ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Κατάτμηση διαστημάτων με εφαρμογές σε ζεύξη και ταξινόμηση

Ζαχαρίας Ν. Χάσπαρης

Επιβλέπων: **Ιωάννης Ιωαννίδης,** Καθηγητής

ΑΘΗΝΑ

ΙΟΥΝΙΟΣ 2015

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Κατάτμηση διαστημάτων με εφαρμογές σε ζεύξη και ταξινόμηση

Ζαχαρίας Ν. Χάσπαρης

Α.Μ.: 1115200800284

ΕΠΙΒΛΕΠΩΝ: **Ιωάννης Ιωαννίδης,** Καθηγητής

ΠΕΡΙΛΗΨΗ

Στην εποχή μας ο όγκος των δεδομένων που υπάρχει διαθέσιμος είναι τεράστιος. Επιπρόσθετα, κάθε λεπτό που περνάει αυτός ο όγκος αυξάνεται συνεχώς. Αυτό έχει πολλές φορές σαν αποτέλεσμα οι δυνατότητες μίας μόνο υπολογιστικής μονάδας να μην μας είναι επαρκείς. Για το λόγο αυτό είναι πολύ χρήσιμο να διαμοιράζουμε τον όγκο των δεδομένων σε συστάδες υπολογιστών (clusters). Με τον τρόπο αυτό καταφέρνουμε να έχουμε γρηγορότερη επεξεργασία και αποδοτικότερη ανάκτηση δεδομένων. Για να γίνει αυτή η κατάτμηση (partitioning) των δεδομένων υπάρχουν αρκετές τεχνικές που εφαρμόζουμε. Οι δύο πιο συνηθισμένες είναι η Τυχαία Κατάτμηση (Random Partitioning) και η Κατάτμηση Κατακερματισμού (Hash Partitioning). Χρησιμοποιούμε τις μεθόδους αυτές ανάλογα με τη διαχείριση και το είδος των ερωτημάτων (query) που θέλουμε να εφαρμόσουμε στα δεδομένα μας. Για παράδειγμα, Τυχαία Κατάτμηση θα χρησιμοποιήσουμε όταν θέλουμε να αποθηκεύσουμε δεδομένα σε διαφορετικούς κόμβους αποθήκευσης χωρίς να ενδιαφερόμαστε για τις τιμές που θα στείλουμε στον καθένα. Από την άλλη, Κατάτμηση Κατακερματισμού θα χρησιμοποιήσουμε όταν τα ερωτήματα που θα εκτελέσουμε είναι ερωτήματα ισότητας (equity queries). Στη συγκεκριμένη πτυχιακή ασχοληθήκαμε με μία διαφορετική τεχνική που εφαρμόζεται σπανιότερα στα διάφορα υπολογιστικά συστήματα αλλά είναι εξίσου αποτελεσματική. Η μέθοδος αυτή είναι η Κατάτμηση Διαστημάτων (Range Partitioning). Η συγκεκριμένη τεχνική διαχωρίζει ένα πίνακα δεδομένων (table) με βάση την κλίμακα των τιμών του σε ένα συγκεκριμένο πεδίο. Η Κατάτμηση Διαστημάτων αυξάνει την απόδοση σε λειτουργίες όπως η παράλληλη ταξινόμηση και οι ζεύξεις διαστημάτων (interval joins). Αναπόσπαστο κομμάτι της Κατάτμησης Διαστημάτων είναι τα Ιστογράμματα (Histograms) τα οποία παρέχουν εικόνα της ποικιλομορφίας των τιμών των δεδομένων. Υλοποιήσαμε τον αλγόριθμο της Κατάτμηση Διαστημάτων με Ιστογράμματα στο σύστημα EXAREME του Madgik Lab. Επίσης κάναμε πειραματική ανάλυση του αλγορίθμου με διαφορετικούς όγκους δεδομένων για να διαπιστώσουμε την αποτελεσματικότητά του.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Κατάτμηση Διαστημάτων με Ιστογράμματα

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: βάσεις δεδομένων, ιστογράμματα, κατανεμημένα συστήματα, κατάτμηση διαστημάτων, μεγάλα δεδομένα

ABSTRACT

Nowadays the volume of the data which are available is huge. Furthermore, every minute that volume is being continuously increased. Thus, many times the possibilities of a unique machine are not enough. For that reason, it is very useful to distribute the volume of data to clusters. In that way we manage to have faster management and more efficient recovery of the data. To achieve the data partitioning there are many techniques which we use. The two most common methods are the Random Partitioning and the Hash Partitioning. We use those methods according to the kind of queries which we would like to execute to our data. For instance, we would use Random Partitioning if we would like to store our data to different storage nodes without caring about the values which we will send to each one. Moreover, we would use Hash Partitioning if the queries, which we will execute, are equity queries. In this dissertation project, we dealt with a different technique which is applied rarely in the compute systems but it is equally efficient. That method is the Range Partitioning. That technique splits a table according to the scale of the values in a specific field. Range Partitioning increases the attribution in certain operations such as the parallel sorting and the interval joins. A very important part of Range Partitioning is the Histograms which give an image of the value distribution of the data. We developed the Range Partitioning algorithm with Histograms upon the EXAREME system of Madgik Lab. Also, we made experimental evaluation of the algorithm with different data volumes to testify its efficiency.

SUBJECT AREA: Range Partitioning with Histograms

KEYWORDS: databases, histograms, distributed systems, range partitioning, big data

ΕΥΧΑΡΙΣΤΙΕΣ

Για τη διεκπεραίωση της παρούσας Πτυχιακής Εργασίας θα ήθελα να ευχαριστήσω τους επιβλέποντες, καθηγητή Ιωάννη Ιωαννίδη, Herald Kllapi, Ελευθέριο Σταματογιαννάκη, για τη συνεργασία και την πολύτιμη συμβολή τους στην ολοκλήρωση της. Επίσης, θα ήθελα να ευχαριστήσω θερμά το Madgik Lab, και πιο συγκεκριμένα τους Αλέξανδρο Παπαδόπουλο και Χριστόφορο Σβίγγο, για τη βοήθεια που μου παρείχαν καθ' όλη τη διάρκεια της Πτυχιακής Εργασίας.

Ακόμα θα ήθελα να ευχαριστήσω την οικογένεια και τους φίλους μου για τη στήριξη που μου προσέφεραν.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΠΡΟΛΟΓΟΣ

Η παρούσα Πτυχιακή Εργασία αναπτύχτηκε στην Αθήνα από τον Ιούλιο 2014 μέχρι τον Ιούνιο 2015, στο πλαίσιο της απόκτησης του πτυχίου του Τμήματος Πληροφορικής και Τηλεπικοινωνιών της Σχολής Θετικών Επιστημών του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών. Χάρη στην υποστήριξη όσων ευχαριστήσαμε παραπάνω, μας δόθηκε η ευκαιρία να διεξαχθεί η Πτυχιακή Εργασία κάτω από άριστες συνθήκες συνεργασίας.

1. ΕΙΣΑΓΩΓΗ

Κάθε μέρα υπολογίζεται πως δημιουργούμε 2.5 πεντάκις εκατομμύρια bytes δεδομένων [1]. Είναι τόσο μεγάλος αυτός ο αριθμός που θεωρείται πως το 90% των δεδομένων που υπάρχουν σήμερα στον κόσμο τα δημιουργήσαμε μόνο μέσα στα δύο τελευταία χρόνια. Τα δεδομένα αυτά προέρχονται από διάφορους τομείς. Αν θέλαμε να αναφέρουμε κάποιους από αυτούς θα ήταν οι αισθητήρες που συλλέγουν κλιματολογικές πληροφορίες, οι ενέργειες από τα δίκτυα κοινωνικής δικτύωσης, οι ψηφιακές φωτογραφίες και βίντεο, οι καταγραφές των συναλλαγών και τα στίγματα από GPS των κινητών μας. Τα δεδομένα αυτά ονομάζονται Μεγάλα Δεδομένα (Big Data) [2] [3] [4].

Για όλα αυτά τα δεδομένα πέρα από το ότι πρέπει να υπάρχει διαθέσιμος αποθηκευτικός χώρος, πρέπει αυτή η αποθήκευση να γίνεται με τέτοιο τρόπο ώστε η ανάκτησή τους και η αξιοποίησή τους να είναι όσο το δυνατόν πιο αποτελεσματικές. Τα συστήματα που επωμίζονται τη λειτουργία αυτή λέγονται Κατανεμημένα Συστήματα (Distributed Systems) [5].

Ένα κατανεμημένο σύστημα είναι μια εφαρμογή που εκτελεί ένα σύνολο από πρωτόκολλα με σκοπό να συντονίσει τις ενέργειες πολλών διεργασιών πάνω σε ένα δίκτυο, έτσι ώστε όλα τα συστατικά του συστήματος να συνεργαστούν σωστά για να διεκπεραιώσουν ένα μικρό ή μεγάλο σύνολο από ζητούμενες ενέργειες. Υπάρχουν πολλά πλεονεκτήματα στη δημιουργία ενός κατανεμημένου συστήματος όπως η δυνατότητα σε απομακρυσμένους χρήστες να συνδέονται στο σύστημα χρησιμοποιώντας τις δυνατότητές του με ένα ανοιχτό και κλιμακωτό τρόπο. Με τον όρο ανοιχτό εννοούμε πως κάθε συστατικό του συστήματος είναι συνεχώς ανοιχτό να αλληλεπιδράσει με άλλα συστατικά. Με τον όρο κλιμακωτό εννοούμε πως το σύστημα μπορεί εύκολα να αλλάξει ώστε να προσαρμοστεί στους αριθμούς των χρηστών, στις πηγές και στις υπολογιστικές οντότητες.

Έτσι, ένα κατανεμημένο σύστημα μπορεί να είναι μεγαλύτερο και ισχυρότερο δεδομένου του συνδυασμού των δυνατοτήτων των κατανεμημένων συστατικών του, σε σχέση με συνδυασμούς αυτόνομων υπολογιστικών μονάδων. Αλλά πέρα από χρήσιμο, ένα κατανεμημένο σύστημα πρέπει να είναι και αξιόπιστο. Το συγκεκριμένο είναι μια δύσκολη διαδικασία λόγω της πολυπλοκότητας που παρουσιάζουν οι συναλλαγές μεταξύ των συνεχώς εκτελούμενων συστατικών των συστημάτων.

Για να θεωρήσουμε πως ένα κατανεμημένο σύστημα είναι πλήρως αξιόπιστο πρέπει να έχει τα εξής χαρακτηριστικά: ανθεκτικό σε λάθη, συνεχώς διαθέσιμο, ανακτήσιμο, συνεπές, κλιμακωτό, ασφαλές και να παρέχει προβλέψιμη απόδοση.

Ο τρόπος και ο σκοπός που τα συστήματα χρησιμοποιούν και επεξεργάζονται τα δεδομένα τους τα διαχωρίζει σε δύο κατηγορίες. Η μια κατηγορία είναι τα συστήματα OLTP (On-Line Transaction Processing - Επεξεργασία Διαδικτυακών Συναλλαγών) και η άλλη τα συστήματα OLAP (On-Line Analytical Processing - Ηλεκτρονική Αναλυτική Επεξεργασία) [6].

Τα συστήματα OLTP χαρακτηρίζονται από μεγάλο αριθμό μικρών σε έκταση συναλλαγών (Εισαγωγή, Ενημέρωση, Διαγραφή). Ο κύριος στόχος των OLTP συστημάτων είναι η ταχύτατη επεξεργασία ερωτημάτων, διατηρώντας την ακεραιότητα των δεδομένων σε περιβάλλοντα που εισέρχονται πολλοί χρήστες και προσφέροντας αποτελεσματικότητα που μετριέται από τον αριθμό των συναλλαγών ανά δευτερόλεπτο. Σε μια OLTP βάση δεδομένων υπάρχουν λεπτομερή και ακριβή δεδομένα, και το σχήμα το οποίο χρησιμοποιείται για την αποθήκευση τα δεδομένα των συναλλαγών είναι το μοντέλο οντότητα (entity model).

Τα συστήματα OLAP χαρακτηρίζονται από σχετικά μικρότερο όγκο συναλλαγών. Τα ερωτήματα που γίνονται είναι συχνά αρκετά πολύπλοκα και εμπεριέχουν ομαδοποιήσεις. Για τα OLAP συστήματα ο χρόνος απόκρισης είναι ένα μέτρο αποτελεσματικότητας. Οι OLAP εφαρμογές χρησιμοποιούνται ευρέως από τεχνικές εξόρυξης δεδομένων. Σε μία OLAP βάση δεδομένων υπάρχει ομαδοποιημένο ιστορικό δεδομένων το οποίο αποθηκεύεται σε πολυδιάστατα σχήματα.

Στη συγκεκριμένη πτυχιακή, ασχοληθήκαμε με τα συστήματα OLAP [7]. Αν θέλουμε να δούμε πιο αναλυτικά τα συγκεκριμένα συστήματα, τα δεδομένα τους προέρχονται από τις διάφορες OLTP βάσεις δεδομένων και είναι πολυδιάστατες οπτικές διαφόρων ειδών από επιχειρησιακές δραστηριότητες. Ο σκοπός τους είναι να βοηθούν στο σχεδιασμό και επίλυση των προβλημάτων και να υποστηρίζουν τις αποφάσεις που χρειάζεται να παρθούν. Οι εισαγωγές και οι ενημερώσεις γίνονται από περιοδικά, μεγάλης διάρκειας σύνολα εργασιών και οι ερωτήσεις είναι τις περισσότερες φορές πολύπλοκες με αρκετές ομαδοποιήσεις. Ο χρόνος που χρειάζεται ένα OLTP σύστημα για επεξεργασία των δεδομένων του εξαρτάται από το μέγεθός τους. Οι ανανεώσεις και τα πολύπλοκα ερωτήματα μπορούν να κρατήσουν ακόμα και αρκετές ώρες. Μία τεχνική που χρησιμοποιείται συχνά είναι η δημιουργία ευρετηρίων. Οι απαιτήσεις σε χώρο που χρειάζεται είναι μεγάλες λόγω της ύπαρξης ομαδοποιημένων δομών και ιστορικών δεδομένων και σίγουρα μεγαλύτερες από των OLTP συστημάτων λόγω της ύπαρξης πολλών ευρετηρίων. Ο σχεδιασμός των βάσεων δεδομένων ομαλοποιείται με τη χρήση λίγων πινάκων. Επίσης αντί για συμβατικά αντίγραφα ασφαλείας, κάποια συστήματα απλά επαναφορτώνουν τα OLTP δεδομένα σαν μια μέθοδο ανάκτησης.

Για τα συστήματα αυτά έχουμε αναπτύξει πολλές τεχνικές και λειτουργίες που βελτιώνουν την απόδοση και αποτελεσματικότητά τους. Μία από τις πιο σημαντικές είναι η κατάτμηση που εφαρμόζεται στις βάσεις δεδομένων των συστημάτων. Εταιρείες, όπως η Oracle [8], η Microsoft [9], η Google [10] και η IBM [11], έχουν ασχοληθεί και εξελίξει σε μεγάλο βαθμό τη συγκεκριμένη τεχνική. Η κατάτμηση προσφέρει μεγάλη ευελιξία βελτιώνοντας τη διαχείριση, απόδοση και διαθεσιμότητα των δεδομένων. Τέλος, λόγω του μεγάλου όγκου δεδομένων που υπάρχουν τα τελευταία χρόνια, είναι το κλειδί για να μπορέσουν να δημιουργηθούν συστήματα που θα έχουν τη δυνατότητα να διαχειριστούν τα δεδομένα αυτά με τον βέλτιστο τρόπο.

Μέσω της κατάτμησης επιτρέπεται σε ένα πίνακα της βάσης να χωριστεί σε μικρότερα κομμάτια κάθε ένα από τα οποία ονομάζεται τμήμα (partition). Κάθε τμήμα αποθηκεύεται σε διαφορετικό αποθηκευτικό κόμβο της συστάδας υπολογιστών.

Υπάρχουν πολλές τεχνικές με τις οποίες γίνεται αυτός ο διαχωρισμός. Το βασικό στοιχείο είναι πως συμβαίνει με βάση ένα χαρακτηριστικό του πίνακα, παραδείγματος χάρη με το πεδίο που αναφέρει την ημερομηνία μιας παραγγελίας. Οι βασικές τεχνικές είναι η Τυχαία Κατάτμηση και η Κατάτμηση Κατακερματισμού. Όμως κάθε τεχνική είναι χρήσιμη για συγκεκριμένες λειτουργίες που θα θελήσουμε να εκτελέσουμε στη συνέχεια. Στην περίπτωση που θέλουμε να εργαστούμε σε διαστήματα τιμών, η πιο σωστή μέθοδος που θα πρέπει να χρησιμοποιήσουμε είναι η Κατάτμηση Διαστημάτων. Η Κατάτμηση Διαστημάτων είναι ο τρόπος να διαχωρίζεται μια βάση δεδομένων με βάση την κλίμακα των τιμών της σε ένα συγκεκριμένο πεδίο που λογίζεται και ως κλειδί. Η διανομή των τιμών είναι συνεχές χωρίς κανένα ενδιάμεσο κενό, δηλαδή το πάνω ανοιχτό άκρο ενός πεδίου καλύπτεται από το επόμενο πεδίο.

Ένα πολύ σημαντικό εργαλείο για σωστότερη υλοποίηση της Κατάτμησης Διαστημάτων είναι τα Ιστογράμματα [12]. Στις βάσεις δεδομένων τα ιστογράμματα τα χρησιμοποιούμε σαν ένα μηχανισμό για πλήρη συμπίεση και προσέγγιση της διασποράς των δεδομένων. Η διασπορά των δεδομένων είναι πολύ χρήσιμη αλλά επειδή είναι πολύ μεγάλη για να αποθηκευτεί επακριβώς, τα ιστογράμματα έχουν ένα ρόλο

προσεγγιστικού μηχανισμού. Μέσω αυτών έχουμε τη δυνατότητα να εκτελούμε τη Κατάτμηση Διαστημάτων με όσο το δυνατό καλύτερα αποτελέσματα στη δημιουργία σωστά διαχειρίσιμων τμημάτων.

Στην παρούσα πτυχιακή ασχοληθήκαμε με τη συγκεκριμένη θεματική περιοχή, δηλαδή με την Κατάτμηση Διαστημάτων με Ιστογράμματα. Το κομμάτι της υλοποίησης το πραγματοποιήσαμε στο σύστημα EXAREME του Madgik Lab. Το EXAREME είναι ένα σύστημα για ελαστική, μεγάλης κλίμακας επεξεργασίας των ροών δεδομένων στο νέφος. Το EXAREME δημιουργήθηκε με σκοπό να είναι παρόμοιο με το υπολογιστικό μοντέλο του νέφους. Αυτό έγινε μέσω τριών αξόνων. Πρώτον με τον ορισμό γλωσσικών αφαιρέσεων που μπορούν δηλωτικά να εκφράσουν παραλληλισμό δεδομένων και πολύπλοκους υπολογισμούς. Δεύτερον με το σχεδιασμό μιας αρχιτεκτονικής με σαφή διαχωρισμό μεταξύ των συστατικών με καλά ορισμένη σημασιολογία. Και τρίτον αξιοποιώντας ταυτόχρονα τις ιδιότητες της ελαστικότητας [13] και οίκο-ελαστικότητας του νέφους με δυναμικούς εντοπισμούς υπολογιστικών πηγών και προσφέροντας συμβιβασμούς μεταξύ του χρόνου εκτέλεσης της ροής των δεδομένων και του χρηματικού κόστους χρήσης των πηγών αυτών. Το EXAREME είναι η πρώτη προσπάθεια υλοποίησης ενός συστήματος που εκμεταλλεύεται και τις δύο ελαστικότητες του νέφους.

Το υπόλοιπο της συγκεκριμένης πτυχιακής οργανώνεται ως εξής. Αρχικά στο Τμήμα 2 γίνεται αναφορά σε παρόμοιες έρευνες και εργασίες που έχουν γίνει πάνω στο συγκεκριμένο κομμάτι. Έπειτα στο Τμήμα 3 έχουμε μια επισκόπηση του συστήματος EXAREME πάνω στο οποίο έγινε η υλοποίηση του προγραμματιστικού μέρους της πτυχιακής. Το Τμήμα 4 αναφέρεται στα ιστογράμματα και ο τρόπος με τον οποίο υλοποιήθηκαν και συνέβαλαν στην υλοποίηση της Κατάτμησης Διαστημάτων. Έπειτα στο τμήμα 5 περιγράφουμε όλες τις υπόλοιπες αλλαγές που έγιναν στο σύστημα για να εφαρμοστεί η Κατάτμηση Διαστημάτων ενώ στο τμήμα 6 παραθέτουμε τα πειράματα που εκτελέστηκαν ώστε να διαπιστωθεί η αποτελεσματικότητα και τα πλεονεκτήματα που προσφέρει. Τέλος στο τμήμα 7 έχουμε τη σύνοψη της πτυχιακής, τα οφέλη που κερδήθηκαν και αναφορές σε πιθανές μελλοντικές επεκτάσεις που μπορούν να γίνουν.

2. Σχετικές εργασίες

Στο συγκεκριμένο τμήμα θα παραθέσουμε εργασίες σχετικές με τον τομέα της Κατάτμησης Διαστημάτων μέσω ιστογραμμάτων σε κατανεμημένα συστήματα, οι οποίες υπήρξαν η βάση για τη μελέτη και υλοποίηση της συγκεκριμένης πτυχιακής.

2.1 Κατανεμημένα συστήματα

Ένα κατανεμημένο σύστημα που χρησιμοποιείται ως βάση σε αρκετά συστήματα είναι το σύστημα Spanner που έχει υλοποιήσει η Google [14]. Το Spanner είναι η κλιμακούμενη, με πολλαπλές εκδοχές, παγκόσμια κατανεμημένη και συγχρονισμένη αντιγραφόμενη βάση δεδομένων της Google. Είναι το πρώτο σύστημα που κατανέμει δεδομένα σε παγκόσμια κλίμακα και υποστηρίζει εξωτερικά συνεπείς κατανεμημένες δοσοληψίες. Το Spanner είναι μία βάση δεδομένων που διαμοιράζει δεδομένα σε ομάδες Paxos [15] στατικών μηχανών σε κέντρα επεξεργασίας σε όλο τον κόσμο. Η αντιγραφή χρησιμοποιείται για παγκόσμια διαθεσιμότητα. Το Spanner αυτόματα διαμοιράζει δεδομένα στα μηχανήματα όσο το μέγεθος των δεδομένων ή ο αριθμός των εξυπηρετητών αλλάζει και αυτόματα αναδιανείμει δεδομένα στα μηχανήματα για να εξισορροπήσει το φορτίο αποφεύγοντας την κατάρρευση. Ένα από τα συστήματα που έχει ως βάση το Spanner και εκμεταλλεύεται τις ιδιότητές του είναι το σύστημα F1 [16] που έχει υλοποιήσει επίσης η Google. Το F1 είναι μία υβριδική βάση δεδομένων που συνδυάζει την υψηλή διαθεσιμότητα, την επεκτασιμότητα των NoSQL συστημάτων όπως το Bigtable [17] και τη συνοχή και χρησιμότητα των κλασσικών SQL βάσεων δεδομένων. Στο F1 η κατάτμηση των δεδομένων γίνεται με Κατάτμηση Κατακερματισμού.

Το κατανεμημένο σύστημα Hadoop (HDFS - Hadoop Distributed File System) [18] της Yahoo! είναι σχεδιασμένο για την αξιόπιστη αποθήκευση μεγάλων όγκων δεδομένων και τη ροή αυτών των δεδομένων με μεγάλο εύρος ζώνης στις εφαρμογές των χρηστών. Σε μία μεγάλη συστάδα υπολογιστών, χιλιάδες διακομιστές φιλοξενούν άμεσα συνδεδεμένες μονάδες ταυτόχρονα με την εκτέλεση εργασιών από τις εφαρμογές των χρηστών. Με την κατανομή του αποθηκευτικού χώρου και τους υπολογισμούς σε πολλούς διακομιστές, οι πόροι μπορούν να μεγαλώσουν ενώ παράλληλα παραμένουν οικονομικοί σε κάθε μέγεθος. Για την κατάτμηση των δεδομένων χρησιμοποιείται ένας υβριδικός αλγόριθμος βασισμένος στην Κατάτμηση Κατακερματισμού.

Το πρόγραμμα Stratosphere [19] είναι μία πλατφόρμα για μαζική παράλληλη ανάλυση δεδομένων. Επιτρέπει την ανάλυση μεγάλων όγκων δεδομένων χρησιμοποιώντας διάφορους τελεστές, όπως map, reduce και join. Αυτοί οι τελεστές μπορούν να συναθροιστούν σε ένα γράφο ροής δεδομένων που αναπαριστά μία εργασία ανάλυσης δεδομένων. Η υποστήριξη επαναληπτικών αλγορίθμων επιτρέπει στο χρήστη να εκτελεί προηγμένες γραφικές αναλύσεις και εργασίες τεχνητής νοημοσύνης σε κατανεμημένα συστήματα. Αν και παρουσιάστηκε σαν μια εναλλακτική μορφή του συστήματος Hadoop, προσφέρει πλήρη συμβατότητα μαζί του.

Τέλος, γνωρίζουμε πως η επικοινωνία μέσω δικτύου είναι το πιο αργό συστατικό των συνιστωσών ενός κατανεμημένου συστήματος βάσεων δεδομένων που είναι υλοποιημένο για μεγάλης κλίμακας αναλύσεις. Ο αλγόριθμος track join [20] είναι ένας αλγόριθμος που ελαχιστοποιεί την διαδικτυακή κίνηση με τη δημιουργία ενός βέλτιστου χρονοδιαγράμματος μεταφοράς για κάθε ξεχωριστό κλειδί ενοποίησης. Ο αλγόριθμος αυτός επεκτείνει τις επιλογές συναλλαγών μεταξύ κεντρικής μονάδας επεξεργασίας (CPU - Central Processing Unit) και του δικτύου.

2.2 Κατάτμηση δεδομένων

Το κομμάτι της Κατάτμησης Δεδομένων είναι μια ευρέως διαδεδομένη τεχνική διαμοιρασμού των δεδομένων στην οποία τα βασικά προβλήματα για την απόδοσή της είναι η σωστή κατάτμηση και τοποθέτηση των δεδομένων της βάσης στους κόμβους πάνω από το δίκτυο.

Ο αλγόριθμος GFF [21] που έχει παρουσιαστεί επικεντρώνεται στην πρώτη διάσπαση και πως αυτή μπορεί να πραγματοποιηθεί σε κομμάτια ικανά για αποθήκευση στους κόμβους. Τα τμήματα που θα αποθηκευτούν επιλέγονται με βάση το μέγιστο όφελος με τη χρήση ενός άπληστου αλγορίθμου. Η ανάθεση στους κόμβους επεξεργασίας χρησιμοποιεί έναν first-fit αλγόριθμο.

Στο [22] εξετάζονται τεχνικές οι οποίες μπορούν να χρησιμοποιηθούν για την κατάτμηση μια βάσης δεδομένων. Οι τεχνικές κατακερματισμού που εξετάζονται είναι ο οριζόντιος κατακερματισμός, οριζόντιος κατά ομάδες, απλός κάθετος, φυσικός κάθετος, κάθετος κατά ομάδες και μικτή κατάτμηση. Στη συγκεκριμένη έρευνα παρουσιάζονται πληροφορίες μέσω μετρήσεων για να επιλέγεται η κατάλληλη τεχνική ανάλογα με τον περιβάλλον στο οποίο θα εφαρμοστεί.

Στο [23] παρουσιάζεται μία προσέγγιση για τη χρήση αλγορίθμου για συστάδες υπολογιστών βασισμένων στην αναζήτηση με σκοπό υψηλής κλίμακας επιδόσεις στο κομμάτι της ανάκτησης των δεδομένων. Ο πυλώνας για αυτή την υλοποίηση είναι το πρόβλημα του πλανόδιου πωλητή (TSP - Travelling Salesman Problem).

Πιο συγκεκριμένα τώρα για το πεδίο της Κατάτμησης Διαστημάτων, στο [24] έχουμε μία πλήρη ανάλυση για τον τρόπο λειτουργίας της συγκεκριμένης μεθόδου και τις ιδιαιτερότητες που παρουσιάζει. Ακόμη στο [25] παρουσιάζονται αποτελεσματικές μεθόδους της Κατάτμησης Διαστημάτων αναζητώντας συχνά μοτίβα σε τεράστιες βάσεις δεδομένων. Βασίζονται στο κλειδί της κάθε βάσης για να ανακαλύψουν τοπικά συχνά μοτίβα (LFP - Local Frequent Pattern). Αφού βρεθεί το τμήμα με το μοτίβο από το υποτμήμα της βάσης δεδομένων, γίνεται η εύρεση του γενικότερου μοτίβου από την τοπική βάση δεδομένων και εκτελείται η κατάτμηση στο σύνολο της βάσης.

Τέλος στο [26] έχουμε μια προσπάθεια σύγκρισης των ειδών της κατάτμησης, προσπαθώντας να υλοποιηθούν και να αξιολογηθούν αλγόριθμοι πάνω σε διαφορετικά επίπεδα λοξότητας των δεδομένων.

2.3 Ιστογράμματα

Τέλος μεγάλο κομμάτι έρευνας και εργασιών υπάρχει στο κομμάτι των Ιστογραμμάτων. Στο [12] υπάρχει η ιστορία των Ιστογραμμάτων και η πιθανή μελλοντική τους πορεία, επισημαίνοντας κυρίως τις περιόδους εκείνες που χαρακτήρισαν την εξέλιξή τους. Στα [27] και [28] παρουσιάζεται η άμεση σχέση που έχουν με το τομέα των βάσεων δεδομένων. Επίσης στο [29] παρουσιάζονται διαφορετικές μορφές Ιστογραμμάτων ανάλογα με το είδος της επεξεργασίας που θέλουμε να πραγματοποιήσουμε στη βάση μας.

Τέλος στο [30] και στα [31] [32] [33] υπάρχουν πειράματα για τη μέθοδο του Scotts και των v-Optimal ιστογραμμάτων αντίστοιχα. Από τη μία η μέθοδος του Scotts είναι μία φόρμουλα που υπολογίζει το βέλτιστο πλάτος κάδου (bucket) του ιστογράμματος και ελαχιστοποιεί ασυμπτωτικά το ολοκληρωμένο μέσο τετραγωνικό σφάλμα. Από την άλλη τα v-Optimal ιστογράμματα, είναι ένας κανόνας κατάτμησης που ορίζει πως τα όρια του κάδου του ιστογράμματος θα παραχθούν με τρόπο ώστε να ελαχιστοποιείται η συσσωρευτική σταθμισμένη διακύμανση των κάδων. Όπως δείχνει και η έρευνα στο [33] από τους Viswanath Poosala και Ιωάννη Ιωαννίδη, η πιο ακριβή εκτίμηση των

δεδομένων γίνεται μέσω των v-Optimal ιστογραμμάτων χρησιμοποιώντας την τιμή του δεδομένου σαν μια παράμετρο ταξινόμησης και τη συχνότητά του σαν μια παράμετρο πηγής.

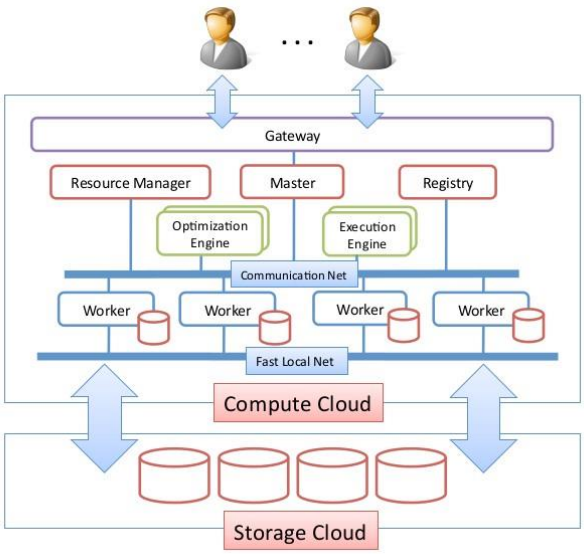
3. Επισκόπηση συστήματος EXAREME

Η υλοποίηση της πτυχιακής έγινε πάνω στο σύστημα EXAREME. Το EXAREME είναι ένα σύστημα επεξεργασίας μεγάλης κλίμακας δεδομένων. Είναι βασισμένο στην SQL γλώσσα αλλά προσφέρει τη δυνατότητα στο χρήστη να το εμπλουτίσει δηλώνοντας τις δικές του συναρτήσεις με απλό και εύκολο τρόπο. Το βασικό του πλεονέκτημα είναι ότι εκμεταλλεύεται την ελαστικότητα του νέφους (cloud) εξισορροπώντας το χρόνο εκτέλεσης με το κόστος των πόρων.

Η ελαστικότητα είναι μία από τις πιο σημαντικές ιδιότητες των κατανεμημένων συστημάτων και του νέφους. Με τον όρο ελαστικότητα χαρακτηρίζουμε το βαθμό με τον οποίο ένα σύστημα είναι ικανό να προσαρμόζεται στις αλλαγές που γίνονται στο φόρτο εργασιών παρέχοντας ή μη πόρους με έναν αυτόματο τρόπο, έτσι ώστε σε κάθε χρονικό σημείο οι πόροι να ταιριάζουν με τις απαιτήσεις μας όσο το δυνατόν περισσότερο. Είναι το βασικό χαρακτηριστικό που διαχωρίζει τέτοιου είδους συστήματα από συστήματα όπως τα άπληστα συστήματα (grid computing). Συνήθως οι τρεις κλίμακες που στοιχειοθετούν την ελαστικότητα ενός συστήματος είναι το κόστος, η ποιότητα και οι διαθέσιμοι πόροι και πόσο αυτοί οι τρεις συμβαδίζουν με τις εκάστοτε απαιτήσεις μας.

Το σύστημα Exareme στοχεύει στο σχεδιασμό ενός συστήματος που να αξιοποιεί τις ικανότητες του νέφους με τον καθορισμό γλωσσικών αφαιρέσεων που μπορούν να δηλώσουν εκφράσεις πολύπλοκων υπολογισμών, σχεδιάζοντας μια αρχιτεκτονική με σαφή διαχωρισμό σε συστατικά με σαφώς καθορισμένες σημασιολογίες και προσφέροντας ανταλλαγές μεταξύ του χρόνου εκτέλεσης των ροών δεδομένων και του χρηματικού κόστους χρήσης των πόρων.

Η αρχιτεκτονική του συστήματος φαίνεται στην Εικόνα 1. Το σύστημα είναι χτισμένο στην κορυφή του επιπέδου IaaS (Infrastructure as a Service) [34] των νεφών χρησιμοποιώντας τις υπολογιστικές και αποθηκευτικές υπηρεσίες που συνήθως προσφέρονται από τους παρόχους νέφους. Στο επίπεδο IaaS, τα νέφη προσφέρουν υπολογιστικούς πόρους με τη μορφή των εικονικών μηχανών (VMs - Virtual Machines). Το κόστος της μίσθωσης μιας εικονικής μηχανής βασίζεται σε ένα καθεστώς χρόνο-κβαντικής τιμολόγησης, και είναι προπληρωμένες, δηλαδή, κάποιος πληρώνει για το σύνολο του ποσοστού ενέργειας ανεξάρτητα από την έκταση της χρήσης των πόρων της εικονικής μηχανής [35].



Εικόνα 1: Αρχιτεκτονική συστήματος EXAREME

Στο συγκεκριμένο σύστημα προσφέρονται δύο γλώσσες σε διαφορετικό επίπεδο αφαίρεσης. Η γλώσσα υψηλού επιπέδου ονομάζεται ExaQL (Exareme Query Language). Η ExaQL βασίζεται στην SQL εμπλουτισμένη με UDFs (User Defined Functions) και συντακτικές επεκτάσεις που καθιστά εύκολη τη χρήση της. Η τριαντάχρονη πορεία της τεχνολογίας βάσεων δεδομένων έδειξε ότι οι δηλωτικές γλώσσες είναι πολύ σημαντικές διότι προσφέρουν ανεξαρτησία των δεδομένων και της πλατφόρμας. Επιλέχθηκε η SQL επειδή είναι μία γλώσσα που χρησιμοποιείται ευρέως και αρκετά κατανοητή, και ως εκ τούτου, οι χρήστες του συστήματος δεν χρειάζεται να μάθουν μια νέα γλώσσα. Η γλώσσα ροής δεδομένων ονομάζεται ExaDFL (Exareme Dataflow Language). Η ExaDFL προσφέρει ένα σύνολο θεμελιακών στοιχείων (primitives) για τη δήλωση του δυναμικού παραλληλισμού των δεδομένων, αφήνοντας το σύστημα να κάνει την πραγματική απόφαση σχετικά με την εκτέλεση. Η λειτουργικότητα του συστήματος μπορεί να επεκταθεί με νέα UDFs χρησιμοποιώντας ένα απλό και σαφές περιβάλλον (interface). Τα UDFs μπορούν να είναι όσο περίπλοκα απαιτείται με αυθαίρετο κώδικα από το χρήστη. Χρησιμοποιώντας αυτές τις αφαιρέσεις, μπορούμε δηλωτικά να εκφράσουμε πολύπλοκους υπολογισμούς δεδομένων.

Από την οπτική πλευρά του χρήστη, το σύστημα χρησιμοποιείται ως παραδοσιακό σύστημα βάσεων δεδομένων: δημιουργεί ή διαγράφει πίνακες, εισάγει εξωτερικά δεδομένα, χτίζει ευρετήρια, παρέχει ερωτήματα, και πολλά άλλα. Ερωτήματα πραγματοποιούνται στο σύστημα από πολλαπλούς χρήστες. Αυτά διαβάζουν τα δεδομένα από διάφορους πίνακες που είναι κατανεμημένοι και τους επεξεργάζεται, ενδεχομένως παράλληλα. Τα ερωτήματα εκφράζονται σε ExaQL ή ExaDFL και μετατρέπονται σε ροές δεδομένων σαν κατευθυνόμενοι άκυκλοι γράφοι (DAG - Directed Acyclic Graph) που έχουν αυθαίρετους υπολογισμούς για κόμβους και αλληλεπιδράσεις μεταξύ παραγωγών - καταναλωτών για ακμές μεταξύ των κόμβων. Τα τυπικά ερωτήματα που στοχεύει το σύστημα είναι πολύπλοκες μεταμορφώσεις εντατικών δεδομένων τα οποία είναι ακριβά για να εκτελεστούν: ερωτήματα μπορούν να διαρκέσουν αρκετά λεπτά ή ώρες.

Το Exareme χωρίζεται στα ακόλουθα στοιχεία: Ο Αρχηγός (Master) είναι το κύριο σημείο εισόδου στο σύστημα και, μεταξύ άλλων, είναι υπεύθυνο για το συντονισμό των υπολοίπων συστατικών. Όλες οι πληροφορίες που σχετίζονται με τα δεδομένα και την παρεχόμενη εικονική μηχανή αποθηκεύονται στο μητρώο. Ο Διαχειριστής Πόρων (Resource Manager) είναι υπεύθυνος για την δέσμευση και αποδέσμευση των εικονικών μηχανών με βάση τη ζήτηση. Η Μηχανή Βελτιστοποίησης (Optimization Engine) μεταφράζει τα ExaQL (ή ExaDFL) ερωτήματα στο διαμερισμένο κώδικα μηχανής του συστήματος λαμβάνοντας υπόψη το κόστος των πόρων. Η Μηχανή Εκτέλεσης (Execution Engine) επικοινωνεί με τον Διαχειριστή Πόρων ο οποίος καταγράφει τους πόρους που απαιτούνται για την εκτέλεση και ξεκινά να προγραμματίζει τους χειριστές (operators) του ερωτήματος σεβόμενος τις εξαρτήσεις τους από το γράφο της ροής δεδομένων και τους διαθέσιμους πόρους. Είναι επίσης υπεύθυνη για την εκ νέου εκτέλεση των χειριστών που απέτυχαν και την αποδέσμευση των πόρων. Τέλος, ο Εργάτης (Worker) εκτελεί χειριστές (σχεσιακές ή UDFs) και μεταφέρει τα ενδιάμεσα αποτελέσματα σε άλλους εργάτες.

Οι πίνακες διαμοιράζονται και αποθηκεύονται στην υπηρεσία αποθήκευσης. Οι πίνακες μπορούν να χωριστούν χρησιμοποιώντας είτε Τυχαία Κατάτμηση, είτε Κατάτμηση Κατακερματισμού. Στην παρούσα πτυχιακή εργασία προσθέσαμε και την Κατάτμηση Διαστημάτων. Κάθε εργάτης φέρνει τις κατατμήσεις που απαιτούνται για την εκτέλεση και τις αποθηκεύει στον τοπικό δίσκο του για την επόμενη χρήση. Οι λόγοι που το κάνουμε αυτό είναι δύο: ευελιξία και κόστος. Αυτό το σύστημα είναι πολύ ευέλικτο, διότι αποσυνδέει τους υπολογισμούς από τους αποθηκευτικούς πόρους βοηθώντας την ελαστικότητα του συστήματος από τη στιγμή που νέες εικονικές μηχανές μπορούν

εύκολα να προστεθούν και να αφαιρεθούν. Επιπλέον, η αποθήκευση των δεδομένων στο υπολογιστικό νέφος είναι πολύ ακριβή. Με τις σημερινές τιμές της Amazon [35], η αποθήκευση 1 TB δεδομένων για 1 μήνα σε S3 [36] θα κοστίσει περίπου $10 και για να το διατηρήσει αποθηκευμένο στο EC2 [35] χωρίς αντιγραφή θα κοστίσει περίπου $2.580 που είναι περισσότερο από δύο τάξεις μεγέθους πιο ακριβό. Η βελτιστοποίηση λαμβάνει υπόψη την τοποθέτηση τμημάτων για να επωφεληθούμε από την τοπικότητα των δεδομένων.

4. Ιστογράμματα και Συγχώνευση

Κάθε τμήμα περιέχει και ένα μέρος του πίνακα της βάσης που θέλουμε να επεξεργαστούμε. Γι αυτό πρέπει να έχουμε μια εικόνα της γενικής μορφής των δεδομένων μας. Έγιναν συναρτήσεις μέσω python, οι οποίες, μέσω του συστήματος EXAREME, έχοντας τα δεδομένα από ένα πίνακα της SQL, δημιουργούν ιστόγραμμα πάνω στο πεδίο που έχουμε επιλέξει, είτε έχουμε αριθμητικές είτε αλφαριθμητικές τιμές (strings). Έτσι για κάθε πίνακα δημιουργούμε ένα ιστόγραμμα που αποτυπώνει την κατανομή των δεδομένων που υπάρχουν στις τιμές. Έπειτα τα ιστογράμματα αυτά συγχωνεύονται (merge) για να αποτυπωθεί η εικόνα που έχει ένας μεγάλος σε έκταση πίνακας μίας βάσης δεδομένων.

4.1 Κατασκευή Ιστογραμμάτων

Αρχικά για την κατασκευή ιστογραμμάτων χρησιμοποιήσαμε την προσέγγιση και το μαθηματικό τύπο του Scotts [30]. Ο συνήθης κανόνας αναφοράς Scotts είναι ο βέλτιστος για τυχαία δείγματα από κανονική κατανομή των δεδομένων, υπό την έννοια ότι ελαχιστοποιεί το ολοκληρωμένο μέσο τετραγωνικό σφάλμα της εκτίμησης πυκνότητας. Η φόρμουλα για το βέλτιστο πλάτος ιστογράμματος υπολογίζεται και ελαχιστοποιεί ασυμπτωτικά το ολοκληρωμένο μέσο τετραγωνικό σφάλμα. Είναι μια διαδικασία βασισμένη σε δεδομένα για την επιλογή της παραμέτρου πλάτος κάδου, η οποία προϋποθέτει μια Gaussian πρότυπο αναφοράς και απαιτεί μόνο το μέγεθος του δείγματος και μια εκτίμηση της τυπικής απόκλισης. Η ευαισθησία της διαδικασίας διερευνάται χρησιμοποιώντας διάφορα μοντέλα πιθανοτήτων οι οποίες παραβιάζουν την Gaussian υπόθεση. Ο τύπος που χρησιμοποιήσαμε για τον υπολογισμό της τιμής του Scotts είναι:

$$h = \frac{3.5\hat{\sigma}}{n^{1/3}}$$

όπου *σ* η τυπική απόκλιση και *n* ο αριθμός των στοιχείων.

Στο κομμάτι της υλοποίησης, λόγω του ότι η λειτουργικότητα της συνάρτησης που θέλαμε να δημιουργήσουμε ήταν η επεξεργασία συνολικά των στοιχείων ενός ή και περισσοτέρων SQL πινάκων, η συνάρτηση θα έπρεπε να είχε τη μορφή μιας συναθροιστικής (aggregate) συνάρτησης του EXAREME και κατ' επέκταση της Python. Πιο συγκεκριμένα, δημιουργούμε μια κλάση (class) η οποία θα έχει κάποια από τα καθιερωμένα πεδία μιας συναθροιστικής συνάρτησης.

Το βασικό όρισμα που δίνουμε για να εκτελεστεί η συνάρτηση αυτή είναι το όνομα της στήλης του πίνακα που θέλουμε να επεξεργαστούμε και περιέχει τα στοιχεία πάνω στα οποία θα δημιουργηθεί το ιστόγραμμα. Επιλέξαμε για τη συνάρτηση αυτή να υπάρχουν δύο τρόποι επιλογής εκτέλεσής της και δημιουργίας του ιστογράμματος. Αν το μοναδικό όρισμα που θα δώσουμε είναι το όνομα της στήλης που αναφέραμε προηγουμένως, τότε η επεξεργασία θα γίνει μόνο με βάση τα στοιχεία που θα μας προκύψουν σύμφωνα με τη μαθηματική προσέγγιση του Scotts. Αν όμως πέρα από το όνομα της στήλης δώσουμε σαν δεύτερο όρισμα έναν ακέραιο αριθμό, τότε ο αριθμός αυτός θα αντιπροσωπεύει τον αριθμό των κάδων που θα έχει το ιστόγραμμα που θα δημιουργηθεί. Έτσι στη συγκεκριμένη δεύτερη λειτουργία, δημιουργούμε το αποτέλεσμα χωρίς να χρησιμοποιήσουμε τον τύπο του Scotts που μας παρέχει όλες τις πληροφορίες για το καταρτισμό του ιστογράμματος, αλλά αντίθετα δημιουργούμε ένα ιστόγραμμα που έχει κάδους ίσου πλάτους, δηλαδή ένα ιστόγραμμα ίσου πλάτους (equi width histogram).

Παραδείγματα των δύο μορφών της εντολής εκτέλεσης της συνάρτησης είναι τα εξής:

- select scottshistogram(C1) from my_table ;
- select scottshistogram(C1, 10) from my_table ;

Αναλύοντας τώρα πιο αναλυτικά τον κώδικα, ξεκινώντας έχουμε τη συνάρτηση αρχικοποίησης (init) στην οποία γίνονται οι απαραίτητες αρχικοποιήσεις που θα χρειαστούν στην πορεία.

Έπειτα έχουμε τη συνάρτηση βήματος (step). Το συγκεκριμένο κομμάτι είναι το “βήμα” μέσω του οποίου παίρνουμε τα δεδομένα από τους πίνακες και θα επαναληφθεί τόσες φορές όσες και τα στοιχεία που επεξεργαζόμαστε για να δημιουργήσουμε το ιστόγραμμα. Με μια πιο γενική ματιά, το συγκεκριμένο κομμάτι κώδικα θα εκτελεστεί όσες είναι οι γραμμές του πίνακα που θέλουμε να παράγουμε το ιστόγραμμά του. Στην αρχή ελέγχουμε εάν έχουμε να επεξεργαστούμε στοιχεία που είναι αριθμητικές ή αλφαριθμητικές τιμές.

Στην περίπτωση που έχουμε αλφαριθμητικές τιμές, πρέπει να τις μετατρέψουμε σε μία μορφή που θα μας παρέχει τη δυνατότητα να μπορούμε να τις συγκρίνουμε και να τις τοποθετήσουμε σε μία κλίμακα. Για να το επιτύχουμε αυτό, παίρνουμε τη δυαδική αναπαράσταση των πρώτων 8 χαρακτήρων της αλφαριθμητικής τιμής σε double word. Σε περίπτωση που έχουμε αλφαριθμητικές τιμές με μήκος μικρότερο των 8 χαρακτήρων, συμπληρώνουμε (padding) με κενά στο τέλος τους. Έτσι από εδώ και πέρα, όταν θα αναφέρετε κάτι ως αριθμητική τιμή θα είναι το ίδιο ανεξάρτητα από το αν αντιπροσωπεύει κάποια αλφαριθμητική τιμή ή όχι.

Επίσης, στο σημείο του ελέγχου του είδος των στοιχείων, ελέγχουμε τις δύο ιδιάζουσες περιπτώσεις της γλώσσας Python, τις λέξεις ‘infinity’ και ‘nan’. Οι συγκεκριμένες επειδή μπορεί να θεωρηθούν και ως αριθμητικές τιμές, έχουμε επιλέξει να τις θεωρούμε ως αλφαριθμητικές.

Στη συνέχεια της συνάρτησης βήματος, ελέγχουμε πόσα ορίσματα έχουμε δεχθεί. Αν έχουμε δεχθεί μόνο ένα, γίνεται υπολογισμός κάποιων τιμών, όπως η μέση τιμή μέχρι εκείνη τη στιγμή, που θα μας χρειαστούν για τον μετέπειτα υπολογισμό του αριθμού του Scotts. Σε περίπτωση δυο ορισμάτων, απλά αποθηκεύουμε τον αριθμό των κάδων που έχει επιλεχθεί. Στο σημείο αυτό ελέγχουμε και αν η τιμή που μας έχει δοθεί για τον αριθμό των κάδων έχει νόημα, είναι δηλαδή μεγαλύτερος του μηδενός. Σε όποια περίπτωση και να είμαστε, είτε ενός είτε δύο ορισμάτων, αποθηκεύουμε την μέχρι στιγμής μέγιστη και ελάχιστη τιμή που μας έχει έρθει. Έπειτα ελέγχουμε αν έχουμε υπερβεί έναν μέγιστο αριθμό δεδομένων που εμείς έχουμε επιλέξει. Στο συγκεκριμένο επιλέχθηκε ο μέγιστος αυτός αριθμός να είναι το 10000. Αν δεν τον έχουμε υπερβεί, τότε αποθηκεύουμε την τιμή που μας ήρθε σε μια λίστα. Αν όμως τον έχουμε υπερβεί, τότε επιλέγουμε τυχαία αν θα ανταλλάξουμε κάποια από τις ήδη αποθηκευμένες τιμές με τη νέα. Το κομμάτι αυτό γίνεται για να μην αποθηκεύουμε όλες τις τιμές σε περίπτωση μεγάλου όγκου δεδομένων και σπαταλάμε άσκοπα μνήμη.

Η τυχαία αυτή επιλογή γίνεται με τον Reservoir Sampling αλγόριθμο [37]. Οι αλγόριθμοι Reservoir Sampling είναι μια οικογένεια τυχαιοποιημένων αλγορίθμων για τυχαία επιλογή ενός δείγματος από μία λίστα στοιχείων, όπου η λίστα περιέχει είτε πολύ μεγάλο είτε άγνωστο αριθμό στοιχείων. Συνήθως το μέγεθος είναι αρκετά μεγάλο με αποτέλεσμα να μη χωράει στη μνήμη. Εμείς χρησιμοποιήσαμε την απλή μορφή του αλγορίθμου με πολυπλοκότητα O(n) που περιγράφεται στο Dictionary of Algorithms and Data Structures [38] και περιέχει τα βήματα που φαίνονται στην Εικόνα 2.

```
array R [ k ] ;   // αποτέλεσμα
integer i, j ;

// γεμίζουμε τον reservoir πίνακα
for each i in 1 to k do
  R [ i ] := S [ i ]
done ;

// αντικαθιστούμε τα στοιχεία με σταδιακά μειούμενη πιθανότητα
for each i in k+1 to length(S) do
  j := random(1, i) ;
  if j <= k then
    R [ j ] := S [ i ]
  fi
done
```

Εικόνα 2: Βήματα Reservoir Sampling αλγορίθμου

Και τέλος έχουμε τη συνάρτηση τέλους (final). Στη συνάρτηση αυτή γίνονται όλες οι απαραίτητες διεργασίες για να έχουμε το τελικό αποτέλεσμα ενός ιστογράμματος. Ξεκινώντας δημιουργούμε τα ονόματα των στηλών που θα έχει ο τελικός πίνακας που θα παράξουμε. Τα πεδία είναι:

- start : εδώ αποθηκεύουμε την αρχή κάθε πεδίου του ιστογράμματος
- end : αντίστοιχα εδώ αποθηκεύουμε το τέλος του κάθε πεδίου. Να σημειωθεί εδώ ότι το τέλος είναι ανοιχτό, δηλαδή τα πεδία έχουν τη μορφή [start, end).
- quantity : εδώ σημειώνουμε την ποσότητα που περιέχει κάθε πεδίο.

Start	End	Quantity
1.0	1.03336653031	2501
1.03336653031	2.00099590929	2137
2.00099590929	3.00199181857	1770
3.00199181857	4.00298772786	1430
4.00298772786	5.00398363715	1071
5.00398363715	6.00497954643	726
6.00497954643	7.00597545572	365
7.00597545572	Infinity	0

Εικόνα 3: Παράδειγμα μορφής δεδομένων συνάρτησης κατασκευής Ιστογραμμάτων

Έπειτα κάνουμε τους απαραίτητους έλεγχους στο αν έχουμε λάβει δεδομένα και σε περίπτωση που ο πίνακας που μας είχε δοθεί για επεξεργασία ήταν κενός επιστρέφεται ένα κενό ιστόγραμμα. Σε αντίθετη περίπτωση συνεχίζουμε και φτάνουμε στο σημείο που δουλεύουμε ανάλογα με ποια από τις δύο περιπτώσεις έχουμε. Αν θα δουλέψουμε με τον τύπο του Scotts, υπολογίζουμε την τυπική απόκλιση των τιμών που έχουμε. Βάση αυτής υπολογίζουμε έπειτα τη τιμή Scotts που είναι και το μέγεθος που θα έχει ο

κάθε κάδος του ιστογράμματος. Τέλος υπολογίζουμε πόσους κάδους θα έχουμε. Από την άλλη, αν δουλέψουμε με συγκεκριμένο αριθμό κάδων, υπολογίζουμε απλά το μέγεθός τους διαιρώντας τη διαφορά της μέγιστης με την ελάχιστη τιμή των τιμών μας με το δοσμένο αριθμό κάδων.

Στη συνέχεια υπολογίζουμε πόσες εγγραφές βρίσκονται σε κάθε κάδο με βάση τις τιμές που υπολογίσαμε προηγουμένως και αποθηκεύουμε τις ποσότητες αυτές. Τέλος, γίνεται η δημιουργία και επιστροφή των στηλών του τελικού πίνακα με όλες τις πληροφορίες που κατασκευάσαμε.

4.2 Συγχώνευση Ιστογραμμάτων

Αφού έχουμε ένα ιστόγραμμα για κάθε τμήμα, επόμενο ζητούμενο είναι να δούμε τη γενικότερη εικόνα που παρουσιάζει ο πίνακας της βάσης μας ώστε να εκτελέσουμε τη λειτουργία που επιθυμούμε. Για να γίνει αυτό χρειαζόμαστε μια συνάρτηση η οποία θα συγχωνεύει όλα τα ιστογράμματα και θα παρέχει ένα γενικό ιστόγραμμα για όλο τον πίνακα. Ακόμη θέλουμε να έχουμε τη δυνατότητα να αναδιοργανώσουμε όλη τη βάση δημιουργώντας τμήματα με ίσο αριθμό εγγραφών, πράγμα που είναι και ο σκοπός της εργασίας αυτής, δηλαδή η Κατάτμηση Διαστημάτων. Άρα σε αυτή την περίπτωση θέλουμε ένα γενικό ιστόγραμμα που να μας δίνει κάδους με ίσο αριθμό εγγραφών.

Για να υλοποιηθούν τα παραπάνω, δημιουργήθηκε η αντίστοιχη συνάρτηση. Όπως και στη συνάρτηση κατασκευής ιστογραμμάτων, έτσι και εδώ θέλουμε να επεξεργαζόμαστε συνολικά τα στοιχεία ενός ή και περισσοτέρων πινάκων SQL. Για το λόγο αυτό και αυτή η συνάρτηση έχει τη μορφή μιας συναθροιστικής συνάρτησης του EXAREME.

Εδώ τα ορίσματα που δίνουμε στη συνάρτηση είναι είτε τρία είτε τέσσερα. Τα τρία βασικά είναι τα ονόματα των στηλών του πίνακα που μας έχει παράξει η συνάρτηση κατασκευής ιστογραμμάτων, δηλαδή start, end και quantity. Σε περίπτωση που δοθούν μόνο αυτά τα τρία ορίσματα το αποτέλεσμα, που θα έχουμε θα είναι το γενικό ιστόγραμμα του πίνακα της βάσης μας. Αν όμως επιθυμούμε να κάνουμε αναδιάρθρωση της βάσης μας και υλοποίησης της Κατάτμησης Διαστημάτων, τότε χρειαζόμαστε τέσσερα ορίσματα και πρέπει να δώσουμε έναν ακόμα αριθμό που θα αντιπροσωπεύει τον αριθμό των τελικών τμημάτων.

Παραδείγματα των δύο μορφών της εντολής εκτέλεσης της συνάρτησης είναι τα εξής:

- select mergehistogram(start, end, quantity) from (select * from table_1 union all select all * from table_2) ;
- select mergehistogram(start, end, quantity, 5) from (select * from table_1 union all select all * from table_2 union all select all * from table_3) ;

Εξετάζοντας πιο αναλυτικά και αυτόν τον κώδικα, ξεκινάμε και πάλι με τη συνάρτηση αρχικοποίησης στην οποία γίνονται όλες οι απαραίτητης αρχικοποιήσεις για τις τιμές που θα χρειαστούμε στη συνέχεια.

Στη συνέχεια έχουμε τη συνάρτηση βήματος. Όπως και προηγουμένως έτσι και τώρα, στο συγκεκριμένο κομμάτι παίρνουμε τα δεδομένα από τους πίνακες και θα εκτελεστεί τόσες φορές όσες και τα στοιχεία που επεξεργαζόμαστε. Εδώ τα στοιχεία αυτά είναι οι τιμές που έχουν προκύψει από τον υπολογισμό των ιστογραμμάτων και οι φορές είναι τόσες όσοι οι κάδοι που έχουν δημιουργηθεί σε αυτά. Στην αρχή ελέγχουμε αν τα δεδομένα μας, δηλαδή τα όρια των κάδων και οι ποσότητες σε αυτούς, είναι αριθμητικές τιμές. Θυμίζουμε εδώ ότι ακόμα και για αλφαριθμητικές τιμές τα ιστογράμματα τα έχουμε δημιουργήσει μετατρέποντας τις τιμές αυτές σε αριθμητικές. Ακόμα ελέγχουμε αν

έχουμε τέσσερα ορίσματα για να αποθηκεύσουμε τον αριθμό των μελλοντικών κάδων. Αν οι τιμές μας περάσουν τον έλεγχο τις αποθηκεύουμε και συνεχίζουμε στις επόμενες.

Έπειτα και πάλι βρίσκεται η συνάρτηση τέλους. Εδώ αρχικά θα δημιουργήσουμε, ανεξάρτητα σε ποια περίπτωση βρισκόμαστε, το γενικό ιστόγραμμα της βάσης. Έχοντας τις επιμέρους τιμές από τα ιστογράμματα του κάθε τμήματος, συγχωνεύοντας τα επιμέρους που μας έχουν δοθεί δημιουργούμε ένα γενικό ιστόγραμμα που παρουσιάζει την κατανομή των δεδομένων της βάσης.

Start	End	Quantity
1.0	8.7306558075	15.7748680246
8.7306558075	14.9203882256	12.6305211917
14.9203882256	16.461311615	3.14434683287
16.461311615	24.1919674225	11.7748680246
24.1919674225	28.8407764511	4.6753959262
28.8407764511	42.7611646767	14.0
42.7611646767	56.6815529023	8.0

Εικόνα 4.α: Παράδειγμα μορφής δεδομένων συνάρτησης συγχώνευσης Ιστογραμμάτων

Start	End	Quantity
1.0	9.57607660617	17.5
9.57607660617	18.7265432077	17.5
18.7265432077	33.3151869522	17.5
33.3151869522	56.6815529023	17.5

Εικόνα 4.β: Παράδειγμα μορφής δεδομένων συνάρτησης συγχώνευσης Ιστογραμμάτων με ίσο αριθμό κάδων

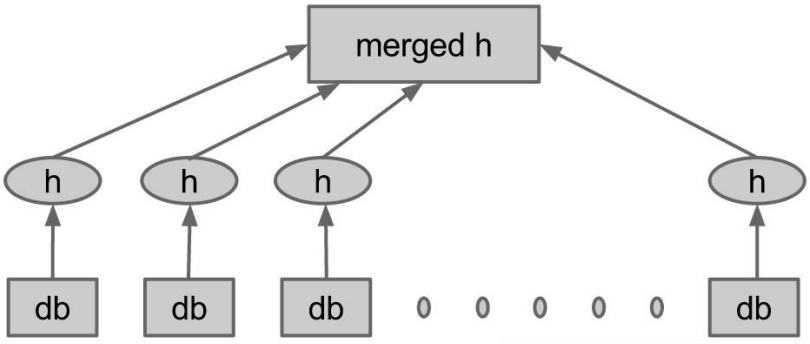
Αφού δημιουργήσουμε το γενικό ιστόγραμμα, φτάνουμε στο σημείο που συνεχίζουμε ανάλογα με τι έχουμε επιλέξει, δηλαδή πιο πρακτικά με το πόσα ορίσματα είχαμε δώσει στην κλήση της συνάρτησης. Αν απλά επιζητούμε ένα γενικό ιστόγραμμα τότε γίνεται η δημιουργία και επιστροφή του τελικού πίνακα με όλες τις πληροφορίες που υπολογίσαμε. Εδώ να σημειώσουμε πως τα πεδία του τελικού πίνακα είναι ίδια και έχουν την ίδια σημασία με αυτά της συνάρτησης κατασκευής ιστογραμμάτων, καθότι και εδώ αυτό που δημιουργούμε είναι ένα ιστόγραμμα. Πιο συγκεκριμένα έχουμε start, end και quantity. Επίσης και πάλι το πεδίο end είναι ανοικτό σε κάθε κάδο, δηλαδή είναι της μορφής [start, end).

Αν αντίθετα θέλουμε να υλοποιήσουμε Κατάτμηση Διαστημάτων, άρα χρειαζόμαστε ένα ιστόγραμμα που να μας δίνει κάδους με ίσο αριθμό εγγραφών, τότε δημιουργούμε ένα νέο ιστόγραμμα με γνώμονα το γενικό ιστόγραμμα που έχουμε ήδη κατασκευάσει. Έτσι υπολογίζουμε τα όρια που θα έχουν οι κάδοι μας ώστε το πλήθος εγγραφών σε κάθε έναν να είναι ίσο, δηλαδή να δημιουργήσουμε ένα ιστόγραμμα ίσου ύψους (equi height histogram). Αφού το κάνουμε αυτό, τότε δημιουργούμε και επιστρέφουμε τον τελικό πίνακα με τις συγκεκριμένες πληροφορίες. Να σημειώσουμε και εδώ πως τα πεδία του πίνακα που επιστρέφουμε, λόγω του ότι επιστρέφουμε και πάλι ένα ιστόγραμμα, είναι start, end και quantity και έχουν τη μορφή [start, end).

Είτε με την μία είτε με την άλλη περίπτωση, στο σημείο αυτό, έχοντας ως δεδομένα τα επιμέρους ιστογράμματα των τμημάτων του πίνακα που μας ενδιαφέρει, έχουμε ένα γενικό ιστόγραμμα που μας δίνει πληροφορίες που αφορούν τον πίνακα της βάσης μας στο σύνολό του.

5. Ολοκλήρωση συστήματος

Η δημιουργία ιστογραμμάτων και η συγχώνευσή τους είναι τα εργαλεία που δημιουργήσαμε για την υλοποίηση της Κατάτμησης Διαστημάτων. Μέχρι στιγμής η υλοποίηση μας παρουσιάζεται στην Εικόνα 5. Για να εντάξουμε όμως στο EXAREME σύστημα τα εργαλεία αυτά, έπρεπε να κάνουμε και αλλαγές στο γενικό σύστημα. Η βασική αλλαγή ήταν αυτή του μεταγλωττιστή του συστήματος. Για να μπορέσει να δέχεται ερωτήματα που εκτελούν την Κατάτμηση Διαστημάτων, πέρα από Τυχαία ή Κατακερματισμού Κατάτμηση που υπήρχε προηγουμένως, έπρεπε να αλλάξει η γλώσσα του συστήματος. Ακόμα έπρεπε να γίνουν οι κατάλληλες προσθήκες ή αλλαγές σε υπάρχουσες συναρτήσεις του συστήματος ώστε να προκύψει το επιθυμητό αποτέλεσμα.



Εικόνα 5: Λειτουργία επί μέρους ιστογραμμάτων και γενικού ιστογράμματος

5.1 Βελτιώσεις μεταγλωττιστή

Ο μεταγλωττιστής του συστήματος EXAREME μπορούσε να αναγνωρίσει ερωτήματα που ζητούσαν είτε Τυχαία είτε Κατακερματισμού Κατάτμηση (Εικόνα 6.α) . Για το λόγο αυτό έπρεπε να γίνουν οι κατάλληλες αλλαγές στη γλώσσα ώστε να υπάρχει και η επιλογή της Κατάτμησης Διαστημάτων (Εικόνα 6.β). Όπως παρατηρούμε στην Εικόνα 6.β, έπρεπε να γίνεται κατανοητό πότε υπάρχει η λέξη “range” και τι πρέπει να γίνει στο σύστημα όταν ανιχνευτεί η λέξη αυτή. Έτσι δημιουργήσαμε νέες λεξικές οντότητες (tokens) και υλοποιήσαμε και αλλαγές στο βασικό κομμάτι που μεταγλωττίζει κάθε ερώτημα.

```
distributed create table emp to 2 on eid as external
select
  cast(C1 as int) as eid,
  cast(C2 as text) as ename,
  cast(C3 as int) as age,
  cast(C4 as float) as salary,
from file('/home/harry/ADPlocalhost/demo/emp.tsv') ;
```

Εικόνα 6.α: Ερώτημα Κατάτμησης Κατακερματισμού

```
distributed create table emp to 3 range on eid as external
select
  cast(C1 as int) as eid,
  cast(C2 as text) as ename,
  cast(C3 as int) as age,
  cast(C4 as float) as salary,
from file('/home/harry/ADPlocalhost/demo/emp.tsv') ;
```

Εικόνα 6.β: Ερώτημα Κατάτμησης Διαστημάτων

5.2 Τροποποιήσεις υπαρχουσών συναρτήσεων

Όπως αναφέραμε, το σύστημα υλοποιούσε είτε Τυχαία είτε Κατακερματισμού Κατάτμηση. Ως εκ τούτου έπρεπε να κάνουμε είτε αλλαγές σε κάποια σημεία του κώδικα είτε προσθήκες νέων τμημάτων για να συμπεριληφθεί η νέα λειτουργία της Κατάτμησης Διαστημάτων με τη χρήση ιστογραμμάτων. Οι τροποποιήσεις αυτές έγιναν σε ήδη υπάρχουσες Python και Java συναρτήσεις του συστήματος.

5.2.1 Συνάρτηση διαίρεσης πίνακα βάσης δεδομένων

Η συγκεκριμένη συνάρτηση του συστήματος είναι γραμμένη σε Python και έχει ως λειτουργία τη διαίρεση του πίνακα μιας βάσης σε νέους πίνακες που αποθηκεύονται σε νέες βάσεις. Δηλαδή είναι η συνάρτηση στην οποία γίνεται η βασική εργασία της κατάτμησης, εκεί που έχοντας όλα τα στοιχεία που χρειάζονται γίνεται ο διαχωρισμός των δεδομένων στα εκάστοτε τμήματα. Ο πίνακας της βάσης που διαχωρίζεται μπορεί να είναι είτε ο κύριος, δηλαδή αυτός που περιέχει όλα τα στοιχεία, είτε κάποιο τμήμα αυτού. Στη δεύτερη περίπτωση, όπως είναι λογικό, αν τυχόν θέλουμε να εκτελέσουμε κατάτμηση μόνο σε εκείνο το τμήμα, τότε η δουλειά μας ολοκληρώνεται σε μία εκτέλεση της συγκεκριμένης συνάρτησης. Αν όμως ο απώτερος σκοπός μας είναι να πραγματοποιηθεί κατάτμηση σε όλη τον πίνακα που μπορεί να εκτείνεται σε διαφορετικά τμήματα, τότε πρέπει η συνάρτηση να εκτελεστεί σε κάθε τμήμα για να έχουμε το τελικό γενικό αποτέλεσμα.

Τα δεδομένα που λαμβάνει η συνάρτηση διαίρεσης ενός πίνακα δίνονται μέσω του ερωτήματος που ετοιμάζεται από τις συναρτήσεις Java του συστήματος και παρέχει όλες τις πληροφορίες που απαιτούνται. Η συγκεκριμένη συνάρτηση έχει διάφορες επιλογές για να πραγματοποιηθεί μια διαίρεση ενός πίνακα. Η αλλαγή που πραγματοποιήσαμε είναι η προσθήκη της επιλογής για Κατάτμηση Διαστημάτων. Η λέξη κλειδί που προστίθετο στην εντολή εκτέλεσης της συνάρτησης μέσω των πληροφοριών από τα ερωτήματα και αναγνωρίζει η συνάρτηση για να εκτελέσει Κατάτμηση Κατακερματισμού είναι το “split”. Στην περίπτωση της Κατάτμησης Διαστημάτων επιλέχθηκε η λέξη αυτή να είναι το “cuts”. Επίσης μετά τη λέξη “cuts” πρέπει να υπάρχει μία αγκύλη που να περιέχει τα σημεία στα οποία θα γίνεται η Κατάτμηση Διαστημάτων, δηλαδή τα σημεία τομής της διάταξης των τιμών που μας δίνουν τους κάδους όπως αυτά υπολογίστηκαν από τα ιστογράμματα.

Ένα παράδειγμα της εντολής εκτέλεσης της συνάρτησης είναι το εξής:

- output cuts:[2500, 5000, 8000] 'my_database.db' select C1,* from my_table ;

Πιο αναλυτικά όσον αναφορά τον κώδικα που προσθέσαμε, αφού γίνει η αναγνώριση της περίπτωσης της Κατάτμησης Διαστημάτων μέσω της λέξης “cuts” κάνουμε εισαγωγή δύο βιβλιοθηκών της Python οι οποίες θα χρειαστούν στη συγκεκριμένη διαδικασία. Οι βιβλιοθήκες αυτές, αν και δεν είναι κομψή ή σωστή πρακτική η εισαγωγή βιβλιοθηκών στη μέση ενός προγράμματος, επιλέχθηκαν να εισαχθούν εδώ για δύο λόγους. Πρώτον είναι βιβλιοθήκες των οποίων συναρτήσεις θα χρησιμοποιηθούν μόνο στο συγκεκριμένο μικρό κομμάτι σε σχέση με τη γενική έκταση της συνάρτησης. Και δεύτερον και βασικότερο, παρατηρήθηκε ότι μερικές φορές λόγω του μεγάλου όγκου τους, σε περιπτώσεις που χρειάζεται να τρέξει πολλαπλές φορές η συνάρτηση, δημιουργούσαν πρόβλημα καθώς έπρεπε σε κάθε εκτέλεση να φορτωθούν ανεξάρτητα αν θα χρειαζόταν ή όχι στη συνέχεια.

Μετά τη φόρτωση των βιβλιοθηκών, έχουμε το κομμάτι των αρχικοποιήσεων των μεταβλητών που θα χρειαστούν στην πορεία. Έπειτα ελέγχουμε αν μετά τη λέξη “cuts”

εμφανίζεται η λίστα των σημείων που αναφέρθηκε προηγουμένως. Επίσης ελέγχουμε αν μέσα στη λίστα υπάρχουν αριθμητικά δεδομένα, γιατί όπως είπαμε στις συναρτήσεις κατασκευής των ιστογραμμάτων, ακόμα και σε περίπτωση που η βάση περιέχει αλφαριθμητικά δεδομένα πάνω στα οποία θα γίνει η Κατάτμηση Διαστημάτων, το σύστημα τα διαχειρίζεται σαν αριθμητικές τιμές. Σε περίπτωση που εμφανιστεί κάτι που είναι διαφορετικό από τις προδιαγραφές που απαιτούνται, τότε το πρόγραμμα τερματίζεται εμφανίζοντας το αντίστοιχο μήνυμα για την εκάστοτε περίπτωση.

Αφού ολοκληρωθεί το κομμάτι με τους ελέγχους των τιμών, υπολογίζουμε πόσα νέα τμήματα του πίνακα, και κατ' επέκταση πόσες νέες βάσεις, θα δημιουργηθούν σύμφωνα με τη λίστα των σημείων. Αν για παράδειγμα η λίστα είναι η [5, 10, 15, 20], έχει δηλαδή 4 σημεία, αυτό σημαίνει πως θα δημιουργηθούν πέντε νέα τμήματα που θα έχουν τιμές (-∞, 5), [5, 10), [10, 15), [15, 20), [20, +∞). Με τον τρόπο αυτό δημιουργούμε τα νέα τμήματα και επιπλέον ένα σύνδεσμο ανοιχτό για το κάθε ένα ώστε να είναι εφικτή η διαχείριση τους στη συνέχεια.

Έπειτα από τη δημιουργία και των νέων τμημάτων, έχουμε το σημείο στο οποίο αποφασίζεται κάθε εγγραφή από τον πίνακα της βάσης που επεξεργάζεται σε ποια από τα νέα τμήματα θα τοποθετηθεί. Η επιλογή αυτή γίνεται με δυαδική αναζήτηση (binary search) πάνω στην τιμή της εγγραφής και τη λίστα των σημείων. Αφού βρεθεί σε ποιο τμήμα θα τοποθετηθεί γίνεται η αντιγραφή όλης της εγγραφής.

Τέλος, αφού έχει γίνει η αντιγραφή όλων των εγγραφών στα νέα τμήματα που αντιστοιχούν, κλείνουμε τους συνδέσμους που είχαμε δημιουργήσει. Μετά από αυτό το σημείο ο πίνακας ή το μέρος του πίνακα της βάσης που επεξεργαστήκαμε έχει υποστεί Κατάτμηση Διαστημάτων και οι εγγραφές του βρίσκονται στα νέα τμήματα.

5.2.2 Συνάρτηση δημιουργίας EXAREME ερωτημάτων

Όπως φαίνεται στην Εικόνα 6.β, στο ερώτημα που δίνει ο χρήστης η μοναδική πληροφορία που αναφέρεται είναι το ότι επιθυμεί να εκτελέσει Κατάτμηση Διαστημάτων και πόσα τελικά τμήματα επιθυμεί να έχει η κατάτμηση που θα πραγματοποιηθεί. Όμως μόλις δοθεί η εντολή αυτή και αναγνωριστεί από το μεταγλωττιστή του συστήματος, τότε το σύστημα εκτελεί αυτόματα διαφορετικές συναρτήσεις για να υλοποιηθεί η Κατάτμηση Κατακερματισμού.

Αρχικά πρέπει να κληθεί η συνάρτηση κατασκευής ιστογραμμάτων με σκοπό να δημιουργηθεί ένα ιστόγραμμα για κάθε ένα από τα ήδη υπάρχοντα τμήματα. Η συνάρτηση αυτή, μέσω της μεθόδου του Scotts, θα δημιουργήσει ένα ιστόγραμμα για κάθε ένα τμήμα ώστε να υπάρχει η εικόνα των δεδομένων του.

Έπειτα θα κληθεί η συνάρτηση συγχώνευσης ιστογραμμάτων. Η συγκεκριμένη, αφού συγχωνεύσει τα ιστογράμματα που παράχθηκαν, θα μας δώσει το γενικό ίσου ύψους ιστόγραμμα για τη μορφή των δεδομένων όλης της βάσης μας. Το γενικό αυτό ιστόγραμμα θα μας δώσει και τα όρια που θα έχουν οι κάδοι μας ώστε να υλοποιηθεί η Κατάτμηση Διαστημάτων.

Τέλος, έχοντας τα όρια των κάδων και κατά συνέπεια τα όρια στα οποία θα χωριστούν τα νέα τμήματα, καλείται η συνάρτηση διαίρεσης πίνακα βάσης. Η συγκεκριμένη θα εκτελέσει το τελικό στάδιο της κατάτμησης διαιρώντας το τμήμα ή τα τμήματα που περιέχουν τα δεδομένα σε νέα τμήματα ώστε να ολοκληρώσουμε την Κατάτμηση Διαστημάτων.

Για να εκτελεστούν όλα αυτά πρέπει να δημιουργηθούν και τα κατάλληλα ερωτήματα που θα τα αναγνωρίσει το σύστημα και θα καλέσει τις αντίστοιχες συναρτήσεις. Για να

γίνει το κομμάτι αυτό έγιναν οι κατάλληλες αλλαγές σε υπάρχουσες συναρτήσεις της Java. Τα ερωτήματα που δημιουργούμε φαίνονται στην Εικόνα 7.

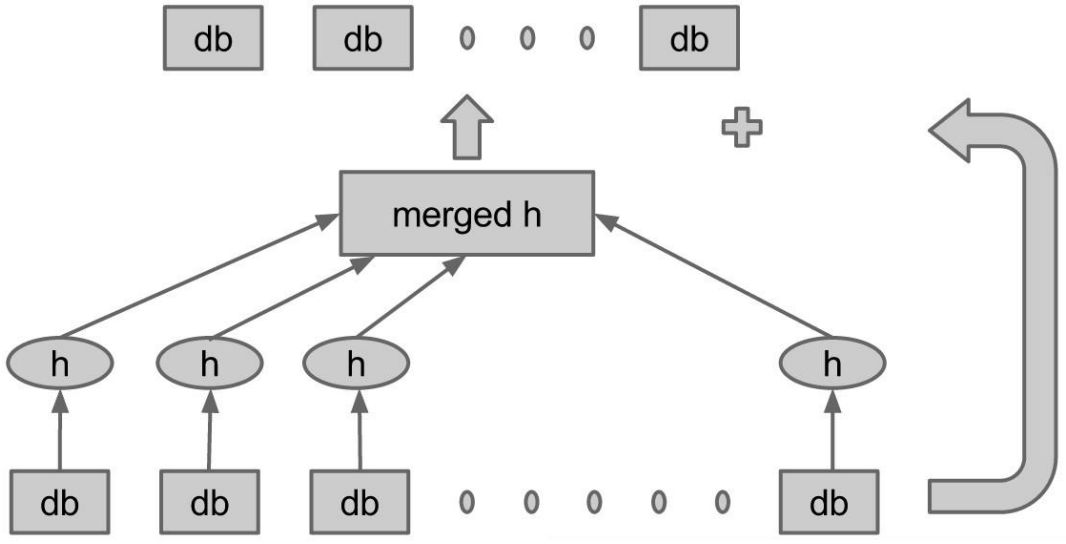
```
distributed create temporary table histograms to 1 as
select
 cast(h1 as float) as start,
 cast(h2 as float) as end,
 cast(h3 as float) as quantity
from (select scottshistogram(my_column) as h from my_table) ;

distributed create temporary  table merged_histogram as direct
select mergehistogram(start,end,quantity,5) as m
from histograms ;

distributed create table repartitioned_table to (list_with_ends_wihout_the_last_one) range on my_column as direct
select column_1, column_2, column_3, column_4, column_5, ... , column_n,
from my_table ;
```

Εικόνα 7: Σειρά ερωτημάτων για εκτέλεση Κατάτμησης Διαστημάτων

Έχοντας όλα τα παραπάνω καταλήγουμε στην τελική μορφή της εφαρμογής μας η οποία υλοποιεί την Κατάτμηση Διαστημάτων. Σχηματικά η μορφή αυτή παρουσιάζεται στην Εικόνα 8.



Εικόνα 8: Γενική εικόνα εφαρμογής

5.3 Δυναμική λειτουργία

Πέρα από τη στατική λειτουργία που περιγράψαμε, η υλοποίησή μας μπορεί να έχει και μια εν μέρει δυναμική λειτουργία. Οι απαιτήσεις που έχουμε για υπολογιστική ισχύ

αυξάνονται καθημερινά. Μαζί με αυτές αυξάνεται και ο όγκος των δεδομένων που επεξεργαζόμαστε. Τα δύο αυτά στοιχεία είναι αλληλένδετα και διαμορφώνουν τις συνθήκες μέσα στις οποίες εργαζόμαστε. Γι αυτό το λόγο στα κατανεμημένα συστήματα είναι πολύ συχνό φαινόμενο να προσθέτουμε σταδιακά καινούρια μηχανήματα στις ήδη υπάρχουσες υπολογιστικές μας μονάδες. Μέσω αυτού αποκτούμε περισσότερη υπολογιστική ισχύ και έχουμε στη διάθεσή μας μεγαλύτερο διαθέσιμο αποθηκευτικό χώρο.

Μια τέτοια προσθήκη δεν έχει προκαθορισμένο χρονοδιάγραμμα για το πότε θα συμβεί. Για παράδειγμα μπορεί να γίνει είτε επειδή έχουμε διαπιστώσει πως το παρόν μας σύστημα είναι ανεπαρκές για τον όγκο της εργασίας που το χρειαζόμαστε είτε γιατί απλά έχουμε την οικονομική δυνατότητα και επιθυμούμε να εξελίξουμε το σύστημά μας. Το σίγουρο είναι ότι δεν είναι επιθυμητό κάθε φορά που κάνουμε μια τέτοια προσθήκη με σκοπό να έχουμε βελτίωση στην απόδοσή μας, να χρειάζεται να πραγματοποιούμε χρονοβόρες εργασίες στα δεδομένα μας ώστε να εκμεταλλευτούμε τις δυνατότητες της νέας μονάδας.

Το πόσο εύκολη είναι η λειτουργικότητα αυτή μας δίνει και το βαθμό της ελαστικότητας του συστήματός μας. Η υλοποίηση μας παρέχει τις βάσεις για να εκμεταλλευόμαστε στο μέγιστο δυνατό την ιδιότητα της ελαστικότητας. Η μέθοδος που έχουμε επιλέξει για το κομμάτι αυτό χωρίζεται σε δύο κομμάτια. Αρχικά επιλέγουμε να υλοποιήσουμε Κατάτμηση Διαστημάτων στον πίνακα της βάσης που επιθυμούμε με τελικό αριθμό τμημάτων μεγαλύτερο από τον αριθμό των μηχανημάτων που διαθέτουμε. Τα επιπλέον τμήματα που δημιουργούνται τα αποθηκεύουμε ισόποσα στα υπάρχοντα μηχανήματα.

Με τον τρόπο αυτό έχουμε διπλό όφελος. Αρχικώς το άμεσο κέρδος μας είναι πως έχουμε εκτελέσει Κατάτμηση Διαστημάτων στα δεδομένα μας. Τα τμήματα αυτά έχουν κομμάτια του πίνακα της βάσης μας και μας είναι ευκολότερη λόγω του ότι είναι στοχευμένη η οποιαδήποτε αναζήτηση και επεξεργασία. Επιπλέον λόγω του ότι αποθηκεύουμε τα τμήματα ισόποσα στα ήδη υπάρχοντα μηχανήματα, δηλαδή δεν αποθηκεύουμε τρία τμήματα σε ένα μηχάνημα και ένα τμήμα σε ένα άλλο, ισοσκελίζουμε το φόρτο εργασίας.

Το δεύτερο όφελος είναι εκείνο που θέτει τις βάσεις για την ελαστικότητα του συστήματός μας. Έχοντας ήδη κατατμήσει τον πίνακά μας σε τμήματα, σε περίπτωση που στο κατανεμημένο σύστημα προστεθεί ένα ακόμα μηχάνημα η ενσωμάτωση σου θα είναι πιο εύκολη. Έχοντας ήδη το τμήμα του πίνακα που του αναλογεί, το μόνο που έχουμε να κάνουμε είναι μια απλή αντιγραφή στο σκληρό του δίσκο. Σε αντίθεση περίπτωση, αν δηλαδή δεν είχαμε έτοιμα επιπλέον τμήματα, θα έπρεπε να εκτελέσουμε ξανά κατάτμηση με αποτέλεσμα και να χρονοτριβούσαμε αρκετά και να χρειαζόταν να πραγματοποιήσουμε αλλαγές σε όλα τα μηχανήματα αφού η προηγούμενη κατάτμηση θα ήταν άχρηστη.

Το δεύτερο σκέλος της μεθόδου μας υλοποιείται μέσω των υπάρχων συναρτήσεων του EXAREME. Μέσω αυτών, μπορούμε εύκολα και γρήγορα να βρούμε και να μεταφέρουμε στο νέο μηχάνημα του κατανεμημένου μηχανήματός μας το τμήμα ή τα τμήματα του πίνακά μας που του αναλογούν. Και λέμε “τα τμήματα” διότι μπορεί η αρχική κατάτμηση να είχε ως αποτέλεσμα τα περισσότερα από τα μηχανήματά μας να είχαν από δύο τμήματα του πίνακα και κάποια λίγα τρία τμήματα. Οπότε εύλογο είναι στο νέο μηχάνημα να χρειαστεί να μεταφέρουμε δύο τμήματα.

Μέσω αυτής της διαδικασίας, επιτυγχάνουμε να μεγιστοποιούμε το χαρακτηριστικό της ελαστικότητας των κατανεμημένων συστημάτων και να παρέχουμε ένα σύστημα το οποίο μπορεί να είναι αξιόπιστο, λειτουργικό και αποτελεσματικό ανεξαρτήτως του υλικού πάνω στο οποίο θα εφαρμοστεί και κυρίως ανεξαρτήτως των οποιοδήποτε αλλαγών πραγματοποιηθούν στο υλικό στην πορεία.

6. Πειραματική αξιολόγηση

Πέρα από το κομμάτι της μελέτης και υλοποίησης της Κατάτμησης Διαστημάτων, πραγματοποιήσαμε και μεγάλο αριθμό πειραμάτων για να αξιολογήσουμε τις δυνατότητες και την αποτελεσματικότητα της υλοποίησής μας. Τα πειράματα αυτά έγιναν και σε κάθε επιμέρους στάδιο της υλοποίησης μας αλλά και στην τελική μορφή του προγράμματός μας.

6.1 Περιγραφή περιβάλλοντος πειραμάτων

Τα πειράματα που εκτελέσαμε μπορούμε να τα χωρίσουμε σε δύο κατηγορίες. Στη μία κατηγορία έχουμε τις μετρήσεις που πήραμε σε κάθε ένα στάδιο της υλοποίησής μας. Τα πειράματα αυτά έγιναν τοπικά σε μία υπολογιστική μονάδα. Η συγκεκριμένη υπολογιστική μονάδα τρέχει Linux μέσω λειτουργικού Ubuntu 14.04 των 32-bit. Ο επεξεργαστής έχει ταχύτητα 2.20 GHz στηριζόμενος σε 8 πυρήνες. Η μνήμη είναι 8Gb τύπου DDR3 και ο σκληρός δίσκος 115Gb με ταχύτητα 5400rpm.

Από την άλλη μεριά, εκτελέσαμε πειράματα με την τελική μορφή της υλοποίησής μας αρχικά τοπικά στην υπολογιστική μονάδα που αναφέραμε, κι έπειτα σε εικονικές μηχανές για να διαπιστώσουμε την αποτελεσματικότητα όταν λειτουργεί σαν κατανεμημένο σύστημα. Είχαμε στη διάθεσή μας τέσσερις μηχανές με δύο πυρήνες ανά μηχανή. Κάθε μηχανή είχε μνήμη 4Gb, σκληρό δίσκο 50Gb και λειτουργικό σύστημα Centos 5.1.

Τα δεδομένα που χρησιμοποιήσαμε είναι τεχνητά δεδομένα από τον μη κερδοσκοπικό οργανισμό TPC (Transaction Processing Performance Council) από την ομάδα TPC-H που περιέχει δεδομένα λήψης αποφάσεων και πιο συγκεκριμένα από το σύνολο δεδομένων lineitem. Το σχήμα του lineitem φαίνεται στην Εικόνα 8.

Name	Type
l_orderkey	int
l_partkey	int
l_suppkey	int
l_linenumber	int
l_quantity	int
l_extendedprice	float
l_discount	float
l_tax	float
l_returnflag	text
l_linestatus	text
l_shipdate	text
l_commitdate	text
l_receiptdate	text
l_shipinstruct	text
l_shipmode	text
l_comment	text

Εικόνα 9: Σχήμα lineitem

6.2 Συνάρτηση κατασκευής Ιστογραμμάτων

Οι πρώτες μετρήσεις έγιναν για να υπολογίσουμε το χρόνο που χρειάζεται ώστε να παράγουμε Ιστογράμματα. Οι μετρήσεις έγιναν στην τοπική μας υπολογιστική μονάδα με το σύνολο δεδομένων lineitem. Ο όγκος των δεδομένων που χρησιμοποιήσαμε ήταν 760MB, έχοντας ο πίνακάς μας 6.001.215 γραμμές. Για να έχουμε καλύτερη εικόνα της υλοποίησης μας, μετρήσαμε τους χρόνους που χρειάζεται η παραγωγή Ιστογραμμάτων έχοντας κάθε φορά διαφορετικό αριθμό τμημάτων στα οποία έπρεπε να υπολογίσει ξεχωριστά το Ιστόγραμμά τους. Επίσης μετρήσαμε τους χρόνους που απαιτούνται και για την κατασκευή Ιστογράμματος με βάση το πεδίο του πίνακα που έχει αριθμητικές τιμές αλλά και με βάση πεδίο που έχει αλφαριθμητικές τιμές. Τα ερωτήματα που χρησιμοποιήσαμε φαίνονται στην Εικόνα 10.

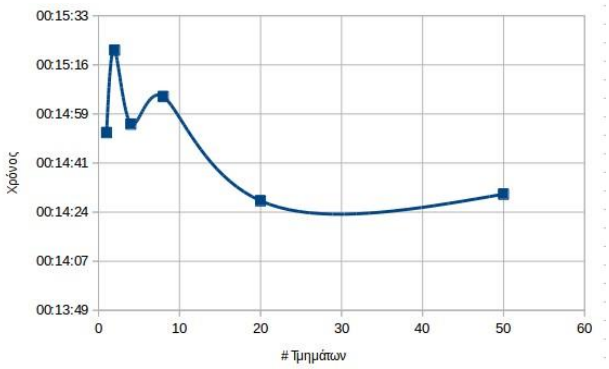
```
distributed create table histograms to 1 as direct
select
  cast(h1 as float) as start,
  cast(h2 as float) as end,
  cast(h3 as float) as quantity
from (select scottshistogram(l_comment) as h from lineitem) ;
```

Εικόνα 10.α: Ερώτημα κατασκευής Ιστογράμματος με βάση αριθμητικές τιμές

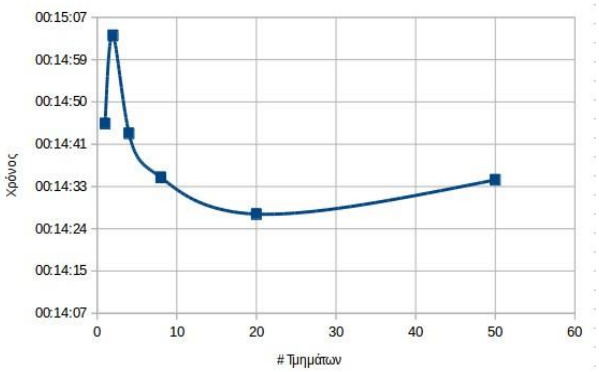
```
distributed create table histograms to 1 as direct
select
  cast(h1 as float) as start,
  cast(h2 as float) as end,
  cast(h3 as float) as quantity
from (select scottshistogram(l_orderkey) as h from lineitem) ;
```

Εικόνα 10.β: Ερώτημα κατασκευής Ιστογράμματος με βάση αλφαριθμητικές τιμές

Οι μετρήσεις μας έδειξαν πως ο χρόνος εκτέλεσης της συνάρτησής μας είναι ανεξάρτητος και του είδους των τιμών του πεδίου που χρησιμοποιήθηκε ως βάση, αλλά και του αριθμού των τμημάτων και κατά συνέπεια του αριθμού των παραγόμενων Ιστογραμμάτων. Τα αποτελέσματα των μετρήσεών μας φαίνονται στο Σχήμα 1.



Σχήμα 1.α: Μετρήσεις χρόνων κατασκευής Ιστογραμμάτων με βάση αριθμητικές τιμές



Σχήμα 1.β: Μετρήσεις χρόνων κατασκευής Ιστογραμμάτων με βάση αλφαριθμητικές τιμές

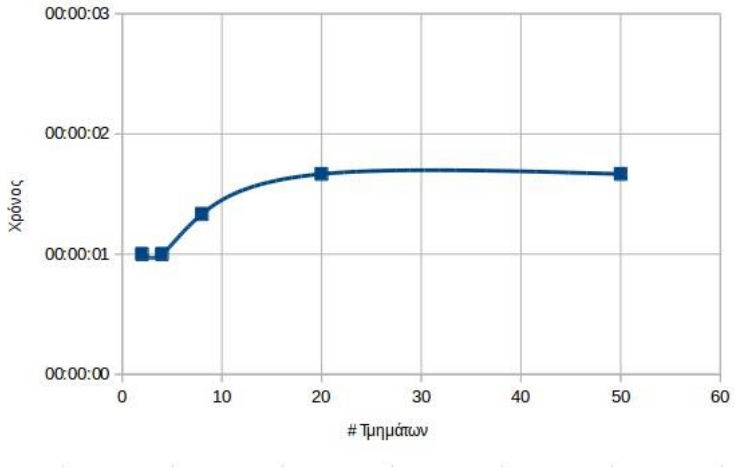
6.3 Συνάρτηση συγχώνευσης Ιστογραμμάτων

Με την ίδια λογική υπολογίσαμε τους χρόνους που χρειαζόμαστε ώστε να παράγουμε το γενικό ιστόγραμμα του πίνακα μιας βάσης. Και εδώ οι μετρήσεις έγιναν στην τοπική μας υπολογιστική μονάδα με το σύνολο δεδομένων lineitem. Ο όγκος των δεδομένων που χρησιμοποιήσαμε ήταν 760MB, έχοντας ο πίνακάς μας 6.001.215 γραμμές. Στις μετρήσεις αυτές προσπαθήσαμε να δούμε αν υπάρχουν μεγάλες διαφορές όσο αυξάνεται ο αριθμός των τμημάτων που πρέπει να συγχωνευτούν. Στο ερώτημα που χρησιμοποιήσαμε, και φαίνεται στην Εικόνα 11, επιλέξαμε να μας δίνει γενικό ιστόγραμμα για κατάτμηση με 5 τμήματα.

```
distributed create table merged as direct
select mergehistogram(start,end,quantity,5) as m
from myhistograms ;
```

Εικόνα 11: Ερώτημα συνάρτησης συγχώνευσης Ιστογραμμάτων

Ο χρόνος που χρειάζεται η συγκεκριμένη συνάρτηση, όπως φαίνεται στο Σχήμα 2 είναι αμελητέος και αυξάνεται ελάχιστα όσο αυξάνονται τα Ιστογράμματα που χρειάστηκε να συγχωνέψει.



Σχήμα 2: Μετρήσεις χρόνων συγχώνευσης Ιστογραμμάτων

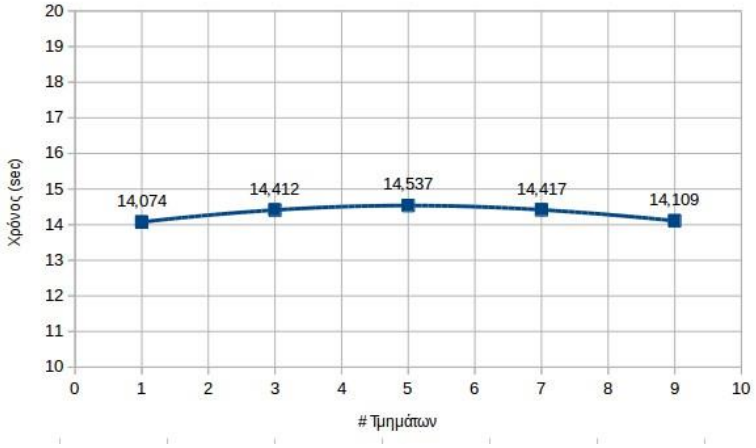
6.4 Συνάρτηση διαίρεσης πίνακα βάσης

Οι μετρήσεις αυτές έγιναν για να διερευνήσουμε πόσο γρήγορα γίνεται η διαίρεση του πίνακα μια βάσης σε επιμέρους βάσεις. Το κομμάτι αυτό είναι το τελευταίο που συμβαίνει κατά τη διαδικασία της Κατάτμησης Διαστημάτων. Για τις μετρήσεις αυτές δώσαμε έτοιμες τις τιμές των σημείων στα οποία θα έπρεπε να γίνει η διαίρεση της βάσης. Ελέγξαμε για τη συνάρτηση αυτή κατά πόσο επηρεάζεται από τις αλλαγές στο μέγεθος του πίνακα που επεξεργάζεται ή από τον αριθμό των τμημάτων που θα

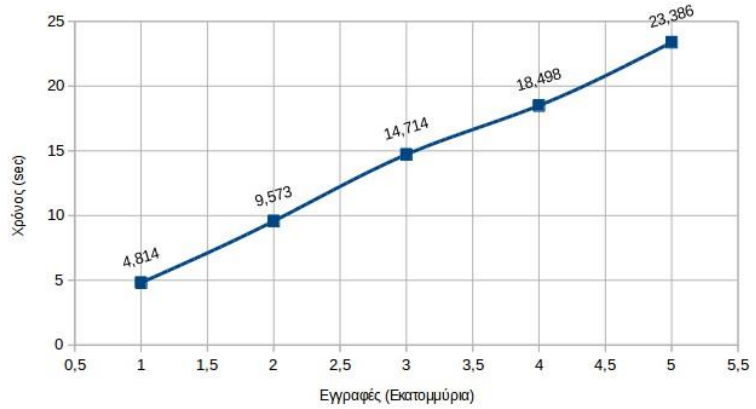
παράξει. Εδώ χρησιμοποιήσαμε πίνακα που δημιουργήσαμε εμείς και ο οποίος είχε μία στήλη με τυχαίους ακεραίους. Η εκτέλεση έγινε στην τοπική μας υπολογιστική μονάδα. Η εντολή της συνάρτησης που χρησιμοποιήσαμε για τις μετρήσεις ήταν η εξής:

- output cuts:[*list_with_points_of_cut*] 'my_database.db' select C1, * from my_table ;

Μέσω των πειραμάτων αυτών διαπιστώσαμε πως καταρχήν ο χρόνος που απαιτείται είναι ανεξάρτητος των τελικών τμημάτων (Σχήμα 3). Από την άλλη, ο χρόνος εκτέλεσης είναι ανάλογος του μεγέθους του πίνακα που επεξεργάζεται (Σχήμα 4).



Σχήμα 3: Μετρήσεις χρόνων με σταθερό αριθμό μεγέθους πίνακα



Σχήμα 4: Μετρήσεις χρόνων με σταθερό αριθμό τελικών τμημάτων

6.5 Κατάτμηση Διαστημάτων μέσω του συστήματος EXAREME

Στο κομμάτι αυτό μετρήσαμε το χρόνο που χρειάζεται για να εκτελεστεί το τελευταίο στάδιο της Κατάτμησης Διαστημάτων, δηλαδή της δημιουργίας των νέων τμημάτων, μέσω όμως του συστήματος EXAREME. Χρησιμοποιήσαμε και εδώ την τοπική

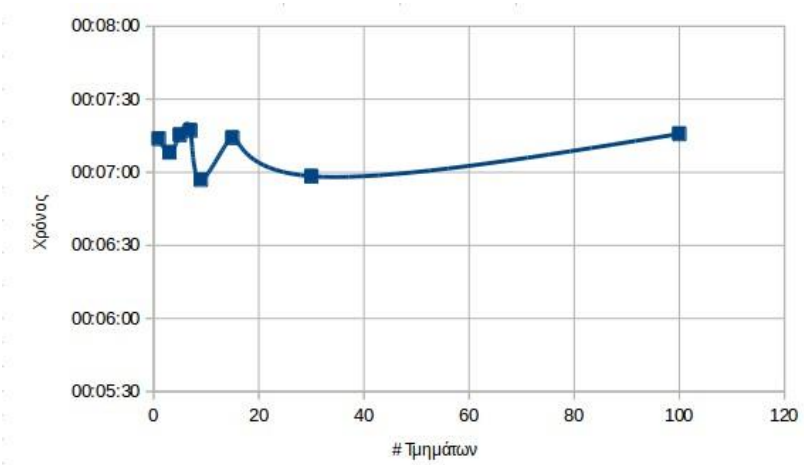
υπολογιστική μονάδα και το σύνολο δεδομένων lineitem. Στο συγκεκριμένο κομμάτι κάναμε αρκετές μετρήσεις με σκοπό να εξετάσουμε και να συγκρίνουμε την απόδοση του συστήματος κάτω από διαφορετικές συνθήκες. Αρχικά με όγκο δεδομένων 760MB και πίνακα με 6.001.215 γραμμές υπολογίσαμε το χρόνο που απαιτείται για διαφορετικό κάθε φορά αριθμό τελικών τμημάτων. Το ερώτημα που χρησιμοποιήσαμε φαίνεται στην Εικόνα 11.

```
distributed create table lineitem to (τιμές) range on l_orderkey as external
select
 cast(c1 as int) as l_orderkey,
 cast(c2 as int) as l_partkey,
 cast(c3 as int) as l_suppkey,
 cast(c4 as int) as l_linenumber,
 cast(c5 as int) as l_quantity,
 cast(c6 as float) as l_extendedprice,
 cast(c7 as float) as l_discount,
 cast(c8 as float) as l_tax,
 cast(c9 as text) as l_returnflag,
 cast(c10 as text) as l_linestatus,
 cast(c11 as text) as l_shipdate,
 cast(c12 as text) as l_commitdate,
 cast(c13 as text) as l_receiptdate,
 cast(c14 as text) as l_shipinstruct,
 cast(c15 as text) as l_shipmode,
 cast(c16 as text) as l_comment
from (file '/home/harry/ADPlocalhost/demo/biglineitem.tbl' delimiter:|) ;

*όπου τιμές τα σημεία στα οποία θα γίνει η Κατάτμηση Διαμερισμάτων.
```

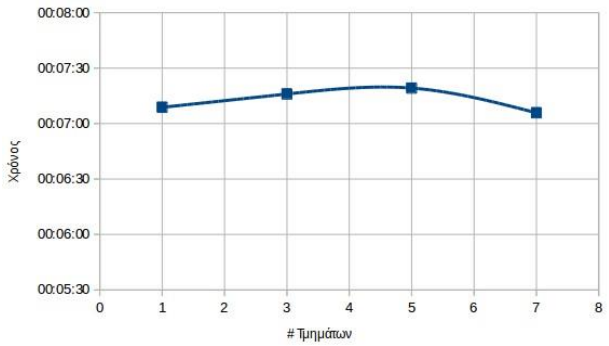
Εικόνα 12: Ερώτημα Κατάτμησης Διαστημάτων μέσω συστήματος EXAREME

Όπως φαίνεται στο Σχήμα 5, ο χρόνος είναι ανεξάρτητος από το πόσα τελικά τμήματα θα έχουμε.

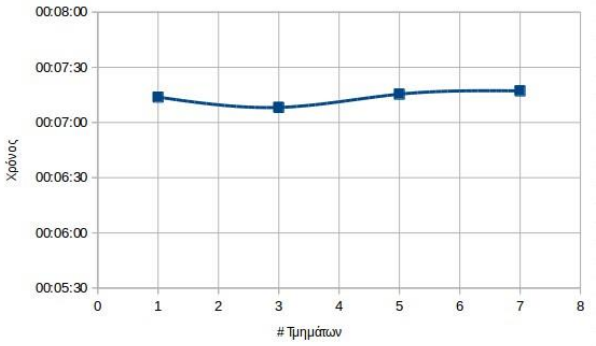


Σχήμα 5: Μετρήσεις χρόνων με σταθερό μέγεθος πίνακα και μεταβλητό αριθμό τελικών τμημάτων

Έπειτα συγκρίναμε τους χρόνους που χρειάζεται για την ίδια εργασία όταν η κατάτμηση συμβαίνει σε πεδίο το οποίο έχει τιμές που είναι ομοιόμορφα κατανεμημένες (l_orderkey) με την περίπτωση να συμβαίνει σε πεδίο που οι τιμές δεν έχουν ομοιόμορφη κατανομή στον άξονα των τιμών (l_linenumber). Τα ερωτήματα είναι τα ίδια με προηγουμένως με τη μόνη διαφορά να βρίσκεται στο όνομα της κολόνας στην οποία εκτελούσαμε την κατάτμηση. Στα δύο διαγράμματα του Σχήματος 6 διαπιστώνουμε πως ούτε αυτό το κριτήριο επηρεάζει τους χρόνους που χρειάζεται η εφαρμογή μας.

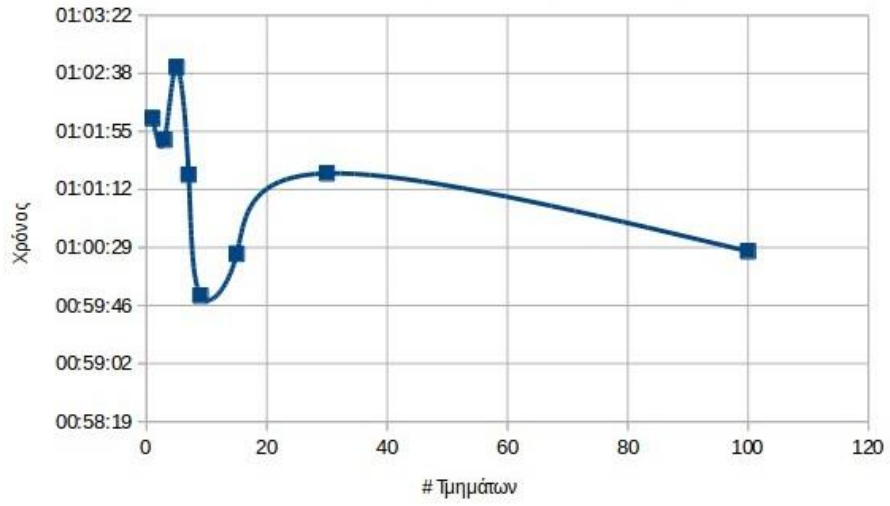


Σχήμα 6.α: Μετρήσεις χρόνων σε πεδίο τιμών πίνακα με ομοιόμορφη κατανομή

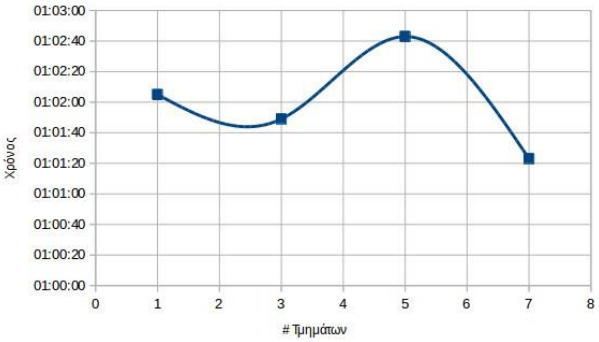


Σχήμα 6.β: Μετρήσεις χρόνων σε πεδίο τιμών πίνακα με ανομοιόμορφη κατανομή

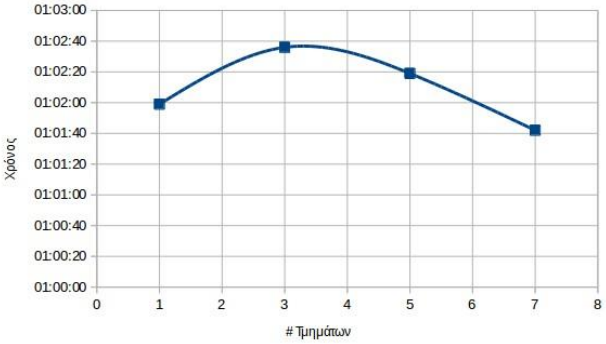
Τέλος, εκτελέσαμε τα ίδια πειράματα με τη διαφορά ότι τώρα ο όγκος των δεδομένων μας ήταν αρκετά μεγαλύτερος. Πάλι με το σύνολο δεδομένων lineitem αλλά αυτή τη φορά είχαμε δεδομένα 12.4Gb που μεταφράζεται σε 96.019.440 εγγραφές. Τα αποτελέσματα των μετρήσεων φαίνονται στα διαγράμματα των Σχημάτων 7 και 8. Και εδώ παρατηρούμε πως ακόμα και με τόσο μεγάλο όγκο δεδομένων, δεν έχουμε αλλαγές ούτε αν αυξάνουμε τον αριθμό των τελικών τμημάτων ούτε αν εκτελούμε κατάτμηση σε πεδίο που είναι ή όχι ομοιόμορφα κατανεμημένο.



Σχήμα 7: Μετρήσεις χρόνων με μεγάλο σταθερό όγκο δεδομένων και μεταβλητό αριθμό τελικών τμημάτων



Σχήμα 8.α: Μετρήσεις χρόνων με μεγάλο σταθερό όγκο δεδομένων σε πεδίο τιμών πίνακα με ομοιόμορφη κατανομή



Σχήμα 8.β: Μετρήσεις χρόνων με μεγάλο σταθερό όγκο δεδομένων σε πεδίο τιμών πίνακα με ανομοιόμορφη κατανομή

6.6 Κατάτμηση Διαστημάτων μέσω κατανεμημένου συστήματος

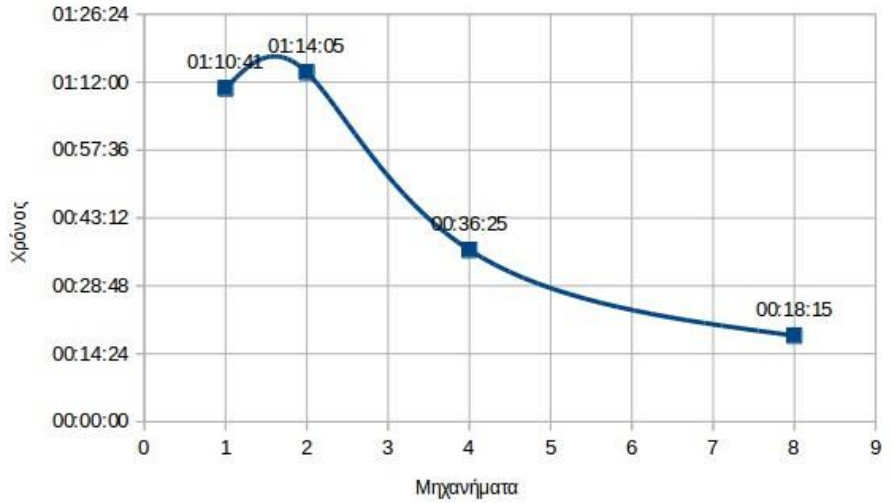
Οι τελευταίες μετρήσεις μας έγιναν εκτελώντας την εφαρμογή ως σύνολο, δηλαδή την παραγωγή των ιστογραμμάτων, της συγχώνευσής τους και τη δημιουργία των νέων τμημάτων, για να εκτελέσουμε την Κατάτμηση Διαστημάτων μέσω εικονικών μηχανών. Ο σκοπός των μετρήσεων αυτών ήταν να διαπιστωθεί κατά πόσο βελτιώνεται η απόδοση του συστήματος όσο αυξάνεται ο αριθμός των υπολογιστικών μηχανών που τρέχουν παράλληλα και επεξεργάζονται τα δεδομένα. Για το σκοπό αυτό εκτελέσαμε πειράματα αρχικά στην τοπική υπολογιστική μονάδα και έπειτα στις εικονικές μηχανές διπλασιάζοντας κάθε φορά των αριθμό των μηχανών που χρησιμοποιούσαμε. Πάλι με το σύνολο δεδομένων lineitem με όγκο δεδομένων 3.1Gb που μεταφράζεται σε 24.004.860 εγγραφές. Επίσης πέρα από το συνολικό χρόνο που καταγράψαμε, μετρήσαμε και επιμέρους πόσο χρειάζεται η παραγωγή και συγχώνευση ιστογραμμάτων και έπειτα η δημιουργία νέων τμημάτων. Το ερώτημα που χρησιμοποιούσαμε φαίνεται στην Εικόνα 12.

```
distributed create table lineitem to (#τμημάτων) range on l_orderkey as external
select
  cast(c1 as int) as l_orderkey,
  cast(c2 as int) as l_partkey,
  cast(c3 as int) as l_suppkey,
  cast(c4 as int) as l_linenumber,
  cast(c5 as int) as l_quantity,
  cast(c6 as float) as l_extendedprice,
  cast(c7 as float) as l_discount,
  cast(c8 as float) as l_tax,
  cast(c9 as text) as l_returnflag,
  cast(c10 as text) as l_linestatus,
  cast(c11 as text) as l_shipdate,
  cast(c12 as text) as l_commitdate,
  cast(c13 as text) as l_receiptdate,
  cast(c14 as text) as l_shipinstruct,
  cast(c15 as text) as l_shipmode,
  cast(c16 as text) as l_comment
from (file 'lineitem.tbl' delimiter:|) ;
```

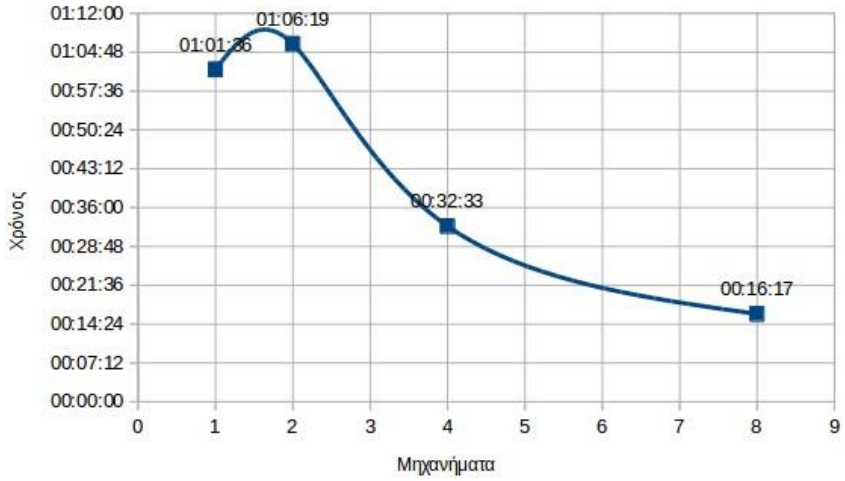
*όπου *#τμημάτων* ο αριθμός των τελικών τμημάτων τα οποία επιζητούμε μετά την υλοποίηση της Κατάτμησης Διαμερισμάτων.

Εικόνα 13: Ερώτημα Κατάτμησης Διαστημάτων μέσω κατανεμημένου συστήματος

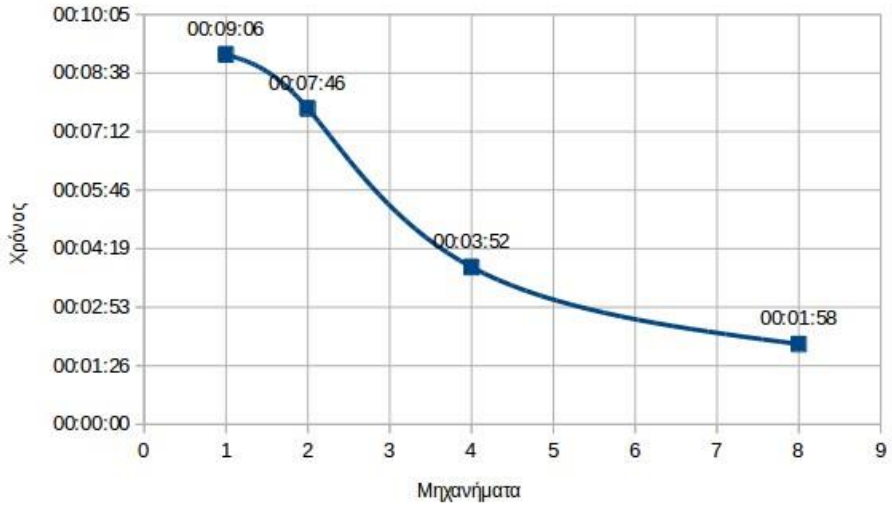
Όπως παρατηρούμε και στο Σχήμα 9, ο χρόνος που χρειάζεται η τοπική υπολογιστική μονάδα να εκτελέσει τη συγκεκριμένη εργασία, ακόμα και στη δική μας περίπτωση που είχαμε στη διάθεσή μας μια αρκετά ισχυρή μονάδα, είναι παρόμοιος με το χρόνο που χρειάζονται 2 παράλληλοι επεξεργαστές σε ένα κατανεμημένο σύστημα. Από εκεί και πέρα όσο αυξάνουμε τον αριθμό των επεξεργαστών στο κατανεμημένο σύστημα τόσο μειώνεται και ο χρόνος ο οποίος απαιτείται για να εκτελεστεί η ίδια εργασία. Η ίδιο εικόνα παρατηρείται και στα Σχήματα 9 και 10 στα οποία έχουμε καταγράψει το χρόνο που απαιτείται για το κομμάτι της παραγωγής των επιμέρους ιστογραμμάτων μαζί με την συγχώνευση αυτών και της διαίρεσης του πίνακα της βάσης μας αντίστοιχα. Προσεγγιστικά μπορούμε να πούμε πως όταν διπλασιάζουμε τον αριθμό των επεξεργαστών στο κατανεμημένο σύστημα, υποδιπλασιάζεται ο χρόνος εκτέλεσης που χρειάζεται ώστε να πραγματοποιηθεί η Κατάτμηση Διαστημάτων. Μέσω των μετρήσεων αυτών διαπιστώνουμε πως η υλοποίησή μας εκμεταλλεύεται τις δυνατότητες της παράλληλης επεξεργασίας παρουσιάζοντας σαφείς βελτιώσεις απόδοσης όσο αυξάνουμε την παραλληλοποίηση.



Σχήμα 9: Μετρήσεις χρόνων εκτέλεσής
Κατάτμησης Διαστημάτων μέσω κατανεμημένου συστήματος



Σχήμα 10: Μετρήσεις χρόνων παραγωγής και συγχώνευσης Ιστογραμμάτων
μέσω κατανεμημένου συστήματος



Σχήμα 11: Μετρήσεις χρόνων για διαίρεση του πίνακα της βάσης μας μέσω κατανεμημένου συστήματος

7. Συμπεράσματα και μελλοντική εργασία

Λόγω του τεράστιου όγκου δεδομένων που υπάρχουν και αυξάνονται συνεχώς σε καθημερινή βάση, είναι απαραίτητη η χρήση τεχνικών που αποθηκεύουν και επεξεργάζονται αποτελεσματικά τα δεδομένα αυτά. Για το λόγο αυτό χρησιμοποιούνται κυρίως συστάδες υπολογιστών που κατανέμουν την εργασία για παράλληλη επεξεργασία. Επίσης για να διαμοιράσουμε τα δεδομένα χρησιμοποιούμε τις διαφορετικών ειδών κατατμήσεις με βασικότερες την Τυχαία Κατάτμηση και την Κατάτμηση Κατακερματισμού. Στη συγκεκριμένη πτυχιακή εργασία ασχοληθήκαμε με την Κατάτμηση Διαστημάτων. Υλοποιήσαμε τις απαραίτητες συναρτήσεις πάνω στο σύστημα EXAREME του Madgik Lab για τη δημιουργία Ιστογραμμάτων και την κατάτμηση των δεδομένων. Μέσω πειραμάτων αξιολογήσαμε την υλοποίησή μας και εξετάσαμε κατά πόσο επηρεάζεται από τον όγκο και τη μορφή των δεδομένων. Ακόμη χρησιμοποιήσαμε εικονικές μηχανές με σκοπό να ελέγξουμε και την αποτελεσματικότητα του συστήματος όταν εκτελείται σαν κατανεμημένο σύστημα.

Διαπιστώσαμε πως η δημιουργία του ιστογράμματος του πίνακα μιας βάσης είναι το πιο χρονοβόρο τμήμα της Κατάτμησης Διαστημάτων και γι αυτό δε θα έπρεπε να γίνονται συχνός υπολογισμός τους αλλά όταν έχουν επέλθει αρκετές αλλαγές στα δεδομένα μας. Ακόμη είδαμε μέσω πειραμάτων και μετρήσεων πόσο μεγάλη βελτίωση επιτυγχάνεται όσο αυξάνεται ο αριθμός των μηχανών που έχουμε στη διάθεσή μας.

Τώρα όσον αφορά μελλοντική εργασία που είναι δυνατό να γίνει, τρεις επεκτάσεις είναι αυτές που μπορούν να πραγματοποιηθούν έχοντας ως βάση όσα υλοποιήθηκαν στην παρούσα πτυχιακή. Η πρώτη είναι η χρησιμοποίηση, πέραν της μεθόδου του Scotts για τη δημιουργία ιστογραμμάτων, των v-Optimal ιστογραμμάτων. Τα v-Optimal ιστογράμματα έχουν τη δυνατότητα να εκτιμούν καλύτερα τον περιεχόμενο των κάδων. Ένα ιστόγραμμα είναι μία εκτίμηση των δεδομένων της βάσης μας, και κάθε ένα θα έχει αρκετά λάθη. Ο κανόνας κατάτμησης που χρησιμοποιείται για τα v-Optimal ιστογράμματα προσπαθεί να έχει τη μικρότερη δυνατή διακύμανση ανάμεσα στους κάδους, το οποίο παρέχει μικρότερα λάθη. Με αυτή την προσθήκη θα υπάρχει η δυνατότητα επιλογής του κατάλληλου τύπου ιστογράμματος με σκοπό την καλύτερη και αποτελεσματικότερη Κατάτμηση Διαστημάτων.

Η δεύτερη επέκταση έχει να κάνει με τη δυνατότητα άμεσου κατακερματισμού στις ενημερώσεις. Αυτό σημαίνει να είναι εφικτό να γίνονται άμεσες αλλαγές στο σύστημα και στα δεδομένα χωρίς την παρέμβαση του χρήστη μετά από κάποια αλλαγή στα δεδομένα των βάσεων. Δηλαδή, έστω ότι γίνεται αφαίρεση ή προσθήκη κάποιας τιμής σε μια από τις βάσεις μας ή ακόμα και να αλλάξει κάποια από τις παρούσες. Τότε θα μπορούσε να γίνεται αυτόματη ενημέρωση του ιστογράμματος που αναπαριστά το συγκεκριμένο τμήμα. Ως εκ τούτου, η αλλαγή αυτή μπορεί να επέφερε επιπλέον αλλαγές σε άλλα επίπεδα του συστήματος, όπως στο γενικό ιστόγραμμα της βάσης ή σε προηγούμενη διαίρεση που έχει γίνει σε τμήματα. Όλες αυτές οι διεργασίας θα μπορούσαν να συμβούν αυτόματα στο παρασκήνιο χωρίς να χρειάζεται ο χρήστης να πραγματοποιήσει κάποια περαιτέρω ενέργεια.

Μία ακόμη επέκταση μπορεί να είναι η διαχείριση χωρικών δεδομένων (spatial data) [39]. Τα χωρικά δεδομένα είναι εκείνα που προσδιορίζουν τη γεωγραφική τοποθεσία αντικειμένων μέσα στο χώρο. Οι περισσότερες χωρικές βάσεις δεδομένων επιτρέπουν την παρουσίαση απλών γεωμετρικών αντικειμένων όπως σημεία, γραμμές και πολύγωνα. Άλλες χωρικές βάσεις δεδομένων μπορούν να διαχειριστούν πιο πολύπλοκες κατασκευές όπως 3D αντικείμενα, τοπολογικές επιφάνειες και γραμμικά δίκτυα. Ενώ οι τυπικές βάσεις δεδομένων είναι σχεδιασμένες ώστε να μπορούν να διαχειριστούν αριθμητικούς και αλφαριθμητικούς τύπους δεδομένων, χρειάζεται επιπλέον λειτουργικότητα να προστεθεί στις βάσεις ώστε να είναι σε θέση να μπορούν

να διαχειριστούν αποτελεσματικά χωρικά δεδομένα. Για να μπορέσουμε να διαχειριστούμε με τη μέθοδο της Κατάτμησης Διαστημάτων τα συγκεκριμένα δεδομένα θα πρέπει να έχουμε αντίστοιχα και δισδιάστατα ιστογράμματα. Μέσω αυτής της διαδικασίας θα μπορούμε να απαντήσουμε σε χωρικά ερωτήματα, όπως για παράδειγμα σε ερώτημα που ζητά ποιες πόλεις είναι πιο κοντά στο σημείο που βρισκόμαστε εκείνη τη στιγμή.

ΠΙΝΑΚΑΣ ΟΡΟΛΟΓΙΑΣ

Ξενόγλωσσος όρος	**Ελληνικός Όρος**
Cluster	Συστάδα Υπολογιστών
Partitioning	Κατάτμηση
Random Partitioning	Τυχαία Κατάτμηση
Hash Partitioning	Κατάτμηση Κατακερματισμού
Query	Ερώτημα
Equity Query	Ερώτημα Ισότητας
Range Partitioning	Κατάτμηση Διαστημάτων
Table	Πίνακας Δεδομένων
Interval Join	Ζεύξη Διαστημάτων
Histogram	Ιστόγραμμα
Big Data	Μεγάλα Δεδομένα
Distributed Systems	Κατανεμημένα Συστήματα
On-Line Transaction Processing	Επεξεργασία Διαδικτυακών Συναλλαγών
On-Line Analytical Processing	Ηλεκτρονική Αναλυτική Επεξεργασία
Entity Model	Μοντέλο Οντότητα
Partition	Τμήμα
Bucket	Κάδος
Cloud	Νέφος
Grid Computing	Άπληστα Συστήματα
Virtual Machine	Εικονική Μηχανή
Primitive	Θεμελιακό Στοιχείο
Interface	Περιβάλλον
Master	Αρχηγός
Resource Manager	Διαχειριστής Πόρων
Optimization Engine	Μηχανή Βελτιστοποίησης
Execution Engine	Μηχανή Εκτέλεσης
Operator	Χειριστής
Worker	Εργάτης
String	Αλφαριθμητική τιμή
Merge	Συγχώνευση
Aggregate	Συναθροιστική
Class	Κλάση
Equi Width Histogram	Ιστόγραμμα Ίσου Πλάτους
Init	Αρχικοποίηση
Step	Βήμα
Padding	Συμπληρώνω
End	Τέλος
Equi Height Histogram	Ιστογράμματα Ίσου Ύψους
Token	Λεξική Οντότητα
Binary Search	Δυαδική Αναζήτηση
Spatial Data	Χωρικά Δεδομένα

ΣΥΝΤΜΗΣΕΙΣ – ΑΡΚΤΙΚΟΛΕΞΑ – ΑΚΡΩΝΥΜΙΑ

GPS	Global Positioning System
OLTP	On-Line Transaction Processing
OLAP	On-Line Analytical Processing
NoSQL	Not only Structured Query Language
SQL	Structured Query Language
HDFS	Hadoop Distributed File System
CPU	Central Processing Unit
TSP	Travelling Salesman Problem
LFP	Local Frequent Pattern
IaaS	Infrastructure as a Service
VM	Virtual Machine
ExaQL	Exareme Query Language
UDF	User Defined Function
ExaDFL	Exareme Dataflow Language
DAG	Directed Acyclic Graph
TPC	Transaction Processing Council

ΑΝΑΦΟΡΕΣ

[1] “IBM - What is big data?” [Online]. Διαθέσιμο: http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html. [Προσπελάστηκε: 22-Jun-2015].

[2] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, and A. H. Byers, “Big data: The next frontier for innovation, competition, and productivity,” McKinsey Global Institute, May 2011.

[3] A. McAfee, E. Brynjolfsson, T. H. Davenport, D. J. Patil, and D. Barton, “Big data,” *Manag. Revolut. Harv. Bus Rev*, vol. 90, no. 10, pp. 61–67, 2012.

[4] “IBM Big Data – What is Big Data – United States.” [Online]. Διαθέσιμο: http://www.ibm.com/big-data/us/en/. [Προσπελάστηκε: 22-Jun-2015].

[5] “Introduction to Distributed System Design - Google Code University - Google Code.” [Online]. Διαθέσιμο: http://www.hpcs.cs.tsukuba.ac.jp/~tatebe/lecture/h23/dsys/dsd-tutorial.html. [Προσπελάστηκε: 22-Jun-2015].

[6] “OLTP vs. OLAP.” [Online]. Διαθέσιμο: http://datawarehouse4u.info/OLTP-vs-OLAP.html. [Προσπελάστηκε: 22-Jun-2015].

[7] S. Chaudhuri and U. Dayal, “An overview of data warehousing and OLAP technology,” *ACM Sigmod Rec.*, vol. 26, no. 1, pp. 65–74, 1997.

[8] “Partitioning Concepts.” [Online]. Διαθέσιμο: http://docs.oracle.com/cd/B28359_01/server.111/b32024/partition.htm. [Προσπελάστηκε: 22-Jun-2015].

[9] “Partitioned Tables and Indexes.” [Online]. Διαθέσιμο: https://msdn.microsoft.com/en-us/library/ms190787.aspx. [Προσπελάστηκε: 22-Jun-2015].

[10] “Partitioning Collections - Cloud Dataflow — Google Cloud Platform.” [Online]. Διαθέσιμο: https://cloud.google.com/dataflow/model/partition. [Προσπελάστηκε: 22-Jun-2015].

[11] “DB2 partitioning features.” [Online]. Διαθέσιμο: https://www.ibm.com/developerworks/data/library/techarticle/dm-0608mcinerney/. [Προσπελάστηκε: 22-Jun-2015].

[12] Y. Ioannidis, “The history of histograms (abridged),” in *Proceedings of the 29th international conference on Very large data bases-Volume 29*, 2003, pp. 19–30.

[13] W. F. Stokey, *Vibration of systems having distributed mass and elasticity*, vol. 7. chapter, 1988.

[14] J. C. Corbett, J. Dean, M. Epstein, A. Fikes, C. Frost, J. J. Furman, S. Ghemawat, A. Gubarev, C. Heiser, P. Hochschild, and others, “Spanner: Google’s globally distributed database,” *ACM Trans. Comput. Syst. TOCS*, vol. 31, no. 3, p. 8, 2013.

[15] L. Lamport, “The part-time parliament,” *ACM Trans. Comput. Syst. TOCS*, vol. 16, no. 2, pp. 133–169, 1998.

[16] J. Shute, R. Vingralek, B. Samwel, B. Handy, C. Whipkey, E. Rollins, M. Oancea, K. Littlefield, D. Menestrina, S. Ellner, and others, “F1: A distributed SQL database that scales,” *Proc. VLDB Endow.*, vol. 6, no. 11, pp. 1068–1079, 2013.

[17] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, “Bigtable: A distributed storage system for structured data,” *ACM Trans. Comput. Syst. TOCS*, vol. 26, no. 2, p. 4, 2008.

[18] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, “The hadoop distributed file system,” in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, 2010, pp. 1–10.

[19] D. Wu, D. Agrawal, and A. El Abbadi, “StratOSphere: mobile processing of distributed objects in Java,” in *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, 1998, pp. 121–132.

[20] O. Polychroniou, R. Sen, and K. A. Ross, “Track join: distributed joins with minimal network traffic,” 2014, pp. 1483–1494.

[21] D. Sacca and G. Wiederhold, “Database partitioning in a cluster of processors,” *ACM Trans. Database Syst.*, vol. 10, no. 1, pp. 29–56, Mar. 1985.

[22] “Selecting a Database Partitioning Technique: Library and Information Science Journal Article | IGI Global.” [Online]. Διαθέσιμο: http://www.igi-global.com/article/journal-database-management-jdm/51123. [Προσπελάστηκε: 22-Jun-2015].

[23] Chun-Hung Cheng, Wing-Kin Lee, and Kam-Fai Wong, “A genetic algorithm-based clustering approach for database partitioning,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 32, no. 3, pp. 215–230, Aug. 2002.

[24] J. Rousu, *Efficient range partitioning in classification learning*. Helsinki: Helsingin Yliopisto. Department of Computer Science, 2001.

[25] R. Gupta and C. S. Satsangi, “An efficient range partitioning method for finding frequent patterns from huge database,” *Int. J. Adv. Comput. Res.*, vol. 2, no. 2, pp. 62–69, 2012.

[26] D. J. DeWitt, J. F. Naughton, D. A. Schneider, and S. Seshadri, *Practical skew handling in parallel joins*. University of Wisconsin-Madison, Computer Sciences Department, 1992.

[27] V. Poosala, V. Ganti, and Y. E. Ioannidis, “Approximate query answering using histograms,” *IEEE Data Eng Bull*, vol. 22, no. 4, pp. 5–14, 1999.

[28] Y. E. Ioannidis and V. Poosala, “Histogram-based approximation of set-valued query-answers,” in *VLDB*, 1999, vol. 99, pp. 174–185.

[29] Y. E. Ioannidis and S. Christodoulakis, “Optimal histograms for limiting worst-case error propagation in the size of join results,” *ACM Trans. Database Syst. TODS*, vol. 18, no. 4, pp. 709–748, 1993.

[30] D. W. Scott, “On optimal and data-based histograms,” Biometrika, 1979, vol. 66, No 3, pp. 605–610.

[31] S. Guha, N. Koudas, and K. Shim, “Approximation and streaming algorithms for histogram construction problems,” *ACM Trans. Database Syst. TODS*, vol. 31, no. 1, pp. 396–438, 2006.

[32] Y. E. Ioannidis and V. Poosala, “Balancing Histogram Optimality and Practicality for Query Result Size Estimation,” in *Proceedings of the 1995 ACM SIGMOD international conference on Management of data - SIGMOD ’95*, 1995, pp. 233–244.

[33] V. Poosala, P. J. Haas, Y. E. Ioannidis, and E. J. Shekita, “Improved histograms for selectivity estimation of range predicates,” in *ACM SIGMOD Record*, 1996, vol. 25, pp. 294–305.

[34] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, “A break in the clouds: towards a cloud definition,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 50–55, 2008.

[35] “AWS | Amazon Elastic Compute Cloud (EC2) - Scalable Cloud Hosting.” [Online]. Διαθέσιμο: http://aws.amazon.com/ec2/. [Προσπελάστηκε: 22-Jun-2015].

[36] “AWS | Amazon Simple Storage Service (S3) - Online Cloud Storage for Data & Files.” [Online]. Διαθέσιμο: http://aws.amazon.com/s3/. [Προσπελάστηκε: 22-Jun-2015].

[37] J. S. Vitter, “Random sampling with a reservoir,” *ACM Trans. Math. Softw. TOMS*, vol. 11, no. 1, pp. 37–57, 1985.

[38] “reservoir sampling.” [Online]. Διαθέσιμο: http://xlinux.nist.gov/dads/HTML/reservoirSampling.html. [Προσπελάστηκε: 22-Jun-2015].

[39] H. Samet, “Spatial Data Structuresi,” 1995.