε βασιζόμενος πάνω σε κάποιους κανόνες. Στην περίπτωση αυτή μιλάμε για σημασιολογική απόρριψη (semantic dropping). Μια άλλη προσέγγιση, είναι η απόρριψη πλειάδων με τη μετατροπή υπαρχόντων τελεστών, έτσι ώστε να επιτευχθεί μείωση του όγκου των δεδομένων που καλείται να χειριστεί ο τελεστής, άρα και το ερώτημα διαρκείας. Για παράδειγμα, είναι δυνατή η αλλαγή της συμπεριφοράς του τελεστή «σύνδεσης» (join operator), έτσι ώστε είτε να απορρίπτει δεδομένα τυχαία, είτε να μη λαμβάνει υπ' όψιν του κάποιες πλειάδες κατά τη διάρκεια εκτέλεσης της σύνδεσης.

Όποια τεχνική και αν ακολουθηθεί, είναι δεδομένο ότι θα πρέπει να ορίζονται κανόνες οι οποίοι θα εξασφαλίζουν τη βέλτιστη συμπεριφορά του συστήματος δεδομένης μιας συγκεκριμένης απαίτησης ρυθμού απόρριψης. Το σύστημα Aurora για παράδειγμα, έχει τη

δυνατότητα υπολογισμού όλων των πιθανών σημείων παρεμβολής τελεστών απόρριψης και χρησιμοποιώντας ευριστικούς αλγορίθμους (heuristics) μπορεί να παράγει μια λίστα (load-shedding road map LSRM) εναλλακτικών πλάνων εκτέλεσης του ερωτήματος διαρκείας. Αφού τροφοδοτηθούν με ικανό όγκο δεδομένων οι ευριστικοί αλγόριθμοι, η λίστα αυτή εξασφαλίζει την ποιότητα παραγωγής αποτελεσμάτων για το συγκεκριμένο ερώτημα, επιλέγοντας το πλάνο εκείνο που απορρίπτει τα λιγότερο κατάλληλα δεδομένα σε σχέση όχι μόνο με το ερώτημα αλλά και με τη διαθεσιμότητα των πόρων του συστήματος.

Η έρευνα στο τομέα αυτό είναι έντονη και επικεντρώνεται κυρίως στις τεχνικές εκείνες οι οποίες θα απορρίπτουν δεδομένα βασισμένες σε πιο εξελιγμένες τεχνικές. Ας πάρουμε για παράδειγμα την περίπτωση του τελεστή σύνδεσης που προαναφέραμε. Ο τελεστής αυτός είπαμε ότι απορρίπτει πλειάδες με τυχαίο τρόπο. Αυτό σημαίνει ότι μεταξύ δύο συνόλων Α και Β από πλειάδες, θα ήταν πιθανό να απορριφτεί κάποια η οποία θα εμφανιζόταν μια φορά στο σύνολο Α, αλλά η οποία θα μπορούσε να παράγει πολλές πλειάδες μετά την ένωση με το σύνολο Β (περίπτωση σχέσης master-detail μεταξύ πλειάδων). Αυτό θα σήμαινε αλλοίωση του αποτελέσματος με πιθανόν σοβαρές συνέπειες. θα μπορούσε να εφαρμοστεί λοιπόν κάποιος αλγόριθμος ο οποίος, θα επέλεγε για απόρριψη τις πλειάδες εκείνες από το σύνολο Α, που θα παρήγαγαν τις λιγότερες πλειάδες μετά την ένωση τους με το σύνολο Β. Τέλος, η προσοχή των ερευνητών δίνεται στις απαιτήσεις σε επεξεργαστική ισχύ και μνήμη για την εφαρμογή αυτών των τεχνικών μιας και θα πρέπει να επηρεάζουν το δυνατό λιγότερο το ρυθμό απόδοσης του συστήματος.

Κλείνοντας την αναφορά μας στις τεχνικές απόρριψης δεδομένων, θα πρέπει να τονίσουμε ότι η προσέγγιση υλοποίησης θα πρέπει να λαμβάνει υπόψη της ότι τα ΣΔΡΔ πιθανόν - και μάλλον αυτό είναι το πλέον πιθανό σενάριο - να είναι δέκτες πολλών ταυτόχρονων ερωτημάτων διαρκείας. Αυτό σημαίνει ότι οι τεχνικές που προαναφέραμε θα πρέπει να έχουν τη δυνατότητα να εξασφαλίζουν την ποιότητα των παρεχόμενων υπηρεσιών σε όλους τους χρήστες, δηλαδή να εξασφαλίζουν ότι δε θα απορρίπτονται δεδομένα τα οποία δεν είναι κρίσιμα για ένα ερώτημα αλλά είναι απαραίτητα για κάποιο άλλο.

2.5. **Προσαρμοστικότητα ΣΔΡΔ**

Ένα ΣΔΡΔ θα πρέπει να είναι σε θέση να προσαρμόζεται ανάλογα με τη δυναμική των δεδομένων του ρεύματος. Έτσι, θα πρέπει να είναι σε θέση να αλλάζει το πλάνο εκτέλεσης ανάλογα με τον εκτιμώμενο χρόνο εκτέλεσης του ερωτήματος διαρκείας, ανάλογα με την επιλεκτικότητα (selectivity) των πλειάδων αλλά και ανάλογα με το ρυθμό άφιξης των πλειάδων στο σύστημα έτσι ώστε να είναι δυνατή κάθε φορά η παραγωγή του βέλτιστου πλάνου εκτέλεσης του ερωτήματος.

Αρχικά το πλάνο εκτέλεσης θα είναι βασισμένο στο μέσο χρόνο υπολογισμού εκτέλεσης του ερωτήματος. Η συνέχεια επιβάλλει την αλλαγή του πλάνου έτσι ώστε αυτό να προσαρμόζεται στα δεδομένα που καταφτάνουν και μάλιστα με τέτοιο τρόπο ώστε να ευνοούνται οι γρήγοροι τελεστές και εν συνεχεία, εάν αυτό υποστηρίζεται από το σύστημα, η δρομολόγηση σε άλλους κόμβους, τελεστών οι οποίοι απαιτούν χρόνο για τον υπολογισμό τους. Ένα παράδειγμα ΣΔΡΔ που εφαρμόζει τέτοιου είδους τεχνικές όπως θα δούμε και στη συνέχεια, είναι το CAPE (**C**onstraint-aware **A**daptive stream **P**rocessing **E**ngine).

Η ιδέα της προσαρμοστικότητας των σύγχρονων ΣΔΡΔ βασίζεται σε δύο προσεγγίσεις: Η πρώτη αναφέρεται στην αναδιάταξη των ερωτημάτων διαρκείας (query scrambling) και η δεύτερη στην τεχνική των Eddies & Stems. Ας δούμε πως υλοποιείται κάθε μια από τις δύο αυτές τεχνικές:

2.5.1. **Αναδιάταξη ερωτημάτων** (query scrambling)

Η τεχνική της αναδιάταξης των υπό εκτέλεση ερωτημάτων διαρκείας, βρίσκει εφαρμογή σε κατανεμημένα συστήματα διαχείρισης ροών δεδομένων και χρησιμοποιείται για να αλλάξει δυναμικά τον τρόπο εκτέλεσης ενός ερωτήματος, όταν το ρεύμα δεδομένων δεν έχει σταθερή συμπεριφορά ως προς το ρυθμό ή το περιεχόμενο του. Η τεχνική αυτή υλοποιείται με μια προσέγγιση δύο φάσεων:

- Πρώτη φάση: Επαναπρογραμματισμός ερωτήματος (query rescheduling). Η σειρά με την οποία εκτελούνται οι τελεστές του ερωτήματος στους διαφορετικούς κόμβους του κατανεμημένου συστήματος, αλλάζει έτσι ώστε να είναι δυνατή η εκτέλεση των τμημάτων/τελεστών εκείνων του ερωτήματος που σε διαφορετική περίπτωση θα έπρεπε να περιμένουν για να ολοκληρωθεί η εκτέλεση κάποιων άλλων. Αυτή η τεχνική έχει εφαρμογή σε περιπτώσεις οπου π.χ. κάποιος από τους κόμβους του ΣΔΡΔ δεν είναι διαθέσιμος.
- Δεύτερη φάση: Σύνθεση τελεστών (operator synthesis). Στη φάση αυτή το ΣΔΡΔ περνά, όταν έχει αποτύχει η πρώτη φάση. Τώρα, εισάγονται στο ερώτημα νέοι τελεστές και διαγράφονται ήδη υπάρχοντες, έτσι ώστε να αλλάξει δραστικά το πλάνο εκτέλεσης.

Οι δύο αυτές προσεγγίσεις εκτελούνται σειριακά για κάθε ερώτημα και επαναληπτικά για το σύνολο των ερωτημάτων που εκτελούνται στο σύστημα. Αυτό σημαίνει ότι το σύστημα προσπαθεί συνεχώς να προσαρμόσει την συμπεριφορά του, ανάλογα με τα ερωτήματα που εκτελούνται, πρώτα με τον επαναπρογραμματισμό των ερωτημάτων και εν συνεχεία με τη σύνθεση τελεστών.

2.5.2. **Χρήση** Eddies & Stems

Η ιδέα πίσω από το μηχανισμό Eddy είναι ο εκφυλισμός του ζητήματος της βελτιστοποίησης ενός ερωτήματος διαρκείας σε ζήτημα δρομολόγησης δικτύου. Αυτό επιτυγχάνεται με το να μην συνδέονται απευθείας οι τελεστές ενός ερωτήματος με το πλάνο εκτέλεσης, αλλά να υπάρχει ένας κεντρικός μηχανισμός ο οποίος να τροφοδοτεί κάθε πλειάδα ενός ρεύματος στον κατάλληλο τελεστή. Όταν η πλειάδα έχει περάσει διαδοχικά από όλους τους τελεστές, τότε είναι έτοιμη να εξαχθεί από το σύστημα ως πλειάδα του ρεύματος εξόδου. Ο κεντρικός αυτός μηχανισμός ονομάζεται Eddy.

Από την άλλη πλευρά έχουμε το μηχανισμό των SteMs (State Modules) ο οποίος είναι ένα είδος on-the-fly δημιουργίας ευρετηρίων (indexing) του εισερχόμενου ρεύματος δεδομένων. Το ευρετήριο που θα δημιουργηθεί εξαρτάται από την ποιότητα του ρεύματος. Για παράδειγμα, ένα ευρετήριο θα μπορούσε να είναι ευρετήριο κατακερματισμού (hash index), ή ένα B-δέντρο (B-tree). Είναι βέβαια πιθανή η δημιουργία περισσότερων του ενός ευρετηρίων πάνω σε ένα ρεύμα όταν βέβαια αυτό επιβάλλεται. Έτσι, θα μπορούσαμε για παράδειγμα να έχουμε δύο ευρετήρια, σε δύο διαφορετικά χαρακτηριστικά (attributes) του ρεύματος, τα οποία θα μετείχαν σε διαφορετικούς τελεστές: ενα ευρετήριο κατακερματισμού για χρήση σε ένωση και ένα ευρετήριο B-δέντρου για χρήση σε επιλογή. Μάλιστα, όπως και στα παραδοσιακά ΣΔΡΔ, τα ευρετήρια αυτά θα μπορούσαν να χρησιμοποιηθούν από πολλά ερωτήματα διαρκείας ταυτόχρονα.

Ας δούμε τώρα πως συνδυάζονται οι Eddies με τα Stems. Το σύστημα δημιουργεί ένα στιγμιότυπο (instance) του κάθε τελεστή που μετέχει σε ένα ερώτημα. Οι τελεστές επικοινωνούν μεταξύ τους κάθε φορά μέσω του Eddy και ποτέ απευθείας μεταξύ τους. Σε κάθε πλειάδα του ρεύματος προστίθεται μια σειρά από bits (query bitmap) τα οποία αντιστοιχούν ένα για κάθε τελεστή του ερωτήματος. Αρχικά όλα τα bits κάθε πλειάδας έχουν την τιμή μηδέν, ενώ στην έξοδο της πλειάδας από το σύστημα, όλα τα bits έχουν την τιμή ένα. Άρα:

1. Με την άφιξη μιας πλειάδας στον Eddy, προστίθεται το query bitmap και δρομολογείται στο κατάλληλο Stem. Εν συνεχεία η πλειάδα επιστρέφει στον Eddy.
2. Όταν ο Eddy λάβει ξανά την πλειάδα, αποφασίζει σε ποιον από τους τελεστές θα πρέπει ακολούθως να την ανακατευθύνει. Ο τελεστής δέχεται την πλειάδα και την επεξεργάζεται, προσπελαύνοντας το αντίστοιχο Stem. Εάν ο τελεστής αποφασίσει ότι η πλειάδα θα πρέπει να παραμείνει στο σύστημα, θέτει την τιμή του αντίστοιχου bit του query bitmap σε ένα και στέλνει την πλειάδα στον Eddy. Σε αντίθετη περίπτωση, η πλειάδα απορρίπτεται. Ένα παράδειγμα απόρριψης είναι όταν η πλειάδα δεν πληροί τα κριτήρια ενός τελεστή επιλογής.
3. Το προηγούμενο βήμα επαναλαμβάνεται μέχρι η πλειάδα να επεξεργαστεί από όλους τους τελεστές που σημαίνει όλα τα bits να πάρουν την τιμή ένα. Τότε η πλειάδα αποστέλλεται από τον Eddy στο ρεύμα εξόδου.

Η καρδία του Eddy είναι η αρχή πάνω στην οποία βασίζεται για να δρομολογήσει τις πλειάδες από τελεστή σε τελεστή. Οι τελεστές που δέχονται πλειάδες και τις απορρίπτουν είναι αυτοί που το σύστημα θα πρέπει να επιλέγει περισσότερο μιας και μειώνουν το ρεύμα δεδομένων. Εάν η επιλεκτικότητα (selectivity) ενός κατηγορήματος (predicate) του ερωτήματος αλλάξει κατά τη διάρκεια εκτέλεσης, τότε οι συγκεκριμένοι τελεστές θα αρχίσουν να επιστρέφουν πλειάδες στον Eddy συχνότερα, οπότε το σύστημα θα προσαρμοστεί στα νέα δεδομένα με αποτέλεσμα τη δημιουργία εκ νέου ενός διαφορετικού πλάνου εκτέλεσης.

2.6. **Ομαδική επεξεργασία**, **δειγματοληψία και συνόψεις**

Για τη βελτιστοποίηση των προσεγγιστικών απαντήσεων των ερωτημάτων διαρκείας είναι πολλέ φορές αναγκαία η επεξεργασία των πλειάδων του ρεύματος εισόδου ανά ομάδες και όχι ανά πλειάδα. Ας δούμε για ποιο λόγο συμβαίνει αυτό και πως λειτουργεί η συγκεκριμένη τεχνική:

Ας υποθέσουμε ότι για να απαντηθεί ένα ερώτημα διαρκείας απαιτούνται δύο λειτουργίες: Η πρώτη (update(tuple)) αφορά στην ενημέρωση του ΣΔΡΔ με τις πλειάδες του ρεύματος δεδομένων και η δεύτερη (computeAnswer()) αφορά στην καθεαυτή εκτέλεση του ερωτήματος, δηλαδή στην σταδιακή παραγωγή της απάντησης. Κατά τη διάρκεια επεξεργασίας ενός ερωτήματος, το βέλτιστο σενάριο θα ήταν και οι δύο λειτουργίες να ήταν τόσο γρήγορες ώστε να μπορούσαν όλες οι πλειάδες να επεξεργαστούν χωρίς εισαγωγή καθυστέρησης στο σύστημα. Σε μια τέτοια περίπτωση, δε θα χρειάζονταν να καταφύγουμε σε κάποια ιδιαίτερη τεχνική, μιας και όλα τα δεδομένα που θα κατέφταναν στο ΣΔΡΔ θα ενημέρωναν τη δομή των ήδη υπαρχόντων και η απάντηση θα υπολογιζόταν σε χρόνο λιγότερο από το μέσο χρόνο άφιξης-ενημέρωσης-επεξεργασίας της κάθε πλειάδας. Εάν όμως για κάποιο λόγο μια ή και οι δύο εκ των λειτουργιών update(tuple) και computeAnswer() καθυστερούσε, με αποτέλεσμα ο κύκλος άφιξης-ενημέρωσης-επεξεργασίας πλειάδας να ήταν μεγαλύτερος του μέσου χρόνου επεξεργασίας, τότε αυτό θα είχε αντίκτυπο στην απόδοση του συστήματος, αφού θα μειωνόταν σταδιακά, άρα το σύστημα θα έπαυε να λειτουργεί ως σύστημα πραγματικού χρόνου.

2.6.1. **Ομαδική επεξεργασία** (batch processing)

Στην περίπτωση όπου η λειτουργία update(tuple) είναι γρήγορη αλλά η computeAnswer() είναι αργή, η λύση θα ήταν να επεξεργασθούν οι πλειάδες ανά ομάδες. Τα δεδομένα στην περίπτωση αυτή, καθώς εισέρχονται στο σύστημα παραμένουν σε αδρανή κατάσταση, μεχρις ότου το σύστημα να είναι σε θέση να τα επεξεργαστεί, όταν δηλαδή δεν υπάρχει φόρτος. Αυτό σημαίνει βέβαια ότι θα πρέπει να υπάρχει προσωρινός χώρος αποθήκευσης (buffer) - ιδανικά στη μνήμη του συστήματος - για να μην επωμιστεί το σύστημα και το κόστος εισόδου/εξόδου σε σκληρούς δίσκους.

Το αποτέλεσμα, ανάλογα με την καθυστέρηση, μπορεί να χαρακτηριστεί ακριβές σε ένα χρονικό σημείο, παλαιότερο ή νεότερο σε σχέση με το παρόν, σε καμία όμως περίπτωση δε θα είναι ακριβές σε πραγματικό χρόνο. Το πλεονέκτημα της μεθόδου είναι ότι εισάγοντας μια μικρή καθυστέρηση στην έκδοση των αποτελεσμάτων, έχουμε τη δυνατότητα να εξάγουμε αποτελέσματα με σχετική ακρίβεια και ταυτόχρονα να μπορούμε να χειριστούμε ρεύματα δεδομένων με μεγάλες αυξομειώσεις του ρυθμού εισόδου.

2.6.2. **Δειγματοληψία** (sampling)

Στην περίπτωση όπου η λειτουργία update(tuple) είναι αργή αλλά η computeAnswer() είναι γρήγορη, το σύστημα δέχεται πλειάδες πιο γρήγορα από ότι μπορεί να τις προωθήσει και να τις επεξεργαστεί. Αυτό σημαίνει ότι θα πρέπει να απορριφθούν κάποιες πλειάδες.

Είδαμε σε προηγούμενες παραγράφους τεχνικές με τις οποίες είναι δυνατόν να απορρίψουμε κάποια δεδομένα από το ρεύμα εισόδου προσπαθώντας να κρατήσουμε όσο το δυνατόν υψηλή την ποιότητα υπηρεσίας του συστήματος, δηλαδή την ακρίβεια του εξαγομένου αποτελέσματος με χρήση μεθόδων που θα απορρίπτουν τις λιγότερο κατάλληλες πλειάδες. Οι τεχνικές αυτές, προσπαθούν να δημιουργήσουν ένα όσο το δυνατόν αντιπροσωπευτικότερο

δείγμα του ρεύματος και είναι όπως είπαμε ένας τομέας έρευνας με πολλές απαιτήσεις αλλά και προσδοκίες.

2.6.3. **Συνόψεις** (synopses)

Στην περίπτωση όπου και οι δύο λειτουργίες είναι αργές, (αλλά και γενικότερα ως εναλλακτική των δύο τεχνικών που προαναφέραμε), θα μπορούσε να χρησιμοποιηθεί η λύση των συνόψεων. Η λύση αυτή εισάγει την έννοια των συναθροιστικών δεδομένων (aggregated data) στα ΣΔΡΔ. Βέβαια, υπάρχουν πολλές διαφορές στον τρόπο με τον οποίο υπολογίζονται σε σχέση με τις συναθροίσεις των κλασσικών ΣΔΒΔ, αλλά και ομοιότητες στον τρόπο με τον οποίο χρησιμοποιούνται. Έτσι, ενώ οι συνόψεις υπολογίζονται παίρνοντας δείγμα δεδομένων, οι συναθροίσεις των παραδοσιακών ΣΔΒΔ αντανακλούν με ακρίβεια τα δεδομένα από τα οποία δημιουργήθηκαν. Άλλη σημαντική διαφορά είναι ότι οι συνόψεις διατηρούνται στη μνήμη του συστήματος και όχι στο δίσκο, όπως επίσης και ότι το μέγεθος των συνόψεων είναι κατά πολύ μικρότερο από αυτό των συναθροίσεων των ΣΔΒΔ.

Οι συνόψεις στα ΣΔΡΔ υπολογίζονται με δύο βασικές τεχνικές σκίτσων (sketches). Τα AMS sketches και τα FM sketches. Οι συνόψεις, όπως συμπεραίνεται από τα παραπάνω, αποτελούν μέθοδο προσεγγιστικής απάντησης ενός ερωτήματος. Αυτό σημαίνει ότι βρίσκουν εφαρμογή σε περιπτώσεις όπου το ζητούμενο είναι η ανάλυση τάσεων, η ανίχνευση ανωμαλιών ή και περιπτώσεων απάτης κτλ, όπου δεν απαιτείται η ακρίβεια σε επίπεδο πλειάδας (όπως για παράδειγμα μιας οικονομικής εφαρμογής που απαιτεί ακρίβεια σε επίπεδο τέταρτου δεκαδικού ψηφίου) αλλά η εύρεση τάσεων μέσα στον όγκο των ρευμάτων δεδομένων. Τα βασικά χαρακτηριστικά των σκίτσων είναι τα παρακάτω:

- Έχουν μικρό ίχνος, που σημαίνει ότι καταλαμβάνουν λίγη μνήμη
- Αποτελούν δομές που προκύπτουν από τυχαία δειγματοληψία πάνω στα δεδομένα εισόδου και μπορούν να διαχειριστούν εύκολα και άμεσα όγκους δεδομένων, υψηλού ρυθμού ρευμάτων δεδομένων.
- Εγγυώνται την ακρίβεια των αποτελεσμάτων, δεδομένων συγκεκριμένων κριτηρίων, και εξασφαλίζουν έτσι την ποιότητα της υπηρεσίας που προσφέρουν.
- Μπορούν να χειριστούν διαγραφές πλειάδων.
- Μπορούν να εφαρμοστούν σε κατανεμημένα ΣΔΡΔ. Αυτό σημαίνει ότι είναι δυνατός ο υπολογισμός σκίτσων σε κάθε κόμβο και εν συνεχεία να παραχθεί η σύνοψη με το συνδυασμό όλων των υπό μέρους σκίτσων και μάλιστα χωρίς να απαιτείται πολύπλοκη επεξεργασία.

Συνοπτικά, τα AMS σκίτσα, ενδείκνυνται για την παροχή πληροφορίας που έχει να κάνει με το γενικότερο όγκο των δεδομένων, όπως συχνότητα ανά λεπτό ή εκτίμηση όγκου ενός τελεστή σύνδεσης, ενώ τα FM σκίτσα, ενδείκνυνται για την εξαγωγή πληροφορίας σχετική με τα χαρακτηριστικά των πλειάδων όπως για παράδειγμα, τον αριθμό των διακριτών τιμών του ρεύματος δεδομένων.

2.7. **Ανασχετικοί τελεστές** (blocking operators)

Ανασχετικοί είναι οι τελεστές οι οποίοι δεν μπορούν να παράξουν έξοδο εάν προηγουμένως δεν έχουν λάβει όλα τα δεδομένα στην είσοδο τους. Τέτοιοι τελεστές είναι για παράδειγμα η ταξινόμηση και οι τελεστές που χρησιμοποιούνται για τη δημιουργία συναθροίσεων όπως min(), max(), avg, count() και sum(). Έτσι, η προσπάθεια να χρησιμοποιηθούν αυτοί οι τελεστές σε ΣΔΡΔ θα αποτύγχανε αφού τα ρεύματα δεδομένων δεν έχουν ουσιαστικά τέλος, άρα και οι τελεστές αυτοί δε θα επέστρεφαν ποτέ αποτέλεσμα. Από την άλλη πλευρά όμως, οι τελεστές αυτοί είναι ιδιαίτερα χρήσιμοι και χρησιμοποιούνται ευρέως στη βελτιστοποίηση του χρόνου απόκρισης των ερωτημάτων. Άρα, τα ΣΔΡΔ δε θα μπορούσαν απλά να αγνοήσουν τη χρησιμότητα αυτή, αλλά να χειριστούν κατάλληλα τις απαιτήσεις υλοποιώντας αντίστοιχους μηχανισμούς. Μάλιστα, αυτός ο τομέας έρευνας είναι ένας από τους πιο ενεργούς στο χώρο των ΣΔΡΔ με πολλές προκλήσεις και υποσχέσεις.

Οι ανασχετικοί τελεστές που βρίσκονται στην αρχή των πλάνων εκτέλεσης μπορούν να εντοπιστούν και να χειρισθούν ευκολότερα, σε αντίθεση με αυτούς που βρίσκονται σε ενδιάμεσα στάδια, όπως για παράδειγμα μια συνάθροιση ενός υπο-ερωτήματος (subquery). Εάν ο τελεστής βρίσκεται στην αρχή του πλάνου εκτέλεσης και παράγει από ένα έως ελάχιστα δεδομένα, τότε πιθανόν να είναι δυνατή η συνεχόμενη εκτέλεση του κάθε φορά που μια πλειάδα του ρεύματος καταφτάνει στην είσοδο του συστήματος. Όταν όμως η έξοδος του τελεστή είναι μεγάλη, (π.χ. σε περίπτωση ταξινόμησης) τότε η επανεκτέλεση του τελεστή θα ήταν χρονοβόρα. Πιο αποτελεσματική προσέγγιση θα ήταν αυτή της αρχικής εκτέλεσης, αποθήκευσης της δομής εξόδου και εν συνεχεία, σταδιακή ενημέρωση του αποτελέσματος με τα δεδομένα του ρεύματος.

Πάντως, καμία από τις δύο αυτές μεθόδους δε θα είχε τα επιθυμητά αποτελέσματα στην περίπτωση που ο ανασχετικός τελεστής βρισκόταν σε κάποιο ενδιάμεσο στάδιο του πλάνου εκτέλεσης. Το βασικό πρόβλημα στην περίπτωση αυτή είναι ότι τα αποτελέσματα που παράγονται από τέτοιους τελεστές θα πρέπει να τροφοδοτήσουν άλλους τελετές του πλάνου, ενώ την ίδια στιγμή, νέα δεδομένα καταφθάνουν στο σύστημα. Έτσι τα αποτελέσματα των ενδιάμεσων αυτών τελεστών δεν θα ήταν αξιόπιστα ενόσω το ερώτημα εκτελείται.

Μια προσέγγιση για την αντιμετώπιση του προβλήματος αυτού είναι η αντικατάσταση των ανασχετικών τελεστών με μη-ανασχετικούς τελεστές (non-blocking operators) οι οποίοι εκτελούν (σχεδόν) την ίδια λειτουργία. Ένα παράδειγμα τέτοιου τελεστή είναι ο τελεστής juggle που αποτελεί την μη-ανασχετική έκδοση του τελεστή της ταξινόμησης. Ο τελεστής αυτός, προσπαθεί να αναδιατάξει τις πλειάδες εισόδου με τέτοιο τρόπο έτσι ώστε τα δεδομένα εξόδου να βρίσκονται σε ταξινομημένη μορφή. Αυτό εμπεριέχει πάντα τον κίνδυνο, κάποιες πλειάδες να βρεθούν εκτός σειράς οπότε και να απορριφθούν. Η χρήση μη-ανασχετικών τελεστών στη θέση των ανασχετικών, είναι ένα ανοιχτό πρόβλημα και ιδιαίτερα για τους τελεστές των συναθροίσεων.

Μια άλλη προσέγγιση στη χρήση μη-ανασχετικών τελεστών είναι η χρήση σημείων στίξης (punctuations) τα οποία λειτουργούν ως ένα είδος περιορισμού (constraint) στο τι είναι αποδεκτό να εμφανιστεί ως πλειάδα στο ρεύμα εισόδου και στο τι όχι. Οι περιορισμοί αυτοί ενσωματώνονται στο ρεύμα εισόδου και έχουν ως αποτέλεσμα την αλλαγή της συμπεριφοράς των τελεστών που συμμετέχουν στο πλάνο εκτέλεσης του ερωτήματος. Για παράδειγμα, ένας τέτοιος περιορισμός θα ήταν η πρόταση: «όλες οι πλειάδες που ακολουθούν θα πρέπει να έχουν στο χαρακτηριστικό daynumber τιμή μεγαλύτερη ή ίση του 10». Αυτό θα είχε ως αποτέλεσμα ένας τελεστής συνάθροισης να απέρριπτε τις πλειάδες εκείνες για τις οποίες θα ίσχυε:

daynumber < 10. Με αυτόν τον τρόπο θα μειώνονταν και οι απαιτήσεις του συστήματος σε μνήμη για τον υπολογισμό του αποτελέσματος.

Ένα ανοιχτό πρόβλημα είναι η συσχέτιση και τυποποίηση τέτοιων σημείων στίξης με τις ανάγκες σε μνήμη για την εκτέλεση ερωτημάτων διαρκείας. Αυτό σημαίνει ότι ένα ερώτημα το οποίο θα απαιτούσε θεωρητικά, απεριόριστο ποσό μνήμης για να εκτελεστεί, θα μπορούσε να αποδειχθεί ότι με τη χρήση σημείων στίξης θα απαιτείτο συγκεκριμένο ποσό μνήμης. Αυτό βεβαίως θα σήμαινε ότι η χρήση των σημείων στίξης θα ήταν υποχρεωτική στο ρεύμα δεδομένων. Σχετική με την ιδέα αυτή είναι και η ιδέα χρήσης σχεσιακών τελεστών στο φυσικό σχεδιασμό της βάσης δεδομένων, έτσι ώστε σε συνεργασία με τα σημεία στίξης, να είχαμε ως αποτέλεσμα την αναίρεση των προβλημάτων χρήσης των ανασχετικών τελεστών.

2.8. Ανάγκη πρόσβασης σε παλαιότερα δεδομένα

Στο μοντέλο των συστημάτων διαχείρισης ρευμάτων δεδομένων από τη στιγμή που μια πλειάδα έχει περάσει από την διαδικασία επεξεργασίας, δεν μπορεί να επαναχρησιμοποιηθεί μιας και όπως αναφέραμε, η επεξεργασία των πλειάδων στα συστήματα αυτά είναι σειριακή.

Αυτός ο περιορισμός σημαίνει ότι τα ερωτήματα διάρκειας τα οποία εκτελούνται στο σύστημα αφού κάποια δεδομένα έχουν ήδη απορριφθεί, είναι δυνατόν είτε να είναι ανακριβή ή ακόμη και να είναι τελείως αδύνατο να απαντηθούν. Μια καλή λύση σε αυτό το πρόβλημα είναι να θεωρήσουμε ότι όλα τα περιστασιακά ερωτήματα επιτρέπεται να αναφέρονται σε μελλοντικά δεδομένα και μόνο. Έτσι, μόνο τα δεδομένα που καταφθάνουν στο σύστημα μετά την έναρξη εκτέλεσης του ερωτήματος θα λαμβάνονται υπόψη ενώ όλα τα προηγούμενα θα αγνοούνται για το συγκεκριμένο ερώτημα. Ακόμη και αν αυτή η λύση μπορεί να μην φαίνεται αρκετά ικανοποιητική, μπορεί τελικά να αποδειχτεί ότι είναι πλήρως αποδεκτή σε πολλές κατηγορίες εφαρμογών.

Μια ακόμα πιο φιλόδοξη προσέγγιση για το χειρισμό των περιστασιακών ερωτημάτων, τα οποία έχουν αναφορές σε παλαιότερα δεδομένα, είναι η δημιουργία συναθροίσεων και συνόψεων γενικής μορφής / σκοπού, πάνω στα ρεύματα δεδομένων. Αυτές θα μπορούν να χρησιμοποιηθούν έτσι ώστε να δίνουν προσεγγιστικές απαντήσεις σε μελλοντικά περιστασιακά ερωτήματα. Ακολουθώντας αυτή την προσέγγιση, θα πρέπει να παρθεί μια απόφαση εκ των προτέρων για το ποιός θα ήταν ο καλύτερος τρόπος με τον οποίον θα μπορούσαν να δημιουργηθούν αυτές οι συνόψεις και συναθροίσεις αλλά και ποιος θα ήταν ο βέλτιστος τρόπος να χρησιμοποιηθεί η μνήμη του συστήματος, έτσι ώστε οι προσεγγιστικές απαντήσεις στα μελλοντικά ερωτήματα διάρκειας να είναι όχι μόνον όσο το δυνατόν πιο ακριβείς αλλά και να καλύπτουν και όσο το δυνατό περισσότερες και διαφορετικές μεταξύ τους ερωτήσεις.

Το πρόβλημα εδώ είναι ίδιο σε κάποιους τομείς του, με το πρόβλημα της βέλτιστης φυσικής σχεδίασης μιας κλασικής βάσης δεδομένων, που καλείται να απαντήσει στο ποια είναι τα καταλληλότερα ευρετήρια και ποιες οι καταλληλότερες υλοποιημένες όψεις (materialized views) οι οποίες θα απαντούν σε όσο το δυνατόν περισσότερες περιστασιακές ερωτήσεις τίθενται από τους χρήστες του συστήματος. Υπάρχει όμως μια σημαντική διαφορά: σε ένα παραδοσιακό σύστημα βάσεων δεδομένων, όταν ένα ευρετήριο η μια υλοποιημένη όψη δεν υπάρχει ή δεν καλύπτει το ερώτημα, είναι δυνατή η δρομολόγηση στα δεδομένα του πίνακα από τα οποία θα μπορούσε να έχει προέλθει η υλοποιημένη όψη η το ευρετήριο (βεβαίως με αυξημένο κόστος προσπέλασης). Από την άλλη πλευρά, σε ένα σύστημα διαχείρισης ροών δεδομένων, αν η συγκεκριμένη συνάθροιση ή σύνοψη δεν υπάρχει, τότε απλά, δεν μπορεί να γίνει περαιτέρω επεξεργασία, άρα το ερώτημα δεν μπορεί να απαντηθεί.

Ακόμη όμως και στην περίπτωση που τα περιστασιακά ερωτήματα έχουν τον περιορισμό να αναφέρονται μόνο σε μελλοντικά δεδομένα, εγείρονται θέματα τα οποία αφορούν στο πως θα δημιουργηθούν οι κατάλληλες δομές, πως θα επεξεργασθούν τα δεδομένα καλύτερα, και πως θα παράγονται οι απαντήσεις με ακρίβεια και χωρίς καθυστέρηση. Σε εφαρμογές ρευμάτων δεδομένων όπου τα ερωτήματα είναι διαρκείας και όχι εφήμερα ερωτήματα στιγμιότυπων (snapshot queries), το κέρδος το οποίο θα μπορούσε να παραχθεί από την βελτιστοποίηση της εκτέλεσης πολλών ερωτημάτων διάρκειας ταυτόχρονα, θα μπορούσε να ήταν κατά πολύ μεγαλύτερο από το αντίστοιχο πιθανό κέρδος βελτιστοποίησης μιας παραδοσιακής βάσης δεδομένων.

Κλείνοντας, θα πρέπει να σημειώσουμε ότι η ιδιαιτερότητα των περιστασιακών ερωτημάτων διαρκείας μετατρέπει το πρόβλημα της εύρεσης του καλύτερου πλάνου εκτελέσεις

από off-line, σε πρόβλημα πραγματικού χρόνου το οποίο όπως είδαμε και παραπάνω, προκαλεί θέματα προσαρμοστικότητας του πλάνου εκτέλεσης στα δεδομένα. Μια λύση είναι βεβαίως η χρήση των eddies και stems η οποία όμως θα πρέπει να επεκταθεί έτσι ώστε να περιλάβει και τον ταυτόχρονο χειρισμό πολλών ερωτημάτων διαρκείας, κάτι το οποίο αποτελεί σύγχρονο τομέα έρευνας στο χώρο των ΣΔΡΔ.

2.9. **Χρονόσημα** (timestamps)

Η φύση των ρευμάτων δεδομένων επιβάλλει την ανάγκη προσδιορισμού χρονικών και ποσοτικών ορίων έτσι ώστε να είναι δυνατή η σταδιακή επεξεργασία τους. Τα όρια αυτά ορίζονται από διαστήματα τιμών που καλούνται «παράθυρα» και τα οποία έχουν σκοπό να χωρίσουν το ρεύμα σε ομάδες πλειάδων, διακριτές χρονικά ή/και ποσοτικά μεταξύ τους. Με το πέρας της επεξεργασίας κάθε ομάδας πλειάδων, που ορίζεται από το αντίστοιχο παράθυρο, καθίσταται δυνατή και η σταδιακή αποτίμηση του ερωτήματος διαρκείας. Στην περίπτωση ορισμού χρονικών ορίων, τα άκρα των παραθύρων ορίζονται από τα χρονόσημα (timestamps) τα οποία ενσωματώνονται και χαρακτηρίζουν την κάθε πλειάδα του εισερχόμενου ρεύματος.

Η προσέγγιση αυτή είναι ξεκάθαρη για τις πλειάδες οι οποίες προέρχονται από ένα μόνο ρεύμα δεδομένων αλλά λιγότερο καθαρή όταν προσπαθούμε να χωρίσουμε χρονικά πλειάδες οι οποίες προέρχονται από πολλαπλά συστήματα. Ποιο θα πρέπει, για παράδειγμα, να είναι το χρονόσημο μιας πλειάδας στο αποτέλεσμα μιας σύνδεσης δύο συνόλων Α και Β, όταν η κάθε πλειάδα που συμμετείχε στην ένωση και ανήκε στο σύνολο Α, είχε διαφορετικό χρονόσημο από την αντίστοιχη πλειάδα του συνόλου Β;

Προβλήματα επίσης παρουσιάζονται όταν καλούμαστε να αποδώσουμε ένα χρονόσημο σε μια πλειάδα η οποία προέκυψε από την ένωση πολλών κατανεμημένων ρευμάτων δεδομένων από διαφορετικά συστήματα, τα οποία συστήματα είχαν διαφορά στον ορισμό του χρόνου. Καταλαβαίνουμε λοιπόν τα προβλήματα που προκύπτουν, αφού ακόμη και η έννοια του χρόνου μεταξύ συστημάτων είναι σχετική. Τα χρονόσημα που μπορούν να οριστούν είναι δύο ειδών:

- *μη-ορισμένα-χρονόσημα:**implicit timestamps:*** είναι τα χρονόσημα τα οποία δεν προκύπτουν από τα συστήματα τα οποία παράγουν τις πλειάδες, αλλά από το ρεύμα εισόδου στο οποίο ενσωματώνεται ο χρόνος άφιξης της πλειάδας στο σύστημα επεξεργασίας και
- *ορισμένα-χρονόσημα:**explicit timestamps:*** είναι τα χρονόσημα τα οποία είναι καθορισμένα μέσα στην πλειάδα, αφού ένα από τα χαρακτηριστικά της πλειάδος είναι ορισμένο ως χρονόσημο.

Τα ορισμένα-χρονόσημα χρησιμοποιούνται όταν κάθε πλειάδα αντιστοιχεί σε ένα γεγονός του πραγματικού κόσμου, το οποίο έλαβε χώρα σε μια συγκεκριμένη χρονική στιγμή, η οποία θεωρείται σημαντική για τον ορισμό της πλειάδας. Τα μη-ορισμένα-χρονόσημα χρησιμοποιούνται όταν η πηγή δεδομένων δεν συμπεριλαμβάνει χρονική πληροφορία στην πλειάδα κι όταν το χρονικό σημείο στο οποίο παρήχθη δεν είναι σημαντικό, αλλά οι σχετικές υποθέσεις του τύπου «πρόσφατο γεγονός» η «παλαιότερο γεγονός» εξακολουθούν να είναι σημαντικές.

Τα ορισμένα-χρονόσημα έχουν το μειονέκτημα ότι οι πλειάδες τις οποίες προσδιορίζουν μπορεί να μην καταφθάσουν στο σύστημα επεξεργασίας με τη χρονική σειρά την οποία παρήχθησαν. Αυτό σημαίνει ότι οι πλειάδες οι οποίες παρήχθησαν χρονικά αργότερα από κάποιες άλλες, μπορεί να καταφθάσουν νωρίτερα στο σύστημα επεξεργασίας. Η αδυναμία εξασφάλισης της σωστής χρονικής σειράς των πλειάδων, καθιστά δύσκολους τους υπολογισμούς σε σχέση με τα ορισμένα-χρονόσημα και γενικότερα την οποιαδήποτε επεξεργασία η οποία βασίζεται στην χρονική σειρά και ταξινόμηση των πλειάδων.

Πάντως, μπορούμε να θεωρήσουμε ότι οι πλειάδες των ρευμάτων εισόδου είναι σχεδόν πάντα ταξινομημένες χρονικά με βάση το χρονόσημο τους, τουλάχιστον κατά το μεγαλύτερο μέρος τους. Έτσι, πλειάδες οι οποίες είναι εκτός χρονικής σειράς μπορούν να ταξινομηθούν κατά την είσοδό τους στο σύστημα με τη χρήση κάποιας ενδιάμεσης μνήμης. Βεβαίως υπάρχει

πάντα η πιθανότητα άφιξης μιας πλειάδας η οποία βρίσκεται εκτός χρονικών ορίων παραθύρου, που η συγκεκριμένη χρονική στιγμή επεξεργασίας ανήκει, με αποτέλεσμα την απόρριψη της από το σύστημα.

Ας δούμε τώρα πως μπορούμε να προσδιορίσουμε χρονόσημα σε πλειάδες που αποτελούν την έξοδο τελεστών όπως για παράδειγμα της σύνδεσης. Στην περίπτωση αυτή υπάρχουν πολλές προσεγγίσεις οι οποίες μπορούν να ακολουθηθούν. Εξετάζουμε δύο από αυτές:

- Η πρώτη προσέγγιση (η οποία θα ταίριαζε περισσότερο με τα μη-ορισμένα-χρονόσημα) είναι να μην υπάρξει εγγύηση για την σειρά ταξινόμησης των πλειάδων του αποτελέσματος της σύνδεσης αλλά απλά, να γίνει η υπόθεση ότι οι πλειάδες οι οποίες φθάνουν νωρίτερα, είναι και πιθανόν να περάσουν από τον τελεστή της σύνδεσης νωρίτερα. Η ακριβής σειρά μπορεί να εξαρτάται είτε από την υλοποίηση, είτε από λεπτομέρειες του χρονοπρογραμματισμού του συστήματος. Κάθε πλειάδα η οποία παράγεται από έναν τελεστή σύνδεσης, χαρακτηρίζεται και από ένα χρονόσημο το οποίο ορίζει τη χρονική στιγμή κατά την οποία παρήχθη από τον τελεστή της σύνδεσης. Αυτή η προσέγγιση έχει το πλεονέκτημα ότι είναι εύκολα υλοποιήσιμη και δεν χρειάζεται μελλοντικές τροποποιήσεις. Έχει όμως το μειονέκτημα ότι δεν μπορεί να χρησιμοποιηθεί για τον ορισμό παράθυρών πάνω στα αποτελέσματα του ερωτήματος αφού τα χρονόσημα δεν αντιστοιχούν στα πραγματικά χρονικά σημεία στα οποία οι πλειάδες παρήχθησαν.
- Η δεύτερη προσέγγιση η οποία μπορεί να χρησιμοποιηθεί και με τις δύο μεθόδους ορισμού χρονοσήμων, είναι να αφήσουμε τον χρήση του συστήματος να ορίσει ποιο θα είναι το χρονόσημο της κάθε πλειάδας του αποτελέσματος του τελεστή της σύνδεσης πολλαπλών ροών. Μια απλή πολιτική θα ήταν να ορίσουμε την προτεραιότητα των χρονοσήμων με βάση τη σειρά με την οποία εμφανίζονται τα ρεύματα στην δήλωση «FROM» του ερωτήματος διαρκείας. Για παράδειγμα, εάν έχουμε δύο ρεύματα s1 και s2, τα οποία μετέχουν στο ερώτημα διαρκείας: SELECT * FROM s1, s2 WHERE s1.x=s2.y και εμφανίζονται στη δήλωση «FROM» με τη σειρά s1, s2, τότε το αποτέλεσμα του τελεστή σύνδεσης των δύο αυτών ρευμάτων θα ήταν μια πλειάδα της οποίας το χρονόσημο θα ήταν εκείνο του ρεύματος s1. Η προσέγγιση αυτή έχει το μειονέκτημα ότι μπορεί να παράγει πολλές πλειάδες στην έξοδο του τελεστή, με το ίδιο χρονόσημο.

Αν απαιτείτο περαιτέρω χρονικός χαρακτηρισμός των πλειάδων εξόδου, θα μπορούσε να χρησιμοποιηθεί και το χρονόσημο του δεύτερου ρεύματος. Οι πλειάδες εξόδου που παράγονται από τον τελεστή της σύνδεσης θα μπορούσαν πρώτα να ταξινομούνται με βάση το χρονόσημο του ρεύματος s1 και εν συνεχεία να ταξινομούνται με βάση το χρόνοσιμο του ρεύματος s2. Η προσέγγιση αυτή μπορεί να έχει όμως και κάποια μειονεκτήματα ειδικά όσον αφορά στην υλοποίηση αφού απαιτεί τη προσωρινή φύλαξη των πλειάδων εξόδου μέχρις ότου να είναι βέβαιο ότι καμία μελλοντική πλειάδα εισόδου δεν αλλάζει την χρονική σειρά του αποτελέσματος. Για παράδειγμα εάν τα χρονόσημα των πλειάδων του ρεύματος s1 έχουν προτεραιότητα σε σχέση με αυτά του ρεύματος s2 και μια πρόσφατη πλειάδα του s1 συνδεθεί με μια πλειάδα του s2, είναι πιθανό μια μελλοντική πλειάδα του s2 να ενωθεί με μια παλαιότερη πλειάδα s1, η οποία εξακολουθεί να είναι μέσα στα χρονικά όρια του παράθυρου που επεξεργάζεται το σύστημα εκείνη τη στιγμή. Σε αυτήν την περίπτωση η πλειάδα η οποία παρήχθη δεύτερη, ανήκει χρονικά πριν την πλειάδα η οποία παρήχθη πρώτη από τον τελεστή της σύνδεσης. Σε ένα ερώτημα με πολλές τέτοιες περιπτώσεις απαιτείται η εκτίμηση όχι μόνο του κόστους χρήσης μνήμης για την προσωρινή αποθήκευση των ροών εξόδου, αλλά και το

κόστος επανα-υπολογισμού της ταξινόμησης του αποτελέσματος. Βεβαίως, εάν δεν υπάρχουν ορισμένα παράθυρα γίνεται εύκολα αντιληπτό ότι, ο τελεστής της σύνδεσης δε θα μπορέσει ποτέ να παράξει αποτελέσματα με εγγυημένη 100% τη σειρά ταξινόμησης των πλειάδων.

Σε κάθε περίπτωση, η σωστή χρήση χρονοσήμων είναι ιδιάζουσας σημασίας για τον ορισμό του ρεύματος και την εξαγωγή αποτελεσμάτων με ακρίβεια σε επίπεδο χρόνου. Τα χρονόσημα, όπως προαναφέραμε αλλά και θα δούμε και στη συνέχεια, εχουν ευρεία εφαρμογή στα παράθυρα των οποίων μονάδα μέτρησης / προσδιορισμού είναι ο χρόνος σε αντίθεση με τα παράθυρα που προσδιορίζονται από το πλήθος των πλειάδων.

2.10. **Ιστογράμματα** (histograms)

Τα ιστογράμματα έχουν ευρεία χρήση σε εμπορικά ΣΔΒΔ. Αποτελούν μέρος των βελτιστοποιητών (optimizers) και παρέχουν στατιστικές πληροφορίες σχετικές με την κατανομή τιμών πεδίων πινάκων, σε διαστήματα τιμών. Για κάθε πεδίο πίνακα που δημιουργείται ένα ιστόγραμμα, είναι δυνατόν να αντληθούν δεδομένα όπως κατανομή τιμών, διασπορά, μέγιστα και ελάχιστα, έτσι ώστε να αποτιμηθεί η επιλεκτικότητα (selectivity) των τιμών του πεδίου. Η πληροφορία αυτή χρησιμοποιείται από τον βελτιστοποιητή για να επιλέξει το καλύτερο πλάνο εκτέλεσης σε σχέση και με την επιλεκτικότητα που προκύπτει από τα ιστογράμματα των τιμών των πεδίων των πινάκων που μετέχουν σε ένα ερώτημα.

Τα ισοβαθή ιστογράμματα (equidepth histograms) αποτελούν τη βάση πολλών δημοφιλών και αρκετά αποτελεσματικών τύπων ιστογραμμάτων. Επιπλέον, είναι σχετικά απλά στον υπολογισμό τους. Τα συμπιεσμένα ιστογράμματα, είναι μια παραλλαγή των ισοβαθών ιστογραμμάτων και είναι κατάλληλα για το χειρισμό δεδομένων των οποίων οι τιμές έχουν μεγάλη διασπορά.

Η χρήση ιστογραμμάτων αν και είναι όπως είπαμε αρκετά διαδεδομένη στα ΣΔΒΔ, εντούτοις, στα ΣΔΡΔ παρουσιάζει θέματα αποδοτικής λειτουργίας αφού απαιτεί υψηλό κόστος σε χρήση μνήμης και χρόνου υπολογισμού. Ακόμη όμως και αν ήταν δυνατός ο αρχικός υπολογισμός των ιστογραμμάτων με το όποιο κόστος, η πολυπλοκότητα και το κόστος επανα-υπολογισμού αυτών, ειδικά όταν οι επιλεκτικότητες των πεδίων τροποποιούνται με την πάροδο του χρόνου, καθιστά προβληματική τη χρήση τους σε περιβάλλοντα πραγματικού χρόνου. Από την άλλη πλευρά τα ιστογράμματα αποτελούν μια ισχυρή βάση πάνω στην οποία θα μπορούσαν να στηριχθούν προσεγγιστικά αποτελέσματα σε ερωτήματα διαρκείας, με την παραγωγή συνόψεων βασισμένων σε αυτά. Για αυτό το λόγο, ο τομέας έρευνας της κατασκευής και χρήσης ιστογραμμάτων σε περιβάλλοντα ΣΔΡΔ είναι αρκετά κινητικός, αφού παρουσιάζει πολλές προκλήσεις αλλά και υποσχέσεις.

2.11. **Κυματίδια** (wavelets)

Τα κυματίδια και οι μετασχηματισμοί κυματιδίων (wavelet transformations) είναι μια ακόμη μέθοδος που αποσκοπεί, τουλάχιστον όσον αφορά στα ΣΔΡΔ, στην κατασκευή συνόψεων με προσεγγιστικές μεθόδους (οι άλλες μέθοδοι όπως είδαμε μέχρι τώρα είναι τα σκίτσα, τα ιστογράμματα και η δειγματοληψία) αλλά και στη βελτιστοποίηση πλάνων εκτέλεσης όπως και τα ιστογράμματα. Πέραν του χώρου των ΣΔΡΔ, τα κυματίδια και οι μετασχηματισμοί κυματιδίων, απαντώνται σε δύο μορφές (απωλεστικοί και μη-απωλεστικοί αλγόριθμοι) και χρησιμοποιούνται κυρίως στη συμπίεση δεδομένων. Τα κυματίδια και οι μετασχηματισμοί τους, εφαρμόζονται σε μονοδιάστατα ή πολυδιάστατα δεδομένα και αποτελούν ευρύ τομέα έρευνας στο χώρο των ΣΔΡΔ. Η έρευνα επικεντρώνεται κυρίως στα παρακάτω:

- Χρήση των κυματιδίων και των μετασχηματισμών κυματιδίων σε πραγματικό χρόνο πάνω σε ρεύματα δεδομένων. Λόγω της πολυπλοκότητας των αλγορίθμων μετασχηματισμών κυματιδίων, δεν είναι δυνατή η χρήση τους σε πραγματικό χρόνο. Αντ' αυτού, ερευνάται η χρήση τεχνικών που θα παράγουν προσεγγιστικά το ίδιο αποτέλεσμα με αυτό των μετασχηματισμών κυματιδίων.
- Βελτιστοποίηση των μετασχηματισμών όσον αφορά στις συνιστώσες ελαστικότητας (wavelet coefficients) από τις οποίες εξερτώνται, έτσι ώστε να περιορίζεται το λάθος υπολογισμού, ο χρόνος επεξεργασίας και ο χώρος που καταλαμβάνουν στη μνήμη, ενώ ταυτόχρονα να προσεγγίζουν με μεγαλύτερη ακρίβεια τα εξαρτώμενα δεδομένα.

2.12. **Σημεία στίξης** (punctuations)

Είδαμε σε προηγούμενο κεφάλαιο ότι μια άλλη προσέγγιση στη χρήση μη-ανασχετικών τελεστών είναι η χρήση σημείων στίξης (punctuations) τα οποία λειτουργούν ως ένα είδος περιορισμού (constraint) στο τι είναι αποδεκτό να εμφανιστεί ως πλειάδα στο ρεύμα εισόδου και στο τι όχι. Οι περιορισμοί αυτοί ενσωματώνονται στο ρεύμα εισόδου και έχουν ως αποτέλεσμα την αλλαγή της συμπεριφοράς των τελεστών που συμμετέχουν στο πλάνο εκτέλεσης του ερωτήματος.

Τα σημεία στίξης είναι αντικείμενα τα οποία ενσωματώνονται μέσα σε ένα ρεύμα δεδομένων και δηλώνουν το τέλος κάποιου υποσυνόλου των δεδομένων. Σε ένα ρεύμα δεδομένων το οποίο περιέχει σημεία στίξης, όλα τα δεδομένα τα οποία συμφωνούν με το συγκεκριμένο σημείο στήριξης θα εμφανίζονται πριν από αυτό. Θα μπορούσαμε να ορίσουμε ένα σημείο στήριξης ως ένα κατηγόρημα του πεδίου τιμών τέτοιο, ώστε οι πλειάδες του ρεύματος δεδομένων οι οποίες πληρούν το κατηγόρημα να συμφωνούν και στο σημείο στίξης. Μπορούμε να ορίσουμε τριών τύπων συμπεριφορές οι οποίες καλούνται «συμπεριφορές σημείων στίξης», για να περιγράψουμε το πως οι τελεστές ερωτημάτων εκμεταλλεύονται τα σημεία στίξης:

1. τη «συμπεριφορά παραγωγής αποτελέσματος» (pass behavior) η οποία ορίζει πότε ένας ανασχετικός τελεστής μπορεί να παράξει δεδομένα, για παράδειγμα, πότε ένας τελεστής συνάθροισης μπορεί να παράξει δεδομένα στην έξοδο του, που σημαίνει ότι όλες οι πλειάδες που έχουν φτάσει μέχρι στιγμής πληρούν ένα συγκεκριμένο κατηγόρημα, άρα δεν περιμένουμε να φθάσουν άλλες πλειάδες που να ανήκουν στο ίδιο υποσύνολο δεδομένων.
2. τη «συμπεριφορά διατήρησης» (keep behavior) η οποία ορίζει πότε ένας ανασχετικός τελεστής διατήρησης κατάστασης (stateful operator) μπορεί ή όχι να ελευθερώσει μέρος της κατάστασης του, όπως δηλαδή και στην προηγούμενη περίπτωση και να παράξει δεδομένα
3. τη «συμπεριφορά αναπαραγωγής στίξεων» (propagation behavior) η οποία ορίζει πότε ένας τελεστής μπορεί να παράξει σημεία στίξης, για παράδειγμα όταν όλα τα δεδομένα ενός υποσυνόλου που πληρούν ένα συγκεκριμένο σημείο στίξης, καταφθάσουν στον τελεστή συνάθροισης, ο τελεστής θα πρέπει να παράξει ένα σημείο στίξης το οποίο θα υποδηλώνει το τέλος του υποσυνόλου αυτού.

Έτσι, σε ένα περιβάλλον αισθητήρων, θα μπορούσαμε να παρεμβάλλουμε σημεία στίξης μέσα στο ρεύμα σε τακτά χρονικά διαστήματα (π.χ. ανά ώρα), δηλώνοντας ότι όλες οι πλειάδες για το συγκεκριμένο πρόθεμα του ταξινομημένου ρεύματος (τα δεδομένα που εξάγονται από έναν αισθητήρα είναι ταξινομημένα με βάση το χρόνο), έχουν καταφθάσει στο σύστημα. Τότε, τα δεδομένα θα παράγονταν στην έξοδο του τελεστή, η διατήρηση της κατάστασης του θα ελευθερωνόταν για τα δεδομένα αυτά και θα παραγόταν ένα σημείο στίξης που θα δήλωνε το τέλος του υποσυνόλου δεδομένων. Η εισαγωγή σημείων στίξης, θα είχε δύο πλεονεκτήματα στους ανασχετικούς τελεστές:

1. Δε θα χρειαζόταν να τροποποιήσουμε τους τελεστές, έτσι ώστε να δέχονται ταξινομημένα δεδομένα στην είσοδο τους (από την άλλη θα πρέπει βεβαίως να τους βελτιώσουμε έτσι ώστε να υποστηρίζουν τα σημεία στίξης) και
2. Δε θα χρειαζόταν πλέον οι τελεστές να παράγουν ταξινομημένα δεδομένα στην έξοδο τους.

Τα σημεία στίξης είναι δυνατόν να παράγονται σε πολλά σημεία του συστήματος. Στο προηγούμενο παράδειγμα του συστήματος αισθητήρων τα πιθανά σημεία παραγωγής και ενσωμάτωσης σημείων στίξης στο ρεύμα δεδομένων θα ήταν:

- Στην πηγή των δεδομένων, δηλαδή στον αισθητήρα. Ο αισθητήρας θα μπορούσε να παράγει ένα σημείο στίξης π.χ. ανά ώρα. Αυτό προσθέτει ένα «ποσό ευφυΐας» στον αισθητήρα.
- Στα σημεία εισόδου των δεδομένων στο ΣΔΡΔ, όπου θα μπορούσε να εκμεταλλευτεί κάποιου είδους πληροφορίας για το αναμενόμενο ρεύμα. Π.χ., αν το σύστημα γνωρίζει τα πεδία τιμών που μπορεί να πάρει η θερμοκρασία, τότε μπορεί να αποκλείσει τιμές που βρίσκονται εκτός ορίων παράγοντας σημεία στίξης που ορίζουν το μέγιστο και ελάχιστο των τιμών που αναμένονται από τους αισθητήρες θερμοκρασίας.
- Σε σχεσιακούς πίνακες βάσεων δεδομένων του ιδίου ή άλλων συστημάτων, οι οποίοι θα μπορούσαν να περιέχουν πληροφορίες για τους αισθητήρες. Με βάση αυτήν την πληροφορία θα μπορούσε να οριστεί το πότε και υπό ποιες συνθήκες θα παράγονται σημεία στίξης. Με τη μέθοδο αυτή, αποφορτίζουμε τους αισθητήρες από το κόστος, και το φόρτο της «ευφυΐας» που αναφέραμε πιο πάνω.
- Στα σημεία εξόδου των τελεστών, οι οποίοι θα μπορούσαν να ενσωματώνουν σημεία στίξης μέσα στο ρεύμα, δηλώνοντας π.χ. το τέλος της σειράς ενός συνόλου ταξινομημένων πλειάδων.

Κλείνοντας την αναφορά μας στα σημεία στίξης, παραθέτουμε μερικά από τα ανοιχτά ζητήματα που προκύπτουν από τη χρήση τους και τα οποία αποτελούν σύγχρονο αντικείμενο έρευνας και ανάπτυξης:

- Ποια θα ήταν εκείνα τα ερωτήματα τα οποία θα ωφελούνταν περισσότερο από τη χρήση των σημείων στίξης; Είναι διαπιστωμένο ότι θεωρητικά δε μπορούν όλα τα ερωτήματα να ωφεληθούν από τη χρήση στίξεων.
- Δεδομένου ότι η εκτέλεση ενός ερωτήματος μπορεί να βελτιστοποιηθεί με τη χρήση σημείων στίξης, ποιο ακριβώς θα ήταν το σύνολο των εκείνων των στίξεων που θα επέφερε τη μέγιστη βελτιστοποίηση; Για παράδειγμα, θα μπορούσαν να ενσωματωθούν στίξεις που να δηλώνου το πέρας ενός υποσυνόλου δεδομένων κάθε μία ώρα, μισή ώρα ή δύο ώρες. Ποια θα ήταν η βέλτιστη προσέγγιση;
- Επιπλέον, σε ποιο σημείο του συστήματος θα έπρεπε να ενσωματωθούν τα σημεία στίξης;
- Τέλος, ποιο θα ήταν το καταλληλότερο πλάνο εκτέλεσης για ένα συγκεκριμένο ερώτημα διαρκείας που θα χρησιμοποιούσε σημεία στίξης;

2.13. **Παράθυρα** (windows)

Μέχρι στιγμής αναλύσαμε διάφορες μεθόδους με τις οποίες χειριζόμαστε ρεύματα δεδομένων με σκοπό τη διάσπαση των μη πεπερασμένων ρευμάτων σε μικρότερα σύνολα τα οποία μπορούν να τροφοδοτηθούν σε μη-ανασχετικούς τελεστές έτσι ώστε να παράξουν, προσεγγιστικά ή μη, προοδευτικό αποτέλεσμα μέσα από το χαοτικό όγκο των ρευμάτων εισόδου. Μια άλλη δομή η οποία χρησιμοποιείται ευρέως στα ΣΔΡΔ για τον ίδιο σκοπό είναι τα παράθυρα (windows). Ο κύριος σκοπός χρήσης των παραθύρων είναι ο ορισμός υποσυνόλων δεδομένων διακριτών μεταξύ τους έτσι ώστε να είναι δυνατή η χρήση των παραδοσιακών σχεσιακών τελεστών πάνω σε αυτά.

Κατά καιρούς έχουν προταθεί και υλοποιηθεί διάφοροι τύποι παραθύρων οι οποίοι έχουν κατηγοριοποιηθεί με βάση συγκεκριμένα κριτήρια τα οποία ορίζουν τα όρια του παραθύρου, τη σχέση του στο χρόνο σε σύγκριση με άλλα παράθυρα κτλ. Ένα παράθυρο ορίζεται από δύο χαρακτηριστικά:

1. το εύρος του, δηλαδή το χρονικό διάστημα εάν πρόκειται για παράθυρο χρονοσήμων ή το πλήθος των πλειάδων και
2. το βήμα, δηλαδή το χρόνο ή τον αριθμό πλειάδων κατά το οποίο μετακινείται το παράθυρο, από το οποίο προκύπτει και αν θα υπάρχει επικάλυψη στα διαστήματα των διαδοχικών παραθύρων. Για παράδειγμα, ένα χρονικό παράθυρο μπορεί να έχει εύρος 10 λεπτών και βήμα 2 λεπτών, άρα τα παράθυρα θα επικαλύπτονται

Ο ρόλος των παραθύρων συνίσταται στην απόσπαση πεπερασμένου πλήθους στοιχείων από το ρεύμα δίνοντας πρόσβαση διαρκώς σε περιορισμένο αλλά σαφώς προσδιορισμένο τμήμα του ρεύματος. Τα περιεχόμενα τους τοποθετούνται χρονικά κοντά στα πιο πρόσφατα στοιχεία των ρευμάτων, ενώ επαναπροσδιορίζονται με κάθε νέα πλειάδα που φτάνει σε αυτά, ανάλογα με τους κανόνες και τις παραμέτρους κάθε παραθύρου. Οι παραθυρικοί τελεστές απαντώνται σε ερωτήματα διαρκείας με στόχο να τροφοδοτήσουν προβληματικούς τελεστές που χρησιμοποιούν όλα τα δεδομένα εισόδου τους για την επιστροφή πλήρους απάντησης, όπως λ.χ. οι *συναθροιστικοί* (***aggregate***) τελεστές *πλήθους* (***COUNT***), *αθροίσματος* (***SUM***), *μέσου όρου* (***AVG***) κ.τ.λ. Όμως όπως εξηγήθηκε σε προηγούμενη ενότητα, δεν είναι ούτε εφικτή ούτε επιθυμητή, η διατήρηση και η επεξεργασία όλων των δεδομένων. Η τροφοδότηση των προβληματικών τελεστών με παράθυρα έχει ως αποτέλεσμα την απεμπλοκή τους, με αντίτιμο την αποδοχή προσεγγίσεων στις επιστρεφόμενες απαντήσεις. Οι τελεστές αυτοί επεξεργάζονται πλέον τα στοιχεία που περιέχονται σε κάθε παράθυρο, εστιάζοντας και αυτοί με την σειρά τους στην πρόσφατη πληροφορία.

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

Σε κάθε περίπτωση, ο τύπος και οι παράμετροι κάθε παραθύρου προσδιορίζουν με αρκετά σαφή τρόπο τον βαθμό προσέγγισης που τυχόν υπεισέρχεται στις απαντήσεις. Με την χρήση τους ωστόσο επιτυγχάνονται: η απαίτηση γρήγορης επεξεργασίας, ο περιορισμός σε σαφώς μικρότερο κομμάτι μνήμης και το φιλτράρισμα της εισερχόμενης πληροφορίας έτσι ώστε πάντοτε να στέλνεται για επεξεργασία ένα πρόσφατο κομμάτι της. Κατά τον τρόπο αυτό, αντισταθμίζονται επαρκώς, οι επιπτώσεις στην ακρίβεια των απαντήσεων που πιθανών να προκαλεί η χρήση των παραθύρων. Η δημοφιλέστερη ίσως κατηγορία, είναι αυτή των κυλιόμενων παράθυρων (sliding windows) με τα οποία παρέχεται πρόσβαση στις πλειάδες που φέρουν χρονόσημο εντός κάποιας χρονικής έκτασης.

Ας δούμε λοιπόν, πιο αναλυτικά, ποιοι είναι οι τύποι των παραθύρων και πως αυτά κατηγοριοποιούνται:

2.13.1. **Κατηγορίες παραθύρων με βάση τον τρόπο ορισμού των πλειάδων**.

2.13.1.1. Παράθυρα ορισμένα με βάση τον αριθμό των πλειάδων (tuple-based windows)

Τα παράθυρα αυτού του τύπου, ορίζονται με βάση ένα συγκεκριμένο πλήθος πλειάδων οι οποίες μπορεί να καλύπτουν κάθε φορά διαφορετικό χρονικό διάστημα ανάλογα με την πηγή εκπομπής του ρεύματος, αλλά και τη χρονική περίοδο. Για παράδειγμα, ένας αισθητήρας ο οποίος συλλέγει πληροφορίες σχετικά με την κυκλοφορία των οχημάτων σε ένα δρόμο, θα αποστέλλει πολλά περισσότερα δεδομένα σε ώρες αιχμής παρά τις νυχτερινές ώρες. Έτσι, ενώ με βάση το πλήθος των πλειάδων, τα παράθυρα θα μπορούσαν να έχουν σταθερό μήκος 100 πλειάδων, χρονικά θα καταλάμβαναν διαφορετικά μεγέθη διαστημάτων, πιθανά λιγότερο από λεπτό τις πρωινές ώρες σε αντίθεση με πολύ μεγαλύτερους χρόνους κατά τις βραδινές ώρες.

2.13.1.2. Παράθυρα ορισμένα με βάση χαρακτηριστικά των πλειάδων (partitioned-based windows)

Τα παράθυρα αυτά ονομάζονται μεριστικά (partitioned) παράθυρα και ορίζονται με βάση κάποιο χαρακτηριστικό (attribute) της πλειάδας. Για παράδειγμα ένα αριθμητικό πεδίο που παίρνει ακέραιες τιμές από 1 έως 10, θα μπορούσε να αποτελεί το κριτήριο διαμέρισης των πλειάδων του ρεύματος σε διακριτά παράθυρα, τα οποία θα περιέχουν το κάθε ένα από αυτά μόνο τις πλειάδες που αντιστοιχούν σε έναν και μόνο ακέραιο του εύρους τιμών. Τα παράθυρα αυτά συνήθως αφορούν πρόσφατα δεδομένα τα οποία φυλάσσονται σε ενδιάμεσες μνήμες μέχρις ότου συγκεντρωθούν όλα τα απαραίτητα δεδομένα για τη διαδικασία της διαμέρισης. Εν συνεχεία γίνεται συνένωση των δεδομένων και ο τελεστής παράγει αποτέλεσμα.

2.13.1.3. Παράθυρα ορισμένα με βάση το χρονικό διάστημα (time-based windows)

Τα παράθυρα αυτού του τύπου ονομάζονται και παράθυρα χρονοσήμου (timestamp). Ορίζονται με βάση κάποιο χρονικό διάστημα, π.χ. μία ώρα. Τα παράθυρα αυτά μπορεί να διαφέρουν στο πλήθος των πλειάδων τους ανάλογα με τη χρονική περίοδο και την πηγή εκπομπής. Έτσι, στο προηγούμενο παράδειγμα του αισθητήρα παρακολούθησης κυκλοφορίας οχημάτων, τα παράθυρα χρονοσήμων που θα ορίζονταν ανά ώρα, άρα το σύστημα θα δεχόταν για επεξεργασία σταθερό πλήθος παραθύρων ανά μέρα, αλλά πιθανά με διαφορετικό πλήθος πλειάδων ανάλογα με την ώρα της μέρας. Η υλοποίηση των παραθύρων αυτού του τύπου μοιάζει με την τεχνική της χρήσης των σημείων στίξης στην περίπτωση που αυτά παράγονται σε καθορισμένα χρονικά διαστήματα.

2.13.2. Κατηγορίες παραθύρων ανάλογα με τη μεταβλητότητα των ορίων τους

2.13.2.1. Παράθυρα σταθερού εύρους (fixed-band windows)

Τα παράθυρα αυτά μπορεί να είναι είτε χρονοσήμου, είτε βασισμένα στο πλήθος των πλειάδων. Σε κάθε περίπτωση, θα παραμένει αμετάβλητη η απόσταση των άκρων του πεδίου ορισμού τους. Εάν για παράδειγμα, πρόκειται για παράθυρα χρονοσήμου τα οποία έχουν οριστεί ανά ώρα, τότε αυτά θα διατηρούν αμετάβλητο το εύρος των άκρων τους, άρα τη «διάρκεια» του παραθύρου στη μία ώρα, και θα παράγουν δεδομένα σταθερά ανά ώρα.

2.13.2.2. Παράθυρα μεταβλητού εύρους (variable-size windows)

Σε αντίθεση με τα παράθυρα σταθερού εύρους, τα παράθυρα μεταβλητού εύρους, έχουν τη δυνατότητα αλλαγής των άκρων των πεδίων τιμών τους. Χρησιμοποιούνται σε περιπτώσεις όπου θέλουμε να πάρουμε σωρευτικές πληροφορίες με αρχή μια συγκεκριμένη χρονική στιγμή. Ένα παράδειγμα τέτοιων παραθύρων είναι τα παράθυρα χρονικού οροσήμου που θα εξετάσουμε στη συνέχεια.

2.13.3. Κατηγορίες παραθύρων ανάλογα με τη δυναμικότητα των ορίων τους

2.13.3.1. Κυλιόμενα παράθυρα (sliding windows)

Τα κυλιόμενα παράθυρα έχουν το χαρακτηριστικό του ότι μπορεί να επικαλύπτουν χρονικά (ή πληθικά εάν πρόκειται για παράθυρα ορισμένα με βάση τον αριθμό πλειάδων) άλλα παράθυρα. Αυτό συμβαίνει στην περίπτωση που το μήκος του παραθύρου είναι μεγαλύτερο του βήματος. Αν για παράδειγμα υποθέσουμε ότι έχουμε ορίσει παράθυρα χρονοσήμων μήκους 1 ώρας με βήμα ανανέωσης 15 λεπτών, τότε το κάθε παράθυρο θα μοιράζεται με το προηγούμενο του, τις πλειάδες 45 λεπτών.

Για το προηγούμενο παράθυρο οι πλειάδες αυτές θα αποτελούν τις πλέον πρόσφατες, ενώ για το τρέχον, τις παλιότερες. Το σχήμα αυτό εφαρμόζεται για την ηπιότερη σταδιακή ανανέωση των δεδομένων, αφού χρησιμοποιώντας και δεδομένα προηγούμενου παραθύρου, ο τελεστής του ερωτήματος διαρκείας, ανάλογα πάντα και με το βήμα ανανέωσης, θα παράγει αποτελέσματα εξομάλυνσης και όχι τόσο δραστικής αλλαγής. Ένα παράδειγμα χρήσης είναι εφαρμογές παρακολουθείς και διαχείρισης δικτύων που δείχνουν μεταξύ άλλων και το ρυθμό χρήσης των πόρων των συστημάτων και των δικτύων.

2.13.3.2. Επάλληλα παράθυρα (tumbling windows)

Τα παράθυρα αυτού του τύπου ορίζονται σε διαδοχικά όρια, άρα τα εύρη τους δεν επικαλύπτονται. Χρησιμοποιούνται στην περίπτωση που το ερώτημα επιβάλλει την ανάλυση διαφορετικών μεταξύ τους χρονικών (ή πληθικών εάν πρόκειται για παράθυρα ορισμένα με βάση τον αριθμό πλειάδων τους) περιόδων. Τα δεδομένα αυτά μπορούν εν συνεχεία να συνδυαστούν συναθροίστηκα σε μεγαλύτερα σύνολα.

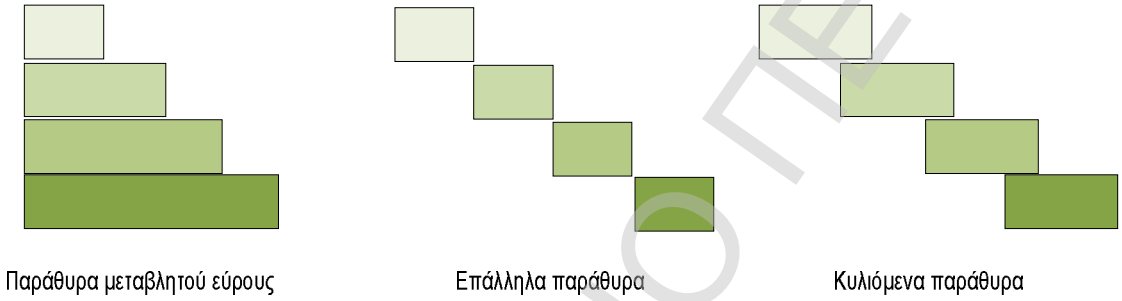
2.13.3.3. Παράθυρα χρονικού οροσήμου (landmark windows)

Τα παράθυρα αυτού του τύπου κρατούν σταθερό το ένα άκρο – συνήθως το κάτω – και μεταβάλλουν το άλλο άκρο. Είναι παράθυρα μεταβλητού εύρους και χρησιμοποιούνται σε περιπτώσεις όπου θέλουμε να πάρουμε σωρευτικές πληροφορίες με αρχή μια συγκεκριμένη χρονική στιγμή. Χρησιμοποιώντας το παράδειγμα του αισθητήρα κυκλοφορίας οχημάτων, μια

ερώτηση που θα ικανοποιούταν με τη χρήση παραθύρων χρονικών οροσήμων θα ήταν, πόσα οχήματα αθροιστικά ανά ώρα, έχουν περάσει από το σημείο που βρίσκεται ο αισθητήρας κατά τη διάρκεια της μέρας. Τα μεγέθη των παραθύρων θα ήταν 1 ώρα για το πρώτο (από 00:00 έως 01:00), 2 ώρες για το δεύτερο (από 00:00 έως 02:00), τρεις ώρες για το τρίτο (από 00:00 έως 03:00) κτλ.

Η χρήση των παραθύρων σε ΣΔΡΔ είναι ευρεία, ως εκ τούτου, πολλά ζητήματα που προκύπτουν βρίσκονται σε έρευνα, όσον αφορά στον τρόπο ορισμού τους, τη βελτιστοποίηση ερωτημάτων με χρήση παραθύρων (και ειδικότερα μάλιστα όταν εμπλέκονται περισσότερα του ενός παραθυρικά ρεύματα), το συνδυασμό τεχνικών απόρριψης δεδομένων σε παραθυρικά ρεύματα και πολλά άλλα.

Το σχήμα 2-2 απεικονίζει τις κατηγορίες παραθύρων ανάλογα με τη μεταβλητότητα των ορίων τους:



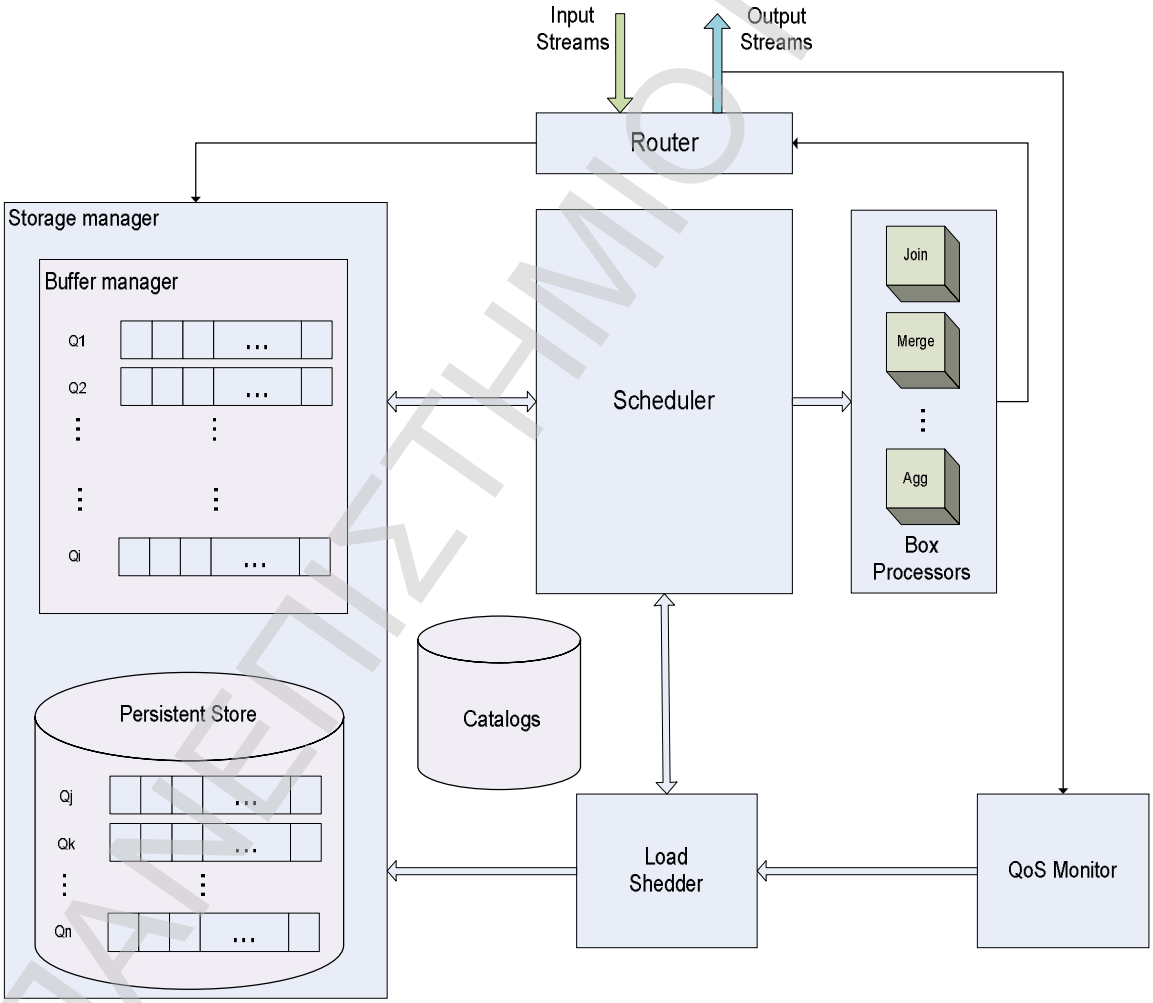
Εικόνα 2-2: Κατηγορίες παραθύρων ανάλογα με τη μεταβλητότητα των ορίων τους

2.14. **Πρωτότυπα ΣΔΡΔ**

Στις επόμενες παραγράφους παρουσιάζουμε συνοπτικά τα πιο διαδεδομένα πρωτότυπα ΣΔΡΔ και τα χαρακτηριστικά τους.

2.14.1. Aurora

Το Aurora είναι ένα γενικού σκοπού Σύστημα Διαχείρισης Ρευμάτων Δεδομένων που σχεδιάζεται και υλοποιείται από τα πανεπιστήμια Brandeis, Brown και M.I.T. Προτείνεται μία προσέγγιση βασισμένη σε ένα γραφικό περιβάλλον με «κουτιά» και «βέλη» (boxes and arrows) θυμίζοντας ένα διάγραμμα ροής δεδομένων αναπαριστώντας το πλάνο εκτέλεσης κάποιου ερωτήματος διαρκείας. Το σχήμα 2-3 παρουσιάζει την αρχιτεκτονική λειτουργίας του συστήματος.



Εικόνα 2-3: Διάγραμμα αρχιτεκτονικής του Aurora

Ο χρονοπρογραμματιστής (scheduler) αποτελεί το συνδετικό κρίκο όλων των υποσυστημάτων του συστήματος. Αποφασίζει για τη σειρά εκτέλεσης των διεργασιών (κουτιά), προσδιορίζοντας τόσο τον αριθμό των πλειάδων που θα δοθούν για επεξεργασία σε κάποιο τελεστή, όσο και την πορεία των ρευμάτων δεδομένων διαμέσου των τελεστών προς την έξοδο. Οι πλειάδες παρακολουθούνται μονίμως από τον Επόπτη Ποιότητας (QoS Monitor), ανατροφοδοτόντας τον χρονοπρογραμματιστή με στατιστικά στοιχεία για την εκτέλεση και τις επιδόσεις του συστήματος. Κάτι αντίστοιχο συμβαίνει και στο σύστημα CAPE το οποίο θα εξετάσουμε στη συνέχεια.

Ένα άλλο σημαντικό υποσύστημα είναι ο διαχειριστής του χώρου αποθήκευσης (Storage Manager) που αναλαμβάνει τη δρομολόγηση των πελιάδων σε ουρές ενδιάμεσης μνήμης (buffer queues) στην περίπτωση που διαπιστωθεί ότι εξαντλείται η διαθέσιμη φυσική μνήμη του συστήματος, κάτι που δεν μπορεί να αποκλειστεί ειδικά όταν ο ρυθμός άφιξης των δεδομένων αυξηθεί 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 χρονόσημο (timestamp), το οποίο χρησιμοποιείται για εποπτεία της παρεχόμενης ποιότητας υπηρεσίας από το σύστημα. Τελικά τα στοιχεία καταλήγουν στην έξοδο, οι πλειάδες της οποίας τροφοδοτούν κάποια εφαρμογή κατά ασύγχρονο τρόπο, ανάλογα δηλαδή προς το ρυθμό παραγωγής τους.

Οι χρήστες έχουν την ευχέρεια να ορίσουν διάφορες απαιτήσεις σχετικά με την ποιότητα υπηρεσιών (Quality of Service QoS) των ερωτημάτων που θέτουν στο σύστημα. Το σύστημα με την σειρά του καλείται να χειριστεί ανάλογα τις απαιτήσεις αυτές με την δημιουργία και παρακολούθηση διαγραμμάτων ποιότητας υπηρεσίας (QoS graphs) όσον αφορά τους χρόνους απόκρισης και την ακρίβεια των αποτελεσμάτων. Ο χρήστης έχει την δυνατότητα να μεταβάλλει την προτεραιότητα εκτέλεσης των ερωτημάτων, με στόχο την τήρηση της ποιότητας υπηρεσιών (QoS) που λαμβάνεται, συνυπολογίζοντας όλους τους τελεστές.

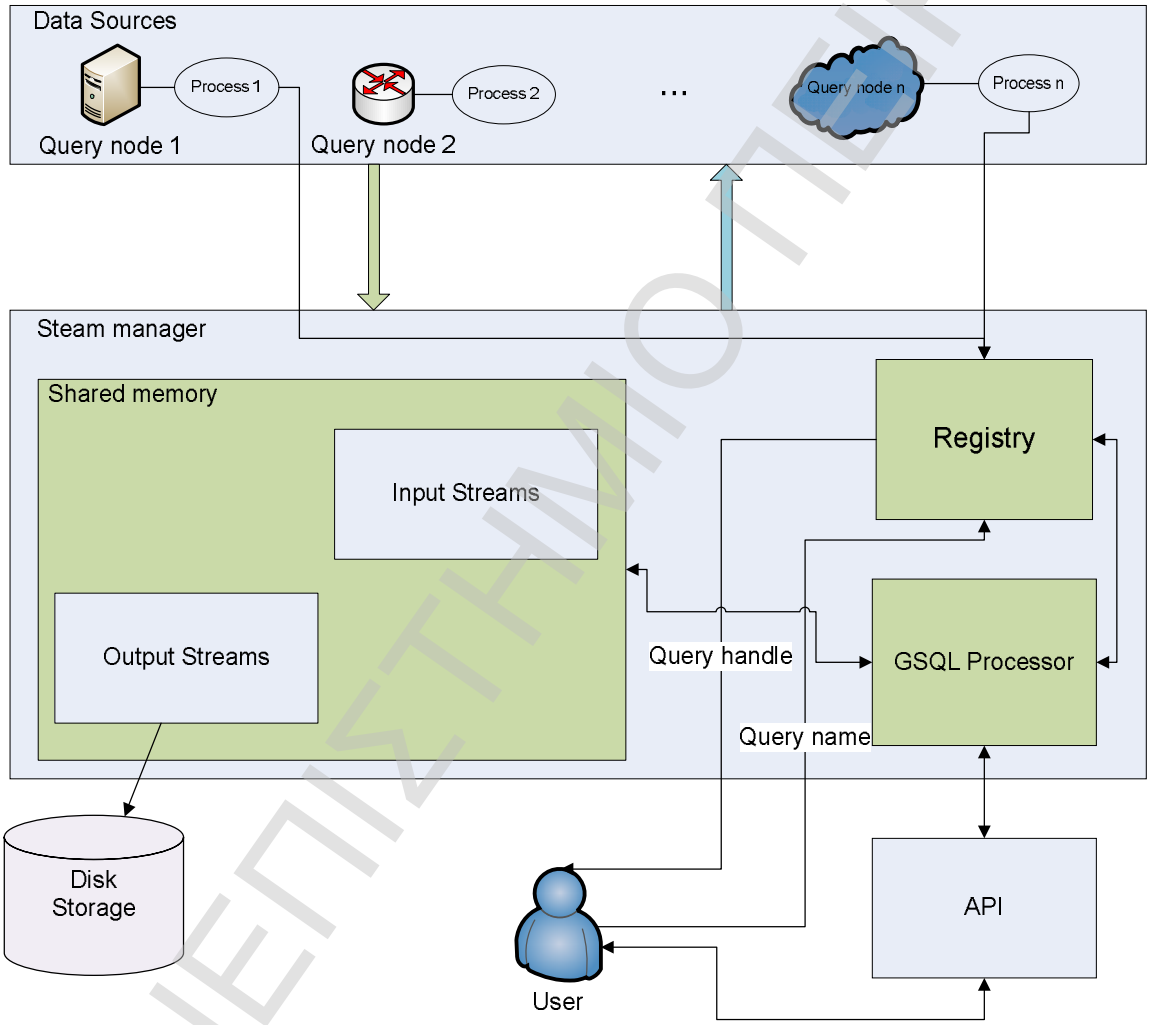
Οι προδιαγραφές των χρηστών και τα διαγράμματα ποιότητας υπηρεσίας χρησιμεύουν στη συνέχεια για να αποφασιστεί με ποιο τρόπο και σε ποιες περιστάσεις θα απορριφτεί κάποιο μέρος των δεδομένων, ελαφρύνοντας το φόρτο του συστήματος (load shedding) επηρεάζοντας την ακρίβεια των απαντήσεων. Έτσι, εάν το σύστημα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

2.14.2. Gigascope

Πρόκειται για ένα σύστημα διαχείρισης ρευμάτων δεδομένων που αναπτύσσεται από την AT&T σε συνεργασία με το Πανεπιστήμιο Carnegie. Το Gigascope χρησιμοποιείται στη διαχείριση και παρακολούθηση δικτύων τηλεπικοινωνιών και γενικότερα υπολογιστικών δικτύων και εφαρμόζεται. Το σύστημα έχει χρησιμοποιηθεί σε διάφορες εφαρμογές, όπως ανάλυση δικτύων, ανάλυση πρωτοκόλλων, ανίχνευση μη εξουσιοδοτημένης πρόσβασης, έλεγχο κίνησης σε ιστοτόπους και δίκτυα κτλ. Στο σύστημα αυτό η διακινούμενη πληροφορία παίρνει την μορφή ρεύματος δεδομένων αποκλείοντας την επεξεργασία στατικών σχέσεων.

Η αρχιτεκτονική λειτουργίας παρουσιάζεται στο διάγραμμα 2-4. Η καρδιά του συστήματος είναι ο διαχειριστής ρευμάτων (stream manager). Κάθε πηγή (κόμβος) δεδομένων για να αποστείλει δεδομένα θα πρέπει καταρχήν να καταχωρηθεί στον κατάλογο (registry) των

πηγών εισόδου. Τότε δημιουργείται μια διεργασία (process) για κάθε κόμβο (query node) η οποία επικοινωνεί με το διαχειριστή ρευμάτων. Κάθε κόμβος επικοινωνεί με το σύστημα μέσω μιας κοινής διαμοιρασμένης μνήμης (shared memory) στην οποία αποστέλλονται τα ρεύματα εισόδου και από την οποία εξάγονται τα ρεύματα εξόδου. Ο χρήστης ή η εφαρμογή η οποία εκτελεί ένα ερώτημα διαρκείας, αποστέλλει καταρχήν στον κατάλογο, ένα όνομα ερωτήματος (query name) και παραλαμβάνει ένα χειριστή ερωτήματος (query handler). Εν συνεχεία επικοινωνεί μέσω του προγραμματιστικού περιβάλλοντος (API) και του επεξεργαστή της γλώσσας διαχείρισης ρευμάτων (GSQL processor) με τις πηγές των ρευμάτων, πάντα μέσω της περιοχής διαμοιρασμένης μνήμης. Τα ρεύματα εξόδου μπορούν να επιστραφούν στο χρήστη ή να αποθηκευτούν για ιστορικούς λόγους και περαιτέρω επεξεργασία.



Εικόνα 2-4: Διάγραμμα αρχιτεκτονικής του Gigascope

Σε όλες τις πλειάδες κατά την επεξεργασία του, προστίθεται ένα ακόμη χαρακτηριστικό για τη λειτουργία του χρονοσήμου και εν συνεχεία τα ρεύματα συνεχίζουν μέχρι την έξοδο τους από το σύστημα. Ειδικά τα ερωτήματα που σχετίζονται με την παρακολούθηση του δικτύου

κάνουν ρητή αναφορά στο χρονόσημο τους. Τα ερωτήματα που υποβάλλονται στο σύστημα, περνούν από τη διαδικασία μεταγλώττισης (compiler) και παράγονται τμήματα κώδικα σε C και C++, τα οποία και τελικά εκτελούνται, επιστρέφοντας στους χρήστες τα εξαγόμενα ρεύματα δεδομένων. Τα ερωτήματα στο Gigascope υποβάλλονται με τη χρήση της γλώσσας GSQL που αποτελεί μια παραλλαγή της SQL. Τα αποτελέσματα των ερωτημάτων μπορούν όπως είπαμε να αποθηκευτούν για περαιτέρω επεξεργασία.

2.14.3. STREAM

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. Η ανάπτυξη έχει τρεις κυρίως στόχους:

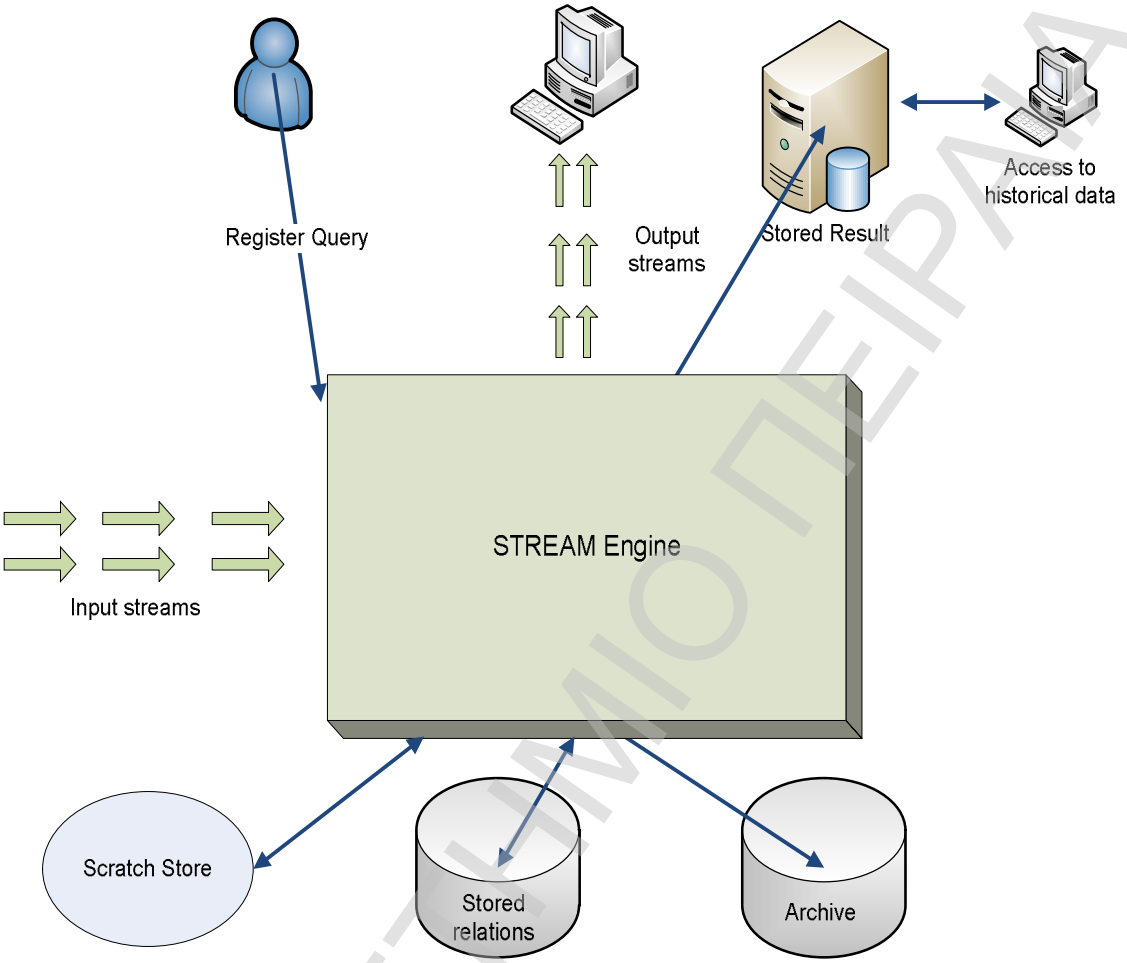
- Ø Ένα ευέλικτο τρόπο διεπαφής (interface) προκειμένου να διευκολύνεται η ανάγνωση και η εγγραφή ρευμάτων δεδομένων, ως μέρος μιας ιεραρχικής διαχείρισης του αποθηκευτικού χώρου.
- Ø Την αποτελεσματική επεξεργασία των ερωτημάτων διαρκείας που διατυπώνονται σε SQL ή με τελεστές περιβάλλον API για την υποβολή των ερωτημάτων διαρκείας και
- Ø 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). Το σχήμα 2-5 παρουσιάζει την αρχιτεκτονική λειτουργίας του συστήματος.

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 πλάνο 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 πλάνων 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.



Εικόνα 2-5: Διάγραμμα αρχιτεκτονικής του STREAM

Ένα πλάνο 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 πλάνων 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.

2.14.4. TelegraphCQ

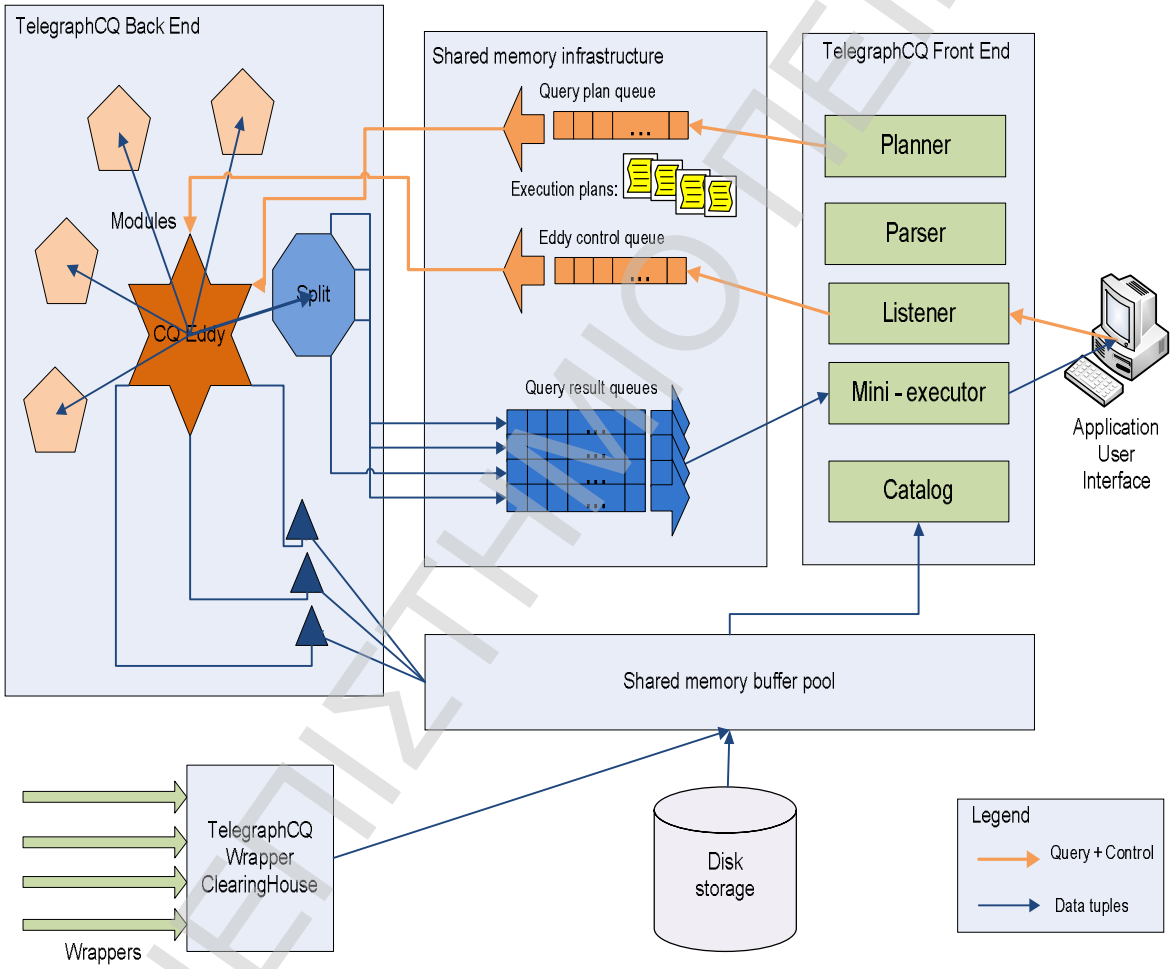
Το σύστημα αυτό αποτελεί μετεξέλιξη του πρωτοτύπου Telegraph που συντονίζεται από το Πανεπιστήμιο Berkeley από το 2000, με κύριο στόχο την ανάπτυξη μιας αρχιτεκτονικής προσαρμοζόμενης στη ροή των δεδομένων κυρίως σε δικτυακά περιβάλλοντα, με έμφαση στα δίκτυα αισθητήρων. Ωστόσο, η αναγκαιότητα αντιμετώπισης 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.

Το TelegraphCQ (όπως και το CAPE), εισάγει μια πωτοτυπία στην αρχιτεκτονική και αντιμετώπιση των ερωτημάτων διαρκείας. Ετσι θεωρείται ότι δεν είναι μόνο τα δεδομένα που εμφανίζουν μεταβλητότητα και ρέουν μέσα στο δίκτυο, αλλά ότι και τα ίδια τα ερωτήματα διαρκείας μπορούν και πρέπει να αλλάζουν δυναμικά με την πάροδο του χρόνου. Ετσι εισάγεται η έννοια της προσαρμοστικότητας των τελεστών και γενικότερα των μηχανισμών δημιουργίας πλάνων εκτέλεσης. 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 πλάνο εκτέλεσης (adaptive query plan). Το σχήμα 2-6 παρουσιάζει την αρχιτεκτονική λειτουργίας του συστήματος.



Εικόνα 2-6: Διάγραμμα αρχιτεκτονικής του TelegraphCQ

Κάθε πλάνο 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

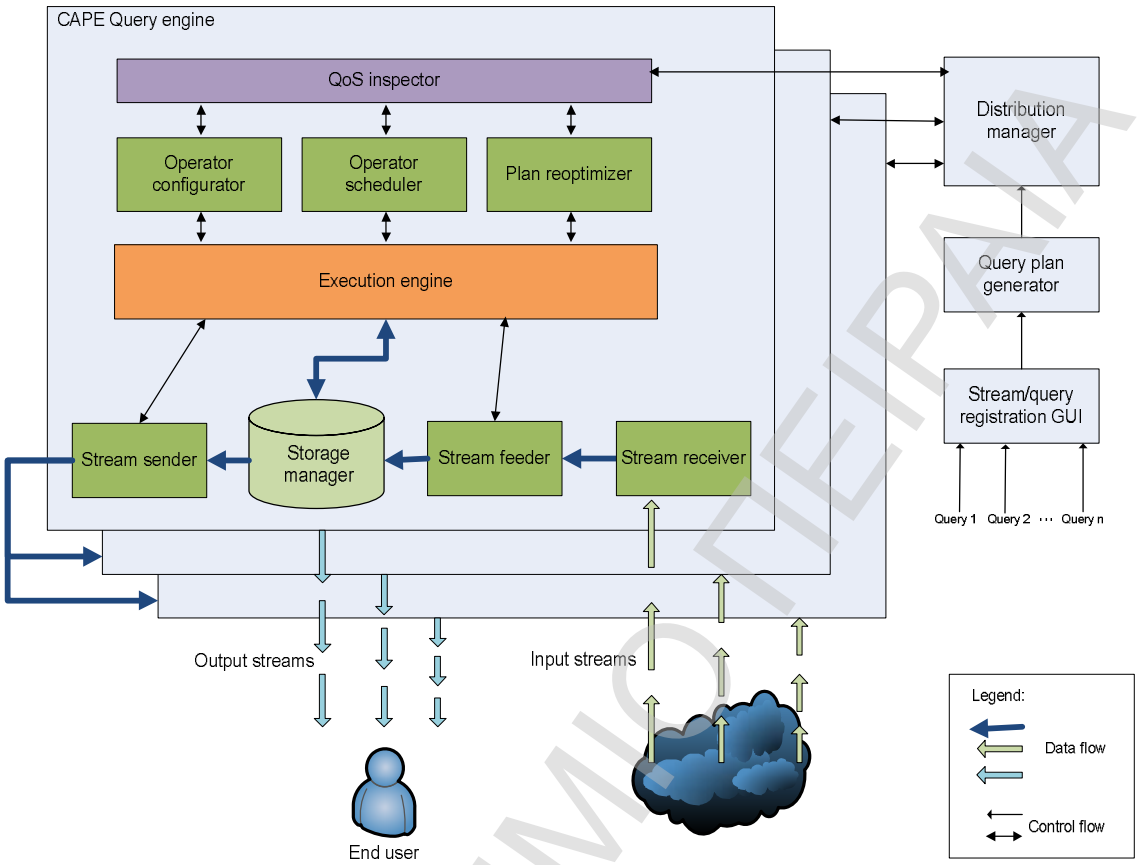
διοχετεύονται αδιαλείπτως στους τελεστές των ερωτημάτων πλειάδα προς πλειάδα. Τα Flux (Fault-tolerant Loadbalancing eXchange) δρομολογούν τις πλειάδες μεταξύ των επεξεργαστών μιας συστοιχίας (cluster) ώστε να επιτύχουν παραλληλισμό της εκτέλεσης με εξισορρόπηση φόρτου (load balancing) και ανοχή σε σφάλματα (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 πλάνο εκτέλεσης του ερωτήματος. Έπειτα, τα προσχέδια διοχετεύονται δυναμικά σε μια ουρά που έχει δημιουργηθεί στο κοινό τμήμα μνήμης του συστήματος. Απ' εκεί, ο Εκτελεστής (Executor) επιλέγει συνεχώς τα πλέον πρόσφατα προσχέδια και τα προσθέτει στα ήδη εκτελούμενα ερωτήματα. Τα αποτελέσματα της επεξεργασίας περιέρχονται σε ουρές εξόδου στο κοινό τμήμα της μνήμης, απ' όπου ο Ακροατής (Listener) τα κατευθύνει στον αντιπρόσωπο (proxy) για να διανεμηθούν τελικά στους χρήστες.

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-ανασχετικών 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.

2.14.5. CAPE

Το CAPE (**C**onstraint Aware **A**daptive Stream **P**rocessing **E**ngine) είναι ένα σύστημα διαχείρισης ρευμάτων δεδομένων το οποίο ξεχώρισε για την προσαρμοστικότητα του και την ικανότητα του να εκτιμά ερωτήματα διαρκείας σε εφαρμογές ροών δεδομένων υψηλών απαιτήσεων. Είναι ένα σύστημα που εκμεταλλεύεται κυρίως συστοιχίες κόμβων (cluster machines) και η αρχιτεκτονική λειτουργίας του προσφέρει ειδικά χαρακτηριστικά για αυτό. Το σχήμα 2-7 παρουσιάζει την αρχιτεκτονική λειτουργίας του συστήματος.



Εικόνα 2-7: Διάγραμμα αρχιτεκτονικής του CAPE

Τα βασικά δομικά στοιχεία του CAPE είναι ο Διαμορφωτής τελεστών (Operator configurator), ο χρονοπρογραμματιστής τελεστών (operator scheduler), ο επανα-βελτιστοποιητής πλάνων (plan reoptimizer) και ο διαχειριστής κατανομής (distribution manager). Τα στοιχεία αυτά αποτελούν τους 4 δομικούς μηχανισμούς προσαρμοστικότητας του συστήματος. Όταν ο μηχανισμός εκτέλεσης (execution engine) ξεκινήσει την εκτέλεση ενός ερωτήματος διαρκείας, ο επιθεωρητής ποιότητας υπηρεσιών (QoS inspector) ξεκινά τη συλλογή στατιστικών δεδομένων τα οποία αποστέλλονται στους 4 μηχανισμούς προσαρμοστικότητας που αναφέραμε, μαζί με προδιαγραφές από τον επιθεωρητή ποιότητας υπηρεσιών, έτσι ώστε να αποφασίσουν αυτοί εάν θα πρέπει να προσαρμόσουν τη συμπεριφορά τους με βάση τα νέα δεδομένα και απαιτήσεις. Η προσαρμογή αυτή λαμβάνει χώρα σε τέσσερα επίπεδα, (όσοι και οι μηχανισμοί προσαρμοστικότητας) ως εξής:

- Καταρχήν ο διαμορφωτής τελεστών αναλαμβάνει την αλλαγή της συμπεριφοράς των ανασχετικών τελεστών (blocking operators) και των τελεστών διατήρησης κατάστασης (stateful operators) έτσι ώστε να καταστεί δυνατή η συνέχιση της εκτέλεσης τους.
- Εν συνεχεία χρονοπρογραμματιστής τελεστών αναλαμβάνει την αλλαγή της σειράς εκτέλεσης των τελεστών με τη βοήθεια των παρεχόμενων στατιστικών από τον επιθεωρητή ποιότητας υπηρεσιών.
- Ακολουθεί ο επανα-βελτιστοποιητής πλάνων ο οποίος είναι σε θέση να αλλάξει σε πραγματικό χρόνο το πλάνο εκτέλεσης του ερωτήματος εάν αυτό κριθεί αναγκαίο.

- Ø Τέλος, ο διαχειριστής κατανομής μπορεί να προωθήσει το πλάνο εκτέλεσης σε άλλο κόμβο όταν ο συγκεκριμένος κόμβος έχει πρόβλημα διαθέσιμων πόρων.

Το CAPE βασίζεται στα παρακάτω χαρακτηριστικά:

1. Τελεστές που αντιδρούν με βάση τους περιορισμούς των ερωτημάτων διαρκείας (Constraint-exploiting reactive query operators). Με το χαρακτηριστικό αυτό, επιτυγχάνεται η βελτιστοποίηση των ερωτημάτων και η μείωση των απαιτήσεων σε υπολογιστική ισχύ, καθώς και η βελτίωση του χρόνου απόκρισης.
2. Πλαίσιο για τον αυτόματο έλεγχο εκτίμηση και χρονοπρογραμματισμό (Introspective execution scheduling framework). Το χαρακτηριστικό αυτό δίνει τη δυνατότητα επιλογής δυναμικά, από ανά πλήθος αλγορίθμων, τον αλγόριθμο εκείνο που βοηθά το ερώτημα να πετύχει τη μέγιστη βελτιστοποίηση.
3. Βελτιστοποίηση πλάνων εκτέλεσης σε πραγματικό χρόνο (Online query plan optimization and migration). Πρόκειται για τη δυνατότητα συνεχούς επαναπροσδιορισμού των πλάνων εκτέλεσης με βάση τα δεδομένα που εισέρχονται στο σύστημα έτσι ώστε κατά περίπτωση να επιλέγεται ο βέλτιστος αλγόριθμος εκτέλεσης.
4. Πλαίσιο για την εκτέλεση των ερωτημάτων διαρκείας σε κατανεμημένα συστήματα (Adaptive query plan distribution framework). Το χαρακτηριστικό αυτό είναι από τα πλέον ισχυρά σημεία του CAPE αφού διανέμει το φόρτο εργασίας σε διαφορετικά συστήματα επιτυχαίνοντας ισοκατανομή φόρτου, παράλληλη εκτέλεση, βελτιστοποίηση σε χρόνους απόκρισης και ελαχιστοποίηση λαθών που πιθανά θα προέκυπταν από δεδομένα που απορρίφθηκαν γιατί δεν πρόλαβαν να επεξεργαστούν μέσα στο διαθέσιμο χρονικό παράθυρο.

Η βελτιστοποίηση των ερωτημάτων διαρκείας στο CAPE επιτυγχάνεται με την ταυτόχρονη χρήση όλων των παραπάνω χαρακτηριστικών του συστήματος κάτι που θα εγκυμονούσε κινδύνους αν το σύστημα δεν είχε τη δυνατότητα να συντονίζει τις αλληλεπιδράσεις μεταξύ τους. Αυτό που συμβαίνει στην πράξη είναι ότι κάθε ένα από τα χαρακτηριστικά αυτά επικοινωνεί με τα υπόλοιπα με ένα τρόπο ενθυλάκωσης – κατηγοριοποίησης σε επίπεδα και αφού αξιολογηθούν από καθένα χωριστά, το περιεχόμενο και ο όγκος των δεδομένων, ο διαθέσιμος και απαιτούμενος χρόνος εκτέλεσης, οι παράμετροι και περιορισμοί του ερωτήματος, εν συνεχεία εκτελούνται τα ερωτήματα.

Η αιχμή του δόρατος στο CAPE είναι το τελευταίο από τα τέσσερα χαρακτηριστικά που αναφέραμε. Η κατανεμημένη επεξεργασία των ερωτημάτων η οποία πρέπει να εφαρμοστεί για να υποστηρίξει την απαραίτητη επεκτασιμότητα σε συστήματα που πρέπει να υποστηρίξουν μεγάλους όγκους δεδομένων και υψηλές απαιτήσεις ανάλυσης. Το σχέδιο ενός κατανεμημένου συστήματος για ερωτήματα διαρκείας χαρακτηρίζεται από μια πρόσθετη πολυπλοκότητα. Σε ένα τέτοιο σύστημα, τα ρεύματα δεδομένων μπορούν να είναι άπειρα και οι αρχικές στατιστικές κόστους για τα ρεύματα δεδομένων να είναι χαρακτηριστικά άγνωστες. Επιπλέον, οι στατιστικές κόστους συνεχίζουν να αλλάζουν κατά τη διάρκεια του χρόνου. Η κατανεμημένη επεξεργασία ρευμάτων εξετάζει δύο κρίσιμες ερωτήσεις:

1. πώς κατανέμονται αρχικά τα πλάνα εκτέλεσης ερωτημάτων για τα οποία δίνονται ελάχιστα ή ενδεχομένως καμία πληροφορία δαπανών, και
2. πώς προσαρμόζεται αποτελεσματικά ένα κατανεμημένο ερώτημα διαρκείας με βάση τις αλλαγές στο περιβάλλον εκτέλεσης κατά τη διάρκεια εκτέλεσης αυτού.

Η απάντηση στα παραπάνω δύο ερωτήματα είναι τουλάχιστον μέχρι σήμερα, η χρήση εμπειρικών κανόνων που προκύπτουν από συστήματα προσομοίωσης.

Αντίθετα από το Aurora και άλλα συστήματα που εστιάζουν στα ερευνητικά ζητήματα ενός δικτύου μεγάλης περιοχής, το CAPE στοχεύει σε ένα τοπικό περιβάλλον συστάδων που συνδέεται με ένα υψηλό δίκτυο ταχύτητας. Η σχετική έρευνα δείχνει ότι η αρχιτεκτονική του CAPE είναι τέτοια ώστε να παράγει ένα ελαφρύ, αλλά αποτελεσματικό πλάνο εκτέλεσης.

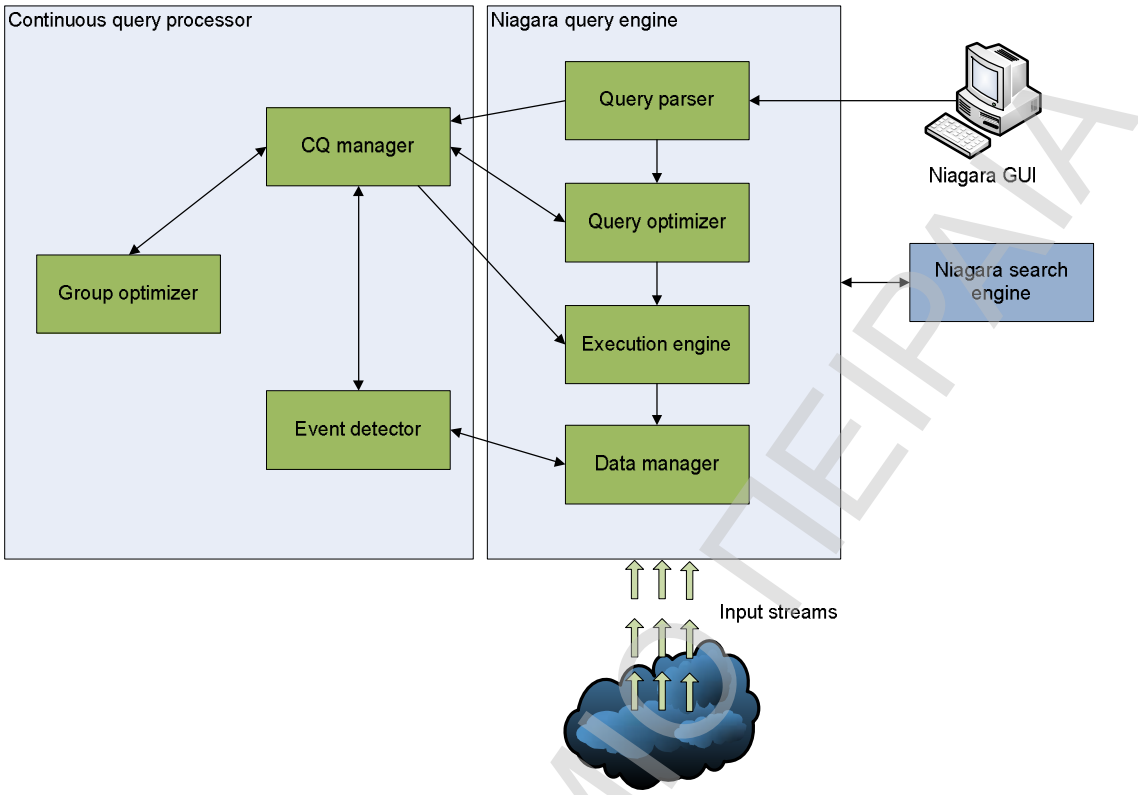
2.14.6. NiagaraCQ

Το NiagaraCQ είναι το υποσύστημα βελτιστοποίησης ερωτημάτων διαρκείας που υπάγεται στο πρόγραμμα Niagara, το οποίο είναι ένα καθαρό σύστημα διαχείρισης δεδομένων που αναπτύσσεται στο πανεπιστήμιο του Wisconsin και του Όρεγκον. Το NiagaraCQ υποστηρίζει τη συνεχή επεξεργασία ερωτημάτων διαρκείας με τη χρήση πολλών κατανεμημένων αρχείων XML σε διαφορετικά συστήματα. Διάφορες τεχνικές χρησιμοποιούνται για να καταστήσουν το NiagaraCQ επεκτάσιμο και αποδοτικό:

- Το NiagaraCQ υποστηρίζει την επαυξητική αξιολόγηση των ερωτημάτων διαρκείας με την εξέταση μόνο της αλλαγμένης μερίδας κάθε ενημερωμένου αρχείου XML και όχι ολόκληρου του αρχείου. Δεδομένου ότι συχνά μόνο μια μικρή μερίδα κάθε αρχείου ενημερώνεται, αυτή η στρατηγική μπορεί να περιορίσει σημαντικά ποσά υπολογισμού. Ένα άλλο πλεονέκτημα της επαυξητικής αξιολόγησης είναι ότι η επαναλαμβανόμενη αξιολόγηση αποφεύγεται και μόνο τα νέα αποτελέσματα επιστρέφονται στους χρήστες.
- Το NiagaraCQ μπορεί να ελέγξει και να ανιχνεύσει τις αλλαγές στις πηγές δεδομένων (data sources) χρησιμοποιώντας τα πρότυπα push και poll σε ετερογενείς πηγές.
- Λόγω της κλίμακας του συστήματος, οι πληροφορίες των ερωτημάτων διαρκείας και των προσωρινών αποτελεσμάτων δεν μπορούν να αποθηκευτούν στη μνήμη. Ένας μηχανισμός διαχείρισης μνήμης (caching mechanism) χρησιμοποιείται για την αποδοτική εκτέλεση των ερωτημάτων με περιορισμένα ποσά μνήμης.

Στο σχήμα 2-8 παρουσιάζεται η αρχιτεκτονική λειτουργίας του συστήματος. Ο διαχειριστής ερωτημάτων διαρκείας (continuous query manager) είναι ο πυρήνας του συστήματος. Παρέχει την κατάλληλη διεπαφή στους χρήστες για τον ορισμό ερωτημάτων διαρκείας αλλά και ενεργοποιεί το μηχανισμό εκτέλεσης ερωτημάτων για την επεξεργασία τους. Ο βελτιστοποιητής ομάδων (group optimizer) εκτελεί λειτουργίες βελτιστοποίησης ερωτημάτων τα οποία ανάλογα με τους τελεστές και τα χαρακτηριστικά τους, ομαδοποιούνται σε ομάδες. Ο ανιχνευτής συμβάντων (event detector), έχει ως αποστολή την ανίχνευση συμβάντων αλλά και αλλαγών στις πηγές δεδομένων. Τέλος, ο διαχειριστής δεδομένων (Niagara data manager) παρέχει τη δυνατότητα της σταδιακής αποτίμησης της εκτέλεσης των ερωτημάτων διαρκείας.

Όταν ένα ερώτημα διαρκείας εκτελείται στο σύστημα, καταρχήν αποστέλλεται στον βελτιστοποιητή ομάδων έτσι ώστε να αποφασιστεί η διαχώριση ή όχι του ερωτήματος σε περισσότερα του ενός. Εν συνεχεία προστίθεται η πληροφορία πηγής (αρχείου) και χρόνου από τον ανιχνευτή συμβάντων και ενεργοποιείται ο διαχειριστής δεδομένων για την έναρξη ανίχνευσης αλλαγών στα αρχεία. Ο διαχειριστής ερωτημάτων διαρκείας εκτελεί το ερώτημα και το αποτέλεσμα επιστρέφεται στο χρήστη.



Εικόνα 2-8: Διάγραμμα αρχιτεκτονικής του NiagaraCQ

Προκειμένου να αντιμετωπιστεί ένας μεγάλος αριθμός χρηστών με διαφορετικά ενδιαφέροντα, ένα σύστημα διαχείρισης ρευμάτων δεδομένων θα πρέπει να είναι ικανό να διαχειρίζεται ένα μεγάλο αριθμό ωθήσεων που εκφράζονται ως σύνθετες ερωτήσεις διαρκείας στα ανά τον ιστό σύνολα δεδομένων. Ο στόχος του προγράμματος Niagara είναι να αναπτυχθεί ένα κατανεμημένο σύστημα βάσεων δεδομένων για τα κατανεμημένα σύνολα δεδομένων XML που χρησιμοποιούν μια γλώσσα διατύπωσης ερωτήσεων όπως η XML-QL. Η στατιστική υπόθεση είναι ότι πολλές ερωτήσεις θα τείνουν να είναι παρόμοιες η μία με την άλλη και έτσι το σύστημα θα είναι σε θέση να χειριστεί τα εκατομμύρια των συνεχών ερωτήσεων με εκτέλεση ουσιαστικά ομαδοποιημένων παρόμοιων ερωτήσεων.

Η βελτιστοποίηση που γίνεται σε ομαδοποιημένα ερωτήματα έχει τα ακόλουθα οφέλη. Κατ' αρχάς, οι ομαδοποιημένες ερωτήσεις μπορούν να μοιραστούν το αποτέλεσμα του υπολογισμού. Δεύτερον, τα κοινά σχέδια εκτέλεσης των ομαδοποιημένων ερωτήσεων μπορούν να συνυπάρξουν στην ίδια περιοχή μνήμης, ελαχιστοποιώντας την πρόσβαση στο σύστημα δίσκων. Τρίτον, η ομαδοποίηση καθιστά πιθανή την εκτέλεση ερωτημάτων διαρκείας ομαδικά και όχι ξεχωριστά, αποφεύγοντας έτσι τις περιττές κλήσεις συστήματος.

Οι ομάδες δημιουργούνται για τις υπάρχουσες ερωτήσεις σύμφωνα με τις υπογραφές τους (expression signatures), οι οποίες αντιπροσωπεύουν παρόμοιες δομές μεταξύ των ερωτήσεων. Οι υπογραφές αντιπροσωπεύουν την ίδια συντακτική δομή, αλλά ενδεχομένως διαφορετικές σταθερές τιμές, σε διαφορετικές ερωτήσεις. Πρόκειται ουσιαστικά για μια συγκεκριμένη εφαρμογή της έννοιας των υπογραφών. Οι ομάδες επιτρέπουν στα κοινά μέρη δύο ή περισσότερων ερωτήσεων να μοιράζονται. Κάθε μεμονωμένη ερώτηση σε μια ομάδα

ερωτημάτων μοιράζεται τα αποτελέσματα από την εκτέλεση του σχεδίου της ομάδας. Όταν υποβάλλεται μια νέα ερώτηση, ο βελτιστοποιητής (optimizer) της ομάδας θεωρεί τις υπάρχουσες ομάδες ως πιθανές επιλογές βελτιστοποίησης. Η νέα ερώτηση συγχωνεύεται σε εκείνες τις υπάρχουσες ομάδες που ταιριάζουν με τις υπογραφές της. Ενώ αυτή η στρατηγική είναι πιθανό να οδηγήσει σε ομάδα μη βέλτιστης απόδοσης (sub-optimal), από την άλλη, μειώνει το κόστος της βελτιστοποίησης ομάδας σημαντικά.

Το πιο σημαντικό είναι το πως εφαρμόζεται η συγκεκριμένη τεχνική σε ένα δυναμικό περιβάλλον. Δεδομένου ότι οι ερωτήματα διαρκείας προστίθενται και αφαιρούνται συχνά, είναι δυνατό οι τρέχουσες ομάδες να καταστούν ανεπαρκείς. Η "δυναμική ανασυγκρότηση" (dynamic re-grouping) θα ήταν χρήσιμη έτσι ώστε να ανασυγκροτείται μέρος των ερωτημάτων ή όλες οι ερωτήσεις, είτε περιοδικά είτε όταν μειώνεται η απόδοση του συστήματος (performace degradation) κάτω από κάποιο όριο (threshold). Αυτό το χαρακτηριστικό αποτελεί τομέα έρευνας για το NiagaraCQ.

2.15. **Γλώσσες Ερωταποκρίσεων ΣΔΡΔ**

2.15.1. **Εισαγωγή**

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 πλάνο 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 πλάνου εκτέλεσης από τον χρήστη, πιθανόν και με την βοήθεια κάποιας κατάλληλης *γραφικής διεπαφής χρήστη* (***Graphical User Interface***). Όπως όμως αναφέρθηκε, τα ερωτήματα διαρκείας εστιάζουν συνήθως στην επεξεργασία της πιο πρόσφατης πληροφορίας, εγείροντας την ανάγκη αφενός για χρονική σήμανση των στοιχείων, αφετέρου για την υλοποίηση κατάλληλων τελεστών που θα αποσπούν και θα επεξεργάζονται τα πλέον πρόσφατα στοιχεία. Η χρονική σήμανση επιτυγχάνεται με την προσθήκη ενός πεδίου *χρονοσήμου* (***timestamp***) στις πλειάδες των ρευμάτων, ενώ εισάγεται η έννοια των παραθύρων για να δηλωθεί η έμφαση των ερωτημάτων στην πιο πρόσφατη πληροφορία.

2.15.2. STREAM: CQL (Continuous Query Language)

Σε αυτήν την επέκταση της γλώσσας SQL υποστηρίζεται η επεξεργασία τόσο ρευμάτων δεδομένων όσο και στατικών σχέσεων. Η CQL αποτελείται από τρία δομικά στοιχεία: α) την σχεσιακή γλώσσα ερωταποκρίσεων SQL για τους τελεστές σχεσιακών πινάκων β) μία γλώσσα προσδιορισμού παραθύρων η οποία ακολουθεί συντακτικά στοιχεία της SQL-99 για την μετατροπή των ρευμάτων σε σχεσιακούς πίνακες και γ) τρεις ειδικούς τελεστές για την μετατροπή σχεσιακών πινάκων σε ρεύματα: ISTREAM για τις νέες πλειάδες του προσωρινού σχεσιακού πίνακα, DSTREAM για τις πλειάδες που διαγράφηκαν από τον πίνακα και RSTREAM για την παροχή όλων των πλειάδων του προσωρινού πίνακα.

2.15.3. Stream Mill System: Expressive Stream Language (ESL)

Υποστηρίζει αποτελεσματικά ένα ευρύ φάσμα εφαρμογών όπως data stream mining, streaming XML processing, time-series queries, and RFID event processing. Η ESL υποστηρίζει φυσικά και λογικά παράθυρα τόσο στις προκαθορισμένες από το σύστημα συναθροίσεις όσο και σε αυτές που ορίζονται από το χρήστη (User Defind Aggregated -UDAs), χρησιμοποιώντας ένα απλό πλαίσιο που εφαρμόζει ομοιόμορφα και στις δύο λειτουργίες. Μάλιστα, πέραν του ότι παρέχει την εγγενή συγγραφή κώδικα για χειρισμό παραθύρων, υποστηρίζει και άλλες γλώσσες έτσι ώστε να μπορούν να ενσωματωθούν εξωτερικές διαδικασίες σε αυτήν.

Η ESL μεταχειρίζεται τα ρεύματα δεδομένων ως απεριόριστες διαταγμένες ακολουθίες πλειάδων. Τα ρεύματα δεδομένων δηλώνονται χρησιμοποιώντας μια εντολή CREATE STREAM. Η εντολή καθορίζει επίσης τον τύπο χρονοσήμανσης που συνδέεται με το ρεύμα δεδομένων. Η ESL υποστηρίζει τους ακόλουθους τρεις τύπους χρονοσημάνσεων:

1. εξωτερικά χρονόσημα (external timestamps)
2. εσωτερικά χρονόσημα (internal timestamps)
3. λανθάνοντα χρονόσημα (latent timestamps)

Τα εξωτερικά χρονόσημα περιλαμβάνονται στις πλειάδες άφιξης, και διευκρινίζονται με μια δήλωση ORDER BY. Τα εσωτερικά χρονόσημα δηλώνονται σε πλειάδες με απουσία ORDER BY δηλώσεων. Τέλος, υπάρχουν περιπτώσεις όπου πλειάδες καταφτάνουν στο σύστημα των οποίων τα χρονόσημα είναι εκτός χρονικών παραθύρων λόγο π.χ. καθυστερήσεων δικτύου. Οι πλειάδες αυτές δηλώνονται με λανθάνοντα χρονόσημα και ανακατευθύνονται σε ένα αυτόνομο ρεύμα δεδομένων το οποίο χειρίζεται με συγκεκριμένο τρόπο έτσι ώστε είτε οι πλειάδες του να απορριφθούν είτε να τοποθετηθούν στα σωστά ρεύματα δεδομένων με τη σωστή χρονική σήμανση. Ο μηχανισμός αυτός έχει αρκετά κοινά με το αντίστοιχο μηχανισμό του συστήματος Aurora.

2.15.4. Aurora: SQuAl (Stream Query Algebra)

Τα ερωτήματα που διατυπώνονται στην γλώσσα SQuAl μετασχηματίζονται σε γράφους κατάλληλα συνδεδεμένων *τόξων* και *κουτιών*. (boxes and arrows) Οι γράφοι μπορούν εναλλακτικά να σχηματιστούν μέσω κατάλληλης γραφικής διεπαφής. Τα *κουτιά* αναπαριστούν τους παρεχόμενους τελεστές, ενώ τα *τόξα* αντιστοιχούν σε ρεύματα δεδομένων και χρησιμοποιούνται για τις συνδέσεις μεταξύ κουτιών. Οι τελεστές λειτουργούν με ρεύματα δεδομένων, ενώ η επεξεργασία στατικών σχέσεων υποστηρίζεται μόνο κατά έμμεσο τρόπο. Τα παράθυρα αποτελούν έμφυτο χαρακτηριστικό κάθε τελεστή που τα χρειάζεται και ορίζονται μέσω κατάλληλων παραμέτρων του τελεστή κατά την διατύπωση του ερωτήματος.

2.15.5. TelegraphCQ StreaQueL (Stream Query Language)

Πρόκειται για μία δηλωτική γλώσσα η οποία αποδίδει ιδιαίτερη έμφαση στον προσδιορισμό διαφόρων τύπων παραθύρων. Κατά βάση, στην τυπική σύνταξη ενός ερωτήματος σε SQL προσαρτάται μια επαναληπτική δομή (FOR-loop) δηλώνοντας την ακολουθία των παραθύρων που θα χρησιμοποιηθούν για να απομονώσουν τα στοιχεία των αντίστοιχων ρευμάτων. Πέρα από τα ρεύματα, υποστηρίζεται η επεξεργασία και σχεσιακών πινάκων.

2.15.6. Gigascope: GSQL

Η γλώσσα αυτή έχει μορφή γλώσσας SQL, αλλά εφαρμόζεται αποκλειστικά σε ρεύματα δεδομένων. Κάθε ρεύμα θεωρείται ότι διαθέτει ένα ή περισσότερα γνωρίσματα βάσει των οποίων μπορεί να οριστεί κάποια χρονική διάταξη. Κατά την διατύπωση των ερωτημάτων ορίζονται *πρωτόκολλα* με την βοήθεια των οποίων: α) ερμηνεύονται τα περιεχόμενα των εισερχόμενων ρευμάτων, β) καθορίζεται ο τρόπος προσδιορισμού της διάταξης τους και γ) προσομοιώνεται η λειτουργία των παραθύρων. Η προσομοίωση των παραθύρων πραγματοποιείται με συνθήκες επιλογής πάνω στα γνωρίσματα που καθορίζουν την διάταξη των στοιχείων σύμφωνα με το επιλεγμένο *πρωτόκολλο*. Τέλος, λαμβάνεται ιδιαίτερη μέριμνα για την ενημέρωση των αποτελεσμάτων ακόμα και κατά την απουσία εισερχόμενων στοιχείων πληροφορίας.

2.15.7. Εντολές Διαχείρισης Ρευμάτων Δεδομένων

Στην ενότητα αυτή περιγράφονται δύο από τις εντολές που παρέχει η γλώσσα διαχείρισης ρευμάτων δεδομένων StreaQueL του συστήματος TelegraphCQ καθώς και κάποια παραδείγματα που θα βοηθήσουν στην κατανόηση της υλοποίησης δομών διαχείρισης ρευμάτων δεδομένων. Οι εντολές που περιγράφονται είναι οι Create Stream και η Select. Να

σημειωθεί ότι το TelegraphCQ, όπως άλλωστε προαναφέραμε, είναι πειραματικό ΣΔΡΔ, έτσι, θα έχει ενδιαφέρον να δούμε πως έγινε η αρχική υλοποίηση των εντολών διαχείρισης και μάλιστα σε σχέση με την εξέλιξη των δομών αυτών στο εμπορικό ΣΔΡΔ StreamBase, το οποίο αναλύεται σε επόμενο κεφάλαιο.

2.15.7.1. CREATE STREAM

Σύνοψη

CREATE STREAM *stream* (*column_name data_type* [TIMESTAMPCOLUMN] [, ...])
TYPE [ARCHIVED | UNARCHIVED]

Περιγραφή

Η εντολή CREATE STREAM δημιουργεί ένα ρεύμα που προέρχεται από αρχειοθετημένο ή μη-αρχειοθετημένο ρεύμα στη βάση δεδομένων. Ένα αρχειακό (archived) ρεύμα δεδομένων δημιουργείται από δεδομένα που ανακτώνται από συσκευές αποθήκευσης (σκληρούς δίσκους). Ένα μη-αρχειακό (unarchived) ρεύμα δεδομένων δημιουργείται από δεδομένα που βρίσκονται στη μνήμη του συστήματος. Ένα μη-αρχειακό ρεύμα, δεν μπορεί να αποθηκευτεί σε σκληρό δίσκο. Σε ένα αρχειακό ρεύμα, επιτρέπεται μόνο η προσθήκη (append) πλειάδων, έτσι το ρεύμα αυξάνεται μονοτονικά.

Το όρισμα TIMESTAMPCOLUMN χρησιμοποιείται για να ορίσει το πεδίο εκείνο της πλειάδας το οποίο αποτελεί το χρονόσημο της. Μια πλειάδα μπορεί να έχει μόνο ένα όρισμα τύπου χρονοσήμου, αλλά μπορεί να έχει πολλά πεδία που να περιέχουν χρονική πληροφορία. Το χρονόσημο – όρισμα χρησιμοποιείται στην περίπτωση εκτέλεσης εντολών επιλογής που περιέχουν παραθυρικές προτάσεις. Τέλος, δεν είναι δυνατή η τροποποίηση των στοιχείων του ρεύματος.

Ορίσματα

Stream: Το όνομα του ρεύματος.

column_name: Όνομα πεδίου

data_type: Τύπος δεδομένων του πεδίου

TIMESTAMPCOLUMN: Το πεδίο που αποτελεί το χρονόσημο της πλειάδας. Στην τρέχουσα έκδοση της γλώσσας πρέπει να έχει την ονομασία: *tcqtime*.

ARCHIVED: Αρχειακό ρεύμα

UNARCHIVED: Μη-αρχειακό ρεύμα

Παραδείγματα

```
CREATE STREAM measurements (
        tcqtime   TIMESTAMP TIMESTAMPCOLUMN,
        stationid INTEGER,
        speed     REAL)
TYPE ARCHIVED;
```

```
CREATE STREAM tinydb (
        tcqtime     TIMESTAMP TIMESTAMPCOLUMN,
        light       REAL,
        temperature REAL)
TYPE UNARCHIVED;
```

2.15.7.2. SELECT

Σύνοψη

Χρησιμοποιείται για την άντληση δεδομένων από ένα ρεύμα δεδομένων.

```
SELECT      <select_list>
        FROM        <relation_and_<pstream_list>
        WHERE       <predicate>
        GROUP BY    <group_by_expressions>
        WINDOW      stream[interval-expr], ...
        ORDER BY    <order_by_expressions>;
```

Περιγραφή

Η εντολή αυτή συντάσσεται και χρησιμοποιείται όπως οι κλασικές εντολές SELECT. Τα σημεία στα οποία διαφοροποιείται είναι στις δηλώσεις *FROM* και *WINDOW*.

- Η δήλωση *FROM*, παίρνει σαν όρισμα όχι μόνο σχεσιακούς πίνακες αλλά και ρεύματα δεδομένων.
- Η δήλωση *WINDOW* χρησιμοποιείται για να δηλώσει την συμμετοχή παραθύρων ρευμάτων στην εντολή. Το όρισμα interval-expr ορίζει ένα χρονικό διάστημα. Για παράδειγμα, '2 seconds', '5 minutes', '3 years' κτλ. Κάθε πλειάδα η οποία καταφθάνει στο σύστημα, εκτιμάται έτσι ώστε να παραχθεί ένα νέο σύνολο πλειάδων το οποίο περιέχει όλες εκείνες τια πλειάδες για τις οποίες το χρονόσημο δεν είναι παλαιότερο από το όρισμα interval-expr. Ένα ερώτημα διαρκείας μπορεί να περιέχει περισσότερα του ενός ρεύματα δεδομένων, το καθένα με διαφορετικό παράθυρο. Στην περίπτωση αυτή το ερώτημα εκτιμάται κάθε φορά που θα καταφθάσει μια πλειάδα σε οποιοδήποτε από τα ρεύματα που συμμετέχουν. Στην περίπτωση της σύνδεσης (join) το χρονόσημο της νέας πλειάδας είναι το χρονόσημο της πλέον πρόσφατης πλειάδας εισόδου εκ των πλειάδων που ενώνονται.

Παραδείγματα

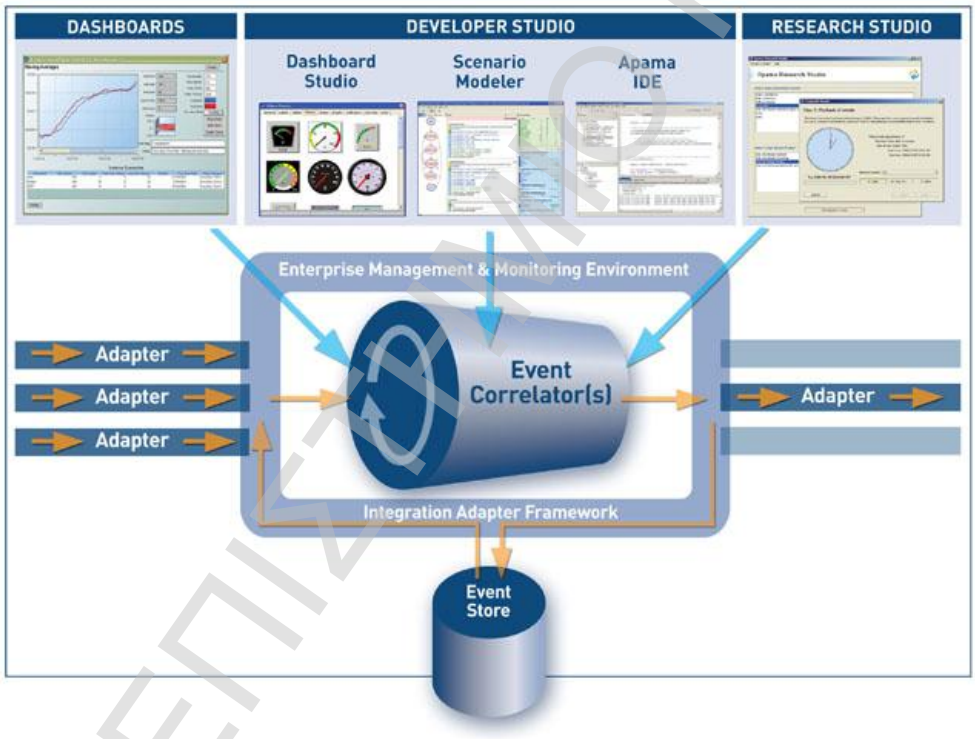
```
SELECT      ms.stationid, s.name, s.highway, s.mile, AVG(ms.speed)
FROM        measurements ms, stations s
WHERE       ms.stationid = s.stationid
GROUP BY    ms.stationid, s.name, s.highway, s.mile
WINDOW      ms ['10 minutes'];
```

2.16. **Σύντομη αναφορά σε εμπορικά ΣΔΡΔ**

Τα ΣΔΡΔ δεν έχουν μείνει όμως στο στάδιο του πειράματος. Κάποια εμπορικά προϊόντα έχουν ήδη αναπτυχθεί και δημιουργήσει μια νέα κατηγορία λογισμικού εφαρμογών, τα Συστήματα Επεξεργασίας Πολύπλοκων Συμβάντων. (CEPS – Complex Event Processing Systems). Στη συνέχεια θα αναφερθούμε επιγραμματικά στα βασικά στοιχεία τριών εξ' αυτών: Του Progress Apama, του Coral8 και του StreamBase το οποίο και χρησιμοποιήθηκε για την ανάπτυξη της εφαρμογής στα πλαίσια του πρακτικού μέρους της διατριβής.

2.16.1. Progress Apama

- Το σύστημα Apama της εταιρείας Progress, παρέχει μια πλατφόρμα για την παρακολούθηση και έλεγχο ρευμάτων δεδομένων. Τα ρεύματα αυτά αναλύονται με σκοπό την ανίχνευση γεγονότων/συμβάντων τα οποία θα ενεργοποιήσουν σε πραγματικό χρόνο την αντίδραση του συστήματος.



Εικόνα 2-9: Διάγραμμα αρχιτεκτονικής του Apama

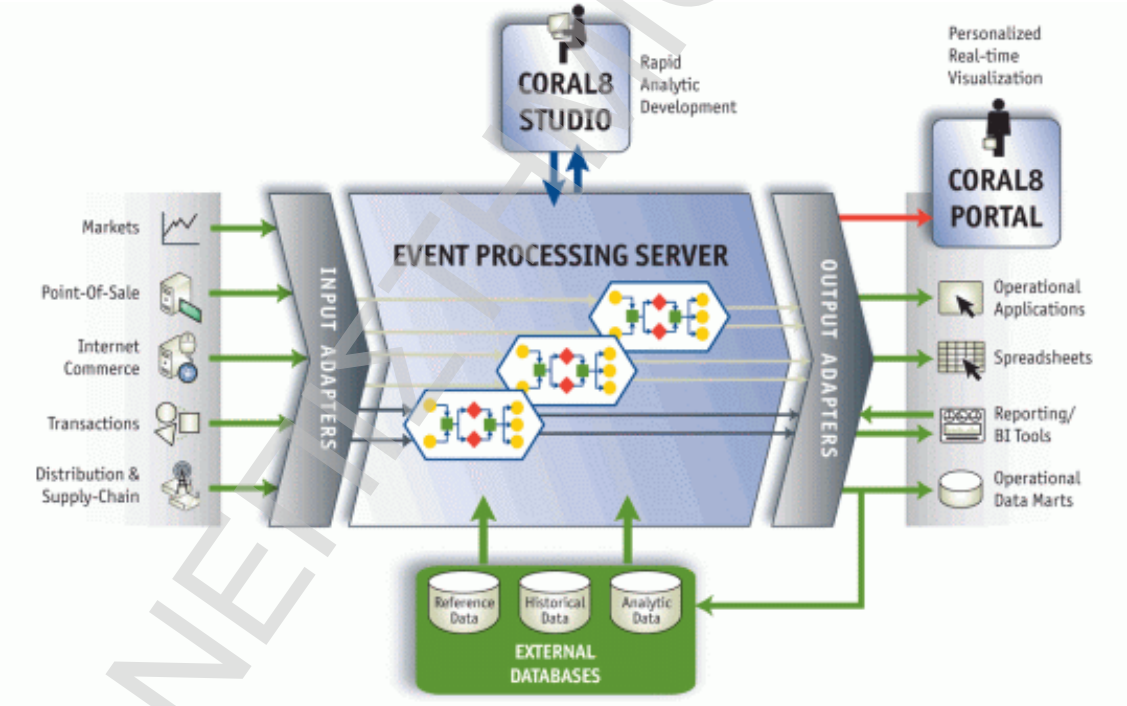
Το σύστημα, αποτελείται από τα εξής υποσυστήματα:

- Apama Event Modeler: Είναι το γραφικό σύστημα διεπαφής με το χρήστη (GUI) μέσω του οποίου σχεδιάζονται οι εφαρμογές από τον προγραμματιστή.
- Event Processing Language (EPL): Αποτελεί τη γλώσσα χειρισμού των ρευμάτων δεδομένων.

- Ø Apama Dashboard Studio: Παρέχει ένα σύνολο εργαλείων για την ανάπτυξη εφαρμογών και γραφικών διεπαφών.
- Ø Apama Research Studio: Πρόκειται για ένα γραφικό περιβάλλον για τη δημιουργία και τον έλεγχο σεναρίων της υπό ανάπτυξη εφαρμογής μέσω μηχανισμών προσομοίωσης.
- Ø Apama EventStore: Πρόκειται για το μηχανισμό μέσω του οποίου γίνεται η συλλογή των ρευμάτων εισόδου.
- Ø Apama Event Manager: Αποτελεί την καρδιά του συστήματος. Πρόκειται για ένα σύστημα διαχείρισης ροών δεδομένων.
- Ø Apama Integration Adapter Framework (IAF): Πρόκειται για μια συλλογή βιβλιοθηκών κώδικα η οποία δίνει τη δυνατότητα σύνδεσης διαφορετικών τύπων πηγών στο σύστημα, όσο και διαφορετικών τύπων παρουσίασης των αποτελεσμάτων στην έξοδο. Έτσι, δίνεται η δυνατότητα άντλησης αλλά και αποστολής της πληροφορίας από διαφορετικών τύπων αρχεία, από βάσεις δεδομένων, κτλ.

2.16.2. Coral8

Το σύστημα Coral8 προσφέρει μια πλατφόρμα ανάπτυξης εφαρμογών ρευμάτων δεδομένων. Το σημαντικό πλεονέκτημα του συστήματος είναι η γλώσσα ανάπτυξης που χρησιμοποιεί σημεία στίξης και τεχνικές pattern matching για να ελέγξει τα δεδομένα εισόδου.



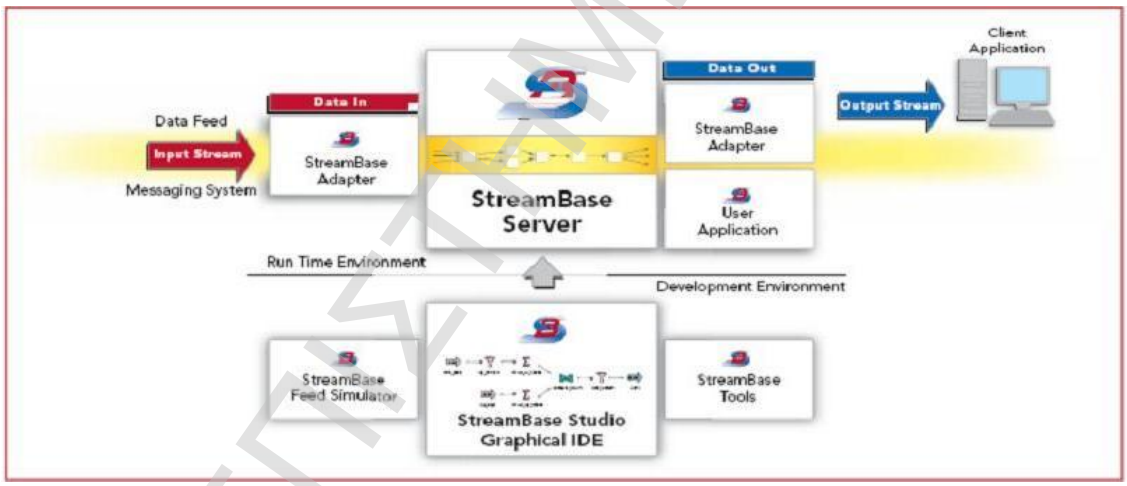
Εικόνα 2-10: Διάγραμμα αρχιτεκτονικής του Coral8

Το σύστημα αποτελείται από τα εξής υποσυστήματα:

- Coral8 Studio: Πρόκειται για το περιβάλλον ανάπτυξης εφαρμογών (GUI) του συστήματος
- Coral8 Server: Πρόκειται για το ΣΔΡΔ που είναι υπεύθυνο για την επεξεργασία των ρευμάτων δεδομένων και το συντονισμό όλων των υπολοίπων στοιχείων του συστήματος
- Coral8 Portal: Πρόκειται για το περιβάλλον διεπαφής του τελικού χρήστη με το σύστημα. Ενσωματώνει μια σειρά εργαλείων για την κατασκευή γραφικών παραστάσεων, διαγραμμάτων κτλ με σκοπό την γρηγορότερη και αμεσότερη οπτικοποίηση των αποτελεσμάτων εξόδου. Adapters: Πρόκειται για βιβλιοθήκες κώδικα και μηχανισμούς οι οποίοι επιτρέπουν τη σύνδεση διαφορετικών συστημάτων στην είσοδο και στην έξοδο του συστήματος.
- Continuous Computation Language (CCL): Είναι η γλώσσα ελέγχου των ρευμάτων δεδομένων που παρέχεται από το σύστημα.

2.16.3. StreamBase

Το σύστημα StreamBase παρέχει μια ολοκληρωμένη πλατφόρμα σχεδίασης, ανάπτυξης, και διαχείρισης ρευμάτων δεδομένων. Παρέχει ένα ολοκληρωμένο περιβάλλον σχεδίασης, ανάπτυξης και ελέγχου της εφαρμογής βασισμένο πάνω στο πρότυπο eclipse και επιτρέπει τη σύνδεση εξωτερικών πηγών/συστημάτων μέσω των ειδικών προσαρμογέων (adapters) που παρέχει.



Εικόνα 2-11: Διάγραμμα αρχιτεκτονικής του StreamBase

Επιγραμματικά το σύστημα StreamBase αποτελείται από τα παρακάτω υποσυστήματα:

- StreamBase Server: Αποτελεί την καρδιά του συστήματος. Πρόκειται για ένα πολυνηματικό (multi-threaded) σύστημα το οποίο μπορεί να αποδώσει δυναμικά σε περιβάλλοντα συμμετρικής πολυεπεξεργασίας (Symmetric Multi-Processing – SMP) ανάλογα με τον αριθμό των επεξεργαστών που διατίθενται/προστίθενται .
- StreamBase Adapters & Interfaces: Πρόκειται για βιβλιοθήκες και έτοιμα προγράμματα τα οποία παρέχουν τη δυνατότητα σύνδεση εξωτερικών ετερογενών συστημάτων στην

είσοδο και την έξοδο του StreamBase server. Επιτρέπουν έτσι τη σύνδεση πηγών όπως αρχείων διαφόρων τύπων, XML δεδομένων, σύνδεση με βάσεις δεδομένων που υποστηρίζουν JDBC (Java Database Connectivity) και άλλα συστήματα.

- StreamBase Studio: Πρόκειται για την πλατφόρμα σχεδίασης, υλοποίησης και ελέγχου των εφαρμογών που αναπτύσσονται.
- StreamSQL: Είναι η γλώσσα χειρισμού ρευμάτων δεδομένων που παρέχει το σύστημα. Είναι χαρακτηριστικό ότι ο προγραμματιστής μπορεί να συνδυάσει την ανάπτυξη εφαρμογών μέσω του StreamBase Studio και τη συγγραφή κώδικα. Το StreamBase Studio παρέχει τη δυνατότητα παραγωγής κώδικα StreamSQL με κάποιους περιορισμούς.
- StreamBase Feed Simulator: Είναι το υποσύστημα το οποίο χρησιμοποιείται για την παραγωγή δοκιμαστικών δεδομένων για την τροφοδότηση του συστήματος κατά τη διάρκεια ανάπτυξης και ελέγχου. Μπορεί να παράγει τυχαία δεδομένα, ή και να τροφοδοτήσει το σύστημα από αρχεία τα οποία περιέχουν τα δοκιμαστικά δεδομένα. Το StreamBase Feed Simulator παρέχει ένα περιβάλλον μέσω του οποίου είναι δυνατή η εισαγωγή δεδομένων από το χρήστη με τη μορφή συγκεκριμένων πλειάδων.

3. Μεθοδολογία υλοποίησης

3.1. Γενικά

Το πρακτικό μέρος της εργασίας, έχει ως στόχο την επίδειξη του τρόπου λειτουργίας ενός Συστήματος Διαχείρισης Ρευμάτων Δεδομένων. Για τους σκοπούς της επίδειξης επιλέχθηκε το σύστημα StreamBase το οποίο όπως προαναφέραμε, παρέχει μια ολοκληρωμένη πλατφόρμα σχεδίασης, ανάπτυξης, ελέγχου και εκτέλεσης εφαρμογών διαχείρισης ρευμάτων δεδομένων.

Η εφαρμογή η οποία αναπτύχθηκε, διαχειρίζεται σε πραγματικό χρόνο δεδομένα τα οποία προσομοιώνουν την κίνηση οχημάτων στην Αττική οδό και πιο συγκεκριμένα τις διελεύσεις των οχημάτων από τα διόδια της Μεταμόρφωσης και της Λεωφόρου Κηφισίας.

Οι παραδοχές οι οποίες έγιναν είναι οι εξής:

- Το σύστημα θα αποτελείται από τέσσερις πηγές εισόδου οι οποίες θα προσομοιώνουν τις διελεύσεις οχημάτων από τους δύο σταθμούς διοδίων με βάση τη χρήση ή όχι e-pass.
- Το σύστημα θα δέχεται τα ρεύματα εισόδου και θα τα επεξεργάζεται έτσι ώστε να παραχθούν αποτελέσματα έτσι όπως αυτά περιγράφονται στο κεφάλαιο της ανάλυσης απαιτήσεων.
- Η έξοδος του συστήματος θα αποτελείται από δέκα ροές εξόδου συναθροιστικών αποτελεσμάτων τα οποία θα δείχνουν στατιστικά στοιχεία κίνησης και χρήσης e-pass στην Αττική Οδό.
- Τα δεδομένα εισόδου θα αποτελούνται από πλειάδες συγκεκριμένης γραμμογράφησης.
- Για την τροφοδοσία του συστήματος επιλέχθηκε η υλοποίηση προσομοιωτών δεδομένων έτσι ώστε να είναι δυνατή η παραγωγή όγκου δεδομένων αλλά και η διαχείριση τους μέσα από την πλατφόρμα ανάπτυξης. Είναι χαρακτηριστικό ότι το σύστημα δίνει τη δυνατότητα μεταβολής του ρυθμού εκπομπής πλειάδων. Έτσι, μπορούμε να εκτελέσουμε την εφαρμογή και να αλλάξουμε το ρυθμό άφιξης δεδομένων εισόδου στο σύστημα, δυναμικά.

Στη συνέχεια περιγράφονται αναλυτικά οι παραδοχές που έγιναν κατά τη σχεδίαση και υλοποίηση των προσομοιωτών.

3.2. **Προσομοίωση ρευμάτων δεδομένων εισόδου**

Για να τροφοδοτηθούν τα ρεύματα εισόδου δεδομένων της εφαρμογής, υλοποιήθηκαν τέσσερις προσομοιωτές διελεύσεων οχημάτων όπως φαίνεται στον ακόλουθο πίνακα:

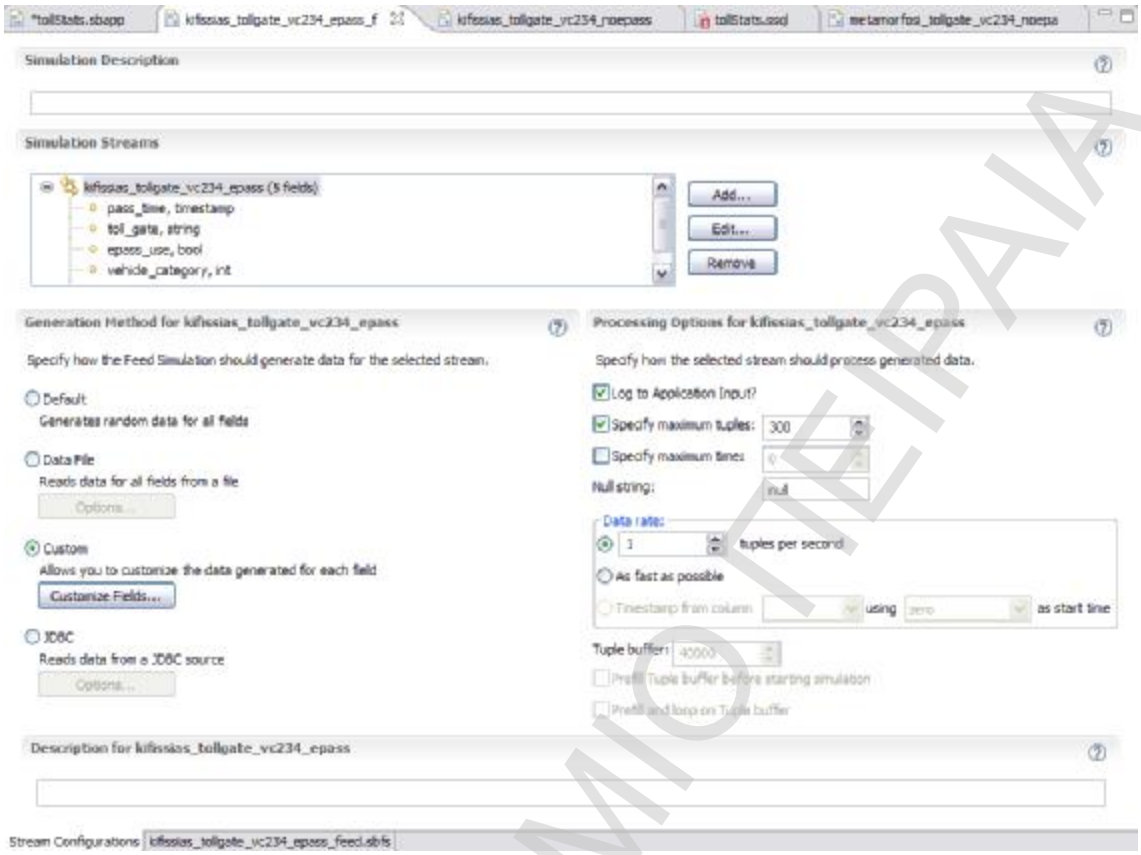
α/α	Ρεύμα Εισόδου	Προσομοιωτής
1	kifissias_tollgate_vc234_epass	kifissias_tollgate_vc234_epass_feed
2	kifissias_tollgate_vc234_noepass	kifissias_tollgate_vc234_noepass
3	Metamorfosi_tollgate_vc234_epass	metamorfosi_tollgate_vc234_epass_feed
4	Metamorfosi_tollgate_vc234_noepass	metamorfosi_tollgate_vc234_noepass

Πίνακας 3-1: Αντιστοιχία Ρευμάτων δεδομένων εισόδου – Προσομοιωτών

Οι προσομοιωτές υλοποιήθηκαν μέσα από το περιβάλλον ανάπτυξης του StreamBase. Η δυνατότητα δημιουργίας τέτοιων εργαλείων είναι πολύ σημαντική μιας και δίνει τη δυνατότητα ελέγχου των εφαρμογών που αναπτύσσονται. Οι προσομοιωτές στο περιβάλλον του StreamBase υλοποιούνται, όπως αναφέραμε και σε προηγούμενο κεφάλαιο, με το εργαλείο Feed Simulation Editor και ονομάζονται feeders αφού σκοπός τους είναι να «ταΐζουν» με δεδομένα τις ροές εισόδου.

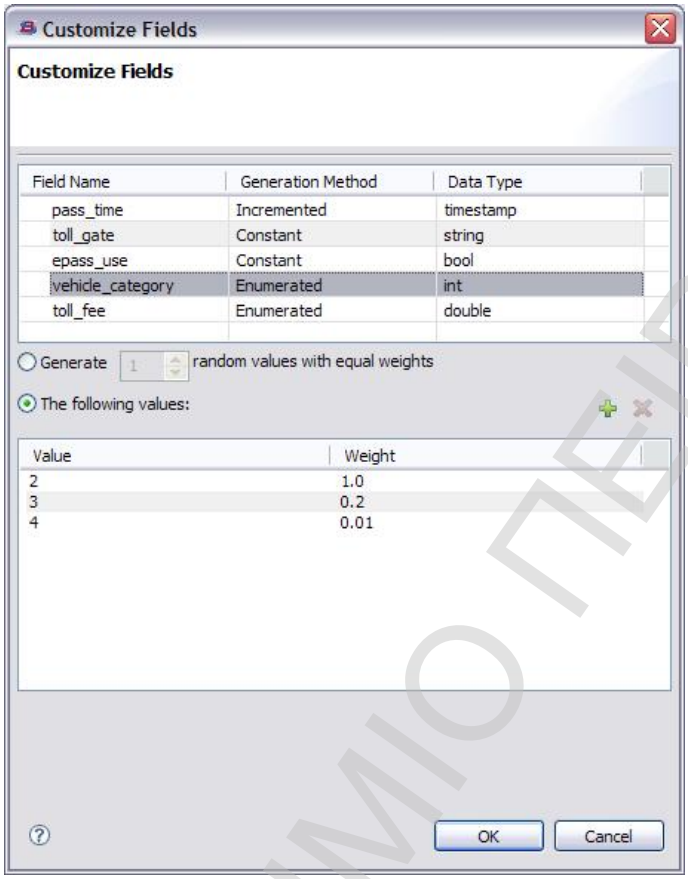
Οι προσομοιωτές της εφαρμογής μας ανήκουν σε δύο ομάδες ανάλογα με το εάν παρέχουν δεδομένα για οχήματα με χρήση e-pass ή όχι. Μια σειρά παραμέτρων ορίσθηκε, έτσι ώστε το δείγμα δεδομένων που προκύπτει από τη χρήση των feeders να είναι κατά το δυνατόν αντιπροσωπευτικό της κυκλοφορίας των διαφορετικών κατηγοριών οχημάτων.

Όπως βλέπουμε και στο σχήμα 3-1, καταρχήν, ο feeder θα πρέπει να συσχετισθεί με μια ροή εισόδου, έτσι ώστε να κληρονομήσει αυτόματα τη δομή των πλειάδων του ρεύματος. Βλέπουμε επίσης ότι ο ρυθμός παραγωγής πλειάδων ορίστηκε σε 1 ανά δευτερόλεπτο με μέγιστο αριθμό πλειάδων τις 300. Αυτό μας δίνει δεδομένα για 5 λεπτά. Κατά τη διάρκεια εκτέλεσης μας δίνεται η δυνατότητα μέσα από το περιβάλλον εκτέλεσης του StreamBase, να αλλάξουμε δυναμικά το ρυθμό εκπομπής πλειάδων.



Εικόνα 3-1: Ορισμός προσομοιωτή με χρήση του Feed Simulation Editor

Ακόμη, όπως φαίνεται και στο σχήμα 3-2, μέσω του γραφικού περιβάλλοντος του Feed Simulation Editor, ήταν εφικτός όχι μόνο ο ορισμός συγκεκριμένων πεδίων τιμών, αλλά και η αντιστοίχιση βαρών (weights) για κάθε χαρακτηριστικό του οποίο το πεδίο τιμών ήταν διακριτή λίστα, σε κάθε τιμή του πεδίου τιμών. Με αυτόν τον τρόπο επηρεάζεται το ποσοστό με το οποίο εμφανίζεται μια τιμή στις πλειάδες του ρεύματος. Βάρη πιο κοντά στην τιμή 1 ή μεγαλύτερα, έχουν ως αποτέλεσμα την εμφάνιση της συγκεκριμένης τιμής περισσότερες φορές στην πλειάδα, σε σύγκριση με μικρότερες τιμές βαρών. Για παράδειγμα, γνωρίζοντας ότι τα οχήματα τύπου 2 (ΙΧ επιβατικά) είναι αναλογικά περισσότερα, η τιμή αυτή πήρε και το μεγαλύτερο βάρος.



Εικόνα 3-2: Ορισμός βαρών για τις τιμές των χαρακτηριστικών της πλειάδας

Με βάση τα όσα είπαμε παραπάνω, ο κάθε προσομοιωτής, παραμετροποιήθηκε όπως δείχνουν οι παρακάτω πίνακες:

Όνομα πεδίου	Τύπος δεδομένων	Πεδίο τιμών
pass_time	Date	Δευτερόλεπτα, αύξηση κατά ένα
toll_gate	String	Σταθερή τιμή: KIFISSIAS
epass_use	Boolean	Σταθερή τιμή: TRUE
vehicle_category	Integer	Λίστα τιμών: 2 με βάρος 1.0 3 με βάρος 0.2 και 4 με βάρος 0.01
toll_fee	Double	Λίστα τιμών: 2.30 με βάρος 1.0 2.00 με βάρος 0.2 1.70 με βάρος 0.1 και 0.00 με βάρος 0.01

Πίνακας 3-2: Παραμετροποίηση του προσομοιωτή kifissias_tollgate_vc234_epass_feed

Όνομα πεδίου	Τύπος δεδομένων	Πεδίο τιμών
pass_time	Date	Δευτερόλεπτα, αύξηση κατά ένα
toll_gate	String	Σταθερή τιμή: KIFISSIAS
epass_use	Boolean	Σταθερή τιμή: FALSE
vehicle_category	Integer	Λίστα τιμών: 2 με βάρος 1.0 3 με βάρος 0.1 και 4 με βάρος 0.01
toll_fee	Double	Σταθερή τιμή: 2.70

Πίνακας 3-3: Παραμετροποίηση του προσομοιωτή kifissias_tollgate_vc234_noepass

Όνομα πεδίου	Τύπος δεδομένων	Πεδίο τιμών
pass_time	Date	Δευτερόλεπτα, αύξηση κατά ένα
toll_gate	String	Σταθερή τιμή: METAMORFOSI
epass_use	Boolean	Σταθερή τιμή: FALSE
vehicle_category	Integer	Λίστα τιμών: 2 με βάρος 1.0 3 με βάρος 0.1 και 4 με βάρος 0.01
toll_fee	Double	Σταθερή τιμή: 2.70

Πίνακας 3-4: Παραμετροποίηση του προσομοιωτή metamorfosi_tollgate_vc234_noepass

Όνομα πεδίου	Τύπος δεδομένων	Πεδίο τιμών
pass_time	Date	Δευτερόλεπτα, αύξηση κατά ένα
toll_gate	String	Σταθερή τιμή: METAMORFOSI
epass_use	Boolean	Σταθερή τιμή: TRUE
vehicle_category	Integer	Λίστα τιμών: 2 με βάρος 1.0 3 με βάρος 0.3 και 4 με βάρος 0.1
toll_fee	Double	Λίστα τιμών: 2.30 με βάρος 1.0 2.00 με βάρος 0.2 1.70 με βάρος 0.3 και 0.00 με βάρος 0.05

Πίνακας 3-5: Παραμετροποίηση του προσομοιωτή metamorfosi_tollgate_vc234_epass_feed

4. Υλοποίηση εφαρμογής

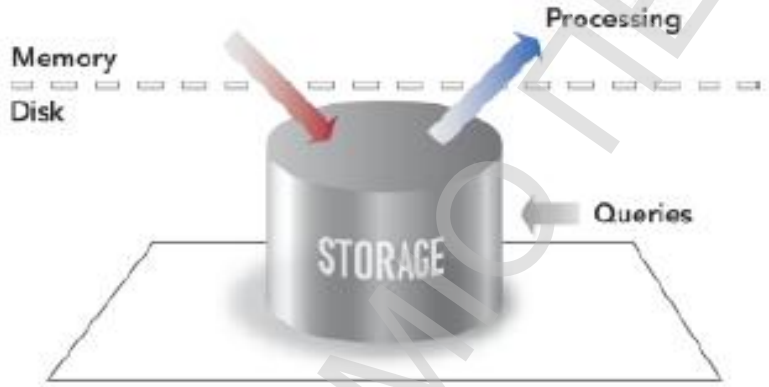
4.1. Πρόταση υλοποίησης

Το πρακτικό μέρος της εργασίας όπως είπαμε, προσπαθεί να επιδείξει, το πως λειτουργεί στην πράξη ένα σύγρονο Σύστημα Διαχείρισης Ρευμάτων Δεδομένων. Πρόκειται για το εμπορικό σύστημα StreamBase. Το μέρος αυτό περιλαμβάνει τα παρακάτω:

1. Εγκατάσταση και παραμετροποίηση του Streambase στο λειτουργικό σύστημα Windows.
2. Συνοπτική παρουσίαση του συστήματος StreamBase (αρχιτεκτονική, περιβάλλον λειτουργίας και ανάπτυξης, δυνατότητες κτλ)
3. Δημιουργία ρευμάτων δεδομένων που θα προσομοιώνουν δεδομένα διέλευσης οχημάτων από τα διόδια της Αττικής Οδού κατά τη διάρκεια μιας μέρας.
4. Υλοποίηση εφαρμογής η οποία θα λαμβάνει στην είσοδο της σε πραγματικό χρόνο, ρεύματα δεδομένων κίνησης οχημάτων (που θα παράγονται από τους προσομοιωτές) και θα ταξινομεί ανάλογα με το είδος και την ώρα, στοιχεία όπως τύπο (δίκυκλο, αυτοκίνητο ΙΧ, φορτηγό ΔΧ κτλ) και πλήθος οχημάτων, έσοδα ανά σταθμό διέλευσης και τύπο οχημάτων. Επίσης, με βάση τα στοιχεία των διελεύσεων, το σύστημα θα είναι δυνατό να παρέχει ειδοποιήσεις αυξημένης κίνησης (alerts) με βάση συγκεκριμένα κατώφλια. Τέλος, το σύστημα θα μπορεί να παρέχει πληροφορίες χρήσης του e-pass.
5. Παρουσίαση αποτελεσμάτων της εφαρμογής σε διαφορετικά χρονικά παράθυρα.

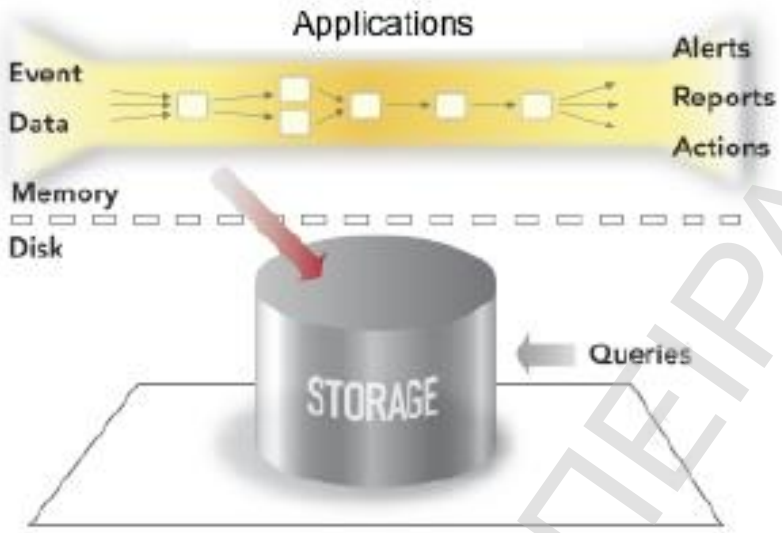
4.2. **Παρουσίαση του** StreamBase

Αναφέραμε σε προηγούμενα κεφάλαια, ότι ένα χαρακτηριστικό των παραδοσιακών ΣΔΒΔ είναι το ότι τα ερωτήματα διατυπώνονται άπαξ (ερωτήματα στιγμιότυπου, one-time queries) σε υπάρχοντα δεδομένα, που σημαίνει ότι αυτά βρίσκονται σε κάποιο αποθηκευτικό μέσο όπως μνήμη συστήματος, σκληρός δίσκος κτλ, και μάλιστα ότι αποτιμούνται με βάση συγκεκριμένο στιγμιότυπο (snapshot) των δεδομένων. Τα αποτελέσματα των ερωτημάτων αυτών πρέπει να αποθηκευτούν για περαιτέρω επεξεργασία. Το μοντέλο αυτό διαχείρισης δεδομένων ονομάζεται «pull model» μιας και οι χρήστες «τραβάνε» στοιχεία από τα αποθηκευμένα δεδομένα. Μια σχηματική παράσταση του μοντέλου αυτού απεικονίζεται στο πιο κάτω σχήμα.



Εικόνα 4-1: «Pull model»

Σε αντίθεση με τα παραπάνω, τα ερωτήματα διαρκείας (continues queries) που υποστηρίζονται από το σύστημα StreamBase, αφορούν, δυναμικά δεδομένα που μπορεί να μην είναι καν αποθηκευμένα σε κάποιο μέσο. Το μοντέλο που υλοποιεί το σύστημα είναι το «push model», όπου τα δεδομένα «σπρώχνουν» τις απαντήσεις στον χρήστη. Μια σχηματική παράσταση του μοντέλου αυτού απεικονίζεται στο πιο κάτω σχήμα.



Εικόνα 4-2: «Push model»

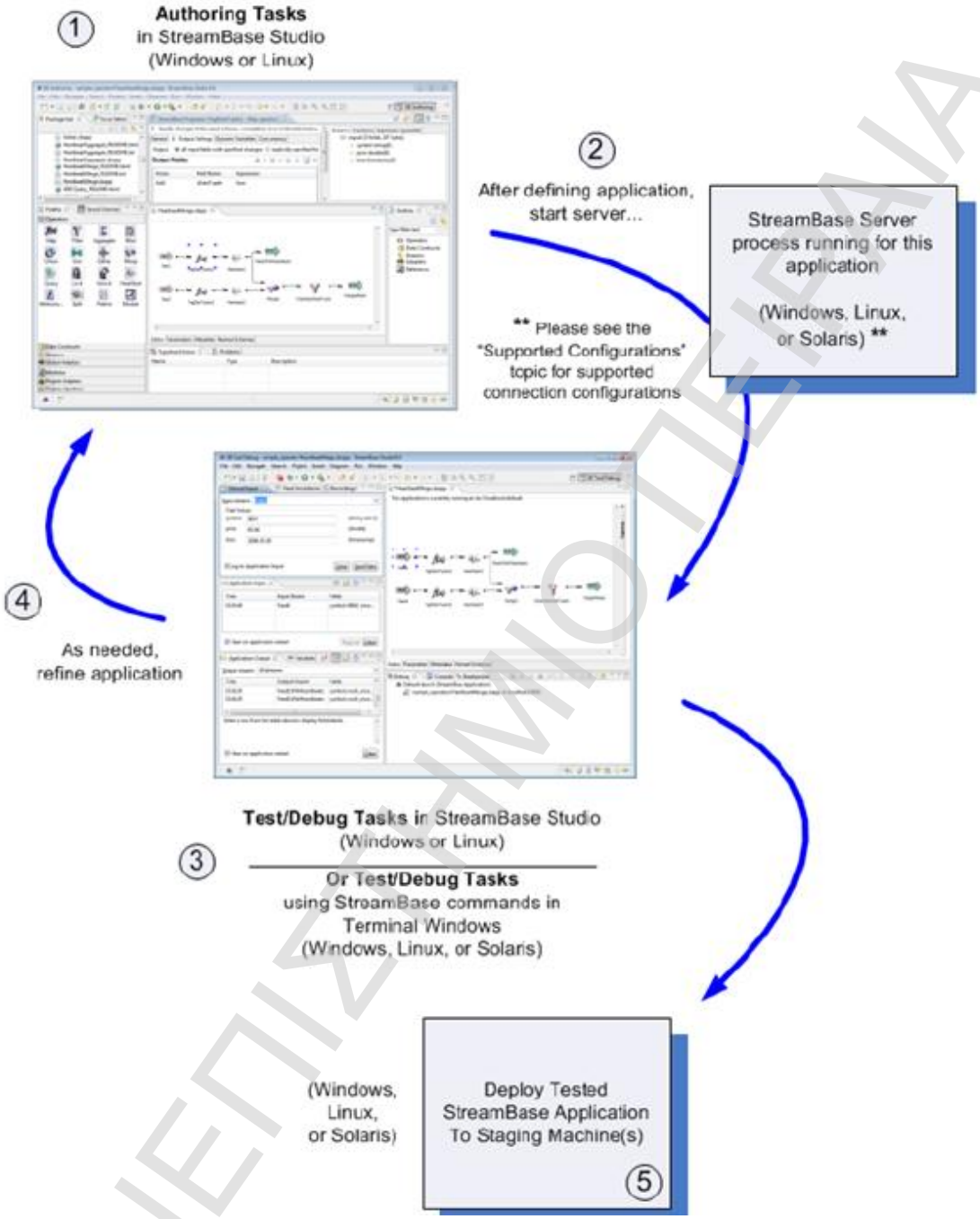
Έτσι, το σύστημα παρέχει λειτουργίες όπως:

1. Δημιουργία συναθροίσεων σε πλειάδες δεδομένων εισόδου σε πραγματικό χρόνο.
2. Εφαρμογή μαθηματικών συναρτήσεων για τον υπολογισμό τιμών, πρόσθεση νέων χαρακτηριστικών στις πλειάδες ή και αφαίρεση
3. Φιλτροποίηση δεδομένων με βάση συγκεκριμένες συνθήκες.
4. Χωρισμός ρευμάτων δεδομένων σε περισσότερα του ενός έτσι ώστε εν συνεχεία να είναι δυνατή η ξεχωριστή επεξεργασία του κάθε ρεύματος
5. Ένωση ρευμάτων δεδομένων εισόδου ή ρευμάτων τα οποία νωρίτερα είχαν χωρισθεί με βάση συγκεκριμένα κριτήρια μέσα στο σύστημα
6. Αποθήκευση πλειάδων σε σχεσιακούς πίνακες για περαιτέρω επεξεργασία και χρήση ως δεδομένα αναφοράς (lookup values)
7. Συνδυασμό δεδομένων προερχόμενων από σχεσιακούς πίνακες και ρεύματα

Το StreamBase παρέχει ένα ολοκληρωμένο περιβάλλον σχεδίασης, ανάπτυξης και ελέγχου εφαρμογών το οποίο ονομάζεται «SB Authoring» και όπως είπαμε και προηγούμενα δίνει τη δυνατότητα στον προγραμματιστή να συνδυάζει την ανάπτυξη εφαρμογών μέσω του StreamBase Studio:

1. γραφικά (EventFlow applications), με τη δημιουργία XML αρχείων μέσω του EventFlow Editor και
2. με τη δημιουργία εφαρμογών με τη συγγραφή κώδικα (StreamSQL applications) με τη χρήση της γλώσσας ερωταποκρίσεων και διαχείρισης ρευμάτων δεδομένων SteamSQL. Οι εφαρμογές αυτές φυλάσσονται σε αρχεία κειμένου με κατάληξη .ssql.

Το παρακάτω διάγραμμα παρέχει μια εικόνα των εργαλείων που χρησιμοποιούνται καθόλη τη διάρκεια του κύκλου ανάπτυξης καθώς και τα βήματα που ακολουθούνται:



Εικόνα 4-3: Εργαλεία και στάδια ανάπτυξης εφαρμογών στο SB Authoring

Τα δεδομένα εισόδου στο σύστημα μπορεί να προέρχονται από διαφορετικών ειδών πηγές και πρωτόκολλα. Οι πιθανοί τρόποι εισόδου δεδομένων είναι:

- Από ένα πρόγραμμα προσαρμογέα (adapter) το οποίο δέχεται στην είσοδο του δεδομένα με βάση συγκεκριμένο πρωτόκολλο (π.χ. TIBCO Rendezvous messages, το

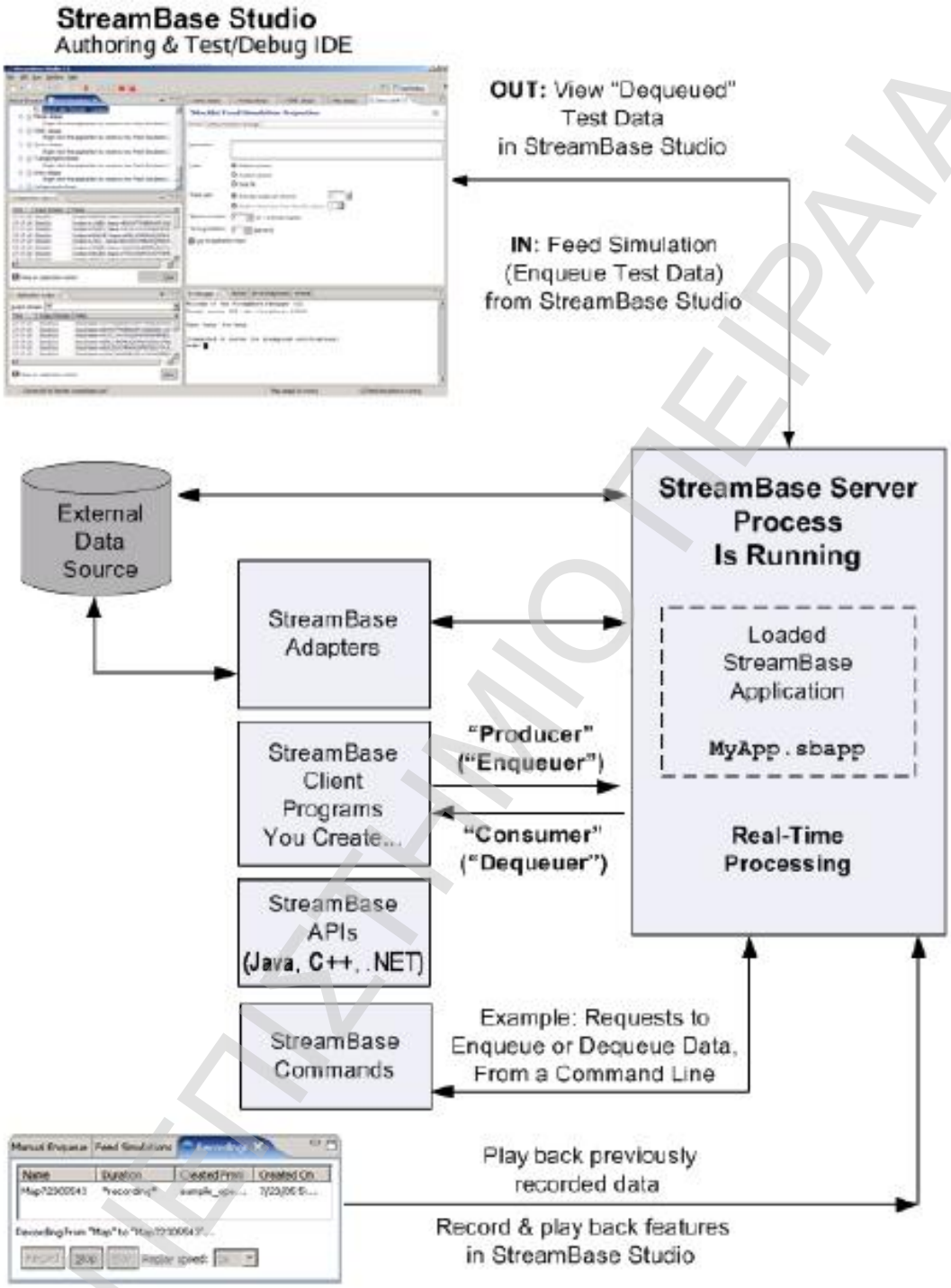
οποίο παρέχει δεδομένα χρηματιστηριακών αγορών) και τα μετατρέπει σε ρεύματα εισόδου σύμφωνα με τα πρωτόκολλα που υποστηρίζει το StreamBase.

- Από ένα πρόγραμμα πελάτη δημιουργίας ουρών δεδομένων (enqueue client program) υλοποιημένο σε Java, C++ ή .NET, το οποίο αποστέλλει δεδομένα σε ένα ρεύμα εισόδου του συστήματος που έχει οριστεί στην εφαρμογή. Η υλοποίηση τέτοιων προγραμμάτων χρησιμοποιεί τις βιβλιοθήκες που παρέχει το StreamBase API.
- Από μια σχεσιακή βάση δεδομένων στην οποία η εφαρμογή συνδέεται μέσω του πρωτοκόλλου JDBC. Απαραίτητη προϋπόθεση εννοείται ότι είναι η πηγή να υποστηρίζει τον τύπο αυτό σύνδεσης. Η σύνδεση αυτή επιτρέπει την παροχή ιστορικής πληροφορίας στο σύστημα σε συνδυασμό με την παροχή δεδομένων ρευμάτων.
- Με τη χρήση εντολών από κάποια τερματική συσκευή, που καθορίζουν επιπλέον επίπεδα λεπτομέρειας στα δεδομένα εισόδου.
- Με τη χρήση του Feed Simulation Editor το οποίο είναι ένα γραφικό εργαλείο για τη δημιουργία τυχαίων πλειάδων ή την ανάκτηση πλειάδων μέσω αρχείων (π.χ. CSV files), με δεδομένα που έχουν δημιουργηθεί έτσι ώστε για να ελεγχθεί η συμπεριφορά της υπό υλοποίηση εφαρμογής. Ο Feed Simulation Editor παρέχει τη δυνατότητα ελέγχου του ρυθμού αποστολής των πλειάδων εισόδου.

Η ανάκτηση πληροφορίας από το StreamBase γίνεται μέσω ρευμάτων εξόδου, τα οποία έχουν οριστεί κατά την υλοποίηση. Οι τρόποι με τους οποίους μπορεί να εξαχθεί η πληροφορία από τα ρεύματα εξόδου είναι οι εξής:

- Με τη χρήση ενός προγράμματος προσαρμογέα ο οποίος μετατρέπει τα δεδομένα εξόδου στη μορφή εισόδου τους στο σύστημα.
- Με τη χρήση ενός προγράμματος πελάτη, το οποίο ελέγχει τα ρεύματα εξόδου της εφαρμογής και συλλέγει τα δεδομένα. Το πρόγραμμα μπορεί να είναι υλοποιημένο σε Java, C++ ή .NET. Η υλοποίηση τέτοιων προγραμμάτων χρησιμοποιεί τις βιβλιοθήκες που παρέχει το StreamBase API.
- Με τη χρήση εντολών εισαγωγής (insert) και τροποποίησης (update) δεδομένων σε σχεσιακούς πίνακες μιας βάσης δεδομένων η οποία είναι συνδεδεμένη μέσω JDBC στο σύστημα.
- Με τη χρήση του γραφικού εργαλείου που παρέχει η πλατφόρμα StreamBase.
- Με τη χρήση εντολών StreamBase για την εξαγωγή δεδομένων από τα ρεύματα εξόδου.

Το επόμενο διάγραμμα δίνει μια εικόνα των τρόπων εισόδου/εξόδου πληροφορίας από το StreamBase.



Εικόνα 4-4: Τρόποι εισόδου/εξόδου ρευμάτων δεδομένων από το StreamBase

.

4.3. **Ανάλυση εφαρμογής - Λειτουργικές προδιαγραφές**

Η Αττική Οδός είναι ένας σύγχρονος αυτοκινητόδρομος μήκους 65 χλμ. Αποτελεί τον περιφερειακό δακτύλιο της ευρύτερης μητροπολιτικής περιοχής της Αθήνας και τη σπονδυλική στήλη του οδικού δικτύου ολόκληρου του Νομού Αττικής. Πρόκειται για έναν αστικού-περιαστικού τύπου αυτοκινητόδρομο, με 3 λωρίδες κυκλοφορίας ανά κατεύθυνση και μια λωρίδα έκτακτης ανάγκης.

Η καρδιά της διαχείρισης της κυκλοφορίας της Αττικής Οδού, «χτυπά» στο Κέντρο Διαχείρισης Κυκλοφορίας (Κ.Δ.Κ.), που βρίσκεται στην Παιανία και λειτουργεί όλο το 24ωρο. Το Κ.Δ.Κ. επιβλέπει διαρκώς τις κυκλοφοριακές συνθήκες που επικρατούν σε όλο το μήκος του αυτοκινητόδρομου και ενημερώνει άμεσα τα λοιπά αρμόδια τμήματα ή/και τους φορείς για την εμφάνιση τυχόν προβλημάτων.

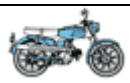
Μια από τις βασικές αποστολές του ΚΔΚ είναι η ανάλυση των δεδομένων που προκύπτουν από τη λειτουργία των σταθμών διοδίων. Σε συνεργασία με τη Διεύθυνση Διοδίων, Εμπορικής Διαχείρισης και Τηλεφωνικής Εξυπηρέτησης της Αττικής Οδού, που έχει ως αντικείμενο το σχεδιασμό της εμπορικής πολιτικής και της πολιτικής διοδίων, καθώς και την είσπραξη των διοδίων και τη διαχείριση των εμπορικών προγραμμάτων, το ΚΔΚ παράγει αναφορές οι οποίες παρουσιάζουν στατιστικά δεδομένα της λειτουργίας των διοδίων και της χρήσης των διαφόρων πακέτων προγραμμάτων (e-pass κτλ). Συνολικά στην Αττική Οδό, λειτουργούν 39 Σταθμοί Διοδίων με 195 συνολικές λωρίδες διοδίων εκ των οποίων:

- Οι 55 είναι ηλεκτρονικές λωρίδες διοδίων
- Ενώ οι 140 είναι λωρίδες διοδίων με εισπράκτορα

Η εφαρμογή που υλοποιείται, δίνει τη δυνατότητα ενημέρωσης σε πραγματικό χρόνο για τις διελεύσεις των διαφόρων τύπων οχημάτων, σε επίπεδο τόσο οικονομικών στοιχείων με βάση τη χρήση των εμπορικών πακέτων, όσο και ποσοτικών δεδομένων, με βάση τον αριθμό διελεύσεων ανά τύπο οχήματος ανά περιόδους, αλλά και συνολικά, καθώς και τη δυνατότητα παραγωγής ειδοποιήσεων (traffic alerts) σε περίπτωση αυξημένης κίνησης. Η εφαρμογή υλοποιείται πιλοτικά στους σταθμούς διοδίων Κηφισίας και Μεταμόρφωσης, αναφέρεται στα οχήματα των κατηγοριών 2, 3 και 4 όπως αυτά περιγράφονται στον πίνακα 4-1 και καλύπτει λειτουργικές ανάγκες ελέγχου των διοδίων με την παραγωγή αναφορών ως εξής:

1. Πλήθος διελεύσεων οχημάτων ανά κατηγορία οχήματος ανά 30 δευτερόλεπτα από κάθε σταθμό διοδίων. Ο αριθμός των διελεύσεων αναφέρεται στο συνολικό αριθμό οχημάτων που πέρασαν από το συγκεκριμένο σταθμό ανά κατηγορία οχήματος, από όλες τις λωρίδες διοδίων (αυτόματες και με εισπράκτορα).
2. Συνολικό πλήθος διελεύσεων οχημάτων ανά 1 λεπτό από κάθε σταθμό διοδίων. Ο αριθμός των διελεύσεων αναφέρεται στο συνολικό αριθμό οχημάτων που πέρασαν από το συγκεκριμένο σταθμό από όλες τις λωρίδες διοδίων (αυτόματες και με εισπράκτορα).
3. Σε περίπτωση που οι συνολικές διελεύσεις είναι περισσότερες από 100, η εφαρμογή παράγει μήνυμα για αυξημένη κυκλοφορία.
4. Ποσό εσόδων με βάση τη χρήση των συσκευών e-pass ανά 30 δευτερόλεπτα από κάθε σταθμό διοδίων. Το ποσό εσόδων αναφέρεται στη χρέωση των συσκευών epass για τις κατηγορίες οχημάτων 2,3 και 4. Ο αριθμός των διελεύσεων αναφέρεται στο συνολικό αριθμό οχημάτων που πέρασαν από το συγκεκριμένο σταθμό από όλες τις αυτόματες λωρίδες διοδίων (αυτόματες και με εισπράκτορα).

5. Συνολικό πλήθος διελεύσεων οχημάτων από την αρχή της ημέρας από όλους τους σταθμούς διοδίων. Ο αριθμός των διελεύσεων αναφέρεται στο συνολικό αριθμό οχημάτων που πέρασαν από όλους αθροιστικά τους σταθμούς διοδίων, από όλες τις λωρίδες διοδίων (αυτόματες και με εισπράκτορα) από την αρχή της μέρας μέχρι τη συγκεκριμένη στιγμή. Η πληροφορία παράγεται ανά λεπτό.
6. Συνολικά έσοδα από την αρχή της ημέρας από όλους τους σταθμούς διοδίων. Το ποσό των εσόδων αναφέρεται στο συνολικό - αθροιστικό ποσό από όλους τους σταθμούς διοδίων, από όλες τις λωρίδες διοδίων (αυτόματες και με εισπράκτορα) από την αρχή της μέρας μέχρι τη συγκεκριμένη στιγμή. Η πληροφορία παράγεται ανά λεπτό.
7. Συνολικά έσοδα με βάση τη χρήση των συσκευών e-pass από όλους τους σταθμούς διοδίων από την αρχή της μέρας μέχρι τη συγκεκριμένη στιγμή. Το ποσό εσόδων αναφέρεται στη χρέωση των συσκευών epass για τις κατηγορίες οχημάτων 2,3 και 4. Ο αριθμός των διελεύσεων αναφέρεται στο συνολικό αριθμό οχημάτων που πέρασαν από το συγκεκριμένο σταθμό από όλες τις αυτόματες λωρίδες διοδίων (αυτόματες και με εισπράκτορα).

Κατ.	**Ενδεικτική Απεικόνιση**	**Τέλη διοδίων**	**Περιγραφή κατηγορίας οχήματος**
1		**1,30 €**	Μοτοποδήλατα, δίκυκλες μοτοσικλέτες ενός τροχού ανά άξονα.
2		**2,70 €**	Επιβατικά (Ι.Χ.) ιδίας χρήσης περιλαμβανομένων επιβατικών με μικρό trailer και σχάρα. Το ύψος τους πρέπει να είναι μικρότερο του 1,30μ. ανεξάρτητα από το αν είναι ενός, δύο ή τριών αξόνων.
3		**2,70 €**	Ελαφρά εμπορικά οχήματα με ύψος μικρότερο του 1,30μ. επάνω από τον πρώτο άξονα και συνολικό ύψος μικρότερο των 2,70μ.
4		**2,70 €**	Αυτοκίνητα με τροχόσπιτα με ύψος μικρότερο του 1,30μ. επάνω από τον πρώτο άξονα και συνολικό ύψος μικρότερο του 2,70μ τα οποία διαθέτουν τρεις το πολύ άξονες καθώς και λεωφορεία (κάτω των 15 θέσεων).
5		**6,70 €**	Μικρά και μεσαία φορτηγά, συνολικού ύψους μεγαλύτερου των 2,70μ. με 2 ή 3 άξονες και μεγάλα λεωφορεία (άνω των 15 θέσεων).

6		**10,80 €**	Μεγάλα φορτηγά με ή χωρίς ρυμουλκούμενο, συνολικού ύψους μεγαλύτερου των 2,70μ. με 4 άξονες και πάνω.

Πίνακας 4-1: Κατηγορίες οχημάτων και τέλη διοδίων Αττικής Οδού

Για την τροφοδότηση του συστήματος ο κάθε σταθμός διοδίων παράγει και αποστέλλει δεδομένα με συγκεκριμένη γραμμογράφηση όπως φαίνεται στον πίνακα 4-2 :

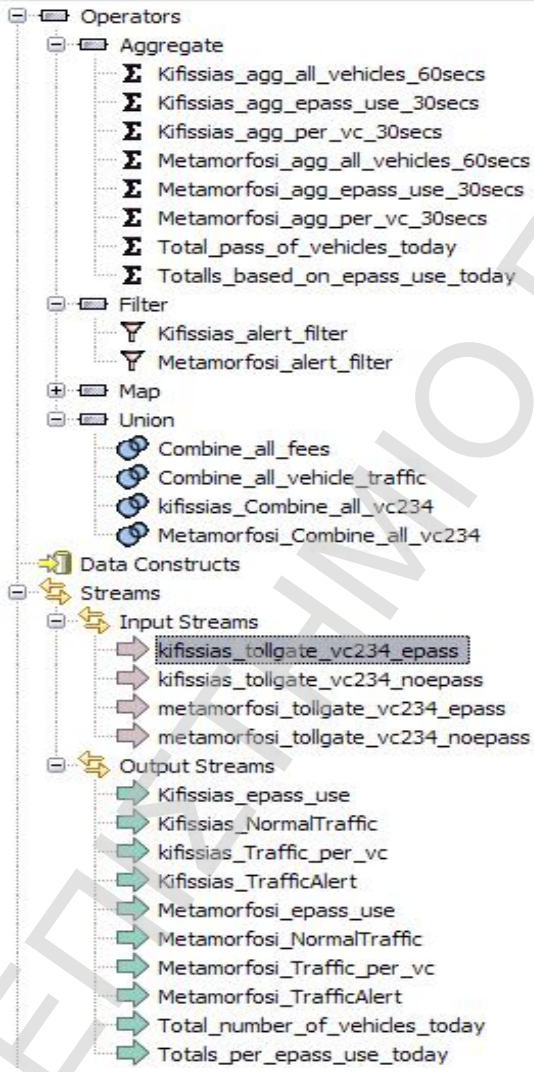
Όνομα πεδίου	Τύπος δεδομένων
pass_time	Date
toll_gate	String
epass_use	Boolean
vehicle_category	Integer
toll_fee	Double

Πίνακας 4-2: Γραμμογράφηση ρευμάτων δεδομένων εισόδου της εφαρμογής

.

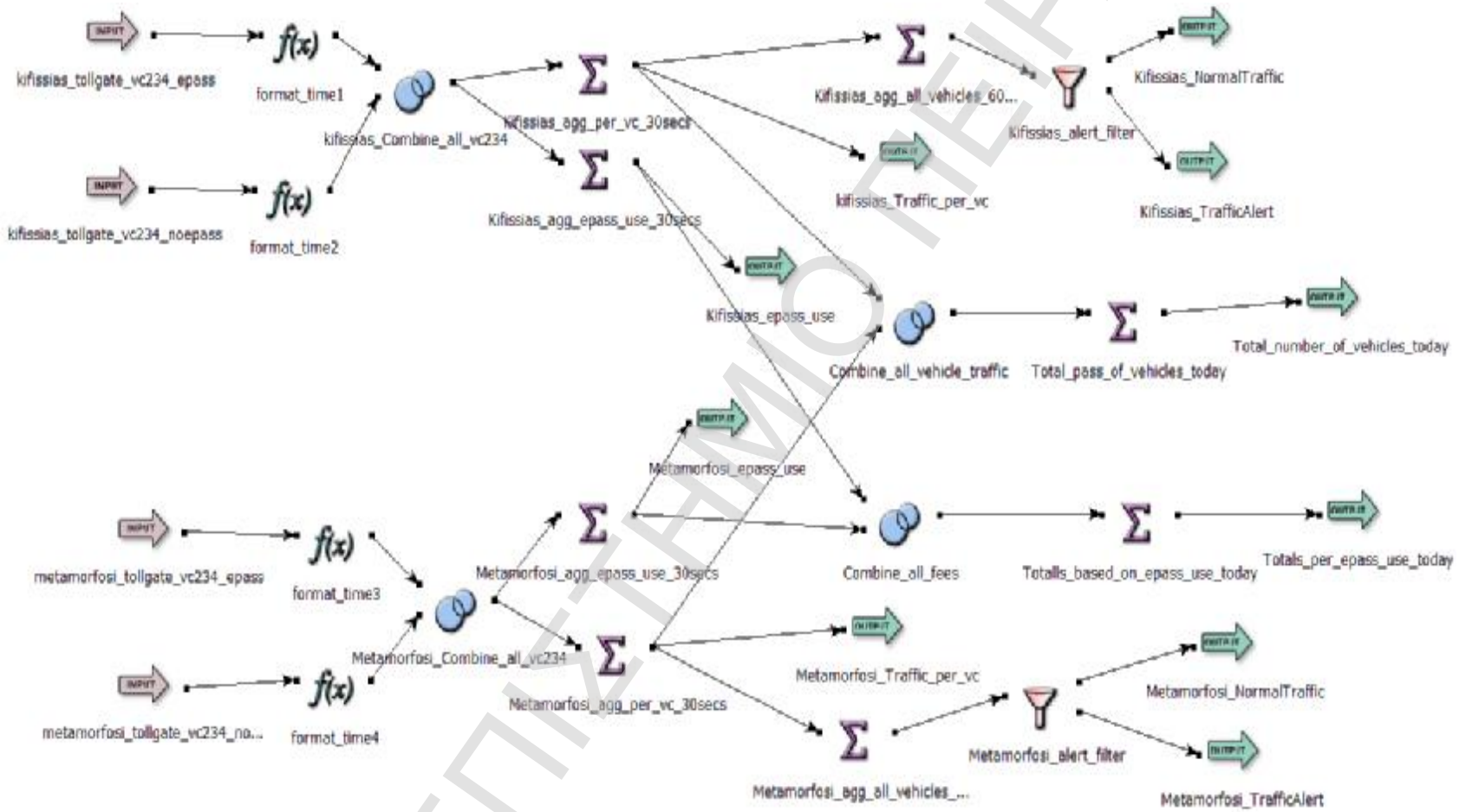
4.4. **Σχεδιασμός και υλοποίηση εφαρμογής**

Η εφαρμογή καλύπτει τις απαιτήσεις έτσι όπως αυτές καταγράφηκαν στην ανάλυση απαιτήσεων και λειτουργικών προδιαγραφών. Όπως φαίνεται και στο σχήμα 4-5, αποτελείται από ένα σύνολο τελεστών συναθροίσεων, ενώσεων και φίλτρων, τεσσάρων ρευμάτων δεδομένων εισόδου και δέκα ρευμάτων δεδομένων εξόδου. Για την τροφοδότηση των ρευμάτων εισόδου με πλειάδες, αναπτύχθηκαν αντίστοιχα, τέσσερις προσομοιωτές δεδομένων.



Εικόνα 4-5: Αντικείμενα από τα οποία αποτελείται η εφαρμογή

Το σύστημα δέχεται στην είσοδο του δεδομένα από τους σταθμούς διοδίων της Αττικής Οδού στην Κηφισίας και στη Μεταμόρφωση. Τα δεδομένα αυτά αφορούν τις κατηγορίες οχημάτων 2,3 και 4 και αποτελούνται από δύο ρεύματα εισόδου για κάθε σταθμό διοδίων. Ένα ρεύμα για τα οχήματα που περνούν αυτόματα (χρησιμοποιούν δηλαδή e-pass) και ένα για τα οχήματα που περνούν από εισπράκτορα. Κάθε ρεύμα δεδομένων τροφοδοτείται από τον αντίστοιχο προσομοιωτή.



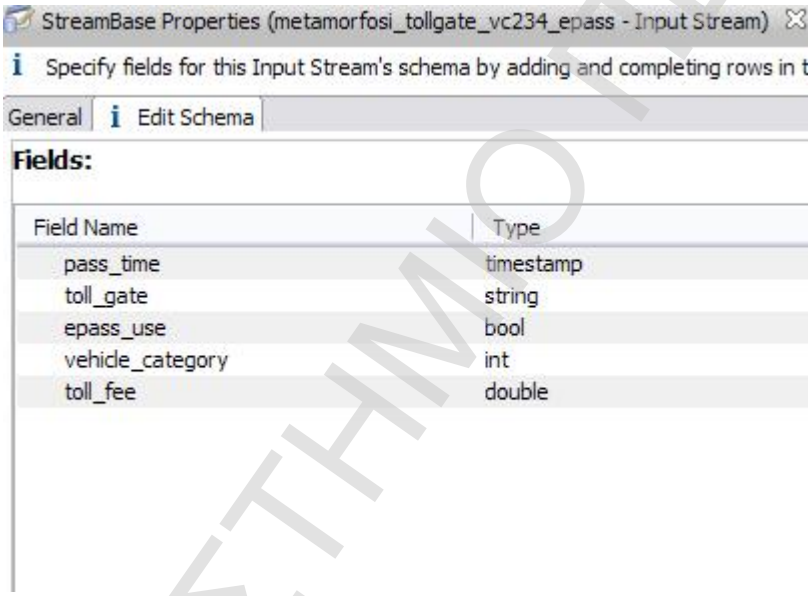
Εικόνα 4-6: Σχηματική απεικόνιση της εφαρμογής

Τα ρεύματα δεδομένων αφού εισαχθούν στο σύστημα, περνούν από μια σειρά τελεστών οι οποίοι επεξεργάζονται τις πλειάδες εισόδου παράγοντας πλειάδες που είτε τροφοδοτούν άλλους τελεστές είτε παράγουν πλειάδες εξόδου. Στο σχήμα 4-6 διακρίνονται τα ρεύματα εισόδου, οι διάφοροι τελεστές, τα ρεύματα εξόδου, καθώς και η ροή των ρευμάτων κατά τη διάρκεια της επεξεργασίας.

Να σημειωθεί ότι στο παράρτημα της εργασίας αυτής παρατίθενται οι εντολές δημιουργίας και ελέγχου των ρευμάτων δεδομένων στη γλώσσα StreamSQL για όλες τις λειτουργίες ρευμάτων, τελεστών, φίλτρων κτλ που περιγράφονται στη συνέχεια.

4.4.1. **Ρεύματα εισόδου δεδομένων εφαρμογής**

Τα ρεύματα εισόδου ορίστηκαν στο StreamBase με βάση τις απαιτήσεις της εφαρμογής, όπως φαίνεται στο παρακάτω σχήμα:



Εικόνα 4-7: Ορισμός ρευμάτων δεδομένων εισόδου της εφαρμογής

Ο ακόλουθος πίνακας περιγράφει τη λειτουργία κάθε ενός εκ των τεσσάρων ρευμάτων δεδομένων εισόδου :

α/α	Ρεύμα Εισόδου	Περιγραφή
1	kifissias_tollgate_vc234_epass	Ρεύμα εισόδου σταθμού διοδίων Κηφισίας για τα οχήματα κατηγορίας 2, 3 και 4 που περνούν αυτόματα (με χρήση e-pass). Το ρεύμα τροφοδοτείται από τον προσομοιωτή δεδομένων kifissias_tollgate_vc234_epass_feed
2	kifissias_tollgate_vc234_noepass	Ρεύμα εισόδου σταθμού διοδίων Κηφισίας για τα οχήματα κατηγορίας 2, 3 και 4 που περνούν από εισπράκτορα. Το ρεύμα τροφοδοτείται από τον προσομοιωτή δεδομένων

		kifissias_tollgate_vc234_noepass
3	metamorfosi_tollgate_vc234_epass	Ρεύμα εισόδου σταθμού διοδίων Μεταμόρφωσης για τα οχήματα κατηγορίας 2, 3 και 4 που περνούν αυτόματα (με χρήση e-pass). Το ρεύμα τροφοδοτείται από τον προσομοιωτή δεδομένων metamorfosi_tollgate_vc234_epass_feed
4	metamorfosi_tollgate_vc234_noepass	Ρεύμα εισόδου σταθμού διοδίων Μεταμόρφωσης για τα οχήματα κατηγορίας 2, 3 και 4 που περνούν από εισπράκτορα. Το ρεύμα τροφοδοτείται από τον προσομοιωτή δεδομένων metamorfosi_tollgate_vc234_noepass

Πίνακας 4-3: Ρεύματα εισόδου

4.4.2. **Ρεύματα εξόδου δεδομένων εφαρμογής**

Τα ρεύματα εξόδου της εφαρμογής ορίστηκαν με ανάλογο τρόπο με αυτόν των ρευμάτων εισόδου και περιγράφονται στον ακόλουθο πίνακα:

α/α	Ρεύμα Εξόδου	Περιγραφή
1	Kifissias_epass_use	Ρεύμα εξόδου που δείχνει τα έσοδα από το σταθμό διοδίων Κηφισίας, ομαδοποιημένα με βάση τη χρήση ή όχι του e-pass. Η ένδειξη ενημερώνεται κάθε 30 δευτερόλεπτα
2	kifissias_Traffic_per_vc	Ρεύμα εξόδου που δείχνει το πλήθος οχημάτων που πέρασαν από το σταθμό διοδίων Κηφισίας, ομαδοποιημένα με βάση τη κατηγορία οχήματος. Η ένδειξη ενημερώνεται κάθε 30 δευτερόλεπτα
3	Kifissias_NormalTraffic	Ρεύμα εξόδου που δείχνει ότι η κυκλοφορία των οχημάτων στο σταθμό Κηφισίας βρίσκεται σε κανονικά επίπεδα. Το συμπέρασμα αυτό εξάγεται από τη χρήση ενός φίλτρου που ελέγχει εάν το πλήθος διελεύσεων των οχημάτων για το προηγούμενο 1 λεπτό από το συγκεκριμένο σταθμό, ήταν μικρότερο των 100 οχημάτων.
4	Kifissias_TrafficAlert	Ρεύμα εξόδου που δείχνει ότι η κυκλοφορία των οχημάτων στο σταθμό Κηφισίας βρίσκεται σε επίπεδα άνω του φυσιολογικού ορίου. Το συμπέρασμα αυτό εξάγεται από τη χρήση ενός φίλτρου που ελέγχει εάν το πλήθος διελεύσεων των οχημάτων για το προηγούμενο 1 λεπτό από το συγκεκριμένο σταθμό, ήταν μεγαλύτερο των 100 οχημάτων. Στην περίπτωση αυτή παράγεται ειδοποίηση (traffic alert) αυξημένης κίνησης
5	Metamorfosi_epass_use	Ρεύμα εξόδου που δείχνει τα έσοδα από το σταθμό διοδίων Μεταμόρφωσης, ομαδοποιημένα με βάση τη

		χρήση ή όχι του e-pass. Η ένδειξη ενημερώνεται κάθε 30 δευτερόλεπτα
6	Metamorfosi_Traffic_per_vc	Ρεύμα εξόδου που δείχνει το πλήθος οχημάτων που πέρασαν από το σταθμό διοδίων Μεταμόρφωσης, ομαδοποιημένα με βάση τη κατηγορία οχήματος. Η ένδειξη ενημερώνεται κάθε 30 δευτερόλεπτα
7	Metamorfosi_NormalTraffic	Ρεύμα εξόδου που δείχνει ότι η κυκλοφορία των οχημάτων στο σταθμό Μεταμόρφωσης βρίσκεται σε κανονικά επίπεδα. Το συμπέρασμα αυτό εξάγεται από τη χρήση ενός φίλτρου που ελέγχει εάν το πλήθος διελεύσεων των οχημάτων για το προηγούμενο 1 λεπτό από το συγκεκριμένο σταθμό, ήταν μικρότερο των 100 οχημάτων.
8	Metamorfosi_TrafficAlert	Ρεύμα εξόδου που δείχνει ότι η κυκλοφορία των οχημάτων στο σταθμό Μεταμόρφωσης βρίσκεται σε επίπεδα άνω του φυσιολογικού ορίου. Το συμπέρασμα αυτό εξάγεται από τη χρήση ενός φίλτρου που ελέγχει εάν το πλήθος διελεύσεων των οχημάτων για το προηγούμενο 1 λεπτό από το συγκεκριμένο σταθμό, ήταν μεγαλύτερο των 100 οχημάτων. Στην περίπτωση αυτή παράγεται ειδοποίηση (traffic alert) αυξημένης κίνησης
9	Total_number_of_vehicles_today	Ρεύμα εξόδου που δείχνει το συνολικό αριθμό οχημάτων που έχουν περάσει και από τους δύο σταθμούς διοδίων (Κηφισίας και Μεταμόρφωσης) από την αρχή της μέρας. Η ένδειξη ενημερώνεται κάθε 1 λεπτό.
10	Totals_per_epass_use_today	Ρεύμα εξόδου που δείχνει τα συνολικά έσοδα και από τους δύο σταθμούς διοδίων (Κηφισίας και Μεταμόρφωσης) από την αρχή της μέρας ομαδοποιημένα ανά χρήση ή όχι του e-pass. Η ένδειξη ενημερώνεται κάθε 1 λεπτό.

Πίνακας 4-4: Ρεύματα εξόδου της εφαρμογής

4.4.3. Τελεστές ένωσης εφαρμογής

Οι τελεστές ένωσης που χρησιμοποιήθηκαν, είχαν ως σκοπό την ένωση των ρευμάτων εισόδου ή ρευμάτων που προέκυψαν από άλλους τελεστές έτσι ώστε να καταστεί δυνατή η επεξεργασία των συνολικών δεδομένων και από τους δύο σταθμούς διοδίων. Ακολουθεί η περιγραφή τους:

α/α	Τελεστής ένωσης	Περιγραφή
1	kifissias_Combine_all_vc234	Ενώνει τις πλειάδες των δύο ρευμάτων εισόδου του σταθμού διοδίων Κηφισίας
2	Metamorfosi_Combine_all_vc234	Ενώνει τις πλειάδες των δύο ρευμάτων εισόδου του σταθμού διοδίων Κηφισίας

3	Combine_all_vehicle_traffic	Ενώνει τις πλειάδες όλων των ρευμάτων εισόδου και των δύο σταθμών διοδίων, για τον υπολογισμό της συνολικής κυκλοφορίας οχημάτων ανά ημέρα
4	Combine_all_fees	Ενώνει τις πλειάδες όλων των ρευμάτων εισόδου και των δύο σταθμών διοδίων, για τον υπολογισμό των συνολικών εσόδων ανά ημέρα

Πίνακας 4-5: Τελεστές ένωσης της εφαρμογής

4.4.4. Τελεστές συνάθροισης εφαρμογής

Η υλοποίηση των τελεστών συνάθροισης, αντιστοιχεί ουσιαστικά στην υλοποίηση παραθύρων. Η εφαρμογή περιλαμβάνει δύο τύπους παραθύρων (επάλληλων και χρονικού οροσήμου). Ο πίνακας που ακολουθεί καταγράφει τους τελεστές συνάθροισης, τον τύπο παραθύρων και τη λειτουργία τους, ενώ τα σχήματα 4-8, 4-9 και 4-10, δείχνουν τον τρόπο ορισμού των παραθύρων μέσα από το περιβάλλον ανάπτυξης του StreamBase.

α/α	Τελεστής συνάθροισης	Περιγραφή
1	Kifissias_agg_per_vc_30secs	Ο τελεστής αυτός, ομαδοποιεί τα δεδομένα του πλήθους διελεύσεων οχημάτων από το σταθμό διοδίων Κηφισίας ανά κατηγορία οχήματος ανά 30 δευτερόλεπτα, ανεξάρτητα από τη χρήση ή όχι του e-pass.
2	Kifissias_agg_epass_use_30secs	Ο τελεστής αυτός, ομαδοποιεί τα έσοδα από το σταθμό διοδίων Κηφισίας με βάση τη χρήση ή όχι του e-pass. Αναφέρεται στο σύνολο των οχημάτων ανεξαρτήτως κατηγορίας. Ο τελεστής παράγει δεδομένα ανά 30 δευτερόλεπτα.
3	Metamorfosi_agg_epass_use_30secs	Ο τελεστής αυτός, ομαδοποιεί τα έσοδα από το σταθμό διοδίων Μεταμόρφωσης με βάση τη χρήση ή όχι του e-pass. Αναφέρεται στο σύνολο των οχημάτων ανεξαρτήτως κατηγορίας. Ο τελεστής παράγει δεδομένα ανά 30 δευτερόλεπτα.
4	Metamorfosi_agg_per_vc_30secs	Ο τελεστής αυτός, ομαδοποιεί τα δεδομένα του πλήθους διελεύσεων οχημάτων από το σταθμό διοδίων Μεταμόρφωσης ανά κατηγορία οχήματος ανά 30 δευτερόλεπτα, ανεξάρτητα από τη χρήση ή όχι του e-pass.
5	Kifissias_agg_all_vehicles_60secs	Ο τελεστής αυτός, ομαδοποιεί τα δεδομένα του πλήθους διελεύσεων οχημάτων από το σταθμό διοδίων Κηφισίας ανά 1 λεπτό ανεξάρτητα από την κατηγορία οχήματος και τη χρήση ή όχι του e-pass.

6	Metamorfosi_agg_all_vehicles_60secs	Ο τελεστής αυτός, ομαδοποιεί τα δεδομένα του πλήθους διελεύσεων οχημάτων από το σταθμό διοδίων Μεταμόρφωσης ανά 1 λεπτό ανεξάρτητα από την κατηγορία οχήματος και τη χρήση ή όχι του e-pass.
7	Total_pass_of_vehicles_today	Ο τελεστής αυτός, αθροίζει το πλήθος διελεύσεων όλων των οχημάτων και από τους δύο σταθμούς διοδίων από την αρχή της μέρας, ανεξάρτητα από την κατηγορία οχήματος και τη χρήση ή όχι του e-pass. Παράγει δεδομένα ανά 1 λεπτό
8	Totalls_based_on_epass_use_today	Ο τελεστής αυτός, αθροίζει τα έσοδα και από τους δύο σταθμούς διοδίων από την αρχή της μέρας, ομαδοποιημένα με βάση τη χρήση ή όχι του e-pass, αλλά ανεξάρτητα από την κατηγορία οχήματος. Παράγει δεδομένα ανά 1 λεπτό

Πίνακας 4-6: Τελεστές συνάθροισης της εφαρμογής

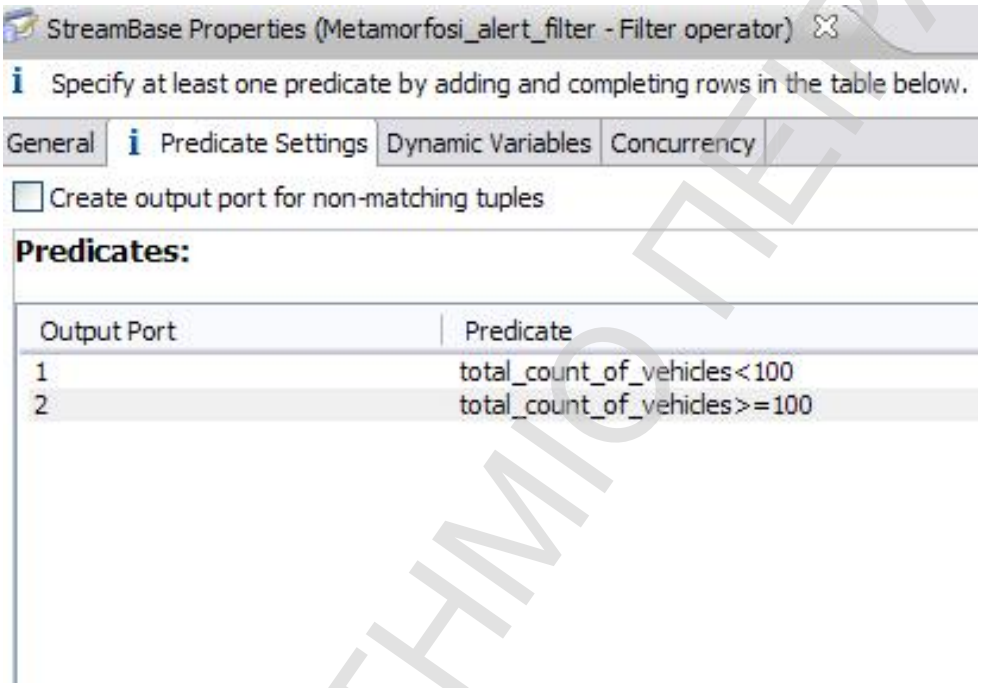
Εικόνα 4-8: Δήλωση επάλληλου παραθύρου εύρους (tumbling window), βήματος 30 δευτερολέπτων

Εικόνα 4-9: Δήλωση επάλληλου παραθύρου εύρους (tumbling window), βήματος 1 λεπτού

Εικόνα 4-10: Δήλωση παραθύρου χρονικού οροσήμου (landmark window)

4.4.5. **Φίλτρα εφαρμογής**

Τα φίλτρα που χρησιμοποιήθηκαν είχαν σκοπό την παραγωγή ειδοποιήσεων για αυξημένη κίνηση στους σταθμούς διοδίων. Οι τελεστές αυτοί, ουσιαστικά υλοποιούν την έννοια του ελέγχου των κατηγορημάτων (predicates) αφού ανιχνεύουν την ικανοποίηση ή όχι συγκεκριμένων τιμών που περιέχονται στις πλειάδες εισόδου τους. Και οι δύο τελεστές φίλτρου έχουν είσοδο πλειάδες που έχουν προκύψει από προηγούμενες συναθροίσεις και ελέγχουν την τιμή συγκεκριμένων πεδίων. Το σχήμα 4-11 δείχνει τον τρόπο ορισμού των φίλτρων μέσα από το περιβάλλον του StreamBase.



Εικόνα 4-11: Ορισμός φίλτρων εφαρμογής

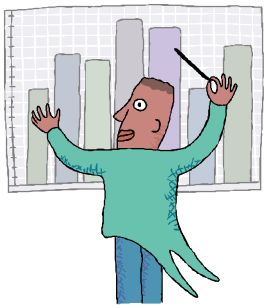
Ακολουθεί πίνακας με την περιγραφή των τελεστών φίλτρων και της λειτουργίας τους:

α/α	Φίλτρο	Περιγραφή
1	Kifissias_alert_filter	Παράγει ειδοποίηση αυξημένης κυκλοφορίας (traffic alert) σε περίπτωση που ο αριθμός διελεύσεων των οχημάτων του σταθμού διοδίων Κηφισίας υπερβεί το όριο των 100 ανά λεπτό.
2	Metamorfosi_alert_filter	Παράγει ειδοποίηση αυξημένης κυκλοφορίας (traffic alert) σε περίπτωση που ο αριθμός διελεύσεων των οχημάτων του σταθμού διοδίων Μεταμόρφωσης υπερβεί το όριο των 100 ανά λεπτό.

Πίνακας 4-7: Φίλτρα της εφαρμογής

5. Εκτέλεση εφαρμογής – Αποτελέσματα

Η εκτέλεση της εφαρμογής έγινε μέσα από το περιβάλλον του συστήματος StreamBase και αφορούσε και στα τέσσερα ρεύματα εισόδου που όπως βλέπουμε και στις οθόνες που παρατίθενται στη συνέχεια, τροφοδοτούνταν από τους αντίστοιχους προσομοιωτές.



Επίσης βλέπουμε ότι, ενώ κατά τον ορισμό των προσομοιωτών ο ρυθμός παραγωγής πλειάδων ήταν μία ανά δευτερόλεπτο για κάθε ρεύμα, μέσω του περιβάλλοντος εκτέλεσης η μείωση ή και αύξηση του ρυθμού, έτσι ώστε να έχουμε διαφορετικές τιμές στο πλήθος των οχημάτων ανά χρονική περίοδο. Τα ρεύματα εξόδου ήταν δυνατό να εμφανίζονται ένα-ένα, όλα μαζί, ή τα επιθυμητά μέσω της οθόνης επιλογής ρευμάτων εξόδου.

Τα ακόλουθα σχήματα (5-1 εως και 5-6) απεικονίζουν την εξέλιξη της εκτέλεσης της εφαρμογής σε διαφορετικά χρονικά σημεία και για διάφορες περιπτώσεις. Τα συνολικά αποτελέσματα (ρεύματα εξόδου) παρατίθενται στον πίνακα 5-1.

Project	Name	Status
tollStats	metamorfosi_tollgate_vc234_noepass.sbfs	running (sent 31)
tollStats	metamorfosi_tollgate_vc234_epass_feed.sbfs	running (sent 34)
tollStats	kifissias_tollgate_vc234_noepass.sbfs	running (sent 37)
tollStats	kifissias_tollgate_vc234_epass_feed.sbfs	running (sent 39)

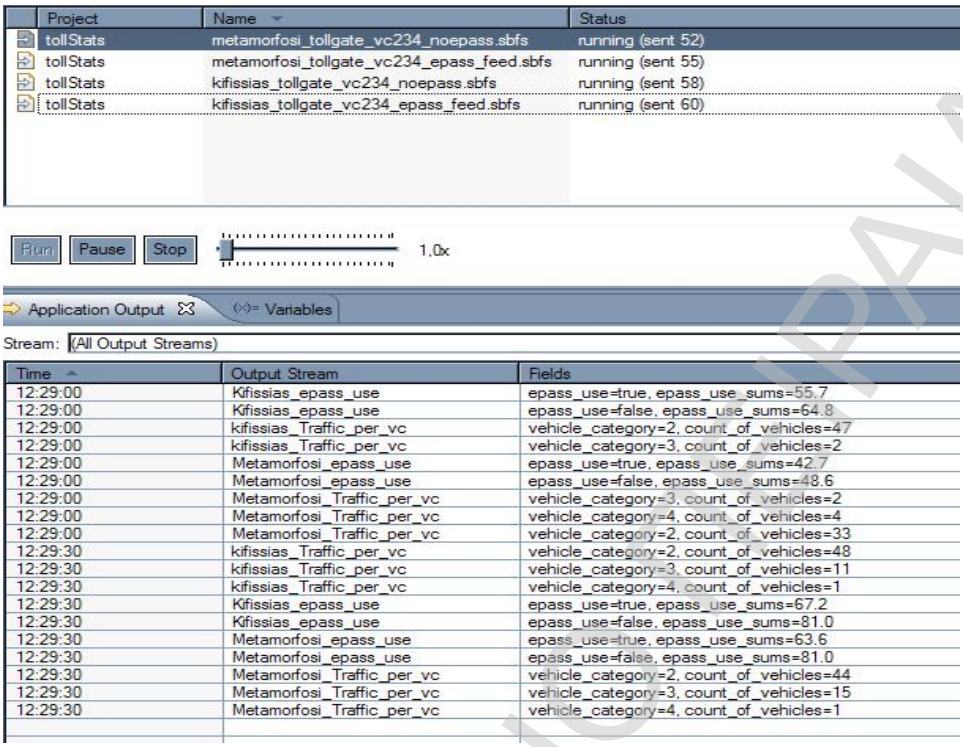
Run Pause Stop 1.0x

Application Output (x)= Variables

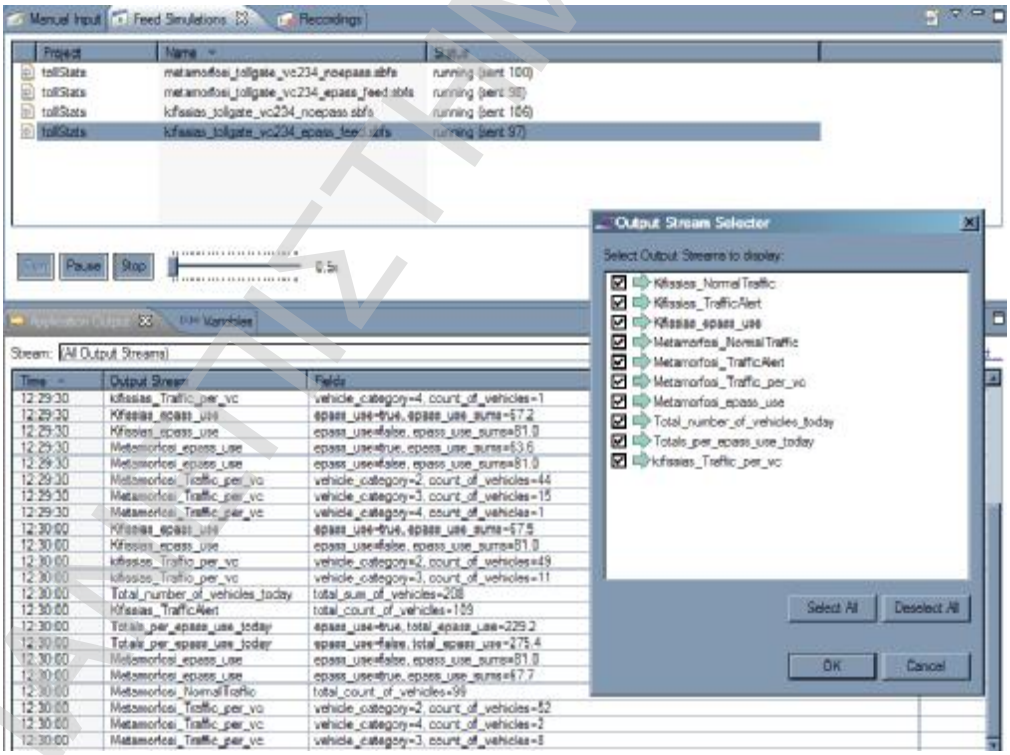
Stream: (All Output Streams)

Time	Output Stream	Fields
12:29:00	Kifissias_epass_use	epass_use=true, epass_use_sums=55.7
12:29:00	Kifissias_epass_use	epass_use=false, epass_use_sums=64.8
12:29:00	kifissias_Traffic_per_vc	vehicle_category=2, count_of_vehicles=47
12:29:00	kifissias_Traffic_per_vc	vehicle_category=3, count_of_vehicles=2
12:29:00	Metamorfosi_epass_use	epass_use=true, epass_use_sums=42.7
12:29:00	Metamorfosi_epass_use	epass_use=false, epass_use_sums=48.6
12:29:00	Metamorfosi_Traffic_per_vc	vehicle_category=3, count_of_vehicles=2
12:29:00	Metamorfosi_Traffic_per_vc	vehicle_category=4, count_of_vehicles=4
12:29:00	Metamorfosi_Traffic_per_vc	vehicle_category=2, count_of_vehicles=33

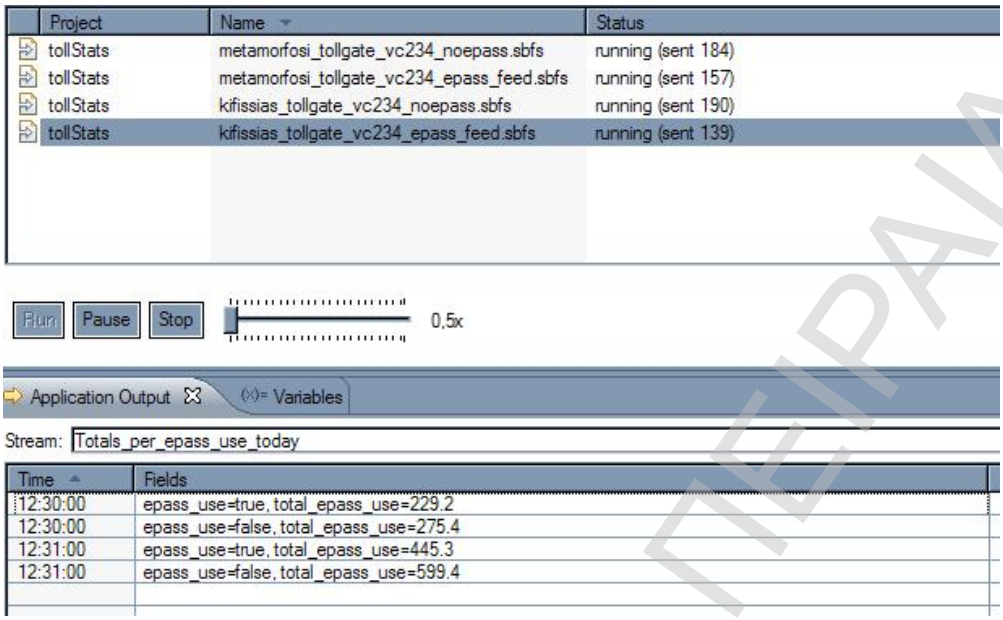
Εικόνα 5-1: Έναρξη εκτέλεσης εφαρμογής



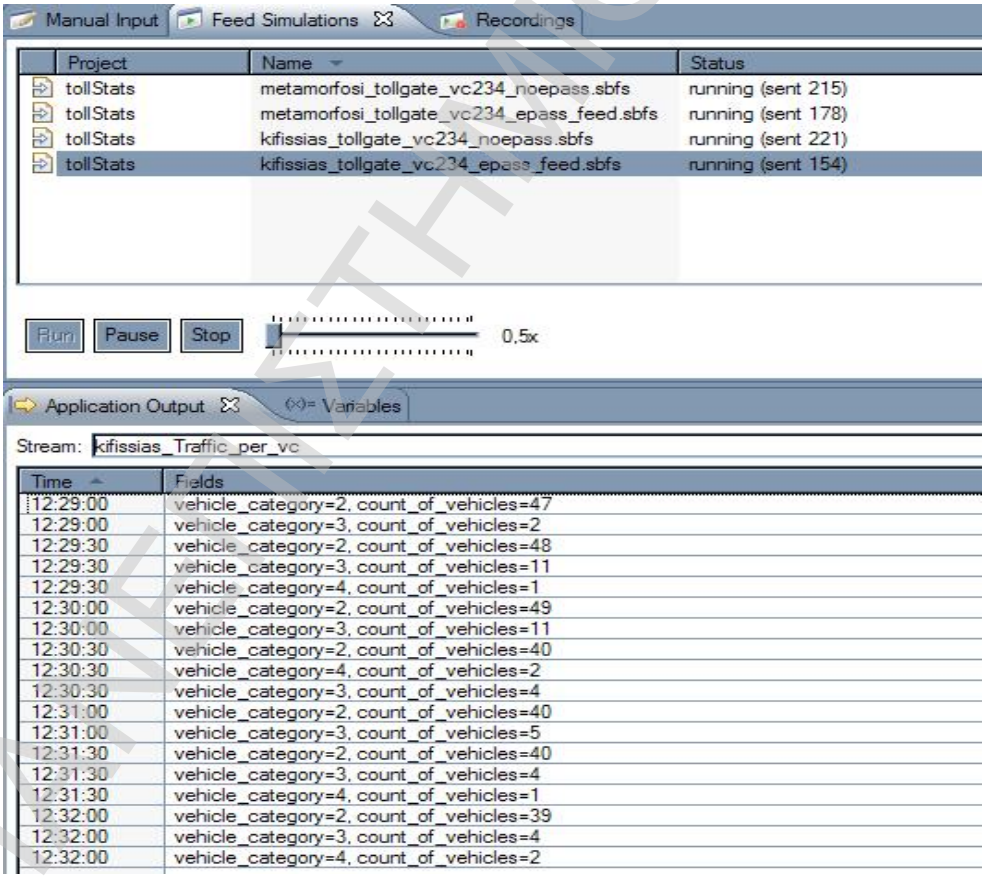
Εικόνα 5-2: Κατάσταση εκτέλεσης μετά από 30 δευτερόλεπτα



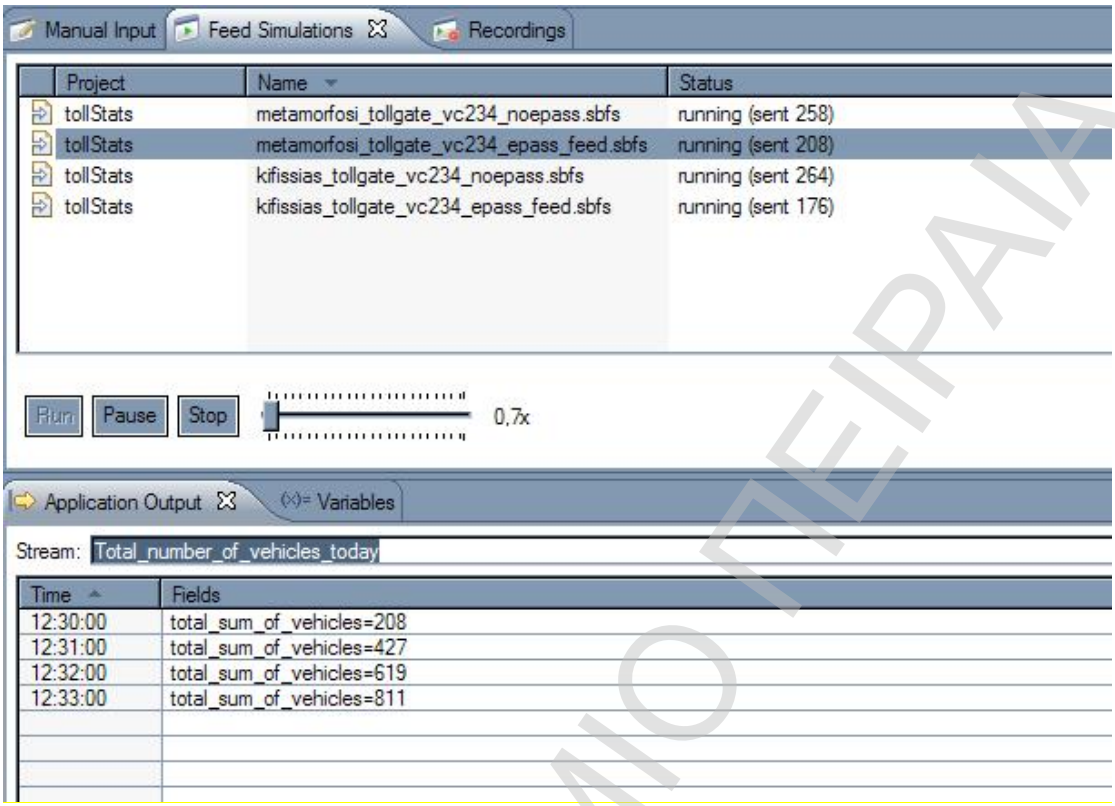
Εικόνα 5-3: Δυνατότητα επιλογής προβολής ρευμάτων εξόδου



Εικόνα 5-4: Μεταβολή (μείωση) ρυθμού παραγωγής πλειάδων



Εικόνα 5-5: Προβολή συγκεκριμένου ρεύματος εξόδου



Εικόνα 5-6: Ρεύμα δεδομένων εξόδου: πλήθος οχημάτων ανά μέρα

Ακολουθούν τα συνολικά αποτελέσματα των ρευμάτων εξόδου για όλη τη διάρκεια εκτέλεσης της εφαρμογής:

Time	Output Stream	Fields
12:29:00	Kifissias_epass_use	epass_use=true, epass_use_sums=55.7
12:29:00	Kifissias_epass_use	epass_use=false, epass_use_sums=64.8
12:29:00	kifissias_Traffic_per_vc	vehicle_category=2, count_of_vehicles=47
12:29:00	kifissias_Traffic_per_vc	vehicle_category=3, count_of_vehicles=2
12:29:00	Metamorfosi_epass_use	epass_use=true, epass_use_sums=42.7
12:29:00	Metamorfosi_epass_use	epass_use=false, epass_use_sums=48.6
12:29:00	Metamorfosi_Traffic_per_vc	vehicle_category=3, count_of_vehicles=2
12:29:00	Metamorfosi_Traffic_per_vc	vehicle_category=4, count_of_vehicles=4
12:29:00	Metamorfosi_Traffic_per_vc	vehicle_category=2, count_of_vehicles=33
12:29:30	kifissias_Traffic_per_vc	vehicle_category=2, count_of_vehicles=48
12:29:30	kifissias_Traffic_per_vc	vehicle_category=3, count_of_vehicles=11
12:29:30	kifissias_Traffic_per_vc	vehicle_category=4, count_of_vehicles=1
12:29:30	Kifissias_epass_use	epass_use=true, epass_use_sums=67.2
12:29:30	Kifissias_epass_use	epass_use=false, epass_use_sums=81.0
12:29:30	Metamorfosi_epass_use	epass_use=true, epass_use_sums=63.6
12:29:30	Metamorfosi_epass_use	epass_use=false, epass_use_sums=81.0
12:29:30	Metamorfosi_Traffic_per_vc	vehicle_category=2, count_of_vehicles=44
12:29:30	Metamorfosi_Traffic_per_vc	vehicle_category=3, count_of_vehicles=15
12:29:30	Metamorfosi_Traffic_per_vc	vehicle_category=4, count_of_vehicles=1

12:30:00	Kifissias_epass_use	epass_use=true, epass_use_sums=67.5
12:30:00	Kifissias_epass_use	epass_use=false, epass_use_sums=81.0
12:30:00	kifissias_Traffic_per_vc	vehicle_category=2, count_of_vehicles=49
12:30:00	kifissias_Traffic_per_vc	vehicle_category=3, count_of_vehicles=11
12:30:00	Total_number_of_vehicles_today	total_sum_of_vehicles=208
12:30:00	Kifissias_TrafficAlert	total_count_of_vehicles=109
12:30:00	Totals_per_epass_use_today	epass_use=true, total_epass_use=229.2
12:30:00	Totals_per_epass_use_today	epass_use=false, total_epass_use=275.4
12:30:00	Metamorfosi_epass_use	epass_use=false, epass_use_sums=81.0
12:30:00	Metamorfosi_epass_use	epass_use=true, epass_use_sums=67.7
12:30:00	Metamorfosi_NormalTraffic	total_count_of_vehicles=99
12:30:00	Metamorfosi_Traffic_per_vc	vehicle_category=2, count_of_vehicles=52
12:30:00	Metamorfosi_Traffic_per_vc	vehicle_category=4, count_of_vehicles=2
12:30:00	Metamorfosi_Traffic_per_vc	vehicle_category=3, count_of_vehicles=8
12:30:30	kifissias_Traffic_per_vc	vehicle_category=2, count_of_vehicles=40
12:30:30	kifissias_Traffic_per_vc	vehicle_category=4, count_of_vehicles=2
12:30:30	kifissias_Traffic_per_vc	vehicle_category=3, count_of_vehicles=4
12:30:30	Kifissias_epass_use	epass_use=true, epass_use_sums=35.9
12:30:30	Kifissias_epass_use	epass_use=false, epass_use_sums=81.0
12:30:30	Metamorfosi_Traffic_per_vc	vehicle_category=3, count_of_vehicles=11
12:30:30	Metamorfosi_Traffic_per_vc	vehicle_category=2, count_of_vehicles=39
12:30:30	Metamorfosi_Traffic_per_vc	vehicle_category=4, count_of_vehicles=1
12:30:30	Metamorfosi_epass_use	epass_use=false, epass_use_sums=81.0
12:30:30	Metamorfosi_epass_use	epass_use=true, epass_use_sums=45.0
12:31:00	Total_number_of_vehicles_today	total_sum_of_vehicles=427
12:31:00	kifissias_Traffic_per_vc	vehicle_category=2, count_of_vehicles=40
12:31:00	kifissias_Traffic_per_vc	vehicle_category=3, count_of_vehicles=5
12:31:00	Kifissias_TrafficAlert	total_count_of_vehicles=106
12:31:00	Totals_per_epass_use_today	epass_use=true, total_epass_use=445.3
12:31:00	Totals_per_epass_use_today	epass_use=false, total_epass_use=599.4
12:31:00	Kifissias_epass_use	epass_use=false, epass_use_sums=81.0
12:31:00	Kifissias_epass_use	epass_use=true, epass_use_sums=33.6
12:31:00	Metamorfosi_Traffic_per_vc	vehicle_category=2, count_of_vehicles=45
12:31:00	Metamorfosi_Traffic_per_vc	vehicle_category=3, count_of_vehicles=5
12:31:00	Metamorfosi_Traffic_per_vc	vehicle_category=4, count_of_vehicles=1
12:31:00	Metamorfosi_epass_use	epass_use=false, epass_use_sums=81.0
12:31:00	Metamorfosi_epass_use	epass_use=true, epass_use_sums=46.5
12:31:00	Metamorfosi_TrafficAlert	total_count_of_vehicles=113
12:31:30	Kifissias_epass_use	epass_use=false, epass_use_sums=81.0
12:31:30	Kifissias_epass_use	epass_use=true, epass_use_sums=31.0
12:31:30	kifissias_Traffic_per_vc	vehicle_category=2, count_of_vehicles=40
12:31:30	kifissias_Traffic_per_vc	vehicle_category=3, count_of_vehicles=4
12:31:30	kifissias_Traffic_per_vc	vehicle_category=4, count_of_vehicles=1
12:31:30	Metamorfosi_epass_use	epass_use=false, epass_use_sums=81.0
12:31:30	Metamorfosi_epass_use	epass_use=true, epass_use_sums=45.3
12:31:30	Metamorfosi_Traffic_per_vc	vehicle_category=3, count_of_vehicles=7
12:31:30	Metamorfosi_Traffic_per_vc	vehicle_category=2, count_of_vehicles=42
12:31:30	Metamorfosi_Traffic_per_vc	vehicle_category=4, count_of_vehicles=2
12:32:00	Kifissias_epass_use	epass_use=false, epass_use_sums=81.0
12:32:00	Kifissias_epass_use	epass_use=true, epass_use_sums=33.6

12:32:00	Totals_per_epass_use_today	epass_use=true, total_epass_use=601.7
12:32:00	Totals_per_epass_use_today	epass_use=false, total_epass_use=923.4
12:32:00	kifissias_Traffic_per_vc	vehicle_category=2, count_of_vehicles=39
12:32:00	kifissias_Traffic_per_vc	vehicle_category=3, count_of_vehicles=4
12:32:00	kifissias_Traffic_per_vc	vehicle_category=4, count_of_vehicles=2
12:32:00	Kifissias_NormalTraffic	total_count_of_vehicles=90
12:32:00	Total_number_of_vehicles_today	total_sum_of_vehicles=619
12:32:00	Metamorfosi_TrafficAlert	total_count_of_vehicles=102
12:32:00	Metamorfosi_Traffic_per_vc	vehicle_category=2, count_of_vehicles=42
12:32:00	Metamorfosi_Traffic_per_vc	vehicle_category=3, count_of_vehicles=6
12:32:00	Metamorfosi_Traffic_per_vc	vehicle_category=4, count_of_vehicles=3
12:32:00	Metamorfosi_epass_use	epass_use=false, epass_use_sums=81.0
12:32:00	Metamorfosi_epass_use	epass_use=true, epass_use_sums=45.3
12:32:30	Kifissias_epass_use	epass_use=false, epass_use_sums=81.0
12:32:30	Kifissias_epass_use	epass_use=true, epass_use_sums=33.0
12:32:30	kifissias_Traffic_per_vc	vehicle_category=2, count_of_vehicles=41
12:32:30	kifissias_Traffic_per_vc	vehicle_category=3, count_of_vehicles=4
12:32:30	Metamorfosi_epass_use	epass_use=false, epass_use_sums=81.0
12:32:30	Metamorfosi_epass_use	epass_use=true, epass_use_sums=42.4
12:32:30	Metamorfosi_Traffic_per_vc	vehicle_category=2, count_of_vehicles=42
12:32:30	Metamorfosi_Traffic_per_vc	vehicle_category=3, count_of_vehicles=9
12:33:00	kifissias_Traffic_per_vc	vehicle_category=2, count_of_vehicles=40
12:33:00	kifissias_Traffic_per_vc	vehicle_category=3, count_of_vehicles=4
12:33:00	kifissias_Traffic_per_vc	vehicle_category=4, count_of_vehicles=1
12:33:00	Kifissias_epass_use	epass_use=false, epass_use_sums=81.0
12:33:00	Kifissias_epass_use	epass_use=true, epass_use_sums=32.1
12:33:00	Kifissias_NormalTraffic	total_count_of_vehicles=90
12:33:00	Total_number_of_vehicles_today	total_sum_of_vehicles=811
12:33:00	Totals_per_epass_use_today	epass_use=true, total_epass_use=756.0
12:33:00	Totals_per_epass_use_today	epass_use=false, total_epass_use=1247.4
12:33:00	Metamorfosi_epass_use	epass_use=false, epass_use_sums=81.0
12:33:00	Metamorfosi_epass_use	epass_use=true, epass_use_sums=45.3
12:33:00	Metamorfosi_TrafficAlert	total_count_of_vehicles=102
12:33:00	Metamorfosi_Traffic_per_vc	vehicle_category=3, count_of_vehicles=5
12:33:00	Metamorfosi_Traffic_per_vc	vehicle_category=2, count_of_vehicles=44
12:33:00	Metamorfosi_Traffic_per_vc	vehicle_category=4, count_of_vehicles=2
12:33:30	Kifissias_epass_use	epass_use=false, epass_use_sums=81.0
12:33:30	Kifissias_epass_use	epass_use=true, epass_use_sums=30.1
12:33:30	kifissias_Traffic_per_vc	vehicle_category=2, count_of_vehicles=38
12:33:30	kifissias_Traffic_per_vc	vehicle_category=3, count_of_vehicles=6
12:33:30	kifissias_Traffic_per_vc	vehicle_category=4, count_of_vehicles=1
12:33:30	Metamorfosi_epass_use	epass_use=false, epass_use_sums=81.0
12:33:30	Metamorfosi_epass_use	epass_use=true, epass_use_sums=43.0
12:33:30	Metamorfosi_Traffic_per_vc	vehicle_category=2, count_of_vehicles=43
12:33:30	Metamorfosi_Traffic_per_vc	vehicle_category=4, count_of_vehicles=1
12:33:30	Metamorfosi_Traffic_per_vc	vehicle_category=3, count_of_vehicles=7
12:34:01	Kifissias_NormalTraffic	total_count_of_vehicles=90
12:34:01	kifissias_Traffic_per_vc	vehicle_category=2, count_of_vehicles=18
12:34:01	kifissias_Traffic_per_vc	vehicle_category=3, count_of_vehicles=2
12:34:01	kifissias_Traffic_per_vc	vehicle_category=4, count_of_vehicles=1

12:34:01	Total_number_of_vehicles_today	total_sum_of_vehicles=1003
12:34:01	Kifissias_epass_use	epass_use=false, epass_use_sums=16.2
12:34:01	Kifissias_epass_use	epass_use=true, epass_use_sums=33.3
12:34:01	Totals_per_epass_use_today	epass_use=true, total_epass_use=906.5
12:34:01	Totals_per_epass_use_today	epass_use=false, total_epass_use=1571.4
12:34:01	Metamorfosi_epass_use	epass_use=false, epass_use_sums=32.4
12:34:01	Metamorfosi_epass_use	epass_use=true, epass_use_sums=46.2
12:34:01	Metamorfosi_TrafficAlert	total_count_of_vehicles=102
12:34:01	Metamorfosi_Traffic_per_vc	vehicle_category=2, count_of_vehicles=27
12:34:01	Metamorfosi_Traffic_per_vc	vehicle_category=3, count_of_vehicles=6
12:34:30	kifissias_Traffic_per_vc	vehicle_category=3, count_of_vehicles=9
12:34:30	kifissias_Traffic_per_vc	vehicle_category=2, count_of_vehicles=16
12:34:30	Kifissias_epass_use	epass_use=true, epass_use_sums=54.5
12:34:31	Metamorfosi_Traffic_per_vc	vehicle_category=2, count_of_vehicles=18
12:34:31	Metamorfosi_Traffic_per_vc	vehicle_category=4, count_of_vehicles=1
12:34:31	Metamorfosi_Traffic_per_vc	vehicle_category=3, count_of_vehicles=2
12:34:31	Metamorfosi_epass_use	epass_use=true, epass_use_sums=45.6
12:35:00	Metamorfosi_epass_use	epass_use=true, epass_use_sums=55.9
12:35:00	Metamorfosi_NormalTraffic	total_count_of_vehicles=54
12:35:00	Metamorfosi_Traffic_per_vc	vehicle_category=2, count_of_vehicles=21
12:35:00	Metamorfosi_Traffic_per_vc	vehicle_category=3, count_of_vehicles=5
12:35:00	Metamorfosi_Traffic_per_vc	vehicle_category=4, count_of_vehicles=1
12:35:00	Kifissias_epass_use	epass_use=true, epass_use_sums=129.7
12:35:00	kifissias_Traffic_per_vc	vehicle_category=2, count_of_vehicles=53
12:35:00	kifissias_Traffic_per_vc	vehicle_category=3, count_of_vehicles=5
12:35:00	kifissias_Traffic_per_vc	vehicle_category=4, count_of_vehicles=2
12:35:00	Kifissias_NormalTraffic	total_count_of_vehicles=46
12:35:00	Total_number_of_vehicles_today	total_sum_of_vehicles=1130
12:35:00	Totals_per_epass_use_today	epass_use=true, total_epass_use=1142.0
12:35:00	Totals_per_epass_use_today	epass_use=false, total_epass_use=1620.0
12:35:34	Kifissias_epass_use	epass_use=true, epass_use_sums=20.1
12:35:34	Kifissias_epass_use	epass_use=null, epass_use_sums=null
12:35:34	kifissias_Traffic_per_vc	vehicle_category=3, count_of_vehicles=2
12:35:34	kifissias_Traffic_per_vc	vehicle_category=2, count_of_vehicles=7

Πίνακας 5-1: Πλειάδες όλων των ρευμάτων δεδομένων εξόδου της εφαρμογής

6. Συμπεράσματα – Ανακεφαλαίωση

6.1. Εισαγωγή

Αντικείμενο της παρούσας εργασίας ήταν να παρουσιάσει την τρέχουσα κατάσταση στην οποία βρίσκεται η ανάπτυξη συστημάτων διαχείρισης ροών δεδομένων και οι σύγχρονες υλοποιήσεις και προοπτικές τους. Στο θεωρητικό μέρος αναφέρθηκαν οι θέσεις και λύσεις που τα πειραματικά ΣΔΡΔ προτείνουν σήμερα για την αντιμετώπιση των διαφόρων θεμάτων και προβλημάτων αρχιτεκτονικής, διαχείρισης ρευμάτων, βελτιστοποίησης, παροχής υπηρεσιών και γενικότερων θεμάτων υλοποίησης. Επίσης παρουσιάστηκαν τα κυριότερα θέματα που παρουσιάζουν ενδιαφέρον και κινητικότητα στους διαφόρους τομείς έρευνας των συστημάτων αυτών και που αποτελούν εν δυνάμει, προοπτικές εξέλιξης. Επίσης έγινε αναφορά σε εμπορικές υλοποιήσεις ΣΔΡΔ, με σκοπό την ανάδειξη των στοιχείων εκείνων που υιοθετήθηκαν από τα πειραματικά συστήματα και αποτέλεσαν τη βάση ανάπτυξης εμπορικών συστημάτων. Στο πρακτικό μέρος της εργασίας, υλοποιήθηκε μια εφαρμογή με χρήση ενός εμπορικού ΣΔΡΔ με σκοπό την επίδειξη και ανάδειξη, πρακτικά πλέον, των λειτουργιών που ένα σύγχρονο σύστημα της κατηγορίας αυτής προσφέρει.

Τα συμπεράσματα που προκύπτουν θα μπορούσαν να συνοψισθούν στα παρακάτω σημεία:

1. Σημαντικά βήματα έχουν γίνει στον τομέα των ΣΔΡΔ. Τα ερευνητικά συστήματα προσφέρουν ήδη σημαντικές δυνατότητες και αποτελούν πυρήνες για περαιτέρω εξέλιξη με τα θέματα που καθημερινά προκύπτουν και ερευνώνται.
2. Οι κυριότεροι τομείς έρευνας στα ΣΔΡΔ είναι η βελτιστοποίηση πλάνων εκτέλεσης, η διαχείριση μνήμης και γενικότερα πόρων συστήματος και η προσαρμοστικότητα στα συνεχώς μεταβαλλόμενα δεδομένα που καλούνται να διαχειριστούν.
3. Η ανάπτυξη εμπορικών ΣΔΡΔ έχει βασιστεί στη χρήση χαρακτηριστικών και δυνατοτήτων που συναντώνται στα ερευνητικά συστήματα. Θα μπορούσαμε να πούμε ότι το σύνολο των χαρακτηριστικών και δυνατοτήτων που τα ερευνητικά συστήματα παρέχουν, είναι υπερσύνολο των λειτουργιών που προσφέρει ένα εμπορικό σύστημα. Από την άλλη πλευρά, οι δυνατότητες που προσφέρονται μέσω των εμπορικών συστημάτων αποτελούν υπερσύνολο ενός οποιουδήποτε ερευνητικού συστήματος. Εξαρτάται πάντα από τον τρόπο που επέλεξε κάθε εμπορικό σύστημα να αφομοιώσει, να παραμετροποιήσει και να εξελίξει ένα χαρακτηριστικό του ερευνητικού συστήματος.
4. Βασικό χαρακτηριστικό των εμπορικών ΣΔΡΔ είναι η δυνατότητα διαχείρισης μεγάλων όγκων δεδομένων, η επεκτασιμότητα καθώς και η ευκολία ανάπτυξης εφαρμογών μέσα από ολοκληρωμένα περιβάλλοντα σχεδίασης, ανάπτυξης, ελέγχου και εκτέλεσης.
5. Τα σύγχρονα ΣΔΡΔ προσφέρουν δυνατότητες διαλειτουργικότητας και συμβατότητας με υπάρχοντα πρωτόκολλα και συστήματα. Έτσι εξασφαλίζεται η προς τα πίσω συμβατότητα και η διασύνδεση με υπάρχοντα συστήματα.

6. Μελλοντικές επεκτάσεις όσον αφορά στα πρωτόκολλα που υποστηρίζονται από τα εμπορικά ΣΔΡΔ θα ήταν ουσιαστική και χρήσιμη. Για παράδειγμα, το StreamBase για τη διασύνδεση με βάσεις δεδομένων υποστηρίζει μόνο το πρωτόκολλο JDBC.
7. Ένα ακόμη στοιχείο που θα μπορούσε να εφαρμοσθεί θα ήταν η δυνατότητα αλλαγής των ορίων ενός παραθύρου. Ένα χρονικό παράθυρο θα μπορούσε για παράδειγμα να αλλάζει μέγεθος (χρονικά όρια) ανάλογα με το διάστημα της ημέρας στο οποίο εκτελείται η εφαρμογή. Το ίδιο θα μπορούσε να συμβαίνει και σε ένα παράθυρο ορισμένο με βάση το πλήθος των πλειάδων.
8. Μια ακόμη πιο προχωρημένη επέκταση θα ήταν η δυνατότητα μετατροπής του τύπου ενός παραθύρου σε κάποιο άλλο. Ένα χρονικό παράθυρο θα μπορούσε να μετατραπεί ή να χρησιμοποιήσει τεχνικές παραθύρων με βάση το πλήθος πλειάδων. Για παράδειγμα θα μπορούσε ένα σύστημα με χρονικά παράθυρα, να παράγει και ενδιάμεσα αποτελέσματα στις ροές εξόδου πριν να ολοκληρωθεί το διάστημα του παραθύρου. Έτσι ο χρήστης μιας εφαρμογής θα μπορεί να παίρνει και ενδιάμεσα αποτελέσματα στην περίπτωση που το εύρος ενός παραθύρου είναι μεγάλο και για κάποιο λόγο, απαιτείται ενημέρωση του ρεύματος εξόδου πριν τη λήξη του παραθύρου.

Στη συνέχεια επιχειρούμε μια αναλυτική συμπερασματική ανακεφαλαίωση των όσων είδαμε σε όλα τα προηγούμενα κεφάλαια.

6.2. **Αρχιτεκτονική**

Είδαμε ότι όσον αφορά στην αρχιτεκτονική των ΣΔΡΔ, οι λειτουργίες και οι δομές τους χωρίζονται στις περιοχές μνήμης και πρόσβασης σε αποθηκευτικό χώρο, με προτίμηση την περιοχή της μνήμης μιας και προσφέρει υψηλές ταχύτητες επεξεργασίας. Τα πεδία έρευνας στον συγκεκριμένο τομέα προσανατολίζονται σε αλγορίθμους οι οποίοι δεν απαιτούν προσπέλαση αποθηκευτικών χώρων, αλλά χρησιμοποιούν αποκλειστικά, ή στο μεγαλύτερο δυνατό μέρος, μόνο τη μνήμη του συστήματος. Το πρόβλημα εύρεσης του μεγέθους μνήμης η οποία απαιτείται για να υπολογιστεί ένα ερώτημα διάρκειας είναι αρκετά δύσκολο και αυτή τη στιγμή η έρευνα δείχνει ότι εάν δεν ξέρουμε τον όγκο των δεδομένων, είναι αδύνατον να θέσουμε ένα όριο στις απαιτήσεις μνήμης, ειδικά σε ερωτήματα τα οποία είναι σύνθετα. Έτσι πρέπει να προχωρήσουμε σε συμβιβαστικές λύσεις οι οποίες παράγουν προσεγγιστικές απαντήσεις. Οι λύσεις αυτές όμως, θα πρέπει να πληρούν κάποιες ελάχιστες προϋποθέσεις ποιότητας των προσφερομένων υπηρεσιών.

Από την άλλη πλευρά όμως, είναι επιθυμητή και η χρήση των αποθηκευτικών χώρων για το συνδυασμό ιστορικών στοιχείων με τα δεδομένα πραγματικού χρόνου των ρευμάτων. Οι προσεγγίσεις στο τομέα αυτό είναι δύο: Είτε υποστήριξη δομών σχεσιακών βάσεων δεδομένων μέσα στο ΣΔΡΔ που συνεπάγεται υποστήριξη πρόσβασης σε αποθηκευτικό χώρο εντός του συστήματος, είτε η διοχέτευση των δεδομένων σε εξωτερικά ΣΔΒΔ μέσω υφισταμένων πρωτοκόλλων σύνδεσης (π.χ.JDBC) που συνεπάγεται ανάκτηση ιστορικών δεδομένων μέσω τρίτων συστημάτων και συνδυασμό αυτών με τα ρεύματα δεδομένων πιθανά μόνο στη μνήμη του συστήματος.

6.3. **Απόρριψη όγκου δεδομένων**

Για την αντιμετώπιση υψηλών ρυθμών ρευμάτων εισόδου, είδαμε ότι προτείνεται η τεχνική της απόρριψης πλειάδων. Η έρευνα στο τομέα αυτό επικεντρώνεται κυρίως στις τεχνικές εκείνες οι οποίες θα απορρίπτουν δεδομένα βασισμένες σε σύνθετες τεχνικές οι οποίες λαμβάνουν υπόψη τους την επιλεκτικότητα των πλειάδων και άλλα σημαντικά χαρακτηριστικά του ρεύματος τα οποία θα είχαν επίπτωση στο αποτέλεσμα και όχι την τυφλή απόρριψη με βάση έναν απλό κανόνα. Η προσοχή των ερευνητών επίσης επικεντρώνεται στις απαιτήσεις επεξεργαστικής ισχύς και μνήμης για την εφαρμογή αυτών των τεχνικών μιας και θα πρέπει να επηρεάζουν το δυνατό λιγότερο το ρυθμό απόδοσης του ΣΔΡΔ.

Ακόμη, οι τεχνικές αυτές θα πρέπει να έχουν τη δυνατότητα να εξασφαλίζουν ότι δε θα απορρίπτονται δεδομένα τα οποία δεν είναι κρίσιμα για ένα ερώτημα αλλά είναι απαραίτητα για κάποιο άλλο σε ένα περιβάλλον όπου εκτελούνται πολλά ερωτήματα διαρκείας ταυτόχρονα. Το κλειδί για την επίτευξη του στόχου αυτού είναι η χρήση αποτελεσματικών μεθόδων δειγματοληψίας για τη δημιουργία ενός όσο το δυνατόν αντιπροσωπευτικότερου δείγματος δεδομένων του ρεύματος.

6.4. Ανασχετικοί τελεστές

Ένας άλλος τομέας έρευνας είδαμε ότι είναι και ο χειρισμός των θεμάτων σχετικών με τους ανασχετικούς τελεστές. Η αντικατάσταση ανασχετικών τελεστών με μη-ανασχετικούς είναι μια προσέγγιση, ειδικότερα στην περίπτωση των τελεστών συναθροίσεων. Έτσι, μη ανασχετικοί τελεστές οι οποίοι εκτελούν (σχεδόν) την ίδια λειτουργία με τον «προβληματικό» σχεσιακό τελεστή αντικαθιστώνται με σκοπό την αποδέσμευση των της κατάστασης άρα και τη δυνατότητα παραγωγής αποτελεσμάτων πριν την ολοκλήρωση της λήψης όλων των δεδομένων στην είσοδο του τελεστή, όπως για παράδειγμα ο τελεστής juggle που είναι μια μη-ανασχετική παραλλαγή τελεστή συνάθροισης.

6.5. Σημεία στίξης

Μια άλλη προσέγγιση στο πρόβλημα των ανασχετικών τελεστών, είναι η χρήση σημείων στίξης τα οποία λειτουργούν ως ένα είδος περιορισμού στο τι είναι αποδεκτό να εμφανιστεί ως πλειάδα στο ρεύμα εισόδου και στο τι όχι. Οι περιορισμοί αυτοί ενσωματώνονται στο ρεύμα εισόδου και έχουν ως αποτέλεσμα την αλλαγή της συμπεριφοράς των τελεστών που συμμετέχουν στο πλάνο εκτέλεσης του ερωτήματος και πιο συγκεκριμένα δίνουν την «άδεια» απελευθέρωσης δεδομένων στην έξοδο του τελεστή χωρίς σε διαδοχικά στάδια.

Ένα ανοιχτό πρόβλημα είναι η συσχέτιση και τυποποίηση τέτοιων σημείων στίξης με τις ανάγκες σε μνήμη για την εκτέλεση ερωτημάτων διαρκείας. Άλλα θέματα στη χρήση σημείων στίξης που ερευνώνται είναι η εύρεση εκείνων των ερωτημάτων που θα ωφελούνταν περισσότερο από τη χρήση των σημείων στίξης, η εύρεση του συνόλου εκείνου των στίξεων που θα επέφερε τη μέγιστη βελτιστοποίηση σε ένα ερώτημα διαρκείας , η εύρεση του καταλληλότερου σημείου του συστήματος όπου θα έπρεπε να ενσωματωθούν σημεία στίξης, και τέλος η θέματα σχετικά με τη βελτιστοποίηση πλάνων εκτέλεσης σε ερωτήματα διαρκείας που χρησιμοποιούν σημεία στίξης.

6.6. Προσαρμοστικότητα

Η ιδιαιτερότητα των περιστασιακών ερωτημάτων διαρκείας μετατρέπει το πρόβλημα της εύρεσης του βέλτιστου πλάνου εκτελέσεις από off-line, σε πρόβλημα πραγματικού χρόνου το οποίο προκαλεί θέματα προσαρμοστικότητας του πλάνου εκτέλεσης στα δεδομένα. Είδαμε ότι μια λύση είναι η χρήση των eddies και stems η οποία όμως θα πρέπει να επεκταθεί έτσι ώστε να χειρίζεται αποδοτικά, πολλά ερωτήματα διαρκείας ταυτόχρονα

6.7. Ιστογράμματα

Η χρήση ιστογραμμάτων αν και είναι όπως είπαμε αρκετά διαδεδομένη στα ΣΔΒΔ, εντούτοις, στα ΣΔΡΔ παρουσιάζει θέματα αποδοτικής λειτουργίας αφού απαιτεί υψηλό κόστος σε χρήση μνήμης και χρόνου υπολογισμού και συντήρησης, ειδικά όταν οι επιλεκτικότητες των πεδίων τροποποιούνται με την πάροδο του χρόνου. Εντούτοις, τα ιστογράμματα αποτελούν μια ισχυρή βάση πάνω στην οποία θα μπορούσαν να στηριχθούν προσεγγιστικά αποτελέσματα σε ερωτήματα διαρκείας, αφού χρησιμοποιούνται για την παραγωγή συνόψεων.

6.8. **Κυματίδια**

Τα κυματίδια και οι μετασχηματισμοί τους, εφαρμόζονται σε μονοδιάστατα ή πολυδιάστατα δεδομένα και αποτελούν ευρύ τομέα έρευνας στο χώρο των ΣΔΡΔ. Το μειονέκτημα είναι ότι δεν μπορούν να εφαρμοστούν σε δυναμικά δεδομένα. Έτσι, η έρευνα επικεντρώνεται κυρίως στην εύρεση των τεχνικών εκείνων που θα παράγουν προσεγγιστικά το ίδιο αποτέλεσμα με αυτό των μετασχηματισμών κυματιδίων αλλά σε περιβάλλοντα πραγματικού χρόνου. Επίσης, ερευνάται η βελτιστοποίηση των μετασχηματισμών όσον αφορά στις συνιστώσες ελαστικότητας από τις οποίες εξαρτώνται, έτσι ώστε να περιορίζεται το λάθος υπολογισμού, ο χρόνος επεξεργασίας και ο χώρος που καταλαμβάνουν στη μνήμη, ενώ ταυτόχρονα να προσεγγίζουν με μεγαλύτερη ακρίβεια τα εξαρτώμενα δεδομένα

6.9. **Παράθυρα**

Η χρήση των παραθύρων σε ΣΔΡΔ είναι ευρεία, ως εκ τούτου, πολλά ζητήματα που προκύπτουν βρίσκονται σε έρευνα, όσον αφορά στον τρόπο ορισμού τους, τη βελτιστοποίηση ερωτημάτων με χρήση παραθύρων (και ειδικότερα μάλιστα όταν εμπλέκονται περισσότερα του ενός παραθυρικά ρεύματα), το συνδυασμό τεχνικών απόρριψης δεδομένων σε παραθυρικά ρεύματα και πολλά άλλα.

6.10. **Υλοποίηση εφαρμογής**

Στο πρακτικό μέρος της εργασίας υλοποιήθηκε μια εφαρμογή διαχείρισης ρευμάτων δεδομένων με σκοπό την ανάδειξη όπως είπαμε των λειτουργιών ενός εμπορικού ΣΔΡΔ αφού ουσιαστικά αποτελεί το παρόν και το μέλλον των ερευνητικών προσπαθειών.

Το σημαντικότατο συμπέρασμα που προκύπτει από την ανάπτυξη της εφαρμογής αυτής είναι ότι τα ΣΔΡΔ είναι πλέον σε κατάσταση όπου μπορούν να υποστηρίξουν το σκοπό για τον οποίο κατασκευάσθηκαν και μάλιστα με πολλές αξιώσεις και προτερήματα σε σχέση με τα υπάρχοντα εμπορικά ΣΔΒΔ.

Ένα άλλο εξίσου σημαντικό συμπέρασμα είναι ότι η χρήση τους είναι αρκετά προσαρμοσμένη στα «σχεσιακά» δεδομένα και αρκετά φιλική προς τον προγραμματιστή και αυτό δείχνει ότι οι προοπτικές σταδιακής ενσωμάτωσης τέτοιων συστημάτων στην αγορά αλλά και περαιτέρω εξέλιξης είναι δεδομένες.

Μια μελλοντική επέκταση της υλοποιημένης εφαρμογής θα κάλυπτε το σύνολο των κατηγοριών οχημάτων και θα είχε προγράμματα πελάτες για την παρακολούθηση (dequeing clients) των αποτελεσμάτων. Θα πρέπει να τονιστεί εδώ ότι η εφαρμογή μπορεί να χρησιμοποιηθεί χωρίς καμία τροποποίηση και με πραγματικά δεδομένα από τη στιγμή που αυτά πληρούν τις προδιαγραφές (γραμμογράφηση εγγραφής) των ρευμάτων εισόδου.

7. **Αναφορές**

[1] Bai Y., Thakkar H.,Wang H., et.al. A Data Stream Language and System Designed for Power and Extensibility

[2] Daniel J. Adabi, Carney D. Cetintemel U., et.al. (2003). Aurora: a new model and architecture for data stream management, The VLDB Journal 2003,12:120–139

[3] Babu S., Widom J. (2001). Continuous Queries over Data Streams. SIGMOD Record , Vol 30, No. 3, Sep 2001

[4] Sutherland M., Liu B., Jbantova M., Rundensteiner E. (2005). DCAPE: Distributed and SelfTuned Continuous Query Processing

[5] Cranor C., Johnson T., Spataschek O. (2003). Gigascope: A Stream Database for Network Applications, SIGMOD 2003, June 9-12, 2003

[6] Cortes C., Fisher K., Pregibon D. (2004).Hancock: A Language For Analyzing Transactional Data Streams. ACM Transactions on Programming Languages and Systems, Vol. 26, No. 2, March 2004, Pages 301–338.

[7] Golab L., Ozsu T. (2003). Issues in Data Stream Management. SIGMOD Record, Vol. 32, No. 2, June 2003

[8] Babcock B.,Babu S.,Datar M. Models and Issues in Data Stream Systems Babcock

[9] Chen J., DeWitt J., et. al. (2000). NiagaraCQ: A Scalable Continuous Query System for Internet Databases

[10] Babcock B., Babu S., et. al. (2004). STREAM: The Stanford Data Stream Management System

[11] Arasu A., Babu S., Widom J. (2006). The CQL continuous query language: Semantic foundations and query execution. The VLDB Journal (2006) 15(2): 121–142

[12] Chandrasekaran S., Cooper O., Deshpande A., et. al. (2003). TelegraphCQ: Continuous Dataflow Processing for an Uncertain World. Proceedings of the 2003 CIDR Conference

[13] Babu S., Munagala K., Widom J.,et. al. Adaptive Caching for Continuous Queries

[14] Jiang Q.,Chakravarthy S. Anatomy of a data stream management system

[15] Das A., Gehrke J., Riedewald M.(2003). Approximate Join Processing Over Data Streams,SIGMOD,June 9-12,2003

[16] Madden S., Shah M., Hellerstein M.J., et. al. (2002). Continuously Adaptive Continuous Queries over Streams.ACM SIGMOD, June 4-6, 2002

[17] Ding L., Rundensteiner A. E. (2004). Evaluating Window Joins over Punctuated Streams. CIKM'04, November 8–13, 2004

[18] Motwani R.,Widom J., Arasu A., et al. Query Processing, Approximation, and Resource Management

[19] Motwani R., Widom J., Arasu A., et al. (2003). Query Processing, Resource Management, and Approximation in a Data Stream Management System. Proceedings of the 2003 CIDR Conference 2003

[20] Wei-bing F., Zhan-huai L. (2008). Technology of Continuous Query Optimization over Data Streams. International Symposium on Information Science and Engieering 2008

[21] Bai Y., Thakkar H., Zaniolo C., et. al. (2008). Time stamp management and query execution in data stream management systems. IEEE Publications 2008

[22] Raman V., Deshpande A., Hellerstein M. J. (2003). Using State Modules for Adaptive Query Processing. 2003

[23] Chaudhry A N. (2005). Introduction to stream data management. Springer publications 2005

[24] Viglas S. (2005). Query execution and optimization. Springer publications 2005

[25] Maier D., Tucker A. P., Garofalakis M. (2005). Filtering, punctuation, windows and synopses. Springer publications 2005

[26] Rundensteiner A.E., Ding L., Zhu Y., et. al. (2005). CAPE: A constraint-aware adaptive stream processing engine. Springer publications 2005

[27] Bai Y., Luo R. C., Thakkar H., et. al. (2005). Efficient support for time series queries in data stream management systems. Springer publications 2005

[28] Namit J., Gehrke J., Balakrishnan H., et. al. (2008). Towards a Streaming SQL Standard. VLDB '08 August 24-30 2008

[29] Kulkarni D., Ravishankar V. C., Cherniack M. (2008). Real-time, Load-Adaptive Processing of Continuous Queries Over Data Streams. DEBS ’08, July 1-4, 2008

[30] Hildrum K., Douglis F., Wolf L J., et al. (2008). Storage Optimization for Large-Scale Distributed Stream-Processing Systems. ACM Transactions on Storage, Vol. 3, No. 4, Article 18,Feb. 2008

[31] Stonebraker M., Çetintemel U., Zdonik S. (2005). The 8 Requirements of Real-Time Stream Processing. SIGMOD Record, Vol. 34, No. 4, Dec. 2005

[32] Stonebraker M., Hachem N., Helland P., et. al. (2007). The End of an Architectural Era (It’s Time for a Complete Rewrite). VLDB ’07, September 23-28, 2007

[33] Salvucci S., Cilia M., Buchmann A. (2007). A Practical Approach for Enabling Online Analysis of Event Streams. DEBS ’07, June 20–22, 2007

[34] Aggarwal C. C. (2003). A Framework for Diagnosing Changes in Evolving Data Streams. SIGMOD 2003, June 912, 2003

[35] Hartzman S. C, Watters R. C. (1990). A relational approach to querying data streams. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 2, NO. 4, DECEMBER 1990

[36] Olteanu D.,Furche T.,Bry F. (2004). An Efficient Single-Pass Query Evaluator for XML Data Streams SAC’04, March 1417, 2004

[37] Jiang Q., Chakravarthy S. Anatomy of a Data Stream Management System

[38] Babcock B., Babu S., Datar M., et. al.(2003). Chain - Operator Scheduling for Memory Minimization in Data Stream Systems. SIGMOD 2003, June 912, 2003

[39] Arasu A., Babcock B., Babu S., et. al. (2004). Characterizing Memory Requirements for Queries Over Continuous Data Streams. ACM Transactions on Database Systems, Vol. 29, No. 1, Pages 162–194, March 2004

[40] Ma L., Nutt W., Taylor H. (2007). Condensative Stream Query Language for Data Streams. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 63. 2007

[41] Mouratidis K., Bakiras S., Papadias D.(2006). Continuous Monitoring of Top-k Queries over Sliding Windows. SIGMOD 2006, June 27–29, 2006

[42] Witkowski A., Bellamkonda S., Li H.G. (2007). Continuous Queries in Oracle. VLDB '07, September 23-28, 2007

[43] Babu S., Widom S. (2001). Continuous queries over Data Streams. SIGMOD Record, Vol. 30, No. 3, Sep. 2001

[44] Lim S.H., Lee G.J., Whang K.Y., et. al. (2006). Continuous Query Processing in Data Streams Using Duality of Data and Queries. SIGMOD 2006, June 27–29, 2006

[45] Sharaf, M., Beaver J., Labrinidis P, et. al. (2004). COUGAR - Balancing energy efficiency and quality of aggregate data in sensor networks. VLDB Journal 13: 384–403, 2004

[46] Fung F.W., Sun D., Gehrke J. (2002). COUGAR - The network is the database. ACM SIGMOD '2002 June 4-6

[47] Goebel V., Plagemann T. (2005). Data stream management systems - a technology for network monitoring and traffic analysis. 8th International Conference on Telecommunications - ConTEL 2005

[48] Koudas N., Srivastava D. (2003). Data Stream Query Processing. Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE'03)

[49] Kumar A., Sung M., Xu J., et. al. (2004). Data Streaming Algorithms for Efficient and Accurate Estimation of Flow Size Distribution. SIGMETRICS/Performance'04, June 12–16, 2004

[50] Hung P. H., Chen M.S. (2006). Efficient Range-Constrained Similarity Search on Wavelet Synopses over Multiple Streams. CIKM'06, November 5–11, 2006

[51] Reiss F., Stockinger K., Wu K., Enabling Real-Time Querying of Live and Historical Stream Data

[52] Chen Y., Davidson B. S., Mihaila G., et. al. (2004). EXPedite - A System for Encoded XML Processing. CIKM'04, November 8–13, 2004

[53] Babu S., Srivastava U., Widom J. (2004). Exploiting k-Constraints to Reduce Memory Overhead in Continuous Queries Over Data Streams. ACM Transactions on Database Systems, Vol. 29, No. 3, Pages 545–580, September 2004

[54] Tucker P., Maier D., Sheard T., et. al. (2003). Exploiting Punctuation Semantics in continuous data streams. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 15, NO. 3, MAY/JUNE 2003

[55] Ghanem M. T., Hammad A. M., Mokbel F. M.(2007). Incremental Evaluation of Sliding-Window Queries over Data Streams. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 19, NO. 1, JANUARY 2007

[56] Berthold H., Schmidt S., Lehner W., et. al. (2005). Integrated Resource Management for Data Stream Systems. ACM Symposium on Applied Computing, SAC'05 March 13-17,2005

[57] Wu L.K., Chen K.S., Yu S.P. (2004). Interval Query Indexing for Efficient Stream Processing. CIKM'04, November 8–13, 2004

[58] Chandrasekaran S., Franklin M.J. (2003). PSoup - A system for streaming queries over streaming data. VLDB Journal (2003) 12: 140–156

[59] Jiang Q., Chakravarthy S. (2003). Queueing Analysis of Relational Operators for Continuous Data Streams. CIKM'03, November 3–8, 2003

[60] Jiang N.,Gruenwald L. (2006). Research Issues in Data Stream Association Rule Mining. SIGMOD Record, Vol. 35, No. 1, Mar. 2006

[61] Balakrishnan H., Balazinska M., Carney D., et. al. (2003). Retrospective on Aurora. VLDB Journal (2004) 13: 370–383

[62] Qiao L., Agrawal D.,Abbadi E.A. (2002). RHist - Adaptive Summarization over Continuous Data Streams. CIKM'02, November 4–9, 2002

[63] Li H.G., Chen S., Tatemura J., et. al. (2006). Safety Guarantee of Continuous Join Queries over Punctuated Data Streams. VLDB '06, September 12-15, 2006

[64] Frahling G., Indyk P., Sohler C. (2005). Sampling in Dynamic Data Streams and Applications. SCG'05, June 6–8, 2005

[65] Abdulla G., Critchlow T., Arrighi W. (2004). Simulation Data as Data Streams. SIGMOD Record, Vol. 33, No. 1, March 2004

[66] STREAM - The Stanford Stream Data Manager User Guide and Design Document

[67] Cormode G., Garofalakis M. (2007). Streaming in a Connected World - Querying and Tracking Distributed Data Streams. SIGMOD'07, June 12–14, 2007

[68] Chandrasekaran S., Cooper O., Deshpande A., et. al. (2003). TelegraphCQ - Continuous Dataflow Processing. SIGMOD 2003, June 912,2003

[69] Demers A., Gehrke J., Rajaraman R., et. al. (2003). The Cougar Project - A work in progress report. SIGMOD Record, Vol. 32, No. 4, December 2003

[70] Yao Y., Gehrke J. (2002). The Cougar approach to in-network query processing in sensor networks. SIGMOD Record, Vol. 31, No. 3, September 2002

[71] Plagemann T., Goebel V., Bergamini A., et. al. Using Data Stream Management Systems for Traffic Analysis - A Case Study

[72] StreamBase Administration Guide, StreamBase Corporarion

[73] StreamBase API Guide, StreamBase Corporarion

[74] StreamBase Authoring Guide, StreamBase Corporarion

[75] StreamBase StreamSQL Guide, StreamBase Corporarion

[76] url: http://telegraph.cs.berkeley.edu

[77] url: http://www.streambase.com

[78] url: http://www.progress.com

[79] url: http://www.coral8.com

[80] url: http://www.oracle.com

[81] url: http://www.aodos.gr

8. Λεξιλόγιο - Συντμήσεις

8.1. Λεξιλόγιο όρων

Ελληνικός Όρος	Αγγλικός Όρος
ακροατής	listener
αναδιάταξη ερωτήματος διαρκείας	query scrambling
ανασχετικός τελεστής	blocking operator
ανοιχτός κώδικας	open source
ανοχή σε σφάλματα	fault-tolerance
αντικείμενο εκτέλεσης	execution object
αντιμεταθετικότητα τελεστών	operator commutativity
αποβολή φόρτου	load shedding
απόρριψη δεδομένων	Data Dropping
αρχειοθέτηση	archive
βάρος	weight
βελτιστοποιητής	optimizer
βελτιστοποιητής ομάδων	group optimizer
γλώσσα ερωταποκρίσεων	query language
δεδομένα αναφοράς	lookup values
δειγματοληψία	sampling
δηλωτική γλώσσα	declarative language
διαμοιρασμένη μνήμη	shared memory
διαχειριστής αποθήκευσης	storage manager
διαχειριστής κατανομής	distribution manager
διεπαφή	interface
διεργασία	process
δοσοληψία	transaction
δρομέας	cursor
δυναμική ανασυγκρότηση	dynamic re-grouping
εισαγωγή	insert
εκτελεστής	executor
ενδιάμεση μνήμη	buffering
ένωση	union
εξισορρόπηση φόρτου	load balancing
εξωτερικό χρονόσημο	external timestamp
επανα-βελτιστοποιητής πλάνων	plan re-optimizer
επαναπρογραμματισμός ερωτήματος	query rescheduling
επιθεωρητής ποιότητας υπηρεσιών	quality of service inspector
επιλογή	selection - select
ερώτημα διαρκείας	continuous query
ερώτημα περιστασιακό	ad-hoc query
ερώτημα προκαθορισμένο	predefined query
ερώτημα στιγμιοτύπου	one-time, snapshot query

εσωτερικό χρονόσημο	internal timestamp
ευρετήριο	index
ευριστικός αλγορίθμους	heuristic algorithm
ισοβαθές ιστόγραμμα	equidepth histogram
ιστόγραμμα	histogram
κατάλογος	registry
κατηγόρημα	predicate
κυματίδιο	wavelet
λανθάνον χρονόσημο	latent timestamp
λανθάνουσα μνήμη	cache
μείωση απόδοσης του συστήματος	performance degradation
μεταγλωττιστής	compiler
μετασχηματισμός κυματιδίων	wavelet transformation
μη βέλτιστη απόδοση	sub-optimal
μη ορισμένα χρονόσημο	implicit timestamp
μη ανασχετικός τελεστής	non-blocking operators
μηχανισμός εκτέλεσης	execution engine
μοντέλο ορισμού και εκτέλεσης ερωτημάτων	query execution model
όριο	threshold
ορισμένα χρονόσημο	explicit timestamp
όψη	view
παράθυρο επάλληλο	tumbling window
παράθυρο ζώνης	band window
παράθυρο κυλιόμενο	sliding window
παράθυρο μεριστικό	partitioned window
παράθυρο μεταβλητού εύρους	variable-size window
παράθυρο ορισμένο με βάση το χρονικό διάστημα	time-based windows
παράθυρο ορισμένο με βάση τον αριθμό πλειάδων	tuple-based window
παράθυρο σταθερού εύρους	fixed-band window
παράθυρο χρονικού οροσήμου	landmark window
περιορισμός	constraint
πλάνο εκτέλεσης ερωτήματος	query execution plan
πολυνηματικό	multi-threaded
προβολή	projection
προσαρμογέας	adapter
προσαρμοζόμενο πλάνο εκτέλεσης	adaptive query plan
προσέγγιση	approximation
προσομοιωτής	simulator
ρεύμα δεδομένων	data stream
ρυθμοαπόδοση	throughput
σημασιολογική απόρριψη	semantic dropping
σημείο στίξης	punctuation
σκίτσο	sketch
στιγμιότυπο	snapshot - instance
συγχώνευση	merge
συγχώνευση δεδομένων	data merging
συμμετρική πολυεπεξεργασία	symmetric multi-processing
συμπεριφορά αναπαραγωγής στίξεων	propagation behavior

συμπεριφορά διατήρησης	keep behavior
συμπεριφορά παραγωγής αποτελέσματος	pass behavior
συνάθροιση	aggregation
συναθροιστικά δεδομένα	aggregated data
σύνδεση	join
συνένωση	concatenation
σύνθεση τελεστών	operator synthesis
συνιστώσες ελαστικότητας	wavelet coefficients
σύνοψη	synopsis
συστάδα	cluster
σχέση	relation
ταξινόμηση	sorting
τελεστής	operator
τελεστής απόρριψης	drop operator
τελεστής διατήρησης κατάστασης	stateful operator
τελεστής σάρωσης	scanner
τομή	intersection
τροποποίηση	update
τυφλή απόρριψη	blind dropping
υλοποιημένη όψη	materialized view
υπο-ερώτημα	subquery
φιλτροποίηση ακριβείας	precise filtering
φυσικό σχέδιο εκτέλεσης	physical query execution plan
χαρακτηριστικό πλειάδας	attribute
χειριστής ερωτήματος	query handler
χρονοπρογραμματιστής	scheduler
χρονόσημα	timestamp

8.2. **Συντμήσεις**

Σύντμηση	**Περιγραφή**
CAPE	Constraint-aware Adaptive stream Processing Engine
CCL	Continuous Computation Language
CEPS	Complex Event Processing Systems
CQL	Continuous Query Language
DAHP	DBMS Active Human Passive
DSMS	Data Stream Management System
EPL	Event Processing Language
ESL	Expressive Stream Language
Flux	Fault-tolerant Load balancing eXchange
GSQL	Gigascope Stream Query Language
GUI	Graphical User Interface
HADP	Human Active DBMS Passive
JDBC	Java Database Connectivity
LSRM	Load-Shedding Road Map
QoS	Quality of Service
SMP	Symmetric Multi-Processing
SQuAl	Stream Query Algebra
SteMs	State Modules
StreamSQL	Stream Structured Query Language
StreaQueL	Stream Query Language
ΣΔΒΔ	Σύστημα Διαχείρισης Βάσεων Δεδομένων
ΣΔΡΔ	Σύστημα Διαχείρισης Ροών Δεδομένων

9. **ΠΑΡΑΡΤΗΜΑ** - **Εντολές** StreamSQL **της εφαρμογής**

```
CREATE INPUT STREAM kifissias_tollgate_vc234_epass (
    pass_time timestamp,
    toll_gate string,
    epass_use boolean,
    vehicle_category int,
    toll_fee double
);
CREATE INPUT STREAM kifissias_tollgate_vc234_noepass (
    pass_time timestamp,
    toll_gate string,
    epass_use boolean,
    vehicle_category int,
    toll_fee double
);
CREATE INPUT STREAM metamorfosi_tollgate_vc234_epass (
    pass_time timestamp,
    toll_gate string,
    epass_use boolean,
    vehicle_category int,
    toll_fee double
);
CREATE INPUT STREAM metamorfosi_tollgate_vc234_noepass (
    pass_time timestamp,
    toll_gate string,
    epass_use boolean,
    vehicle_category int,
    toll_fee double
);
CREATE OUTPUT STREAM Kifissias_epass_use ;
CREATE OUTPUT STREAM kifissias_Traffic_per_vc ;
CREATE OUTPUT STREAM Metamorfosi_epass_use ;
CREATE OUTPUT STREAM Metamorfosi_Traffic_per_vc ;
CREATE OUTPUT STREAM Kifissias_NormalTraffic ;
CREATE OUTPUT STREAM Kifissias_TrafficAlert ;
CREATE OUTPUT STREAM Metamorfosi_NormalTraffic ;
CREATE OUTPUT STREAM Metamorfosi_TrafficAlert ;
CREATE OUTPUT STREAM Total_number_of_vehicles_today ;
CREATE OUTPUT STREAM Totals_per_epass_use_today ;

CREATE STREAM out__format_time1_1 ;
SELECT format_time(pass_time, "dd/mm/yyyy hh:mm:ss") AS
pass_time,
       toll_gate AS toll_gate,
       epass_use AS epass_use,
       vehicle_category AS vehicle_category,
       toll_fee AS toll_fee
 FROM   kifissias_tollgate_vc234_epass
 INTO   out__format_time1_1
```

```
;

CREATE STREAM out__format_time2_1 ;
SELECT format_time(pass_time, "dd/mm/yyyy hh:mm:ss") AS
pass_time,
       toll_gate AS toll_gate,
       epass_use AS epass_use,
       vehicle_category AS vehicle_category,
       toll_fee AS toll_fee
 FROM   kifissias_tollgate_vc234_noepass
 INTO   out__format_time2_1
;

CREATE STREAM out__format_time3_1 ;
SELECT format_time(pass_time, "dd/mm/yyyy hh:mm:ss") AS
pass_time,
       toll_gate AS toll_gate,
       epass_use AS epass_use,
       vehicle_category AS vehicle_category,
       toll_fee AS toll_fee
 FROM   metamorfosi_tollgate_vc234_epass
 INTO   out__format_time3_1
;

CREATE STREAM out__format_time4_1 ;
SELECT format_time(pass_time, "dd/mm/yyyy hh:mm:ss") AS
pass_time,
       toll_gate AS toll_gate,
       epass_use AS epass_use,
       vehicle_category AS vehicle_category,
       toll_fee AS toll_fee
 FROM   metamorfosi_tollgate_vc234_noepass
 INTO   out__format_time4_1
;

CREATE STREAM out__Metamorfosi_Combine_all_vc234_1 ;
SELECT * FROM out__format_time3_1
UNION SELECT * FROM out__format_time4_1
INTO out__Metamorfosi_Combine_all_vc234_1
;

CREATE STREAM out__kifissias_Combine_all_vc234_1 ;
SELECT * FROM out__format_time1_1
UNION SELECT * FROM out__format_time2_1
INTO out__kifissias_Combine_all_vc234_1
;

SELECT epass_use AS epass_use, round(sum(toll_fee),2) AS
epass_use_sums
FROM out__kifissias_Combine_all_vc234_1[SIZE 30,000 ADVANCE
30,000 TIME OFFSET 0,000]
```

```
GROUP BY epass_use
INTO Kifissias_epass_use;

SELECT vehicle_category AS vehicle_category, count(toll_gate) AS
count_of_vehicles
FROM out__kifissias_Combine_all_vc234_1[SIZE 30,000 ADVANCE
30,000 TIME OFFSET 0,000]
GROUP BY vehicle_category
INTO kifissias_Traffic_per_vc;

SELECT epass_use AS epass_use, round(sum(toll_fee),2) AS
epass_use_sums
FROM out__Metamorfosi_Combine_all_vc234_1[SIZE 30,000 ADVANCE
30,000 TIME OFFSET 0,000]
GROUP BY epass_use
INTO Metamorfosi_epass_use;

SELECT vehicle_category AS vehicle_category, count(toll_gate) AS
count_of_vehicles
FROM out__Metamorfosi_Combine_all_vc234_1[SIZE 30,000 ADVANCE
30,000 TIME OFFSET 0,000]
GROUP BY vehicle_category
INTO Metamorfosi_Traffic_per_vc;

CREATE STREAM out__Combine_all_fees_1 ;
SELECT * FROM Kifissias_epass_use
UNION SELECT * FROM Metamorfosi_epass_use
INTO out__Combine_all_fees_1
;

CREATE STREAM out__Combine_all_vehicle_traffic_1 ;
SELECT * FROM kifissias_Traffic_per_vc
UNION SELECT * FROM Metamorfosi_Traffic_per_vc
INTO out__Combine_all_vehicle_traffic_1
;

CREATE STREAM out__Kifissias_agg_all_vehicles_60secs_1 ;
SELECT sum(count_of_vehicles) AS total_count_of_vehicles
FROM kifissias_Traffic_per_vc[SIZE 60,000 ADVANCE 60,000 TIME
OFFSET 0,000]
INTO out__Kifissias_agg_all_vehicles_60secs_1;

CREATE STREAM out__Metamorfosi_agg_all_vehicles_60secs_1 ;
SELECT sum(count_of_vehicles) AS total_count_of_vehicles
FROM Metamorfosi_Traffic_per_vc[SIZE 60,000 ADVANCE 60,000 TIME
OFFSET 0,000]
INTO out__Metamorfosi_agg_all_vehicles_60secs_1;

SELECT * FROM out__Kifissias_agg_all_vehicles_60secs_1
WHERE total_count_of_vehicles<100 INTO Kifissias_NormalTraffic
WHERE total_count_of_vehicles>=100 INTO Kifissias_TrafficAlert
```

```
;

SELECT * FROM out__Metamorfosi_agg_all_vehicles_60secs_1
WHERE total_count_of_vehicles<100 INTO Metamorfosi_NormalTraffic
WHERE total_count_of_vehicles>=100 INTO Metamorfosi_TrafficAlert
;

SELECT sum(count_of_vehicles) AS total_sum_of_vehicles
FROM out__Combine_all_vehicle_traffic_1[TIME VALID EVERY 60,000
OFFSET 0,000]
INTO Total_number_of_vehicles_today;

SELECT epass_use AS epass_use, round(sum(epass_use_sums),2) AS
total_epass_use
FROM out__Combine_all_fees_1[TIME VALID EVERY 60,000 OFFSET
0,000]
GROUP BY epass_use
INTO Totals_per_epass_use_today;
```