ΔΙΑΧΕΙΡΙΣΗ ΠΟΡΩΝ ΣΕ ETL ΔΙΑΔΙΚΑΣΙΕΣ

Η
ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

Υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης
του Τμήματος Πληροφορικής
Εξεταστική Επιτροπή

από τον

Παναγιώτη Δομουχτσίδη

ως μέρος των Υποχρεώσεων

για τη λήψη

του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΟ ΛΟΓΙΣΜΙΚΟ

Ιούνιος 2009

ΑΦΙΕΡΩΣΗ

Αφιερώνω την εργασία αυτή στους γονείς μου που με βοήθησαν να ολοκληρώσω τις σπουδές μου.

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστώ πολύ θερμά τον καθηγητή μου Παναγιώτη Βασιλειάδη για την βοήθειά του και την συνεργασία που είχαμε στην συγγραφή αυτής της εργασίας. Οι συμβουλές και οι γνώσεις που μου μετέδωσε είναι πολύτιμες για τη συνέχεια.

Στο τέλος θα ήθελα να ευχαριστήσω θερμά τους συμφοιτητές μου στο εργαστήριο για την ουσιαστική και σημαντική τους υποστήριξη κατά την διάρκεια της εκπόνησης της μεταπτυχιακής μου διατριβής. Πιο συγκεκριμένα ευχαριστώ τη Μπαϊκούση Ευτυχία, τη Παπαγεωργίου Χαρά, τη Πιλαλίδου Αλίκη και το Ρογκάκο Γιώργο.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ

ΠΕΡΙΛΗΨΗ

Παναγιώτης Δομουχτσίδης του Αποστόλου και της Ζωής. MSc, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Ιούνιος 2009, Διαχείριση Πόρων σε ETL Διαδικασίες. Επιβλέπων: Παναγιώτης Βασιλειάδης.

Οι Αποθήκες Δεδομένων είναι συλλογές δεδομένων που προέρχονται από διαφορετικές πηγές και χρησιμοποιούνται κυρίως για τη λήψη αποφάσεων σε ένα οργανισμό. Για να τροφοδοτηθεί μια αποθήκη με νέα δεδομένα, όπως αυτά παράγονται στις πηγές, χρησιμοποιούνται εργαλεία Εξαγωγής – Μετασχηματισμού – Φόρτωσης δεδομένων (Extract – Transform – Load tools, ETL), τα οποία οργανώνουν τα επί μέρους βήματα της όλης διαδικασίας σαν μία ροή εργασίας. Μία ροή εργασίας ETL μπορεί να θεωρηθεί ως ένας κατευθυνόμενος ακυκλικός γράφος που χρησιμοποιείται για να αναπαραστήσει τη ροή δεδομένων από τις πηγές δεδομένων προς την αποθήκη δεδομένων. Οι κόμβοι του γράφου είναι διαδικασίες καθαρισμού/ μετασχηματισμού δεδομένων ή σύνολα εγγραφών και οι ακμές σχέσεις εισόδου/εξόδου μεταξύ των κόμβων. Η ροή εργασίας είναι ένα αφηρημένο σχήμα σε λογικό επίπεδο, το οποίο πρέπει να υλοποιηθεί σε φυσικό επίπεδο, δηλαδή να αντιστοιχηθεί σε ένα συνδυασμό από εκτελέσιμα προγράμματα που εκτελούν την ETL ροή εργασίας.

Σε επίπεδο λογισμικού, μεταβίβαση των δεδομένων μεταξύ των κόμβων της ροής εργασίας υλοποιείται με ουρές που επικοινωνούν μεταξύ τους. Η ρύθμιση των παραμέτρων των ουρών επηρεάζει την εκτέλεση της ροής. Στην εργασία αυτή, μελετάται η όσο το δυνατό αποδοτικότερη ρύθμιση της απόδοσης πόρων σε ροές εργασίας ETL και συγκεκριμένα, μελετώνται εναλλακτικοί τρόποι διαχείρισης των πόρων που διατίθενται στη ροή εργασίας ώστε να επιταχυνθεί η λειτουργία της.

EXTENDED ABSTRACT IN ENGLISH

Domouchtsidis Panagiotis, Msc, Computer Science Department, University of Ioannina, Greece. June 2009. Resource Management for ETL Workflows. Thesis Supervisor: Panos Vassiliadis.

Data Warehouses (DW) are collections of data coming from different sources, used mostly to support decision-making and data analysis in an organization. To populate a data warehouse with up-to-date records that are extracted from the sources, special tools are employed, called Extraction – Transform – Load (ETL) tools, which organize the steps of the whole process as a workflow. An ETL workflow can be modeled as a directed acyclic graph that specifies the order of the operations applied to the source data, before being loaded to the data warehouse. The nodes of the graph denote either cleansing/transformation data activities or recordsets of data. The edges of the graph denote input/output relationships among the nodes. An ETL workflow is an abstract schema at a logical level that can be implemented at a physical level. In other words, the logical-level workflow is refined at the physical level as a combination of executable programs/scripts that perform the ETL workflow.

From a software level perspective, data are being transferred between the nodes of the ETL workflow through the usage of queues that communicate with each other. The configuration of the queues' parameters affects the performance of the workflows. The objective of this work is to regulate the efficiency of the resource allocation in ETL workflows and specifically, to study the alternative ways of managing the memory resources allocated to the workflow in order to accelerate its operation.

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

1.1. Εισαγωγή

Μία **Αποθήκη Δεδομένων (Data Warehouse-DW)** είναι μία υποδομή πληροφοριών που συλλέγει, ενοποιεί και αποθηκεύει δεδομένα. Το πιο σημαντικό χαρακτηριστικό μιας Αποθήκης Δεδομένων είναι ότι κάνει ακριβή και έγκαιρη διαχείριση πληροφοριών. Έτσι οι επιχειρήσεις αξιοποιούν τις αποθήκες δεδομένων καθιστώντας ικανούς τους εργαζομένους τους να παίρνουν καλύτερες και γρηγορότερες αποφάσεις.

Σύμφωνα με τον W.H.Inmon του [Inmo02], μία Αποθήκη Δεδομένων είναι μία συλλογή από θεματικώς προσανατολισμένα, ολοκληρωμένα, ενοποιημένα, επί μακρόν διατηρούμενα και χρονικά εξαρτώμενα δεδομένα στην υποστήριξη των αποφάσεων διαχείρισης. Επίσης παρουσιάζει ένα τυπικό ορισμό για την Αποθήκη Δεδομένων, σαν μία βάση δεδομένων η οποία περιέχει δεδομένα που είναι οργανωμένα στην βέλτιστη υποστήριξη παρουσίασης και ανάλυσης των δραστηριοτήτων.

Μία Αποθήκη Δεδομένων έχει τα εξής τέσσερα χαρακτηριστικά:

- Είναι προσανατολισμένη προς ένα θέμα, που σημαίνει ότι τα δεδομένα σε μία Αποθήκη Δεδομένων είναι οργανωμένα, έτσι ώστε όλα τα στοιχεία των

δεδομένων να σχετίζονται με τον πραγματικό κόσμο ή τα αντικείμενα να είναι συνδεδεμένα μεταξύ τους.

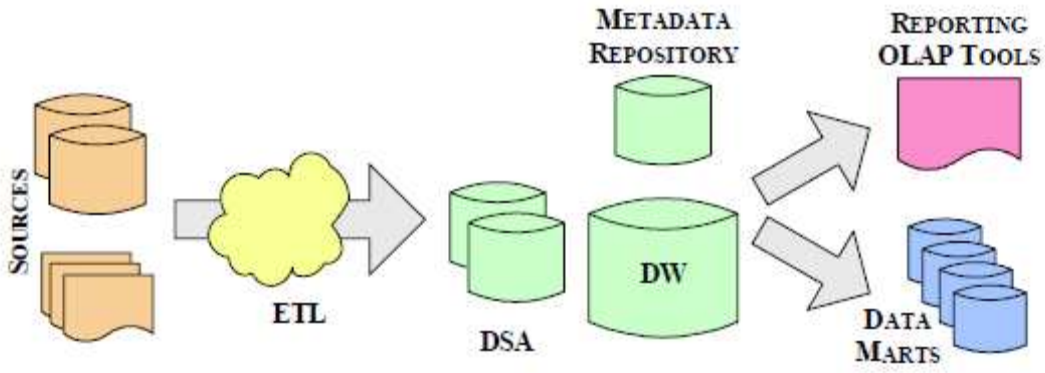
- Είναι ολοκληρωμένη-ενωποιημένη, που σημαίνει ότι η βάση δεδομένων περιέχει δεδομένα από όλες ή τις περισσότερες εφαρμογές μία επιχείρησης και τα δεδομένα συγκεντρώνονται σε μία σταθερή περιοχή.
- Είναι μη-πτητική, δηλαδή τα δεδομένα σε μία βάση δεδομένων ποτέ δεν επανεγράφονται ή διαγράφονται, αλλά παραμένουν για μελλοντική χρήση.
- Είναι χρονικά εξαρτώμενη, δηλαδή οι αλλαγές στα δεδομένα μίας βάσης δεδομένων καταγράφονται έτσι ώστε οι αναφορές που παράγονται να δείχνουν χρονικές αλλαγές.

Πολλά είναι τα πλεονεκτήματα της χρήσης μίας Αποθήκης Δεδομένων. Πρώτα από όλα, μια Αποθήκη Δεδομένων είναι σε θέση να συνδυάσει ποικιλία δεδομένων από διαφορετικές πηγές σε ένα μόνο μέρος. Ενδιαφέρουσες πληροφορίες εξάγονται από κατανεμημένες πηγές, οι οποίες είναι συνήθως ετερογενείς. Αυτό σημαίνει ότι τα ίδια δεδομένα, αναπαρίστανται διαφορετικά στις πηγές, για παράδειγμα μέσω διαφορετικών σχημάτων βάσεων δεδομένων. Η Αποθήκη Δεδομένων έχει να αναγνωρίσει διαφορετικές οντότητες, που αναπαρίστανται με διαφορετικούς τρόπους στις πηγές και μοντελοποιείται σε ένα μοναδικό σχήμα βάσης δεδομένων. Αυτό σημαίνει ότι τα δεδομένα σε μία Αποθήκη Δεδομένων πρέπει να περάσουν από μια σειρά από μετασχηματισμούς που πρέπει να είναι συνεπείς και ενημερωμένοι. Η διαδικασία αυτή αναφέρεται συχνά ως *σημασιολογική ενοποίηση* και αποτελεί μία σημαντική ιδιότητα της Αποθήκης Δεδομένων. Ένα ακόμη πλεονέκτημα της Αποθήκης Δεδομένων είναι ότι μπορεί να υποστηρίξει τις αλλαγές στα δεδομένα, δεδομένου ότι οι τροποποιήσεις σε αυτά μέσα σε μία Αποθήκη Δεδομένων παρακολουθούνται και καταγράφονται. Η Αποθήκη Δεδομένων διατηρεί επίσης ένα ιστορικό αρχείο των δεδομένων που φορτώνονται. Τέλος, η ποιότητα των δεδομένων είναι ένα σημαντικό ζήτημα, δεδομένου ότι όταν εισέρχονται στην Αποθήκη Δεδομένων είναι στις περισσότερες περιπτώσεις, ευμετάβλητα. Τα παραπάνω χαρακτηριστικά δείχνουν ότι μια Αποθήκη Δεδομένων αναμένεται να περιλαμβάνει πάντα ενημερωμένα, σταθερά και ολοκληρωμένα δεδομένα με σκοπό να υποστηρίζει την λήψη αποφάσεων και την ανάλυση των δεδομένων.

Η αρχιτεκτονική μίας Αποθήκης Δεδομένων αποτελείται από διάφορα επίπεδα, όπου σε κάθε επίπεδο τα δεδομένα του προέρχονται από δεδομένα χαμηλότερου επιπέδου (Σχήμα 1.1). Τα κύρια συστατικά μίας Αποθήκης Δεδομένων είναι: *Πηγές Δεδομένων (Data Sources), Περιοχή Ανασυγκρότησης Δεδομένων (Data Staging Area), Αγορά Δεδομένων (Data Marts), Αποθήκη Μετα-δεδομένων (Metadata Repository), ETL εργαλεία, OLAP εργαλεία.*

- **Πηγές Δεδομένων (Data Sources)**: Είναι βάσεις δεδομένων που αποθηκεύουν δομημένα και αδόμητα δεδομένα σαν κομμάτι του λειτουργικού περιβάλλοντος της επιχείρησης. Τα δεδομένα αντλούνται από διάφορες Πηγές που συνήθως είναι ετερογενής.
- **Περιοχή Ανασυγκρότησης Δεδομένων (Data Staging Area)**: Είναι μία βοηθητική βάση δεδομένων, που χρησιμοποιείται για την αποθήκευση ενδιάμεσων αποτελεσμάτων που παράχθηκαν με τις τεχνικές καθαρισμού και μετασχηματισμού από τις Πηγές των Δεδομένων.
- **Αγορά Δεδομένων (Data Marts)**: Ένα λογικό υποσύνολο της συνολικής Αποθήκης Δεδομένων. Συχνά βλέπουμε την Αγορά Δεδομένων σαν ένα περιορισμό της Αποθήκης Δεδομένων σε μία απλή διαδικασία ή σαν ένα σύνολο σχετικών διαδικασιών που στοχεύουν προς ένα λεπτομερή σύνολο.
- **Αποθήκη Μετα-δεδομένων (Metadata Repository)**: Είναι ένα υποσύστημα που αποθηκεύει πληροφορία που σχετίζεται με την δομή και την λειτουργία του συστήματος. Η πληροφορία αυτή ονομάζεται Μετα-δεδομένα και αφορά τον σχεδιασμό των ETL.
- **ETL (Extraction-Transformation-Loading) εργαλεία**: Είναι εργαλεία που είναι υπεύθυνα για την εξαγωγή δεδομένων από διαφορετικές πηγές, τον καθαρισμό, την προσαρμογή και την εισαγωγή στην Αποθήκη Δεδομένων.
- **OLAP (On-Line Analytical Processing) εργαλεία**: Είναι σχεδιασμένα για να προσφέρουν στους αναλυτές την δυνατότητα να εκτελέσουν την απόφαση υποστήριξης πάνω σε δεδομένα συναλλαγών. Τα OLAP εργαλεία είναι βασισμένα σε πολυδιάστατες όψεις δεδομένων, όπου οι αναλυτές μπορούν

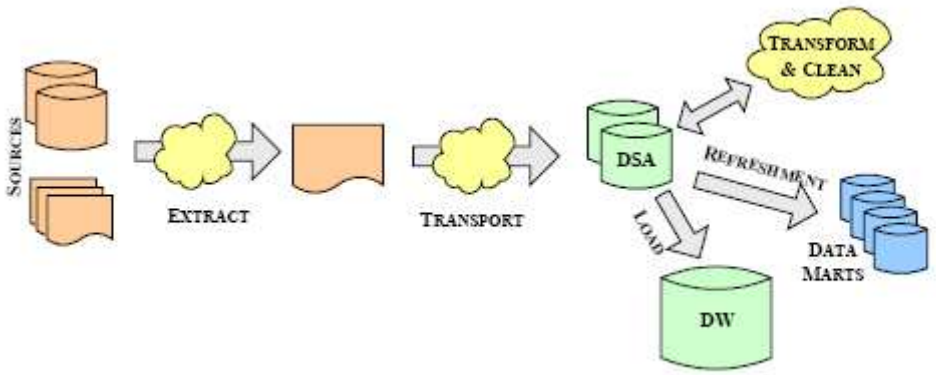
εύκολα να εκτελέσουν συναθροίσεις στα δεδομένα με διάφορους τρόπους και να εξάγουν χρήσιμη πληροφορία.



Σχήμα 1.1 Αρχιτεκτονική μίας Αποθήκης Δεδομένων

Η διαδικασία της μετακίνησης δεδομένων από τις Πηγές Δεδομένων στην Αποθήκη Δεδομένων περιλαμβάνει τρία βήματα:

- **Εξαγωγή (Extraction)**: είναι η διαδικασία καθορισμού ποια δεδομένα που είναι αποθηκευμένα στις πηγές θα επεξεργαστούν περαιτέρω.
- **Μετασχηματισμός (Transformation)**: είναι η διαδικασία στην οποία τα δεδομένα προσαρμόζονται στην μορφή που απαιτείται από την Αποθήκη Δεδομένων.
- **Φόρτωση (Loading)**: είναι η διαδικασία φορτώματος των δεδομένων στην Αποθήκη Δεδομένων.



Σχήμα 1.2 ETL Διαδικασίες

Όπως έχει ειπωθεί προηγουμένως, οι Αποθήκες Δεδομένων είναι συλλογές δεδομένων που προέρχονται από διαφορετικές πηγές και χρησιμοποιούνται κυρίως για τη λήψη αποφάσεων σε ένα οργανισμό. Για να τροφοδοτηθεί μια αποθήκη με νέα δεδομένα, όπως αυτά παράγονται στις πηγές, χρησιμοποιούνται εργαλεία Εξαγωγής – Μετασχηματισμού – Φόρτωσης δεδομένων (Extract – Transform – Load tools, ETL), τα οποία οργανώνουν τα επί μέρους βήματα της όλης διαδικασίας σαν μία ροή εργασίας. Μία ροή εργασίας ETL μπορεί να θεωρηθεί ως ένας κατευθυνόμενος ακυκλικός γράφος που χρησιμοποιείται για να αναπαραστήσει τη ροή δεδομένων από τις πηγές δεδομένων προς την αποθήκη δεδομένων. Οι κόμβοι του γράφου είναι διαδικασίες καθαρισμού/ μετασχηματισμού δεδομένων ή σύνολα εγγραφών και οι ακμές σχέσεις εισόδου/εξόδου μεταξύ των κόμβων. Η ροή εργασίας είναι ένα αφηρημένο σχήμα σε λογικό επίπεδο, το οποίο πρέπει να υλοποιηθεί σε φυσικό επίπεδο, δηλαδή να αντιστοιχηθεί σε ένα συνδυασμό από εκτελέσιμα προγράμματα που εκτελούν την ETL ροή εργασίας.

Σε επίπεδο λογισμικού, μεταβίβαση των δεδομένων μεταξύ των κόμβων της ροής εργασίας υλοποιείται με ουρές που επικοινωνούν μεταξύ τους. Στην μέχρι τώρα βιβλιογραφία δεν έχει προταθεί κάποια αποδοτική μέθοδος για την ρύθμιση των πόρων σε ροές εργασίας ETL. Όλες οι μέθοδοι που έχουν προταθεί μέχρι σήμερα αφορούν πλάνα εκτέλεσης κυρίως για αριστεροβαθή [HuSS00] δένδρα και δεν

λαμβάνουν υπόψη τους τις ιδιαιτερότητες των ροών εργασίας ETL ή αφορούν πολιτικές για τον καλύτερο χρονοπρογραμματισμό αυτών [Kara07]. Έτσι, στην εργασία αυτή μελετάται η όσο το δυνατό αποδοτικότερη ρύθμιση της απόδοσης πόρων σε ροές εργασίας ETL, δηλαδή η αποδοτικότερη διανομή της διαθέσιμης μνήμης ανάμεσα στις διάφορες δραστηριότητες. Η διανομή της διαθέσιμης μνήμης γίνεται με διάφορες τεχνικές όπως: εύνοια δραστηριοτήτων συνένωσης, εύνοια δραστηριοτήτων συνάθροισης, εύνοια δραστηριοτήτων εγγραφής και εύνοια δραστηριοτήτων ταξινόμησης.

Η εργασία αυτή συνεισφέρει στον τομέα των ETL εργαλείων τα εξής:

- Μοντελοποιείται το θεωρητικό πρόβλημα που θέλουμε να επιλύσουμε
- Αναπτύσσονται μέθοδοι ρύθμισης με μικρο-ελέγχους.
- Εξετάζονται εναλλακτικές στρατηγικές απόδοσης της μνήμης.
    - Δίκαιη διαμοίραση μνήμης, δηλαδή σε όλες τις δραστηριότητες μίας ροής εργασιών ETL διανέμεται το ίδιο ποσοστό μνήμης.
    - Εύνοια δραστηριοτήτων συνένωσης: σε αυτή την περίπτωση δίνεται περισσότερη μνήμη σε όσες δραστηριότητες της ροής εργασίας ETL είναι δραστηριότητες συνένωσης. Δηλαδή, οι ουρές που συνδέονται με αυτού του τύπου τις δραστηριότητες θα πάρουν περισσότερη μνήμη από ότι θα πάρουν οι υπόλοιπες μέσα στο γράφημα.
    - Εύνοια δραστηριοτήτων συνάθροισης: εδώ δίνεται εύνοια σε όσες από τις δραστηριότητες του γραφήματος που κάνουν κάποιου είδους συναθροίσεις, δηλαδή δίνεται περισσότερη μνήμη στις ουρές που συνδέονται με δραστηριότητες συνάθροισης.
    - Εύνοια δραστηριοτήτων εγγραφής: στην περίπτωση αυτή ευνοούνται οι δραστηριότητες που γράφουν τα δεδομένα τους σε μία Αποθήκη Δεδομένων. Αυτό έχει σαν αποτέλεσμα να παίρνουν και μεγαλύτερο ποσοστό μνήμης από ότι οι υπόλοιπες του γραφήματος.
    - Εύνοια δραστηριοτήτων ταξινόμησης: σε αυτή την στρατηγική ευνοούνται οι δραστηριότητες ταξινόμησης, είτε αυτές λειτουργούν μόνες τους, είτε ως μέρος άλλων δραστηριοτήτων (όπως στην περίπτωση των δραστηριοτήτων συνάθροισης ή συνένωσης). Έτσι, σε

αυτή την περίπτωση διανέμεται μεγαλύτερο ποσοστό μνήμης στις ουρές που συνδέονται με αυτού του τύπου τις δραστηριότητες από ότι στις υπόλοιπες.

- Τέλος, συγκρίνονται πειραματικά όλες αυτές οι διάφορες τεχνικές.
    - Για το Σενάριο Δένδρου καλύτεροι είναι οι: Δίκαιος Αλγόριθμος, Αλγόριθμος Προτεραιότητας Ταξινομήσεων και Αλγόριθμος Προτεραιότητας Συναθροίσεων.
    - Για το Σενάριο Πεταλούδας καλύτεροι είναι ο Δίκαιος Αλγόριθμος, ο Αλγόριθμος Προτεραιότητας Ταξινομήσεων και ο Αλγόριθμος Προτεραιότητας Συναθροίσεων.
    - Για το Γραμμικό Σενάριο κερδίζει ο Αλγόριθμος Προτεραιότητας Δραστηριοτήτων Εγγραφής για παράγοντες κλιμάκωσης 0.1 και 0.5, ενώ για παράγοντα κλιμάκωσης 1.0 κερδίζει ο Αλγόριθμος Προτεραιότητας Συναθροίσεων.
    - Τέλος, για το Σενάριο Πολλών Εκροών ο Δίκαιος Αλγόριθμος έχει την καλύτερη συμπεριφορά από όλους τους άλλους.

1.2. Δομή της Διατριβής

Η παρούσα διατριβή περιέχει πέντε κεφάλαια τα οποία οργανώνονται ως εξής: Το Κεφάλαιο 2 παρουσιάζει το θεωρητικό υπόβαθρο πάνω στην περιοχή των ETL εργαλείων. Επίσης, περιέχει και κάποιες τεχνικές για την βέλτιστη διαχείριση μνήμης. Στο Κεφάλαιο 3 παρουσιάζεται το πρόβλημά μας μοντελοποιώντας το σαν μία παραλλαγή του “knapsack” προβλήματος και επίσης παρουσιάζονται οι αλγόριθμοι κατανομής της μνήμης ανάμεσα στις λειτουργίες ενός ETL γραφήματος. Στο Κεφάλαιο 4 παρουσιάζονται τα πειράματα που έγιναν με βάση τους αλγορίθμους κατανομής μνήμης και η σύγκριση ανάμεσα τους. Τέλος, στο Κεφάλαιο 5 συνοψίζονται τα αποτελέσματα που παρήχθησαν και αναφέρονται ενδιαφέροντα θέματα προς επίλυση για μελλοντική δουλειά.

ΚΕΦΑΛΑΙΟ 2. ΘΕΩΡΗΤΙΚΟ ΥΠΟΒΑΘΡΟ

2.1. ETL (Εξαγωγή-Μετασχηματισμός-Φόρτωμα/Extraction-Transformation-Loading)

Τα ETL εργαλεία είναι κομμάτι λογισμικού υπεύθυνο για την εξαγωγή δεδομένων από διάφορες πηγές, τον καθαρισμό τους, την παραμετροποίηση και την εισαγωγή τους σε μία Αποθήκη Δεδομένων.

Οι λειτουργίες των εργαλείων αυτών είναι οι εξής:

1. *Εξαγωγή (Extraction)* κατάλληλης πληροφορίας από την πλευρά της πηγής.
2. *Μεταφορά (Transportation)* αυτής της πληροφορίας στην Περιοχή Ανασυγκρότησης Δεδομένων.
3. *Μετασχηματισμός (Transformation)* της πληροφορίας που προέρχεται από πολλαπλές πηγές σε μία κοινή μορφή.
4. *Καθαρισμός (Cleaning)* του συνόλου των δεδομένων.
5. *Διάδοση (Propagation) και Φόρτωση (Loading)* των δεδομένων στην Αποθήκη Δεδομένων και στις Αγορές Δεδομένων.

Το 90% των προβλημάτων στις Αποθήκες Δεδομένων προκύπτουν κατά την διάρκεια του νυχτερινού κύκλου φορτώματος των δεδομένων. Σε αυτή την περίοδο οι διαχειριστές του

συστήματος έχουν να αντιμετωπίσουν προβλήματα όπως: α) αποδοτικό φόρτωμα των δεδομένων, και β) συνδυασμός και εξαρτήσεις παράλληλων εργασιών. Επιπλέον οι ETL διαδικασίες έχουν χρονικούς περιορισμούς, συμπεριλαμβανομένου του χρόνου εκκίνησης και τερματισμού. Στις περισσότερες περιπτώσεις υπάρχει ένα περιορισμένο χρονικό παράθυρο κατά την διάρκεια της νύχτας, όπου προωθείται η ανανέωση στην Αποθήκη Δεδομένων, επειδή αυτή την χρονική περίοδο οι πηγές του συστήματος είναι εκτός λειτουργίας ή δεν χρησιμοποιούνται ευρέως [VaSi09].

Συμπερασματικά το κυριότερο πρόβλημα προκύπτει στον χρονοπρογραμματισμό όλων των διαδικασιών. Έτσι ο διαχειριστής του συστήματος πρέπει να βρει την σωστή σειρά εκτέλεσης για τις εξαρτώμενες εργασίες.

Κατά την διάρκεια των ETL διαδικασιών το πρώτο πράγμα που εκτελείται είναι η *εξαγωγή* της σχετικής πληροφορίας, η οποία στην συνέχεια θα προωθηθεί στην Αποθήκη Δεδομένων.

Οι περιορισμοί που προκύπτουν κατά την διάρκεια αυτής της διαδικασίας είναι οι εξής:

- η πηγή πρέπει να προκαλεί το ελάχιστο “overhead” κατά την διάρκεια της εξαγωγής, όσο άλλες διαχειριστικές διαδικασίες λαμβάνουν χώρα κατά την ίδια χρονική περίοδο, και
- για τεχνικούς λόγους αλλά και λόγους πολιτικής, οι διαχειριστές του συστήματος είναι απρόθυμοι να δεχτούν παρεμβάσεις στην διαμόρφωση του συστήματος. Επομένως θα πρέπει να υπάρχει η ελάχιστη ανάμειξη με την διαμόρφωση του λογισμικού από την πλευρά του κώδικα.

Με βάση τους παραπάνω περιορισμούς έχουν προταθεί οι εξής τεχνικές *εξαγωγής* σχετικής πληροφορίας:

1. Εξαγωγή όλων των δεδομένων που περιέχει η πηγή και επεξεργασία αυτών σαν να ήταν τα αυθεντικά που φορτώνονται πρώτα στην Αποθήκη Δεδομένων.
2. Εξαγωγή ενός στιγμιότυπου των δεδομένων, το οποίο διαδοχικά συγκρίνεται με το προηγούμενο στιγμιότυπο των δεδομένων και εισαγωγές, διαγραφές και ανανεώσεις ανιχνεύονται.
3. Μία τεχνική που περιέχει την χρήση ερεθισμάτων (triggers) στην πηγή, τα οποία ενεργοποιούνται όταν μία τροποποίηση συμβαίνει στα δεδομένα της πηγής.

4. Τεχνική του “log sniffing”, κατά την οποία γίνεται σάρωση του αρχείου και ανακατασκευάζονται οι αλλαγές που έγιναν μετά το τελευταίο σάρωμα. Η τεχνική αυτή είναι αδύνατον να συμβεί στις περισσότερες πηγές δεδομένων.

Η επόμενη φάση που λαμβάνει χώρα είναι ο *μετασχηματισμός και ο καθαρισμός*. Εξαρτώμενες από την εφαρμογή και τα εργαλεία που χρησιμοποιούνται, οι ETL διαδικασίες μπορεί να περιλαμβάνουν πληθώρα μετασχηματισμών. Τα προβλήματα που προκύπτουν στην περίπτωση των μετασχηματισμών χωρίζονται σε δύο κατηγορίες:

- συγκρούσεις και προβλήματα σε επίπεδο σχήματος
- μετασχηματισμοί σε επίπεδο δεδομένων

Στην περίπτωση των προβλημάτων σε επίπεδο σχήματος έχουμε α) **συγκρούσεις στο όνομα**, όπου ίδιο όνομα χρησιμοποιείται για διαφορετικά αντικείμενα (ομώνυμα) ή διαφορετικά ονόματα χρησιμοποιούνται για το ίδιο αντικείμενο (ετερώνυμα), β) **δομικές συγκρούσεις**, όπου ένα πρέπει αντιμετωπίζεται με διαφορετικές αναπαραστάσεις του ίδιου αντικειμένου σε διαφορετικές πηγές. Επιπρόσθετα υπάρχουν πολλές διαφορετικές συγκρούσεις σε επίπεδο δεδομένων, όπως διπλές ή αντιφατικές εγγραφές, διαφορετικές τιμές αναπαράστασης κ.α..

Στην συνέχεια παρουσιάζονται τρεις κοινοί ETL μετασχηματισμοί:

- *Σημασιολογική Εναρμόνιση και Αποκανονικοποίηση (Semantic Reconciliation and Denormalization)*: Είναι ο μετασχηματισμός πληροφορίας που είναι οργανωμένη σε γραμμές, σε πληροφορία που είναι οργανωμένη σε στήλες.
- *Ανάθεση Υποκατάστατου Κλειδιού (Surrogate Key Assignment)*: Συνήθης τεχνική στις Αποθήκες Δεδομένων είναι η αντικατάσταση κλειδιών του συστήματος με ένα ομοιόμορφο κλειδί που ονομάζεται *υποκατάστατο (surrogate)*. Αυτό γίνεται για λόγους απόδοσης και εννοιολογικής ομοιογενοποίησης.
- *Προβλήματα στα Αλφαριθμητικά (String Problems)*: Είναι ο καθαρισμός και η ομοιογενοποίηση αλφαριθμητικών δεδομένων.

Τέλος το *φόρτωμα* στην Αποθήκη Δεδομένων έχει τις δικιές του τεχνικές απαιτήσεις. Ένα τυπικό δίλημμα που συναντάται είναι η επιλογή ανάμεσα στο φόρτωμα μεγάλου όγκου δεδομένων στο Σ.Δ.Β.Δ. (Σύστημα Διαχείρισης Βάσεων Δεδομένων) ή εισαγωγή των δεδομένων σαν ακολουθία γραμμών. Ένα δεύτερο πρόβλημα προέρχεται από την πιθανότητα εισαγωγής αποδοτικών εγγραφών για πρώτη φορά, με εγγραφές που ενημερώνονται από

προηγούμενο φόρτωμα δεδομένων. Τέλος ένα τρίτο πρόβλημα είναι η θεώρηση από την διαχειριστική ομάδα που έχει να ασχοληθεί με την ύπαρξη ευρετηρίων, την ύπαρξη όψεων ή και με τα δύο που ορίζονται πάνω στην Αποθήκη Δεδομένων [SiVS05a].

2.2. ETL Εργαλεία

Παρόλο που υπάρχει μία πληθώρα από εμπορικές λύσεις που προσφέρουν δυνατότητες για την δημιουργία ενός ETL σεναρίου στην αγορά, ένας σχεδιαστής-διαχειριστής χρειάζεται μία συμπαγή μέθοδο για να αναπτύξει ένα αποδοτικό και εύρωστο ETL γράφημα ροής δεδομένων (ETL workflow). Έτσι, σε αυτή την ενότητα παρουσιάζονται θέματα που αφορούν την κατασκευή και την βελτιστοποίηση ETL εργασιών. Γι' αυτό τον λόγο, παρουσιάζονται ETL τεχνολογίες που προτάθηκαν α) από εμπορικές μελέτες και β) από την ερευνητική κοινότητα.

2.2.1. Εμπορικές Μελέτες και Εργαλεία

Σε ό,τι αφορά τις εμπορικές μελέτες που έχουν γίνει, το κύριο χαρακτηριστικό τους είναι ότι χρησιμοποιούν τις παραδοσιακές βάσεις δεδομένων με τις λύσεις των ETL σε ένα Σ.Δ.Β.Δ. (Σύστημα Διαχείρισης Βάσεων Δεδομένων). Οι τρεις μεγάλες τέτοιου είδους βάσεις δεδομένων με ETL λύσεις είναι: η Oracle με το Oracle Warehouse Builder 11 [Oracle09], η Microsoft με το SQL Server 2005 Integration Services (SSIS) [SSIS09] και η IBM με το InfoSphere DataStage [IBM07]. Επίσης υπάρχουν σε αυτή την περιοχή και τα εξής: της Informatica το Informatica Powercenter 8.6 [Infrm09] και της Ascential το Ascential DataStage Integration [Asc09] (το τελευταίο αποτελεί κομμάτι του IBM με τις ETL λύσεις). Τα προηγούμενα τρία έχουν το πλεονέκτημα της ελαχιστοποίησης του κόστους, επειδή οδηγούνται από την βάση δεδομένων, ενώ τα τελευταία δύο έχουν το πλεονέκτημα να στοχεύουν σε σύνθετες και βαθιές λύσεις που δεν είναι εύκολα ορατές από τα γενικά προϊόντα.

2.2.2. Ερευνητικού Τύπου Εργαλεία

Στο παρελθόν έχουν γίνει ερευνητικές προσπάθειες στον σχεδιασμό και την βελτιστοποίηση των ETL εργασιών. Εδώ αναφέρονται τρεις από αυτές: α) AJAX, β) Potter's Wheel, γ) ARKTOS II. Τα δύο πρώτα είναι βασισμένα στην άλγεβρα, ενώ το τελευταίο αναφέρεται στην μοντελοποίηση των ETL διεργασιών με προσαρμόσιμο και επεκτάσιμο τρόπο.

2.2.2.1. AJAX

Το σύστημα AJAX [GFSS00] είναι ένα εργαλείο καθαρισμού δεδομένων που αναπτύχθηκε στην INRIA της Γαλλίας. Ασχολείται με τυπικά προβλήματα ποιότητας δεδομένων όπως για παράδειγμα το πρόβλημα ταυτοποίησης ενός αντικειμένου, το πρόβλημα των λαθών που οφείλονται σε μη πληκτρολόγησή τους και ανακολουθίες δεδομένων ανάμεσα σε ταιριάσματα εγγραφών. Το σύστημα AJAX μοντελοποιεί τον καθαρισμό δεδομένων σαν ένα κατευθυνόμενο γράφημα, στο οποίο μετασχηματίζονται τα δεδομένα τα οποία προέρχονται από κάποιες πηγές δεδομένων. Επίσης παρέχει μία σαφώς καθορισμένη γλώσσα για τα προγράμματα του καθαρισμού των δεδομένων, η οποία αποτελείται από SQL προτάσεις που χρησιμοποιούνται για να εκφράσουν μετασχηματισμούς αντιστοίχησης, ταιριάσματος, ομαδοποίησης και συγχώνευσης. Τέλος, παρέχεται στον χρήστη ένα διαδραστικό περιβάλλον για να επιλυθούν τα σφάλματα και οι αντιφάσεις που δεν μπορούν να διεκπεραιωθούν αυτόματα και για να υποστηρίξει την σταδιακή βελτιστοποίηση του σχεδιασμού των προγραμμάτων καθαρισμού των δεδομένων.

2.2.2.2. Potter's Wheel

Το σύστημα Potter's Wheel [RaHe01] σχεδιάστηκε με σκοπό να παρέχει στον χρήστη του ένα διαδραστικό καθαρισμό των δεδομένων. Το σύστημα αυτό δίνει την δυνατότητα να εκτελούνται διάφορες αλγεβρικές πράξεις πάνω σε ένα μέρος του

συνόλου των δεδομένων. Επίσης παρέχονται αλγόριθμοι βελτιστοποίησης για τον τρόπο χρήσης της CPU για συγκεκριμένους τύπους πράξεων. Η γενική ιδέα στο σύστημα Potter's Wheel είναι ότι οι χρήστες δημιουργούν μετασχηματισμούς δεδομένων με επαναληπτικό και διαδραστικό τρόπο. Από την άλλη το σύστημα αυτόματα δημιουργεί δομές για τις τιμές των δεδομένων σε πεδία που καθορίζονται από τον χρήστη και για την παραβίαση περιορισμών. Έτσι οι χρήστες σταδιακά δημιουργούν μετασχηματισμούς για τον καθαρισμό των δεδομένων προσθέτοντας ή αναιρώντας μετασχηματισμούς όπως γίνεται σε ένα περιβάλλον λογιστικού φύλλου. Αυτοί οι μετασχηματισμοί καθορίζονται είτε από απλές σχηματικές λειτουργίες, είτε παρουσιάζοντας τις επιθυμητές επιδράσεις σε παραδείγματα με δεδομένα τιμών

2.2.2.3. ARKTOS II

Το ARKTOS II [VaSS02], [VSGT03] αποτελεί ένα σύστημα για τον εννοιολογικό, λογικό και φυσικό σχεδιασμό ETL διαδικασιών. Στόχος του συστήματος είναι να διευκολύνει την έρευνα, την διαχείριση και την βελτιστοποίηση του σχεδιασμού και της υλοποίησης των ETL διαδικασιών, τόσο στο αρχικό στάδιο της ανάπτυξης όσο και κατά την διάρκεια της συνεχούς εξέλιξης μίας αποθήκης δεδομένων. Έτσι γι' αυτό τον σκοπό το σύστημα ARKTOS II προσφέρει διάφορα πρότυπα που αντιστοιχούν σε ETL μετασχηματισμούς, που περιλαμβάνουν την σημασιολογία και την διασύνδεσή τους. Με αυτό τον τρόπο διευκολύνεται η κατασκευή των ETL σεναρίων. Επιπλέον λαμβάνει υπόψη την βελτιστοποίηση των ETL σεναρίων με κύριο στόχο την βελτίωση των χρονικών επιδόσεων μίας ETL διαδικασίας. Τέλος το σύστημα ARKTOS II αντιμετωπίζει αποτελεσματικά το πρόβλημα του πώς το λογισμικό σχεδιασμού ενός ETL σεναρίου μπορεί να βελτιωθεί χωρίς να έχει επιπτώσεις στην συνέπειά του.

2.3. ETL Γραφήματα Ροής Εργασιών (ETL Workflows)

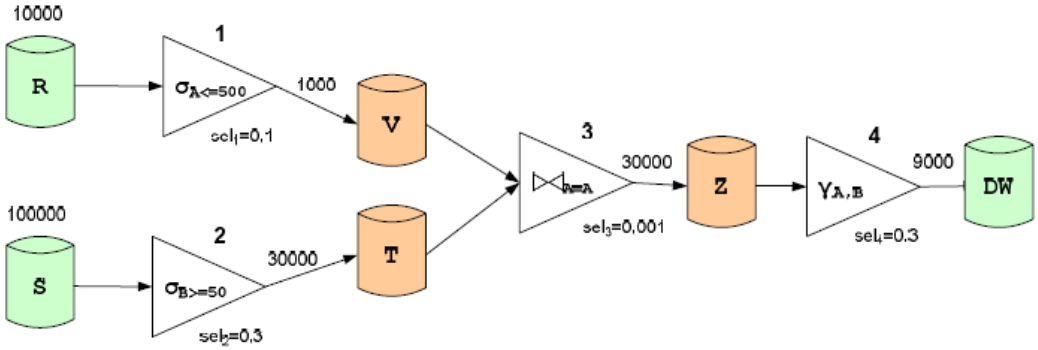
Κάθε ξεχωριστός μετασχηματισμός ή καθαρισμός μίας εργασίας σε ένα ETL σενάριο θεωρείται σαν μία *δραστηριότητα* σε ένα *γράφημα ροής εργασιών (workflow).*Ένα ETL workflow αναπαριστά γραφικά την αλληλοσύνδεση ανάμεσα στους μετασχηματισμούς ενός ETL σεναρίου και σχεδιάζει την ροή των δεδομένων από τις πηγές στην αποθήκη δεδομένων, διαμέσου αυτών των μετασχηματισμών. Το μοντέλο του ETL workflow σύμφωνα με τους συγγραφείς του [SiVS05b] είναι ένα *ακυκλικός κατευθυνόμενο γράφημα (Directed Acycle Gracph - DAG),* **G(V,E)** που περιέχει δύο είδη κόμβων: α) *τις δραστηριότητες (activities **A**)* και β) *τα σύνολα-εγγραφών (recordsets **RS**)*. Επομένως ισχύει ότι το σύνολο των κόμβων περιλαμβάνει όλες τις δραστηριότητες και όλα τα σύνολα εγγραφών (**V=AURS**) , ενώ το σύνολο των ακμών περιέχει μία λίστα από τις σχέσεις που διέπουν τους κόμβους μεταξύ τους (**E=Pr**). Οι δραστηριότητες είναι μονάδες λογισμικού που εκτελούν τις διαδικασίες μετασχηματισμού ή καθαρισμού στα δεδομένα. Ενώ τα σύνολα των εγγραφών χρησιμοποιούνται για την αποθήκευση των δεδομένων. Αυτά μπορεί να είναι είτε Πηγές Δεδομένων εάν είναι ο αρχικός κόμβος ενός μονοπατιού, είτε Περιοχές Ανασυγκρότησης Δεδομένων εάν βρίσκονται στο μέσο ενός μονοπατιού, είτε Αποθήκες Δεδομένων ή όψεις εάν είναι ο τερματικός κόμβος ενός μονοπατιού. Τέλος οι ακμές του γραφήματος χρησιμοποιούνται για να δείξουν την ροή των δεδομένων από τις πηγές στην Αποθήκη Δεδομένων.

Κάθε κόμβος χαρακτηρίζεται από ένα ή περισσότερα σχήματα (λίστα από γνωρίσματα). Τα σύνολα εγγραφών έχουν μόνο ένα σχήμα, ενώ από την άλλη οι δραστηριότητες έχουν τουλάχιστον δύο. Οι δραστηριότητες αποτελούνται από ένα σύνολο *σχημάτων εισόδου (input schemata),* τα οποία φέρνουν τις εγγραφές στην δραστηριότητα για επεξεργασία και ένα ή περισσότερα *σχήματα εξόδου (output schemata),* τα οποία είναι υπεύθυνα να προωθήσουν τα δεδομένα στον επόμενο κόμβο (δραστηριότητα ή σύνολο εγγραφών). Μία δραστηριότητα με ένα σχήμα εισόδου ονομάζεται *μοναδιαία (unary),* ενώ μία δραστηριότητα με δύο σχήματα εισόδου ονομάζεται *δυαδική (binary).*

Τυπικά κάθε δραστηριότητα αποτελείται από μία τετράδα **A(Id, I, O, S)**.

- **Id**: Ένα μοναδικό αναγνωριστικό για κάθε δραστηριότητα.

- **I**: Ένα πεπερασμένο σύνολο από ένα ή περισσότερα σχήματα εισόδου.
- **O**: Ένα πεπερασμένο σύνολο από ένα ή περισσότερα σχήματα εξόδου.
- **S**: Μία ή περισσότερες εκφράσεις στην σχεσιακή άλγεβρα, που χαρακτηρίζουν την σημασιολογία της ροής δεδομένων για κάθε σχήμα εξόδου.



Σχήμα 2.1 Παράδειγμα ενός ETL Γραφήματος Ροής Εργασιών

Το Σχήμα 2.1 παρουσιάζει ένα απλό σενάριο ETL που αναπαριστάνεται γραφικά σε ένα ETL γράφημα ροής εργασιών. Το σενάριο περιλαμβάνει δύο πηγές με πίνακες με ονόματα R και S, μία κεντρική Αποθήκη Δεδομένων και τρεις Περιοχές Ανασυγκρότησης Δεδομένων (V, T, Z). Τα σχεσιακά σχήματα των δεδομένων στις πηγές είναι R(A,B) και S(A,B) αντίστοιχα. Οι δραστηριότητες είναι σχολιασμένες με νούμερα από το 1 ως 4 και έχουν σαν ετικέτα την περιγραφή της λειτουργίας τους. Επίσης στις πηγές των δεδομένων είναι σημειωμένο το πλήθος των πλειάδων που εξάγονται και σε κάθε ακμή είναι σημειωμένο το πλήθος των πλειάδων που προωθούνται από τον προμηθευτή προς τον καταναλωτή. Τέλος σε κάθε δραστηριότητα σημειώνεται και η *επιλεκτικότητά (selectivity)* της.

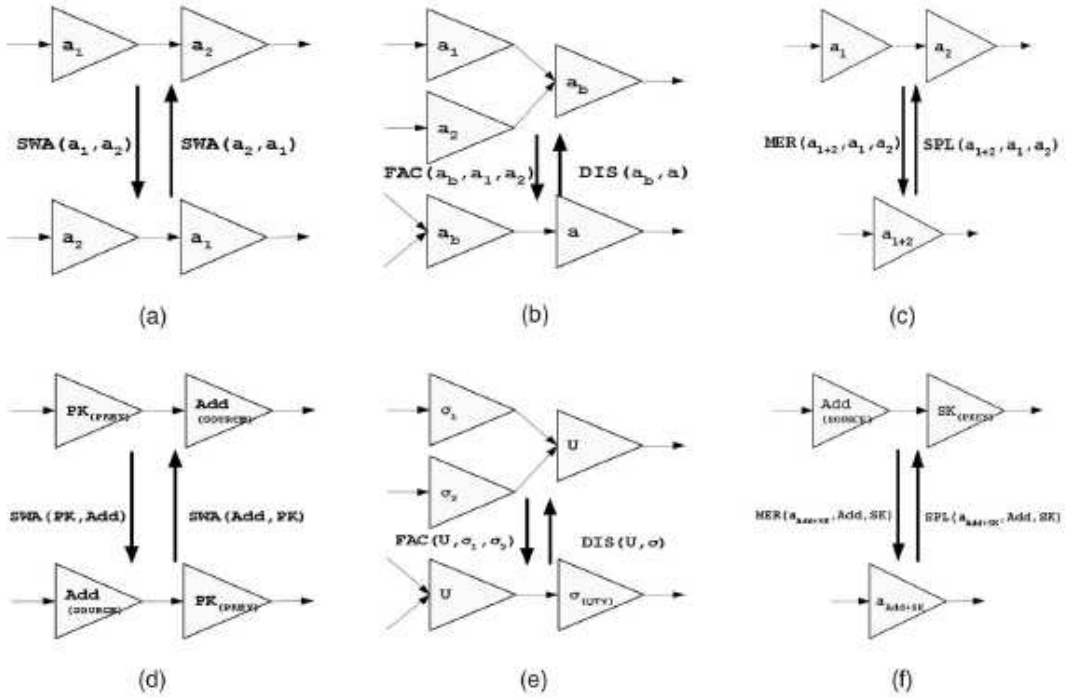
Η λειτουργία των παραπάνω δραστηριοτήτων είναι η εξής: Η δραστηριότητα 1 είναι ένα φίλτρο που επιτρέπει να περάσουν μόνο οι πλειάδες με τιμή μικρότερη ή ίση από το 500 στο γνώρισμα A. Η δραστηριότητα 2 είναι ένα φίλτρο που επιτρέπει να περάσουν οι πλειάδες που έχουν τιμή στο γνώρισμα B μεγαλύτερη ή ίση του 50. Η δραστηριότητα 3 είναι ένα “join” που συνενώνει τα δύο μονοπάτια που προέρχονται από τις δραστηριότητες 1 και 2. Και τέλος, η δραστηριότητα 4 είναι ένα “group by”, δηλαδή ομαδοποιεί τις πλειάδες πάνω στα γνωρίσματα A και B.

2.4. Βελτιστοποίηση ETL Γραφήματος Ροής Δεδομένων

Η ελαχιστοποίηση του κόστους εκτέλεσης ενός ETL workflow παρουσίασε ένα ερευνητικό πρόβλημα με πρακτικές συνέπειες. Οι συγγραφείς [SiVS05c] χειρίζονται το πρόβλημα βελτιστοποίησης των ETL διεργασιών, μοντελοποιώντας το σαν ένα πρόβλημα αναζήτησης σε ένα χώρο καταστάσεων. Θεωρούν κάθε ETL γράφημα σαν μία κατάσταση και κατασκευάζουν τον χώρο των καταστάσεων διαμέσου ενός συνόλου μεταβολών των καταστάσεων. Οι μεταβολές χρησιμοποιούνται για να παράγουν καινούριες ισοδύναμες καταστάσεις. Προτάθηκαν 5 διαφορετικοί τύποι μεταβολών, που εφαρμόζονται στους κόμβους του γραφήματος για να παράγουν ισοδύναμες καταστάσεις.

Αυτές είναι:

- *Αντιμετάθεση (Swap)*: Ένα ζευγάρι μοναδιαίων δραστηριοτήτων ανταλλάσσει τις ακολουθίες τους (Σχήμα 2.1 (a),(d)).
- *Παραγοντοποίηση (Factorize)*: Αντικαθιστά δύο μοναδιαίες δραστηριότητες οι οποίες έχουν την ίδια λειτουργία και δρουν σαν προμηθευτές της ίδιας δυαδικής δραστηριότητας με μία καινούργια μοναδιαία δραστηριότητα. Αυτό γίνεται με σκοπό να εκμεταλλευτούμε το γεγονός ότι η συγκεκριμένη δραστηριότητα θα υπολογιστεί μία μόνο φορά (Σχήμα 2.1 (b),(e)).
- *Κατανομή (Distribute)*: Η κατανομή είναι η αντίστροφη διαδικασία από αυτή της παραγοντοποίησης. Εάν μία δραστηριότητα ενεργεί πάνω σε μία απλή ροή δεδομένων μπορεί να κατανεμηθεί σε δύο διαφορετικές ροές δεδομένων, δηλ. σε δύο διαφορετικά μονοπάτια. Με τον τρόπο αυτό δραστηριότητες με μεγάλη επιλεκτικότητα προωθούνται προς την αρχή του γραφήματος (Σχήμα 2.1 (b),(e)).
- *Ένωση (Merge)*: Η διαδικασία της ένωσης εφαρμόζεται πάνω σε ένα ζευγάρι δραστηριοτήτων που ενώνονται σε μία δραστηριότητα (Σχήμα 2.1 (c),(f)).
- *Διαχωρισμός (Split)*: Ένα ζευγάρι ομαδοποιημένων δραστηριοτήτων μπορούν να διασπαστούν-διαχωριστούν σε δύο ξεχωριστές δραστηριότητες (Σχήμα 2.1 (c),(f)).



Σχήμα 2.2 Τύποι Μεταβολών των Δραστηριοτήτων

Για τον σκοπό των μεταβολών των καταστάσεων εκτός από τα σχήματα εισόδου-εξόδου υπάρχουν και τα ακόλουθα σχήματα που χαρακτηρίζουν κάθε δραστηριότητα:

1. **Λειτουργικό Σχήμα (Functionality Schema):** Είναι ένα σύνολο γνωρισμάτων που είναι υποσύνολο των σχημάτων εισόδου, που δηλώνει τα γνωρίσματα που συμμετέχουν στον υπολογισμό που εκτελείται από την δραστηριότητα.
2. **Παραγόμενο Σχήμα (Generated Schema):** Το σχήμα αυτό περιλαμβάνει όλα τα γνωρίσματα του σχήματος εξόδου που παράγονται από την επεξεργασία της δραστηριότητας.
3. **Αφαιρούμενο Σχήμα (Projected-out Schema):** Περιλαμβάνει τα γνωρίσματα που ανήκουν στα σχήματα εισόδου και δεν διαδίδονται περαιτέρω από την δραστηριότητα.

Αφού έχουν υπολογιστεί τα παραπάνω σχήματα από τις δραστηριότητες, κατασκευάζονται αυτόματα τα σχήματα εισόδου-εξόδου. Η διαδικασία αυτή ονομάζεται παραγωγή σχήματος (schema generation) και γίνεται ως εξής:

- Βήμα 1: Τοπολογική ταξινόμηση του γραφήματος.

- Βήμα 2: Ακολουθώντας την τοπολογική διάταξη, ανατίθενται τα σχήματα εισόδου σε κάθε δραστηριότητα έτσι ώστε να είναι ίδια με τα σχήματα εξόδου των προμηθευτών τους.
- Βήμα 3: Τα σχήματα εξόδου κάθε δραστηριότητας είναι ίσα με την ένωση των σχημάτων εισόδου και των παραγόμενων σχημάτων, αφαιρώντας τα γνωρίσματα που υπάρχουν στο αφαιρούμενο σχήμα.

Μετά από κάθε αλλαγή υπολογίζονται όλα τα σχήματα με βάση τον νέο γράφημα που έχει προκύψει. Αυτή η αυτόματη παραγωγή των σχημάτων που περιγράφηκε παραπάνω απεικονίζεται με την μορφή κώδικα στον αλγόριθμο Schema Generation (SGen) στο Πίνακα 2.1.

Πίνακας 2.1 Αλγόριθμος Παραγωγής Σχήματος

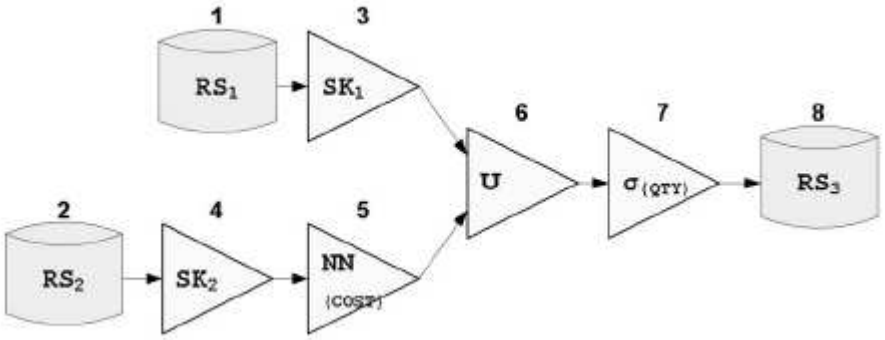
Algorithm Schema Generation (SGen)

1. **Input**: A state S, i.e. a graph **G=(V,E)** in involving n activities
2. **Output**: The state S annotated with fully defined input and output activity schemata
3. **Begin**
4. topologically sort **S;**
5. assign a priority **p**(i) to each activity and left a_i denote the activity with priority **p**(i);
6. **for** (i=1;i<n;i++) **{** //i.e., for each activity
7. **for** each input schema $s_{i,j}$ of a_i **{**
8. $s_{i,j}$ =output_schema(provider($s_{i,j}$));
9. **}**
10. output_schema(a_i) = $U_{(j)}s_{(i,j}$ U generated_schema(a_i) – projected_out(a_i);
11. **}**
12. **End.**

Τοπικές ομάδες (Local Groups): Μία τοπική ομάδα είναι ένα υποσύνολο του γραφήματος, τα στοιχεία της οποίας είναι ένα γραμμικό μονοπάτι από μοναδιαίες δραστηριότητες. Π.χ. στο παράδειγμα του Σχήματος 2.3 τοπικές ομάδες είναι οι {3}, {4,5}, {6}.

Ομόλογες Δραστηριότητες (Homologous Activities): Δύο δραστηριότητες είναι ομόλογες όταν: 1) βρίσκονται σε σύγκλιση τοπικές ομάδες, 2) έχουν την ίδια σημασιολογία (ίδια

αλγεβρική έκφραση) και 3) έχουν τα ίδια λειτουργικά, παραγόμενα και αφαιρούμενα σχήματα. Στο παράδειγμα του Σχήματος 2.3 οι ομόλογες δραστηριότητες είναι οι 3 και 4.



Σχήμα 2.3 Παράδειγμα ETL Γραφήματος

Πρέπει να καθοριστούν κανόνες που επιτρέπουν ή απαγορεύουν την εφαρμογή μετασχηματισμών πάνω σε ένα γράφημα. Μετά την παρουσίαση της παραγωγής του σχήματος ενός γραφήματος, η καταλληλότητα των μεταβολών εξαρτάται από την ορθότητα της παραγωγής ενός σχήματος.

Αντιμετάθεση: Επιτρέπεται η αντιμετάθεση δύο δραστηριοτήτων, α_1 και α_2,εάν:

- και οι δύο δραστηριότητες είναι γειτονικές στο γράφημα
- α_1 και α_2 έχουν απλά σχήματα εισόδου και εξόδου και το σχήμα εξόδου και των δύο δραστηριοτήτων έχει ακριβώς έναν καταναλωτή.
- το λειτουργικό σχήμα των α_1 και α_2 είναι υποσύνολο του σχήματος εισόδου, πριν και μετά την αντιμετάθεση
- τα σχήματα εισόδου των α_1 και α_2 είναι υποσύνολα των προμηθευτών, πριν και μετά την αντιμετάθεση.

Παραγοντοποίηση: Όταν κάνουμε παραγοντοποίηση δύο δραστηριότητες α_1 και α_2 τις αντικαθιστούμε με μία καινούρια δραστηριότητα α_b. Οι συνθήκες που πρέπει να ελέγξουμε για να γίνει ο μετασχηματισμός αυτός είναι οι εξής:

- Οι δραστηριότητες α_1 και α_2 να έχουν την ίδια λειτουργία σε σχέση με την αλγεβρική έκφραση αυτών, και το μόνο πράγμα στο οποίο να διαφέρουν να είναι τα σχήματα εισόδου και εξόδου.
- Οι δραστηριότητες α_1 και α_2 να έχουν κοινό καταναλωτή α_b που έχει δυαδική λειτουργία.

Κατανομή: Στην κατανομή οι συνθήκες που πρέπει να ελεγχθούν είναι παρόμοιες με πριν. Μία δραστηριότητα α μπορεί να κλωνοποιηθεί σε δύο μονοπάτια εάν:

- Μία δυαδική δραστηριότητα είναι προμηθευτής της α και δύο κλώνοι α_1 και α_2 παράγονται από κάθε μονοπάτι που οδηγούν στην α_b.
- Οι δραστηριότητες α_1 και α_2 έχουν την ίδια λειτουργία σε σχέση με την αλγεβρική έκφραση αυτών.

Ένωση: Η ένωση δεν διέπεται από κάποιους κανόνες. Το σχήμα εξόδου της καινούργιας δραστηριότητας είναι η έξοδος της δεύτερης δραστηριότητας και τα σχήματα εισόδου είναι η ένωση των σχημάτων εισόδου των δραστηριοτήτων, αφαιρώντας το σχήμα εισόδου της δεύτερης δραστηριότητας που συνδέεται στην έξοδο της πρώτης.

Διαχωρισμός: Ο διαχωρισμός απαιτεί η καινούρια δραστηριότητα να προέρχεται από ένωση άλλης.

Ένα προφανές ερώτημα που προκύπτει αφορά την ορθότητα των νέων μεταβολών που δημιουργούνται. Δηλαδή, με άλλα λόγια να μπορούμε να εγγυηθούμε ότι κάθε φορά που εφαρμόζεται μία μεταβολή σε μία ορισμένη κατάσταση του γραφήματος που έχουμε, η νέα κατάσταση που δημιουργείται να παράγει ακριβώς τα ίδια δεδομένα με την αρχική στο τέλος της εκτέλεσης.

Όταν ένας γράφος ροής εργασιών έχει εκτελεστεί σωστά, τότε όλες οι δραστηριότητες βεβαιώνονται ότι είναι αληθείς. Έτσι

Προ-συνθήκη Γράφου (Workflow postcondition): Κάθε γράφημα χαρακτηρίζεται από μία workflow postcondition, $Cond_{WF}$, η οποία είναι μία Boolean έκφραση που διατυπώνεται σαν σύνδεση προ-συνθηκών στις δραστηριότητες του γράφου, διευθετημένες σε διάταξη όπως γίνεται η εκτέλεσή τους.

Ισοδύναμοι Γράφοι: Δύο γράφοι ροής εργασιών W1 και W2 είναι ισοδύναμοι όταν:

- το σχήμα των δεδομένων που διαδίδεται προς κάθε στόχο είναι πανομοιότυπο
- $Cond_{w1} \equiv Cond_{w2}$.

Θεώρημα 1: Έχουμε μία κατάσταση S σε ένα γράφο **G=(V,E)**, όπου όλες οι δραστηριότητες έχουν ακριβώς μία έξοδο και έναν καταναλωτή για κάθε σχήμα εξόδου. Επίσης έχουμε μία μεταβολή Τα που παράγει μία καινούργια κατάσταση S', π.χ. ένα γράφος **G'=(V',E')**, που επηρεάζει ένα σύνολο δραστηριοτήτων $\mathbf{G_A \subset V \cup V'}$. Τότε τα σχήματα για τις δραστηριότητες $\mathbf{V\text{-}G_A}$ είναι όμοια με τα αντίστοιχα σχήματα των δραστηριοτήτων $\mathbf{V'\text{-}G_A}$.

Θεώρημα 2: Όλες οι μεταβολές παράγουν ισοδύναμους γράφους.

2.5. Τεχνικές Διαχείρισης Μνήμης

Πολύπλοκες ερωτήσεις χρησιμοποιούν αρκετά συχνά λειτουργίες συνένωσης, ταξινόμησης και συνάθροισης, οι οποίες καταναλώνουν πολλή μνήμη. Οι συνήθεις βελτιστοποιητές θεωρούν ότι όλη η μνήμη είναι διαθέσιμη σε κάθε λειτουργία μέσα στην ερώτηση. Όταν όμως υπάρχουν περιπτώσεις διασωλήνωσης κατά την εκτέλεση της ερώτησης, η μνήμη μοιράζεται ανάμεσα σε όλες τις λειτουργίες που τρέχουν ταυτόχρονα. Το κόστος της κάθε λειτουργίας εξαρτάται από την διαθέσιμη μνήμη. Γι' αυτό τον λόγο εάν η μνήμη που διατίθεται σε κάθε λειτουργία είναι λιγότερη από αυτή που θεωρεί ο βελτιστοποιητής, τότε το κόστος που υπολογίζεται από αυτόν είναι λάθος. Έτσι η βελτιστοποίηση μίας ερώτησης και η κατανομή της μνήμης είναι αλληλοεξαρτώμενα και εάν γίνουν ξεχωριστά δεν θα υπάρξουν τα βέλτιστα αποτελέσματα. Τέλος ο βελτιστοποιητής μίας ερώτησης δεν θα πρέπει μόνο να είναι υπεύθυνος για την συνολική διαθέσιμη μνήμη αλλά θα πρέπει να αποφασίζει το πώς θα μοιράζεται αυτή βέλτιστα ανάμεσα στις λειτουργίες.

Προηγούμενες δουλειές που έχουν γίνει πάνω σε αυτό το θέμα ασχολούνται περισσότερο με τον αποδοτικό προγραμματισμό των διαθέσιμων πόρων για την

εκτέλεση ενός δοσμένου πλάνου μίας ερώτησης. Αυτός ο προγραμματισμός των διαθέσιμων πόρων δεν αλληλεπιδρά με την βελτιστοποίηση μίας ερώτησης, αλλά τα δύο προβλήματα εκτελούνται ξεχωριστά.

Υπάρχουν δύο θέματα προς συζήτηση όπως αυτά προτάθηκαν στον [HuSS00]. Πρώτον, το κόστος ενός πλάνου εκτέλεσης μπορεί να αλλάξει όταν αλλάζει η υποδιαίρεση της μνήμης και δεύτερον, η ίδια η επιλογή του πλάνου εκτέλεσης πρέπει να περιλαμβάνει την μελέτη της κατανομής μνήμης.

Έχουν προταθεί από τους συγγραφείς του [HuSS00] δύο προσεγγίσεις προς την επίλυση του παραπάνω προβλήματος: η προσέγγιση των δύο φάσεων και η προσέγγιση της μίας φάσης. Στην προσέγγιση των δύο φάσεων αρχικά βελτιστοποιείται η ερώτηση χρησιμοποιώντας έναν συνήθη βελτιστοποιητή και στην συνέχεια μοιράζεται η μνήμη ανάμεσα στις λειτουργίες σε κάθε διασωλήνωση. Έτσι κάθε διασωλήνωση τρέχει βέλτιστα με βάση την διαθέσιμη μνήμη που έχουμε. Από την άλλη μεριά υπάρχει και η μέθοδος της μίας φάσης, στην οποία ο βελτιστοποιητής γίνεται γνώστης της διαθέσιμης μνήμης. Έτσι λαμβάνει υπόψη την διαμοίραση της μνήμης ανάμεσα στις λειτουργίες κατά την διάρκεια επιλογής του καλύτερου πλάνου εκτέλεσης.

Έχουν προταθεί επίσης διάφορες ακόμη τεχνικές για την διαχείριση της μνήμης. Οι περισσότερες από αυτές ασχολούνται με την προσέγγιση των δύο φάσεων. Οι συγγραφείς του [YuCo93] θεωρούν ένα περιβάλλον με παράλληλες εκτελέσεις ερωτήσεων και εκεί μελετούν την κατανομή της μνήμης σε ανεξάρτητες ερωτήσεις. Ορίζουν την αρχή της «επιστροφής στην κατανάλωση» (ROC), για να μελετήσουν την συνολική μείωση στον χρόνο απόκρισης λόγω της επιπρόσθετης κατανομής μνήμης σε μία ερώτηση. Οι [MeDe93] θεωρούν το πρόβλημα της διαμοίρασης της μνήμης σε ένα περιβάλλον με πολλαπλές ερωτήσεις, στο οποίο διαιρούν τις ερωτήσεις σε διαφορετικές κατηγορίες ανάλογα με τις απαιτήσεις τους σε μνήμη και παρέχουν ένα πλήθος ευρεστικών μεθόδων για την κατανομή της μνήμης στην εκάστοτε κατηγορία. Τέλος οι [BoKV98] πρότειναν διάφορα στατικά και δυναμικά σχέδια χρονοδρομολόγησης για το δέντρο εκτέλεσης μίας ερώτησης. Χωρίζουν το

δέντρο εκτέλεσης μίας ερώτησης σε ένα σύνολο μέγιστων αλυσίδων διασωλήνωσης και τις χρονοδρομολογούν ξεχωριστά. Στην στατική χρονοδρομολόγηση προτείνουν διάφορες ευριστικές τεχνικές στον διαμοιρασμό της μνήμης ανάμεσα στις λειτουργίες ανάλογα με τις ελάχιστες και μέγιστες απαιτήσεις κάθε λειτουργίας.

ΚΕΦΑΛΑΙΟ 3. ΑΛΓΟΡΙΘΜΟΙ ΚΑΤΑΝΟΜΗΣ ΜΝΗΜΗΣ ΣΤΙΣ ΛΕΙΤΟΥΡΓΙΕΣ ΕΝΟΣ ETL ΓΡΑΦΗΜΑΤΟΣ

Σε αυτό το κεφάλαιο παρουσιάζεται ο γενικός ορισμός του προβλήματος που διαπραγματευόμαστε. Στην συνέχεια, περιγράφονται οι αλγόριθμοι που υλοποιήθηκαν για την αποδοτικότερη διαμοίραση της μνήμης ανάμεσα στις δραστηριότητες μίας ροής εργασιών. Τέλος, παρουσιάζονται διάφορα παραδείγματα που δείχνουν διάφορους τρόπους λειτουργίας των αλγορίθμων.

3.1. Γενικός Ορισμός του Προβλήματος

Θεωρείστε ένα ETL σενάριο που μοντελοποιείται σαν ένα γράφημα *G(V,E)* με *k* κόμβους (|*V*| = *k*). Το πρόβλημα περιγράφεται σε διαφορετικές παραλλαγές:

1. Το γράφημα μοντελοποιείται σαν μία ροή δεδομένων (data flow) που μεταφέρει τα δεδομένα από ένα σύνολο πηγών στον τελικό προορισμό. Στην περίπτωση αυτή οι κόμβοι του γραφήματος είναι δραστηριότητες και σημεία στάσης, ενώ οι ακμές του γραφήματος είναι προμηθευτές (εισόδου/εξόδου) των σχέσεων.
2. Το γράφημα μοντελοποιείται σαν μία μεγάλη ροή ελέγχου (control flow) ενός ETL σεναρίου που συνδυάζει διάφορες ροές δεδομένων (ή άλλες ενέργειες)

σε παράλληλη κίνηση και σε σημεία συνάντησης. Οι κόμβοι του γραφήματος (όπως έχει προαναφερθεί) είναι ροές δεδομένων αυτού του μεγαλύτερου σεναρίου, ενώ οι ακμές του γραφήματος δηλώνουν μηνύματα αρχικοποίησης. Πιο συγκεκριμένα, μία ακμή σε μία ροή δεδομένων δείχνει εάν είναι επιτυχής ή προβληματική η εκτέλεση σε ένα συγκεκριμένο σημείο του γραφήματος και επίσης εισάγει τον επόμενο κόμβο στο γράφημα που μπορεί να είναι είτε μία άλλη ροή δεδομένων είτε κάποια άλλη ενέργεια υποστήριξης (αποστολή ηλεκτρονικού ταχυδρομείου, επανάληψη έκτατου ανάγκης ή κάποια άλλη).

Παρόλο που κάθε ρύθμιση έχει τα δικά της χαρακτηριστικά, προς το παρόν ορίζεται η ακριβής φύση του προβλήματος. Επίσης, χρησιμοποιείται ο όρος «δραστηριότητα» (module) για τους κόμβους του γραφήματος.

Ο σκοπός του προβλήματός μας είναι να βρούμε μία κατανομή της μνήμης σε διαφορετικές δραστηριότητες έτσι ώστε: α) το συνολικό κόστος της εκτέλεσης του γραφήματος να ελαχιστοποιηθεί και β) το συνολικό πλήθος της μνήμης που ανατίθεται στις δραστηριότητες να μην υπερβαίνει το μέγιστο διαθέσιμο και προκαθορισμένο ποσό μνήμης.

Το πρόβλημα που παρουσιάζεται βασίζεται στην υπόθεση ότι έχουμε μία γενική ένδειξη για την συμπεριφορά της κάθε δραστηριότητας, όταν ανατίθεται επιπλέον μνήμη σε αυτή. Με άλλα λόγια, στηρίζεται σε μία αρχική συγκριτική αξιολόγηση των διαφόρων δραστηριοτήτων όταν βρίσκονται μόνες τους. Για την απλούστευση τόσο της συζήτησης όσο και των συμβολισμών, θεωρείται ένα συγκεκριμένο μέγεθος στο σύνολο δεδομένων *D* (όλοι οι ακόλουθοι τύποι μπορούν εύκολα να ενσωματώσουν την μετρική του κόστους, η οποία κανονικοποιείται από το μέγεθος εισόδου).

Υποθέστε την ύπαρξη μίας συνάρτησης $cost_i(m)$ για κάθε δραστηριότητα *i* η οποία επιστρέφει μία πρόβλεψη για τον χρόνο που παίρνει στην δραστηριότητα *i* να επεξεργαστεί όλα τα δεδομένα που παίρνει σαν είσοδο έχοντας αναθέσει σε αυτή ένα ποσό μνήμης *m*. Αν και δεν υπάρχει κανένας συγκεκριμένος περιορισμός σχετικά με την φύση της συνάρτησης κόστους στο πλαίσιο της διαμόρφωσης του προβλήματος,

όπως θα παρουσιαστεί παρακάτω το πρόβλημα πολύ γρήγορα περιπλέκεται ακόμη και για απλές γραμμικές συναρτήσεις.

Θεωρείστε επίσης ένα δοσμένο ποσό μνήμης $M_{\max}$ που ανατίθεται σε διάφορες δραστηριότητες. Η μνήμη που ανατίθεται σε κάθε δραστηριότητα *i* δηλώνεται ως *mem(i)*. Μία πρακτική μετατροπή του προβλήματος είναι να αποφεύγεται ο υπολογισμός της μνήμης σε bytes, αλλά να υιοθετείται μία προσέγγιση κβαντικής κατανομής. Στην περίπτωσή μας η κατανομή της μνήμης σε μπλοκ ανάγεται σε κβάντα μεγέθους *B*. Με αυτό τον τρόπο αντί να δουλεύουμε με $M_{\max}$, έχουμε ένα μέγιστο πλήθος διαθέσιμων μπλοκ μνήμης *N*. Επίσης θεωρείται ότι όλες οι συναρτήσεις κόστους ορίζονται στο διάστημα $[0,N_{\text{dom}}]$.

Ο τυπικός ορισμός του προβλήματος εκφράζεται σαν μία παραλλαγή του knapsack προβλήματος. Πιο συγκεκριμένα πρέπει να γίνει ελαχιστοποίηση της ποσότητας $\sum_{i=1}^{k} \cos t_i(mem(i))$, έτσι ώστε $\sum_{i=1}^{k} mem(i) \le N$ και το *mem(i)>0*.

Για να αποφύγουμε την στέρηση (starvation) και να απλοποιήσουμε το σενάριο εκτέλεσης, δίνουμε αρχικά ένα ελάχιστο ποσό μνήμης σε κάθε δραστηριότητα (N_{min} μπλοκ). Αυτό μετριάζει την ανάγκη ελέγχου του ότι *mem(i)>0*, για κάθε δραστηριότητα *i*. Αυτή η απλή επέκταση προφανώς οδηγεί σε μία μικρότερη τιμή για το συνολικό ποσό μνήμης το οποίο δηλώνεται ως *N** και συνεπώς $N^* = N - k * N_{\min}$.

Πίνακας 3.1 Ορισμοί Συμβόλων

Σύμβολο	Περιγραφή
D	Το μέγεθος του συνόλου των δεδομένων που επεξεργάζεται κάθε δραστηριότητα
$cost_i(m)$	Ο χρόνος που κάνει κάθε δραστηριότητα *i* να επεξεργαστεί τα δεδομένα εισόδου δοσμένου ενός ποσού μνήμης *m*
M_{max}	Συνολικό ποσό της μνήμης
mem(i)	Ποσό της μνήμης που ανατίθεται σε κάθε δραστηριότητα *i*
N	Μέγιστο πλήθος των διαθέσιμων μπλοκ μνήμης
N_{dom}	Μέγιστη τιμή στο πεδίο ορισμού όλων των συναρτήσεων κόστους

3.2. Θεωρητικό Υπόβαθρο

Όπως αναφέρθηκε προηγουμένως το πρόβλημα που παρουσιάστηκε είναι πρακτικά μία παραλλαγή του "knapsack" προβλήματος. Έτσι το πρόβλημα διατυπώνεται ως εξής:

Ελαχιστοποίηση $\sum_{i=1}^{k} \cos t_i(mem(i))$

Έτσι ώστε, $\sum_{i=1}^{k} mem(i) \le N$ και το $mem(i)>0$

Μία πιθανή ενδιάμεση κατάσταση του προβλήματος είναι η εξής: $s^n = [m_0, m_1, ..., m_k]$ με:

- m_0, είναι το ποσό της μνήμης που σκόπιμα δεν χρησιμοποιείται (η χρησιμοποίηση του επιδεινώνει την συμπεριφορά του ETL γραφήματος ροής εργασιών, $cost_0(m)=0$, για κάθε *m*)
- τα υπόλοιπα m_i δηλώνουν το ποσό της μνήμης (σε μπλοκ) που ανατίθεται σε κάθε μονάδα *k*

- $n = \sum_{i=0}^{k} m_i$.

Μπορούμε να χρησιμοποιήσουμε δυναμικό προγραμματισμό για την επίλυση του προβλήματός μας. Έτσι κάθε φορά που έρχεται μία κατάσταση $s^n = [m_0, m_1, ..., m_k]$ οφείλεται σε μία από τις ακόλουθες πιθανές μεταβολές:

- $s^{n-1} = [m_0 - 1, m_1, ..., m_k]$, και η προσθήκη ενός καινούργιου μπλοκ μνήμης αποκλείει ($n\text{-}1 \leq N$) την παραγωγή επιπλέον κέρδους.
- $s^{n-1} = [m_0, m_1 - 1, ..., m_k]$, ένα καινούργιο μπλοκ προστίθεται στην πρώτη δραστηριότητα και το καινούργιο κόστος είναι: $\cos t(s^{n-1}) + \cos t_1(m_1) - \cos t_1(m_1 - 1)$
- …
- $s^n = [m_0, m_1, ..., m_k]$, ένα καινούργιο μπλοκ μνήμης προστίθεται στην *k-οστή* δραστηριότητα και το καινούργιο κόστος είναι: $\cos t(s^{n-1}) + \cos t_k(m_k) - \cos t_k(m_k - 1)$

Θα χρησιμοποιήσουμε το *K(w)* για να δηλώσουμε την “knapsack” συνάρτηση που επιστρέφει το ελάχιστο κόστος που μπορεί να επιτευχθεί όταν η μνήμη που ανατίθεται είναι *w* μπλοκ. Σε αυτή την περίπτωση η “knapsack” συνάρτηση είναι της μορφής:

$$K(w) = \min\{K(w-1) + \cos t_i(m_i) - \cos t_i(m_i - 1)\},\ \forall i = 0,1,...,k$$

Υπάρχει ένα απλός αλγόριθμος που υπολογίζει την παραπάνω “knapsack” συνάρτηση, όπως περιγράφεται στο Σχήμα4.1.

Πίνακας 3.2 Αλγόριθμος Υπολογισμούς Ανάθεσης Μνήμης σε ETL Μονάδες

```
K(0)=N_min;
For w=1 to N
   For m_0=1 to N_dom
      …
      For m_k=1 to N_dom
         K(w,m_0,…,m_k)=min(K(w-1,m_0-1,m_1,…,m_k),
              K(w-1,m_0,m_1-1,…,m_k)+cost(m_1)–cost(m_1-1),…,
              K(w-1,m_0,m_1,…,m_k-1)+cost(m_k)–cost(m_k-1)
))
```

Ο αλγόριθμος μπορεί εύκολα να τροποποιηθεί στις αρχικοποιήσεις των παραμέτρων, έτσι ώστε να διευκολύνει στις παραλλαγές της ανάθεσης της ελάχιστης μνήμης σε όλες τις δραστηριότητες του γραφήματος.

Η ιδιοσυγκρασία του προβλήματος έχει να κάνει με το γεγονός ότι το κόστος είναι παραμετρικό ως προς την κατανομή μνήμης. Αυτό πρακτικά μετατρέπει το πρόβλημά μας σε ένα παραμετρικό “knapsack” πρόβλημα, το οποίο έχει εκθετική πολυπλοκότητα σε σχέση με το μέγεθος των ροών και για απλές συναρτήσεις γραμμικές συναρτήσεις της μορφής *a*m+b*, με το *a* και το *b* να είναι οι συντελεστές της συνάρτησης [BuPf95].

3.3. Εξαντλητικός Αλγόριθμος Κατανομής Μνήμης

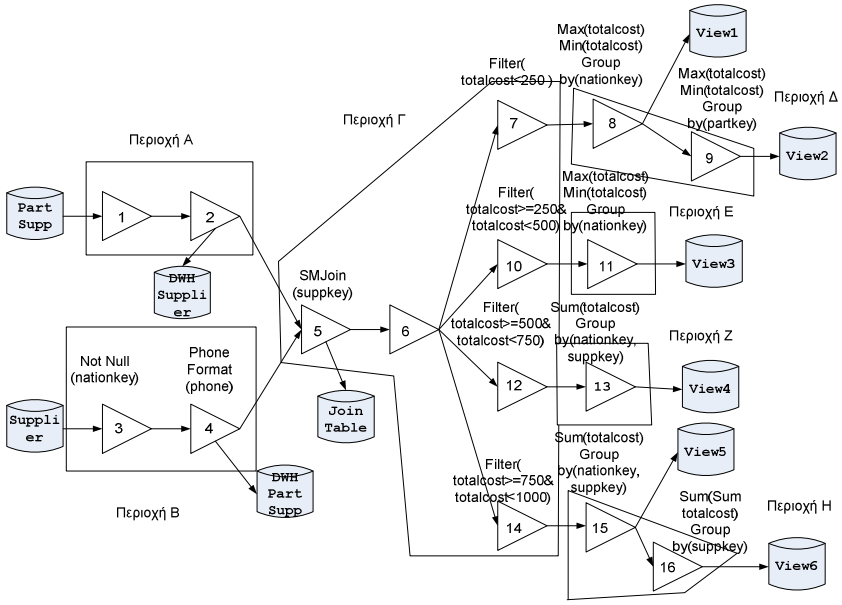
Ο προαναφερθείς αλγόριθμος Δυναμικού Προγραμματισμού έχει το μειονέκτημα ότι παράγει πολλές καταστάσεις (ή εναλλακτικά, χρησιμοποιεί πολύ μεγάλο χώρο μνήμης). Αν έχουμε k δραστηριότητες, τότε ο συνολικός αριθμός δραστηριοτήτων είναι k^n . Μπορούμε να χρησιμοποιήσουμε μια τεχνική «διαίρει και βασίλευε» στο πρόβλημά μας, παρατηρώντας ότι υπάρχουν σημεία μπλοκαρίσματος σε μια ροή εργασίας.

Συγκεκριμένα, ορίζουμε ως σημείο μπλοκαρίσματος (α) κάθε κόμβο που είναι αρχείο εγγραφής και (β) κάθε δραστηριότητα που χρειάζεται να καταναλώσει όλη την είσοδό της προτού αρχίσει να παράγει αποτελέσματα. Τυπικά παραδείγματα τέτοιων δραστηριοτήτων είναι οι δραστηριότητες (α) συνένωσης, (β) συνάθροισης, (γ) ταξινόμησης (στην συγκεκριμένη υλοποίηση, δε, όλες οι παραπάνω δραστηριότητες υλοποιούνται με αλγορίθμους που βασίζονται στην ταξινόμηση και κατά συνέπεια απαιτούν την προσωρινή εγγραφή της εισόδου και την συνεπακόλουθη εξωτερική ταξινόμησή της). Ορίζουμε ως περιοχή (stage) το μονοπάτι ανάμεσα σε δύο σημεία μπλοκαρίσματος. Ο εξαντλητικός αλγόριθμος που μπορεί να χρησιμοποιήσει εκμεταλλεύεται ότι κάθε περιοχή τερματίζει λίγο μετά τον τερματισμό της πιο αργής δραστηριότητάς της. Έτσι, το πρόβλημα μπορεί να εκφρασθεί ως «δοθείσης μιας ανάθεσης μνήμης σε όλες τις δραστηριότητες της ροής εργασίας, βρες την πιο αργή δραστηριότητα για κάθε περιοχή και εκτίμησε το συνολικό χρονικό κόστος ως το άθροισμα των επί μέρους περιοχών».

Για να βρεθούν οι περιοχές μέσα σε ένα γράφημα αρχικά εντοπίζουμε όλους του κόμβους του γραφήματος που είναι σημεία μπλοκαρίσματος. Στην συνέχεια, για κάθε μονοπάτι του γραφήματος που ξεκινάει με ένα κόμβο που είναι σημείο μπλοκαρίσματος εισάγεται κάθε κόμβος και ακμή στην περιοχή μέχρι να βρεθεί άλλος κόμβος που είναι σημείο μπλοκαρίσματος. Στην περιοχή εισάγεται και η τελευταία ακμή που ενώνει τον τελευταίο κόμβο με τον κόμβο που είναι σημείο μπλοκαρίσματος. Τέλος, συνενώνονται περιοχές που περιέχουν κοινά υπογραφήματα.

Για παράδειγμα στο Σχήμα 3.1 έχουμε τις εξής περιοχές: A={2,3}, B={3,4}. Γ={5,6,7,10,12,14}, Δ={8,9}, E={11}, Z={13}, H={15,16}. Έτσι, κάθε φορά παίρνουμε τον κάθε συνδυασμό μνήμης που μπορεί να υπάρξει ανάμεσα στις δραστηριότητες μίας περιοχής και κρατάμε την δραστηριότητα που καθυστερεί περισσότερο. Με αυτό τον τρόπο προκύπτει ο συνολικός χρόνος μίας περιοχής. Έπειτα, αθροίζονται οι χρόνοι όλων των περιοχών έτσι ώστε να πάρουμε τον συνολικό χρόνο του γραφήματος (Γραμμή 15) με την συγκεκριμένη ανάθεση μνήμης. Τέλος (Γραμμές 18-23) ελέγχουμε ένα η ανάθεση μνήμης που έχει γίνει δεν υπερβαίνει την συνολική διαθέσιμη μνήμης και εάν όχι ελέγχουμε για να δούμε εάν ο

χρόνος που μας έδωσε η συγκεκριμένη ανάθεση είναι ο ελάχιστος μέχρι εκείνη την στιγμή, εάν ναι κρατάμε την συγκεκριμένη ανάθεση και τον χρόνο που κάνει, αλλιώς την αγνοούμε.



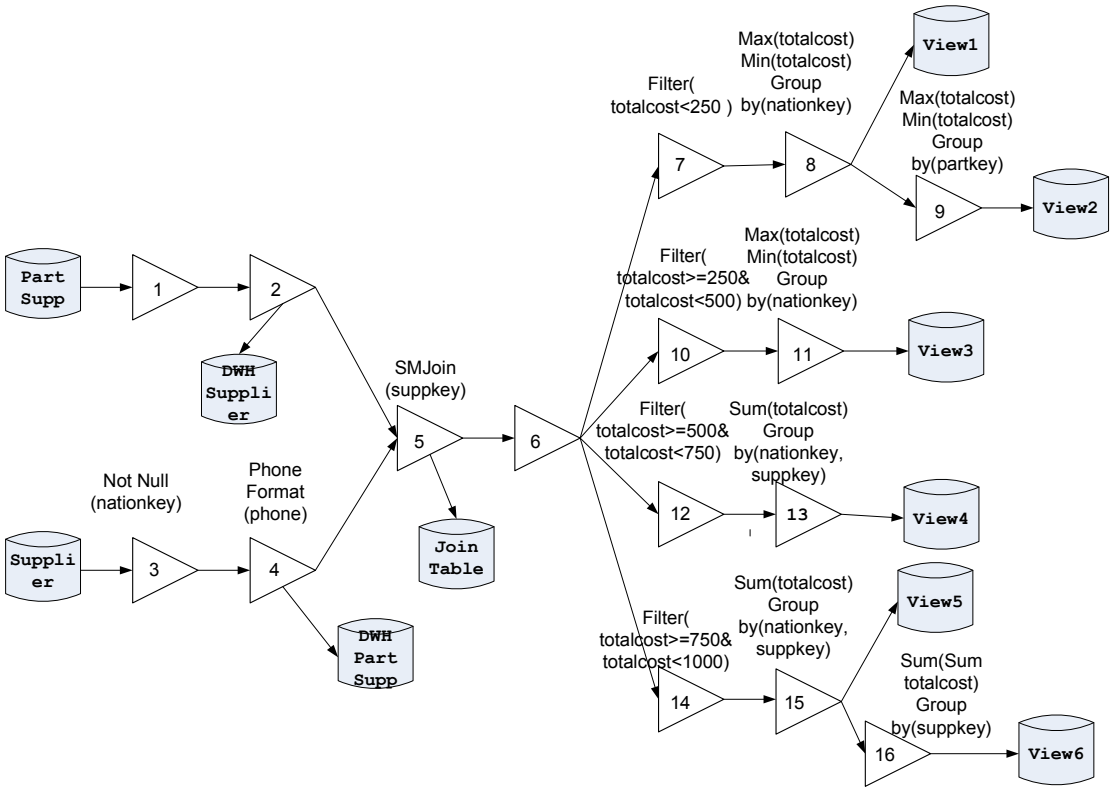
Σχήμα 3.1 Παράδειγμα Χωρισμού των Περιοχών

Πίνακας 3.3 Εξαντλητικός Αλγόριθμος

```
1.  time_min = MAX_value
2.  For i=1 to N
3.  {
4.     For k1=1 to Ndom
5.    {
6.        For k2=1 to Ndom
7.       {
8.       …
9.          For kn=1 to Ndom
10.         {
11.            time = 0
12.            For j=1 to #stages
13.            {
14.               time(m(stagej)) = max(time(m( ∀ki ∈ stagej ))
15.               total_time = total_time + time(m(stagej))
16.               sum_mem = m(stagej)
17.            }
18.            If sum_mem ≤ Mmax  && total_time < time_min Then
19.                  time_min = total_time
20. }…}
```

3.4. Αλγόριθμοι Κατανομής Μνήμης στις Λειτουργίες ενός ETL Γραφήματος

Υπάρχουν διάφοροι αλγόριθμοι οι οποίοι μπορούν να κατανείμουν την διαθέσιμη μνήμη ανάμεσα στις μονάδες ενός ETL γραφήματος. Οι αλγόριθμοι αυτοί είναι οι εξής: *Δίκαιος Αλγόριθμος*, *Αλγόριθμος Προτεραιότητας Συνενώσεων*, *Αλγόριθμος Προτεραιότητας Συναθροίσεων* και *Αλγόριθμος Προτεραιότητας Δραστηριοτήτων Εγγραφής*. Καθένας από αυτούς έχει τα πλεονεκτήματά του και τα μειονεκτήματά του. Παρακάτω παρουσιάζονται οι αλγόριθμοι αυτοί μέσα από ένα παράδειγμα ενός ETL σεναρίου (Σχήμα 3.2).



Σχήμα 3.2 Παράδειγμα ενός ETL Σεναρίου

3.4.1. Δίκαιος Αλγόριθμος

Ο *Δίκαιος Αλγόριθμος* είναι πολύ απλός στην υλοποίησή του. Το μόνο που έχει να κάνει είναι να διανέμει την προκαθορισμένη διαθέσιμη μνήμη ισότιμα σε όλες τις ακμές του γραφήματος. Δηλαδή, εάν είναι $M_{\max}$ η διαθέσιμη προκαθορισμένη μνήμη, N είναι το μέγιστο πλήθος των διαθέσιμων μπλοκ μνήμης (όπως έχει αναφερθεί προηγουμένως) και υπάρχουν k ακμές στο γράφημα, τότε $mem(i)= N/k$ για κάθε ακμή $i=0,1,..,k$.

Εάν εφαρμόσουμε τον παραπάνω αλγόριθμο στο παράδειγμα του Σχήματος 3.2 το οποίο έχει $k=26$ ακμές και με την υπόθεση ότι έχουμε $N=1000$ μπλοκ μνήμης διαθέσιμα, τότε έχουμε ότι $mem(i)=1000/26=38$ μπλοκ μνήμης για κάθε ακμή του σεναρίου (με $i=0,1,..,26$).

3.4.2. Αλγόριθμος Προτεραιότητας Συνενώσεων

Ο *Αλγόριθμος Προτεραιότητας Συνενώσεων* όπως το λέει και το όνομά του δίνει προτεραιότητα στις λειτουργίες του γραφήματος που είναι συνενώσεις. Δηλαδή, δίνει μεγαλύτερο ποσοστό από την προκαθορισμένη διαθέσιμη μνήμη στις ακμές του γραφήματος που έχουν στην είσοδό τους μία συνένωση παρά σε οποιαδήποτε άλλη ακμή.

Έτσι εάν $M_{\max}$ η διαθέσιμη προκαθορισμένη μνήμη, N είναι το μέγιστο πλήθος των διαθέσιμων μπλοκ μνήμης, x είναι το αρχικό ποσοστό μνήμης που μοιράζεται δίκαια σε όλες τις ακμές, k είναι το πλήθος των ακμών και j είναι το πλήθος των ακμών που έχουν στην είσοδο τους μία λειτουργία συνένωσης στο γράφημα, τότε διανέμεται δίκαια ένα ποσοστό της $M_{\max}$ σε όλες τις ακμές και το υπόλοιπο που περισσεύει διανέμεται ισότιμα στις ακμές που έχουν σαν είσοδο μία συνένωση. Επομένως οι ακμές που δεν έχουν στην είσοδο τους μία λειτουργία συνένωσης θα πάρουν συνολικά $(x*N)/k$ μπλοκ μνήμης, ενώ οι ακμές που έχουν στην είσοδό τους μία λειτουργία συνένωσης θα πάρουν $\frac{x*N}{k}+\frac{(1-x)*N}{j}$ μπλοκ μνήμης.

Κατά την εφαρμογή του παραπάνω αλγορίθμου στο παράδειγμα του Σχήματος 3.2 έχουμε ότι $k=26$ ακμές, $j=2$ (είναι οι ακμές που ενώνουν τον κόμβο 5 με τον κόμβο 6 και τον κόμβο 5 με τον κόμβο join_table), και αν υποθέσουμε $N=1000$ και $x=0.2$ (το ποσοστό που μοιράζεται δίκαια σε όλες τις ακμές του γραφήματος), τότε $mem(i)=\frac{0.2*1000}{26}=7$ μπλοκ μνήμης σε όλες τις ακμές εκτός των δύο που ενώνονται με την λειτουργία της συνένωσης και $mem(i)=\frac{0.2*1000}{26}+\frac{0.8*1000}{2}=407$ μπλοκ μνήμης στις δύο ακμές που αναφέρθηκαν προηγουμένως και ενώνονται με την λειτουργία της συνένωσης.

3.4.3. Αλγόριθμος Προτεραιότητας Συναθροίσεων

Παρόμοιος με τον *Αλγόριθμο Προτεραιότητας Συνενώσεων* είναι και ο *Αλγόριθμος Προτεραιότητας Συναθροίσεων* μόνο που σε αυτή την περίπτωση δίνουμε προτεραιότητα στους κόμβους του γραφήματος που είναι συναθροίσεις.

Έτσι διανέμεται δίκαια σε όλες τις ακμές του γραφήματος ένα ποσοστό της προκαθορισμένης μνήμης και το υπόλοιπο ποσοστό που περισσεύει διανέμεται δίκαια στις ακμές που έχουν στην είσοδό τους μία συναθροιστική λειτουργία. Επομένως εάν έχουμε $M_{\max}$ τη διαθέσιμη προκαθορισμένη μνήμη, N το μέγιστο πλήθος των διαθέσιμων μπλοκ μνήμης, x το αρχικό ποσοστό μνήμης που μοιράζεται δίκαια σε όλες τις ακμές, k το πλήθος των ακμών και j το πλήθος των ακμών που έχουν στην είσοδό τους μία συναθροιστική λειτουργία, τότε $(x*N)/k$ μπλοκ μνήμης θα διανεμηθούν στις ακμές του γραφήματος που δεν έχουν στην είσοδό τους μία λειτουργία συνάθροισης και $\frac{x*N}{k}+\frac{(1-x)*N}{j}$ μπλοκ μνήμης θα διανεμηθούν στις ακμές που έχουν στην είσοδό τους μία λειτουργία συνάθροισης.

Στο παράδειγμα του Σχήματος 3.2 όπου έχουμε *k=26* ακμές, *j=8* ακμές που έχουν στην είσοδό τους μία λειτουργία συνάθροισης και είναι οι εξής: 8→View1, 8→9, 9→View2, 11→View3, 13→View4, 15→View5, 15→16, 16→View6 και εάν υποθέσουμε *N=1000* και *x=0.2* (το ποσοστό που μοιράζεται δίκαια σε όλες τις ακμές του γράφου) τότε, εφαρμόζοντας τον *Αλγόριθμο Προτεραιότητας των Συναθροίσεων* η κατανομή της μνήμης ανάμεσα στις ακμές του γραφήματος είναι η εξής: $mem(i)=\frac{0.2*1000}{26}=7$ μπλοκ μνήμης για τις ακμές που δεν έχουν στην είσοδό τους μία λειτουργία συνάθροισης και $mem(i)=\frac{0.2*1000}{26}+\frac{0.8*1000}{8}=107$ μπλοκ μνήμης σε κάθε ακμή που έχει στην είσοδό της μία λειτουργία συνάθροισης.

3.4.4. Αλγόριθμος Προτεραιότητας Δραστηριοτήτων Εγγραφής

Ο *Αλγόριθμος Προτεραιότητας Δραστηριοτήτων Εγγραφής* κινείται στο ίδιο ύφος με τους προηγούμενους δύο αλγορίθμους, μόνο που αυτή την φορά δίνουμε προτεραιότητα στις ακμές που γράφουν τα δεδομένα που έχουν σε μία Αποθήκη Δεδομένων ή σε μία Όψη.

Επομένως, πάλι διανέμεται ένα ποσοστό της προκαθορισμένης διαθέσιμης μνήμης σε όλες τις ακμές δίκαια και έπειτα διανέμεται το ποσοστό της μνήμης που περισσεύει στις ακμές που καταλήγουν σε μία Αποθήκη Δεδομένων ή σε μία Όψη. Έτσι εάν έχουμε, $M_{\max}$ τη διαθέσιμη προκαθορισμένη μνήμη, N το μέγιστο πλήθος των διαθέσιμων μπλοκ μνήμης, x το αρχικό ποσοστό μνήμης που μοιράζεται δίκαια σε όλες τις ακμές, k το πλήθος των ακμών και j το πλήθος των ακμών που έχουν στην έξοδό τους ένα πίνακα, τότε $(x * N)/k$ μπλοκ μνήμης θα διανεμηθούν στις ακμές του γραφήματος που δεν καταλήγουν να γράψουν τα δεδομένα τους σε ένα πίνακα και $\frac{x*N}{k}+\frac{(1-x)*N}{j}$ μπλοκ μνήμης θα διανεμηθούν στις ακμές που έχουν στην έξοδό τους μία Αποθήκη Δεδομένων ή μία Όψη.

Εφαρμόζοντας τον *Αλγόριθμο Προτεραιότητας Δραστηριοτήτων Εγγραφής* στο παράδειγμα του Σχήματος 3.2 όπου έχουμε *k=26* ακμές, *j=9* ακμές που γράφουν τα δεδομένα τους σε ένα πίνακα και είναι οι εξής: 2→DWH_Supplier, 4→DWH_PartSupp, 5→Join_Table, 8→View1, 9→View2, 11→View3, 13→View4, 15→View5, 16→View6 και υποθέτοντας *N=1000* και *x=0.2* (το ποσοστό που μοιράζεται δίκαια σε όλες τις ακμές του γραφήματος) τότε, η κατανομή της μνήμης ανάμεσα στις ακμές του γραφήματος είναι η εξής: $mem(i)=\frac{0.2*1000}{26}=7$ μπλοκ μνήμης για τις ακμές που δεν γράφουν τα δεδομένα τους σε ένα πίνακα και $mem(i)=\frac{0.2*1000}{26}+\frac{0.8*1000}{9}=95$ μπλοκ μνήμης σε κάθε ακμή που γράφει τα δεδομένα της σε ένα πίνακα.

3.4.5. Αλγόριθμος Προτεραιότητας Ταξινομήσεων

Τέλος, έχουμε τον Αλγόριθμο Προτεραιότητας Ταξινομήσεων όπου σε αυτή την περίπτωση δίνουμε προτεραιότητα στις δραστηριότητες ταξινόμησης του γραφήματος. Αυτού του τύπου οι δραστηριότητες μπορούν να λειτουργούν ανεξάρτητα από άλλες ή μπορεί να είναι μέρος άλλων δραστηριοτήτων. Δραστηριότητες που χρησιμοποιούν ταξινόμηση σε ένα μέρος της λειτουργίας τους είναι η δραστηριότητα συνένωσης και η δραστηριότητα συνάθροισης.

Έτσι, σε αυτή την περίπτωση όπως και στις τρεις προηγούμενες (για τους διαφορετικούς αλγορίθμους) διανέμεται το αρχικό ποσοστό της προκαθορισμένης διαθέσιμης μνήμη δίκαια σε όλους τις ακμές του γραφήματος και το υπόλοιπο ποσοστό που περισσεύει στην συνέχεια διανέμεται δίκαια στις δραστηριότητες της ταξινόμησης ή στις δραστηριότητες που χρησιμοποιούν την ταξινόμηση που είναι κάθε φορά ενεργές.

Εφαρμόζοντας τον *Αλγόριθμο Προτεραιότητας Ταξινομήσεων* στο παράδειγμα του Σχήματος 3.2 όπου έχουμε *k=26* ακμές, και υποθέτοντας *N=1000* και *x=0.2* (το ποσοστό που μοιράζεται δίκαια σε όλες τις ακμές του γραφήματος) τότε, η κατανομή της μνήμης ανάμεσα στις ακμές του γραφήματος είναι η εξής: $mem(i) = \frac{0.2 * 1000}{26} = 7$ μπλοκ μνήμης για τις ακμές που δεν περιέχουν δραστηριότητα ταξινόμησης, ενώ όταν εκτελείται το η δραστηριότητα συνένωσης 5 οι ακμές 2→5 και 4→5 θα πάρουν το υπόλοιπο 80% της συνολικής διαθέσιμης μνήμης, στην συνέχεια όταν θα πάρουν αυτό το υπόλοιπο 80% της μνήμης οι ακμές 7→8, 10→11, 12→13, 14→15 και τέλος οι ακμές 8→9 και 15→16 θα πάρουν και πάλι το 80% της μνήμης που περισσεύει.

ΚΕΦΑΛΑΙΟ 4. ΜΕΘΟΔΟΛΟΓΙΑ ΠΕΙΡΑΜΑΤΩΝ

Στο παρόν κεφάλαιο παρουσιάζονται πειραματικές εκτιμήσεις, οι οποίες προσεγγίστηκαν χρησιμοποιώντας διαφορετικού είδους γραφήματα ροής εργασιών. Παρουσιάζεται ένας αριθμός πειραμάτων που διεξήχθησαν με σκοπό να υποστηρίξουν τα αποτελέσματα της θεωρητικής δουλειάς που έγινε και να καταδείξουν την αποδοτικότητα της προτεινόμενης μεθόδου.

4.1. Μετρικές και Παράμετροι

Οι μετρικές που μας ενδιαφέρουν είναι οι ακόλουθες:

Χρόνος Εκτέλεσης (Execution Time): είναι ο χρόνος που χρειάζεται για να εκτελεστεί το κάθε πείραμα. Ο χρόνος εκτέλεσης είναι το βασικό μέτρο για την απόδοση του συστήματος.

Οι παράμετροι που χρησιμοποιούνται για να μετρηθούν τα παραπάνω μέτρα είναι οι εξής:

- **Είδος Σεναρίου:** το είδος του σεναρίου επηρεάζει τα παραπάνω μέτρα. Εάν για παράδειγμα είναι γραμμικό θα έχει περισσότερα φίλτρα σε αντίθεση με το σενάριο δέντρου που θα έχει περισσότερες συνενώσεις, με αποτέλεσμα να επηρεάζεται ο χρόνος εκτέλεσης ενός πειράματος.
- **Μέγεθος Σεναρίου:** το πόσες δραστηριότητες έχει ένας γράφος ροής εργασιών.
- **Μέγεθος Δεδομένων (Data Size):** το πόσο μεγάλο ή μικρό είναι το μέγεθος των δεδομένων που μπαίνουν σαν είσοδο στις πηγές του σεναρίου.
- **Χωρητικότητα Μνήμης (Capacity):** το ποσοστό της μνήμης που δίνεται στο σενάριο για να εκτελεστεί σε σχέση με την μνήμη του υπολογιστή που εκτελεί το σενάριο.
- **Ποσοστό Μνήμης Ανά Δραστηριότητα:** το ποσοστό μνήμης που δίνεται σε κάθε δραστηριότητα του γράφου σε σχέση με την αρχική χωρητικότητα μνήμης που δόθηκε στο σενάριο.
- **Επιλεκτικότητα (Selectivity):** αναφέρεται στο ποσοστό των πλειάδων που επιστρέφονται από μία δραστηριότητα. Μία δραστηριότητα θεωρείται ότι έχει χαμηλή επιλεκτικότητα όταν επιστρέφει ένα μικρό αριθμό πλειάδων. Ενώ όταν επιστρέφει ένα μεγάλο ποσοστό των πλειάδων που μπήκαν σαν είσοδο θεωρείται ότι η δραστηριότητα έχει υψηλή επιλεκτικότητα. Αναφέρεται κυρίως στις δραστηριότητες που είναι φίλτρα.

Στα πειράματα που θα παρουσιαστούν παραμένουν σταθεροί οι εξής όροι:

- **Αλγόριθμος Εκτέλεσης:** όλα τα πειράματα εκτελούνται με τον Minimum Cost αλγόριθμο δρομολόγησης.

- **Χρονικό Κενό (Time Slot):** το χρονικό διάστημα που μεσολαβεί ανάμεσα από τον τερματισμό μίας δραστηριότητας μέχρι να ξεκινήσει να εκτελείται η επόμενη. Στο τέλος αυτού του χρόνου ο δρομολογητής επιλέγει ποια θα είναι η επόμενη δραστηριότητα που θα εκτελεστεί.
- **Μέγεθος Πακέτων (Row Pack Size):** το επιλεγμένο μέγεθος των πακέτων, το οποίο μετριέται σε πλήθος πλειάδων.

Στα πειράματα που διεξήχθησαν μετρήθηκε ο χρόνος εκτέλεσης τους σε δευτερόλεπτα (άξονας y) σε συνάρτηση με το ποσοστό χρήσης της συνολικής μνήμης που είναι διαθέσιμη (άξονας x), δηλαδή την χωρητικότητα μνήμης όπως ορίστηκε προηγουμένως.

Τέλος, όλα τα πειράματα διεξήχθησαν σε Ηλεκτρονικό Υπολογιστή με τα εξής χαρακτηριστικά: Επεξεργαστής Intel Dual Core 3.2Ghz, Σκληρός Δίσκος 250GB, Κύρια Μνήμη 1GB.

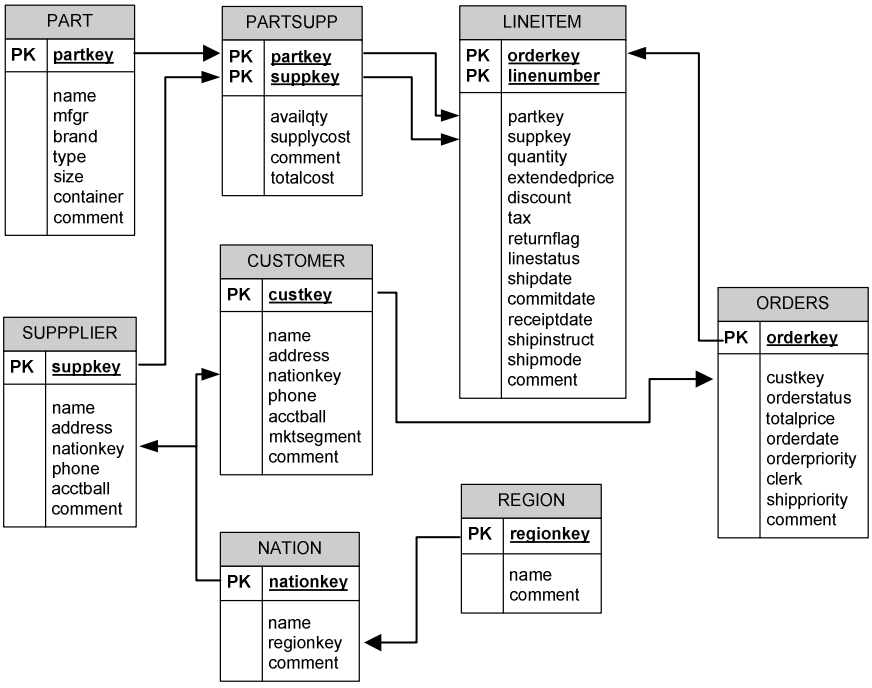
4.2. Σύνολο Δεδομένων

Το σύνολο δεδομένων που χρησιμοποιείται στα πειράματα είναι το TPC Benchmark™H (TPC-H) [Tpch09]. Το TPC-H είναι ένα μέτρο σύγκρισης στην υποστήριξη της λήψης αποφάσεων. Αποτελείται από ένα σύνολο επιχειρηματικών προσανατολισμένων προς αυτό το σκοπό ερωτημάτων και ταυτόχρονη τροποποίηση των δεδομένων. Οι ερωτήσεις και τα δεδομένα που υπάρχουν στην βάση δεδομένων επιλέχθηκαν έτσι ώστε να έχουν μεγάλο εύρος συνάφειας. Το TPC-H απεικονίζει συστήματα λήψης αποφάσεων που εξετάζουν μεγάλου μεγέθους δεδομένα, εκτελούν ερωτήσεις υψηλού βαθμού πολυπλοκότητας και δίνουν απαντήσεις σε κρίσιμες ερωτήσεις.

Το σχεσιακό σχήμα των δεδομένων του TPC-H αποτελείται από 8 ξεχωριστούς πίνακες όπως φαίνεται στο Σχήμα 4.1, το οποίο μπορεί να παραχθεί σε διάφορα

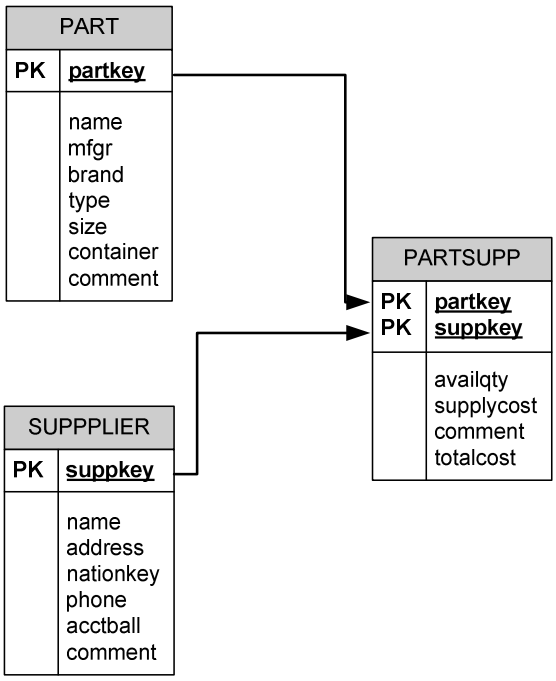
μεγέθη μέχρι και τα 100TB. Το σύνολο των δεδομένων του TPC-H περιγράφει ένα σύστημα πωλήσεων. Η πληροφορία που διατηρείται είναι για τα τμήματα και τους προμηθευτές, ενώ επίσης διατηρείται πληροφορία για τις παραγγελίες που κάνουν οι προμηθευτές και κάποια δεδομένα για τους πελάτες. Τα σενάρια που χρησιμοποιούνται στα πειράματα καθαρίζουν και μετασχηματίζουν τα δεδομένα που εμπεριέχονται στους πίνακες που δίνονται σαν είσοδο στο σενάριο και αποθηκεύονται στις επιθυμητές αποθήκες δεδομένων. Τα σχήματα των Αποθηκών Δεδομένων εξαρτώνται από τους πίνακες:

- **PART**(s_partkey, name, mfgr, brand, type, size, container, comment)
- **SUPPLIER**(s_suppkey, name, address, nationkey, phone, actball, comment, totalcost)
- **PARTSUPP**(s_partkey, s_suppkey, availqty, supplycost, comment)
- **CUSTOMER**(s_custkey, name, address, nationkey, phone, acctball, mktsegment, comment)
- **ORDER**(s_orderkey, custkey, orderstatus, totalpice, orderdate, orderpriority, clerk, shippriority, comment)
- **LINEITEM**(s_orderkey, partkey, suppkey, linenumber, quantity, extendedprice, discount, tax, returnflag, linestatus, shipdate, commitdate, receiptdate, shipinstruct, shipmode, comment, profit)



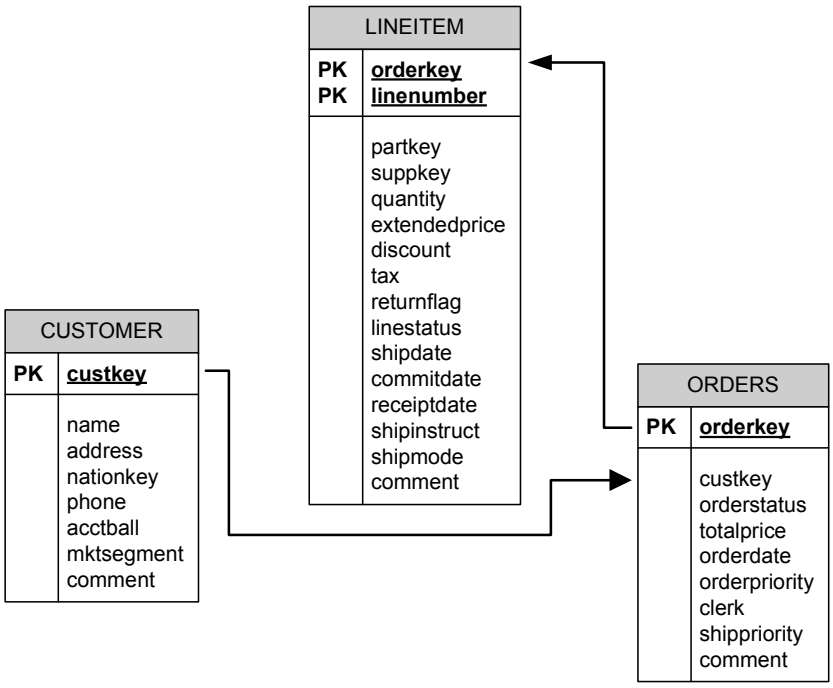
Σχήμα 4.1 Σχεσιακό Σχήμα TPC-H

Οι πηγές των δεδομένων που χρησιμοποιούνται στα σενάριά μας είναι δύο ειδών: τα σημεία αποθήκευσης και τα σημεία πώλησης. Κάθε σημείο αποθήκευσης διατηρεί δεδομένα για τους προμηθευτές και τα τμήματά τους, ενώ κάθε σημείο πώλησης διατηρεί δεδομένα για τους πελάτες και τις παραγγελίες τους. Το σχήμα του σημείου αποθήκευσης περιέχει τον πίνακα **PART**(s_partkey, name, mfgr, brand, type, size, container, comment), τον πίνακα **SUPPLIER**(s_suppkey, name, address, nationkey, phone, actball, comment, totalcost) και τον πίνακα **PARTSUPP**(s_partkey, s_suppkey, availqty, supplycost, comment) που σχετίζεται με τους άλλους δύο (Σχήμα 4.2).



Σχήμα 4.2 Σχεσιακό Σχήμα Σημείου Αποθήκευσης

Από την άλλη μεριά το σχήμα του σημείου πώλησης περιέχει τον πίνακα **CUSTOMER**(s_custkey, name, address, nationkey, phone, acctball, mktsegment, comment) τον πίνακα **ORDER**(s_orderkey, custkey, orderstatus, totalpice, orderdate, orderpriority, clerk, shippriority, comment) και τον πίνακα **LINEITEM**(s_orderkey, partkey, suppkey, linenumber, quantity, extendedprice, discount, tax, returnflag, linestatus, shipdate, commitdate, receiptdate, shipinstruct, shipmode, comment, profit) (Σχήμα 4.3)



Σχήμα 4.3 Σχεσιακό Σχήμα Σημείου Πώλησης

4.3. ETL Σενάρια

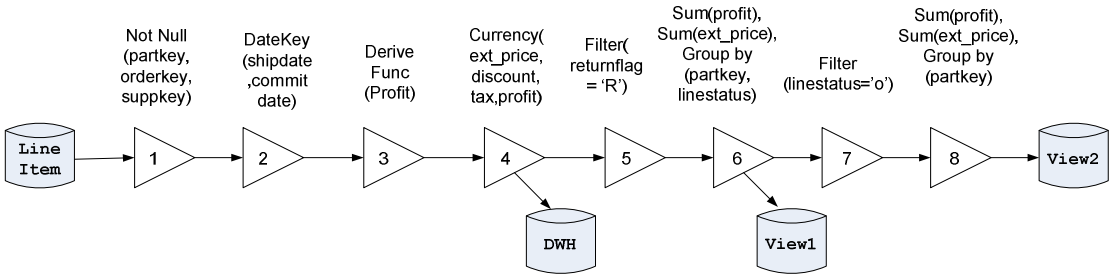
Τα πειράματα που έγιναν περιέχουν διάφορους τύπους γράφων ροής εργασιών, όπως αυτοί έχουν παρουσιαστεί με λεπτομέρειες στο [VKTS07]. Τα σενάρια που παρουσιάζονται παρακάτω χρησιμοποιήθηκαν για την αποτίμηση του συστήματός μας.

4.3.1. Γραμμικό Σενάριο (Line Scenario)

Το Γραμμικό Σενάριο είναι μία ακολουθία από δραστηριότητες που ξεκινάει και τελειώνει με ένα σύνολο εγγραφών, έτσι ώστε όλες οι δραστηριότητες έχουν ακριβώς μία είσοδο και μία έξοδο. Περιέχουν ένα πλήθος από φίλτρα, μετασχηματισμούς και συναθροίσεις σε ένα απλό πίνακα. Αυτού του τύπου τα σενάρια χρησιμοποιούνται για να φιλτράρουν τα δεδομένα του πίνακα εισόδου και να εγγυηθούν ότι ικανοποιούνται όλοι οι περιορισμοί στα δεδομένα της Αποθήκης Δεδομένων. Το σενάριο που

προτείνεται είναι αυτό που φαίνεται παρακάτω στο Σχήμα 4.4 και χρησιμοποιεί σαν πίνακα εισόδου τον πίνακα “LINEITEM”:

1. Αρχικά, ελέγχουμε τα πεδία “partkey”, “orderkey”, “suppkey” για NULL τιμές.
2. Στην συνέχεια, μετατρέπουμε τις ημερομηνίες στα πεδία “shipdate” και “commitdate” σε ένα αναγνωριστικό ημερομηνίας, το οποίο είναι μοναδικό για κάθε ημερομηνία.
3. Στην τρίτη δραστηριότητα υπολογίζεται η τιμή για το πεδίο “profit” ως εξής: η τιμή του πεδίου “extendedprice” αφαιρώντας τις τιμές των πεδίων “tax” και “discount”.
4. Έπειτα, στην τέταρτη δραστηριότητα αλλάζεται η μονάδα μέτρησης των πεδίων “extendedprice”, “profit”, “tax”, “discount”. Τα αποτελέσματα αυτής της δραστηριότητας αποθηκεύονται σε μία αποθήκη δεδομένων.
5. Αυτή η δραστηριότητα διατηρεί μόνο εκείνα τα δεδομένα που έχουν τιμή ίση με ‘R’ στο πεδίο “returnflag”.
6. Στην συνέχεια, υπάρχει μία συναθροιστική δραστηριότητα, η οποία αθροίζει τα τις τιμές των πεδίων “extendedprice” και “profit” ομαδοποιημένα με βάση τα πεδία “partkey” και “linestatus”. Τα αποτελέσματα αυτής της συνάθροισης αποθηκεύονται και σε μία όψη.
7. Η έβδομη δραστηριότητα διατηρεί μόνο τις πλειάδες που έχουν τιμή ‘o’ στο πεδίο “linestatus”.
8. Τέλος έχουμε μία ακόμη συναθροιστική δραστηριότητα, η οποία αθροίζει τα πεδία “profit” και “extendedprice” ομαδοποιημένα με βάση το πεδίο “partkey” και τα αποτελέσματα τα αποθηκεύει εκ’ νέου σε μία καινούργια όψη.



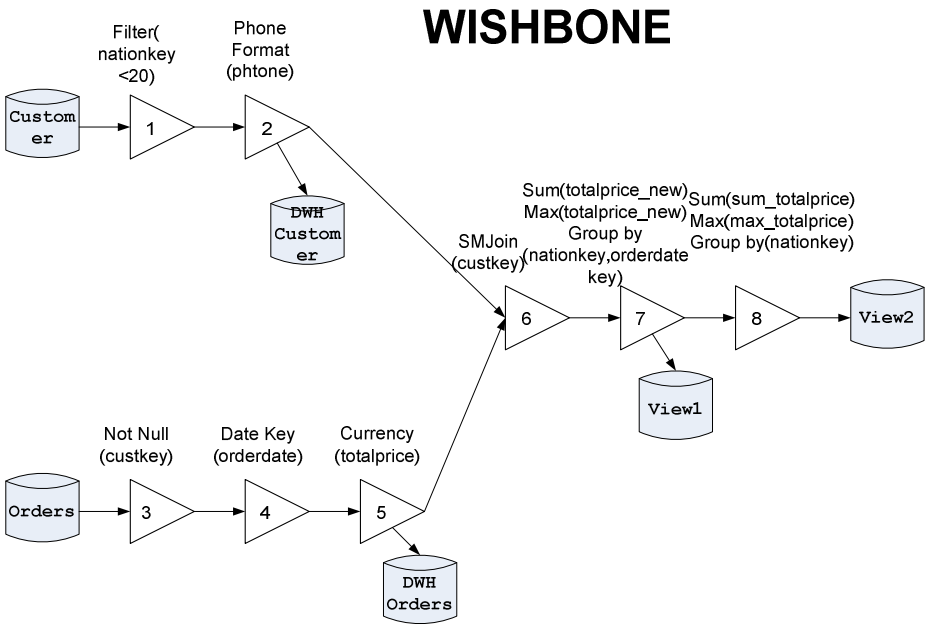
Σχήμα 4.4 Γραμμικό Σενάριο

4.3.2. Σενάριο Σύζευξης Εισερχόμενων Υποροών (Wishbone Scenario)

Στο Σενάριο Σύζευξης Εισερχόμενων Υποροών συνενώνονται δύο παράλληλα Γραμμικά Σενάρια σε ένα. Το σενάριο αυτό χρησιμοποιείται στις περιπτώσεις που θέλουμε να συνενώνουμε δύο πίνακες και να τους αποθηκεύσουμε στο τέλος της μέρας σε μία Αποθήκη Δεδομένων. Το σενάριο που προτείνεται είναι παρουσιάζεται στο Σχήμα 4.5 και χρησιμοποιεί σαν πίνακες εισόδου τους πίνακες “CUSTOMERS” και “ORDERS”:

1. Αρχικά, φιλτράρονται τα δεδομένα του πίνακα CUSTOMER και διατηρούνται μόνο αυτά που έχουν τιμή στο πεδίο “nationkey” μικρότερη από 20.
2. Έπειτα έχουμε μετατροπή του πεδίου “phone” που διατηρεί τηλεφωνικούς αριθμούς αντικαθιστώντας τον χαρακτήρα ‘+’ με δύο μηδενικά και τα αποτελέσματα αυτού του μετασχηματισμού αποθηκεύονται σε μία αποθήκη δεδομένων.
3. Η τρίτη δραστηριότητα διατηρεί τα δεδομένα που δεν έχουν NULL τιμές στο πεδίο “custkey” του πίνακα “ORDERS”.
4. Η δραστηριότητα αυτή εφαρμόζεται πάνω στο πεδίο “orderdate”.
5. Στην συνέχεια, εφαρμόζεται αλλαγή νομίσματος πάνω στο πεδίο “totalprice” και τα αποτελέσματά αυτής της δραστηριότητας αποθηκεύονται σε μία αποθήκη δεδομένων.

6. Η έκτη δραστηριότητα συνενώνει τους δύο πίνακες με βάση το πεδίο “custkey”.
7. Έπειτα, στα αποτελέσματα της συνένωσης εφαρμόζεται μία συναθροιστική δραστηριότητα, στην οποία υπολογίζονται το άθροισμα των δεδομένων του και το μέγιστο πεδίου “totalprice” ομαδοποιημένα με βάση τα πεδία “nationkey” και “orderdatekey” και τα αποτελέσματα αποθηκεύονται σε μία όψη.
8. Τέλος, εφαρμόζεται εκ’ νέου μία συναθροιστική δραστηριότητα η οποία υπολογίζει το άθροισμα και το μέγιστο του πεδίου “totalprice” ομάδοποιημένα με βάση το πεδίο “nationkey” και τα αποτελέσματα αποθηκεύονται σε μία νέα όψη.

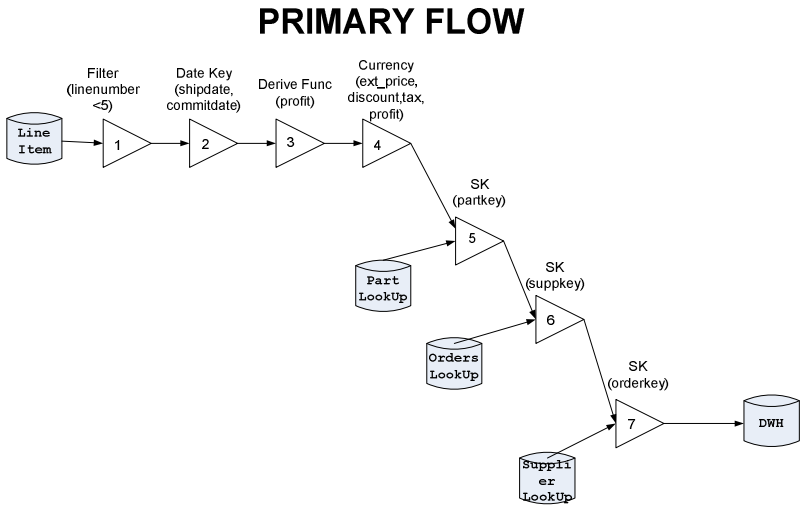


Σχήμα 4.5 Σενάριο Σύζευξης Εισερχόμενων Υπορροών

4.3.3. Σενάριο Πρωτεύουσας Ροής (Primary Flow Scenario)

Το Σενάριο Πρωτεύουσας Ροής είναι ένα σύνηθες σενάριο στις περιπτώσεις όπου ο γράφος περιέχει υποκατάστατα κλειδιών. Το σενάριο που προτείνεται φαίνεται στο Σχήμα 4.6 και χρησιμοποιεί σαν πίνακα εισόδου τον πίνακα “LINEITEM”:

1. Αρχικά, φιλτράρονται τα δεδομένα και διατηρούνται μόνο εκείνα που έχουν τιμή μικρότερη από 5 στο πεδίο “linenumber”.
2. Στην συνέχεια, μετατρέπουμε τις ημερομηνίες στα πεδία “shipdate” και “commitdate” σε ένα αναγνωριστικό ημερομηνίας, το οποίο είναι μοναδικό για κάθε ημερομηνία.
3. Στην τρίτη δραστηριότητα υπολογίζεται η τιμή για το πεδίο “profit” ως εξής: η τιμή του πεδίου “extendedprice” αφαιρώντας τις τιμές των πεδίων “tax” και “discount”.
4. Στην τέταρτη δραστηριότητα αλλάζεται η μονάδα μέτρησης των πεδίων “extendedprice”, “profit”, “tax”, “discount”.
5. Έπειτα, εφαρμόζεται το υποκατάστατο κλειδί στο πεδίο “partkey”.
6. Στην έκτη δραστηριότητα εφαρμόζεται το υποκατάστατο κλειδί στο πεδίο “suppkey”.
7. Τέλος, εφαρμόζεται εκ’ νέου το υποκατάστατο κλειδί στο πεδίο “orderkey” και αποθηκεύονται τα αποτελέσματα σε μία αποθήκη δεδομένων.



Σχήμα 4.6 Σενάριο Πρωτεύουσας Ροής

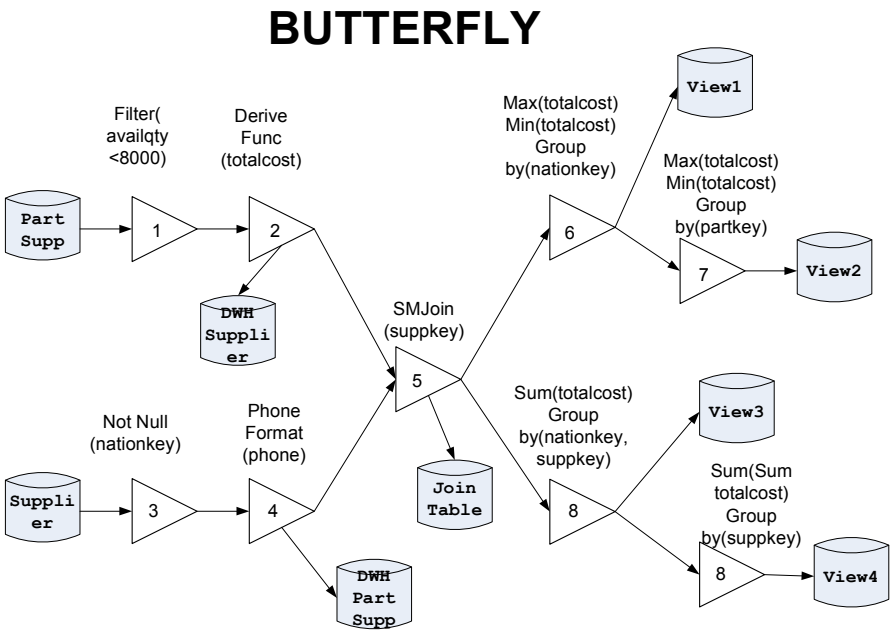
4.3.4. Σενάριο Πεταλούδας (Butterfly Scenario)

Το Σενάριο της Πεταλούδας χωρίζεται σε δύο μέρη, στο δεξί και στο αριστερό φτερό. Στο δεξί φτερό συνενώνονται δύο ή περισσότεροι πίνακες εισόδου, ενώ στο αριστερό φτερό εφαρμόζονται οι επιθυμητές συναθροιστικές δραστηριότητες αποθηκεύοντας τα αποτελέσματα τους στις αντίστοιχες όψεις. Το σενάριο που προτείνεται χρησιμοποιεί σαν πίνακες εισόδου τους πίνακες “PARTSUPP” και “SUPPLIER” και παρουσιάζεται στο Σχήμα 4.7:

1. Η πρώτη δραστηριότητα φιλτράρει τα δεδομένα και διατηρεί μόνο εκείνα που έχουν τιμή μικρότερη από 8000 στο πεδίο “availqty”.
2. Έπειτα, υπολογίζεται η τιμή για το πεδίο “totalcost” και τα αποτελέσματά της αποθηκεύονται σε μία αποθήκη δεδομένων.
3. Στην τρίτη δραστηριότητα φιλτράρονται τα δεδομένα που προέρχονται από τον πίνακα “SUPPLIER” και διατηρούνται μόνο αυτά που δεν έχουν NULL τιμή στο πεδίο nationkey.
4. Στην συνέχεια, γίνεται μετατροπή του πεδίου “phone” που διατηρεί τηλεφωνικούς αριθμούς αντικαθιστώντας τον χαρακτήρα ‘+’ με δύο μηδενικά και τα αποτελέσματα αυτού του μετασχηματισμού αποθηκεύονται σε μία νέα αποθήκη δεδομένων.
5. Στην πέμπτη δραστηριότητα γίνεται συνένωση των αποτελεσμάτων των δραστηριοτήτων 2 και 4, με βάση το πεδίο “suppkey” και τα αποτελέσματά της αποθηκεύονται σε μία αποθήκη δεδομένων.
6. Στα αποτελέσματα της συνένωσης εφαρμόζεται μία συναθροιστική δραστηριότητα, όπου υπολογίζεται το μέγιστο και το ελάχιστο του πεδίου “totalcost” ομαδοποιημένο με βάση το πεδίο “nationkey” και τα αποτελέσματά του αποθηκεύονται σε μία όψη (View1).
7. Στην συνέχεια εφαρμόζεται εκ’ νέου μία συναθροιστική δραστηριότητα, όπου υπολογίζεται το μέγιστο και το ελάχιστο του πεδίου “totalcost” ομαδοποιημένο με βάση το πεδίο “partkey” και αποθηκεύονται σε μία νέα όψη (View2).
8. Η δραστηριότητα οκτώ εφαρμόζει μία συναθροιστική δραστηριότητα στα αποτελέσματα της συνένωσης (δραστηριότητα 5) υπολογίζοντας το άθροισμα των τιμών στο πεδίο “totalcost” ομαδοποιώντας τα με βάση τα πεδία

"nationkey" και "suppkey" και τα αποτελέσματα αυτής αποθηκεύονται σε μία όψη (View3).

9. Τέλος, εφαρμόζεται μία ακόμη συναθροιστική δραστηριότητα, η οποία υπολογίζει το άθροισμα του "totalcost" που προέκυψε από την προηγούμενη δραστηριότητα ομαδοποιημένη βάση του πεδίου "suppkey" και αποθηκεύει τα αποτελέσματα της στην όψη View4.



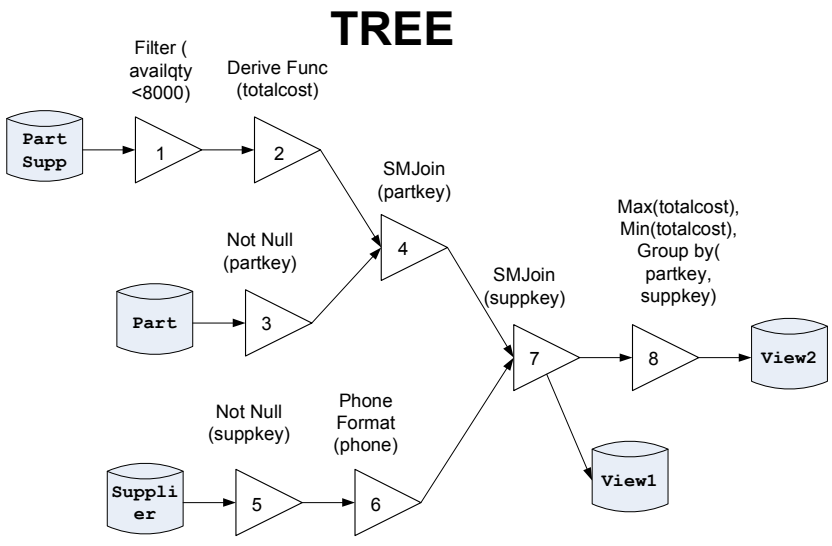
Σχήμα 4.7 Σενάριο Πεταλούδας

4.3.5. Σενάριο Δένδρου (Tree Scenario)

Το Σενάριο Δένδρου συνενώνει πολλούς πίνακες εισόδου και εφαρμόζει έπειτα συναθροιστικές δραστηριότητες στο αποτέλεσμά τους. Οι συνενώσεις μπορούν να γίνουν ετερογενείς σχέσεις που τα περιεχόμενά τους συνδυάζονται. Στην περίπτωσή μας προτείνεται το σενάριο που φαίνεται στο Σχήμα 4.8 που χρησιμοποιεί σαν πίνακες εισόδου τους πίνακες "PARTSUPP", "PART" και "SUPPLIER":

1. Ξεκινώντας φιλτράρονται τα δεδομένα του πίνακα εισόδου "PARTSUPP" και διατηρούνται αυτά που έχουν τιμή μικρότερη του 8000 στο πεδίο "availqty".
2. Έπειτα, υπολογίζεται η τιμή για το πεδίο "totalcost".

3. Η δραστηριότητα 3 διατηρεί μόνο εκείνα τα δεδομένα του πίνακα εισόδου “PART” που έχουν τιμή διαφορετική του NULL στο πεδίο “partkey”.
4. Στην συνέχεια, η δραστηριότητα 4 συνενώνει τα αποτελέσματα των δραστηριοτήτων 2 και 3 με βάση το πεδίο “partkey”.
5. Η δραστηριότητα 5 φιλτράρει τα δεδομένα που προέρχονται από τον πίνακα εισόδου “SUPPLIER” και διατηρεί μόνο εκείνα που έχουν διαφορετική τιμή από NULL στο πεδίο “suppkey”.
6. Στην δραστηριότητα 6 γίνεται μετατροπή του πεδίου “phone” που διατηρεί τηλεφωνικούς αριθμούς αντικαθιστώντας τον χαρακτήρα ‘+’ με δύο μηδενικά.
7. Έπειτα, συνενώνονται (δραστηριότητα 7) τα δεδομένα που προέκυψαν από τις δραστηριότητες 4 και 6 βάση του πεδίου “suppkey” και τα αποτελέσματα αποθηκεύονται σε μία αποθήκη δεδομένων.
8. Τέλος, εφαρμόζεται μία συναθροιστική δραστηριότητα, η οποία υπολογίζει το μέγιστο και το ελάχιστο του πεδίου “totalcost” ομαδοποιημένα με βάση τα πεδία “partkey” και “suppkey” και αποθηκεύει τα αποτελέσματα στην όψη View2.



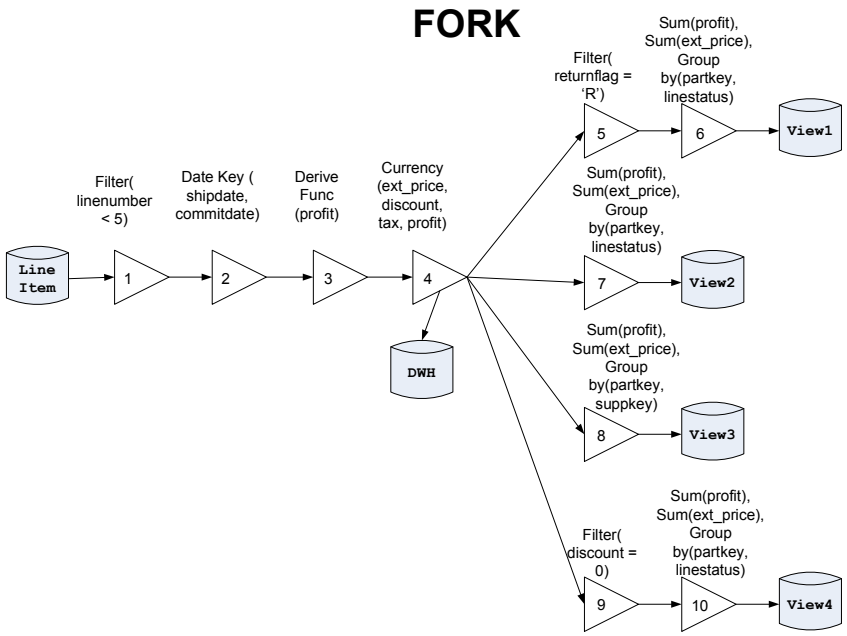
Σχήμα 4.8 Δενδρικό Σενάριο

4.3.6. Σενάριο Πολλών Εκροών (Fork Scenario)

Το Σενάριο Πολλών Εκροών αποτελείται από ένα σύνολο συναθροιστικών δραστηριοτήτων πάνω σε ένα απλό πίνακα εισόδου. Αρχικά ο πίνακας εισόδου καθαρίζεται όπως γίνεται σε ένα Γραμμικό Σενάριο και το αποτέλεσμα που βγαίνει χρησιμοποιείται για να δημιουργηθούν ένα σύνολο από όψεις. Στην περίπτωσή μας προτείνεται το εξής σενάριο όπως αυτό παρουσιάζεται στο Σχήμα 4.9 και χρησιμοποιεί σαν πίνακα εισόδου τον πίνακα “LINEITEM”:

1. Αρχικά, φιλτράρουμε τα δεδομένα του πίνακα εισόδου και διατηρούνται μόνο εκείνα που έχουν τιμή μικρότερη του 5 στο πεδίο “linenumber”.
2. Στην συνέχεια, μετατρέπουμε τις ημερομηνίες στα πεδία “shipdate” και “commitdate” σε ένα αναγνωριστικό ημερομηνίας, το οποίο είναι μοναδικό για κάθε ημερομηνία.
3. Στην τρίτη δραστηριότητα υπολογίζεται η τιμή για το πεδίο “profit” ως εξής: η τιμή του πεδίου “extendedprice” αφαιρώντας τις τιμές των πεδίων “tax” και “discount”.
4. Στην τέταρτη δραστηριότητα αλλάζεται η μονάδα μέτρησης των πεδίων “extendedprice”, “profit”, “tax”, “discount”. Τα αποτελέσματα αυτής της δραστηριότητας αποθηκεύονται σε μία αποθήκη δεδομένων.
5. Έπειτα, φιλτράρονται τα δεδομένα που προέρχονται από την δραστηριότητα 4 και διατηρούνται μόνο αυτά που έχουν τιμή ίση με ‘R’ στο πεδίο “returnflag” (δραστηριότητα 5).
6. Εν’ συνεχεία, τα αποτελέσματα της προηγούμενης δραστηριότητας περνούν από μία συναθροιστική δραστηριότητα, στην οποία υπολογίζονται το άθροισμα των τιμών στα πεδία “profit” και “extended_price” ομαδοποιημένα με βάση τα πεδία “partkey” και “linestatus”, που αποθηκεύονται στην όψη View1.
7. Η δραστηριότητα 7 παίρνει σαν είσοδο τα αποτελέσματα της δραστηριότητας 4 και υπολογίζει το άθροισμα των τιμών στα πεδία “profit” και “extended_price” ομαδοποιημένα με βάση τα πεδία “partkey” και “linestatus” και αποθηκεύονται στην όψη View2.

8. Ομοίως με την δραστηριότητα 7, η δραστηριότητα 8 υπολογίζει το άθροισμα των τιμών στα πεδία “profit” και “extended_price” ομαδοποιημένα με βάση τα πεδία “partkey” και “suppkey” και αποθηκεύει το αποτέλεσμα στην όψη View3.
9. Η δραστηριότητα 9 παίρνει σαν είσοδο τα αποτελέσματα της δραστηριότητας 4 και διατηρεί όσα έχουν τιμή ίση με το ‘0’ στο πεδίο “discount”.
10. Τέλος, η δραστηριότητα 10 υπολογίζει το άθροισμα των τιμών στα πεδία “profit” και “extended_price” ομαδοποιημένα με βάση τα πεδία “partkey” και “linestatus” και αποθηκεύει το αποτέλεσμα στην όψη View4.



Σχήμα 4.9 Σενάριο Πολλών Εκροών

4.4. Συγκριτική Αξιολόγηση Στοιχειωδών Σεναρίων

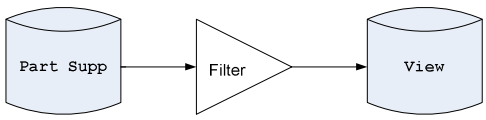
Όπως έχει αναφερθεί στο Κεφάλαιο 3 σκοπός μας είναι να ελαχιστοποιήσουμε το άθροισμα των συναρτήσεων κόστους κάθε δραστηριότητας συναρτήσει του ποσού μνήμης που διανέμεται σε κάθε μία από αυτές. Η συνάρτηση κόστους κάθε

δραστηριότητας ορίζει τον χρόνο που κάνει κάθε δραστηριότητα για να επεξεργαστεί τα δεδομένα εισόδου δοσμένου ενός συγκεκριμένου ποσού μνήμης. Γι' αυτό τον λόγω θέλουμε να παρατηρήσουμε την συμπεριφορά της κάθε δραστηριότητας ξεχωριστά δίνοντας της ολοένα και περισσότερη μνήμη. Με άλλα λόγια, θέλουμε να δούμε μία συγκριτική αξιολόγηση των διαφόρων δραστηριοτήτων που χρησιμοποιούνται στα σενάριά μας όταν αυτές βρίσκονται μόνες τους. Έτσι ώστε να χρησιμοποιηθούν αυτά τα αποτελέσματα για να έχουμε πλήρη γνώση της συμπεριφοράς των δραστηριοτήτων αυτών σε ολοκληρωμένα σενάρια και να μπορούν εξηγηθούν τα όποια αποτελέσματα μας δώσουν.

Γι' αυτό τον λόγο έγιναν τα παρακάτω πειράματα που παρουσιάζουν αυτή ακριβώς την συμπεριφορά των δραστηριοτήτων του φίλτρου, της συνένωσης και της συνάθροισης σε διαφορετικού μεγέθους δεδομένα κάθε φορά με παράγοντες κλιμάκωσης ίσο με 0.5 και 1.0.

4.4.1. Φίλτρο

Τα πειράματα που έγιναν για την δραστηριότητα του φίλτρου έγιναν σε ένα ETL σενάριο στο οποίο υπήρχε μόνο μία δραστηριότητα φίλτρου που είχε σαν είσοδο ένα πίνακα από όπου έπαιρνε τα δεδομένα και σαν έξοδο μία όψη στην οποία έγγραφε τελικά τα δεδομένα (Σχήμα 4.10). Στα πειράματα έγινε δίκαιη διαμοίραση μνήμης και στις δύο ακμές του γράφου και μετρήθηκε ο χρόνος εκτέλεσης του σεναρίου με διαφορετική κάθε φορά ποσότητα μνήμης.



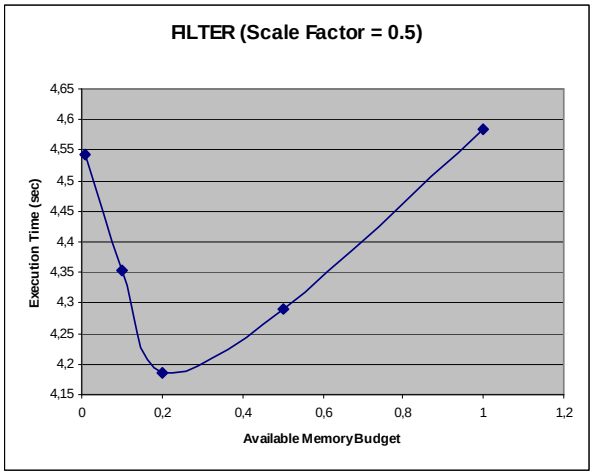
Σχήμα 4.10 Παράδειγμα Στοιχειώδους Σεναρίου Φίλτρου

Παρατηρούμε ότι για το πείραμα με παράγοντα κλιμάκωσης ίσο με 0.5 (Σχήμα 4.11) έχουμε μείωση στον χρόνο εκτέλεσης του πειράματος που φθάνει στο 10% μέχρι και

την περίπτωση που εκτελείται το σενάριο με το 20% της διαθέσιμης μνήμης του Η/Υ. Από εκείνο το σημείο και μέχρι το 100% της συνολικής διαθέσιμης μνήμης έχουμε αύξηση στον χρόνο εκτέλεσης του σεναρίου της τάξης του 9%.

Αντίθετα όταν το πείραμα εκτελέστηκε για παράγοντα κλιμάκωσης ίσο με 1.0 (Σχήμα 4.12) έχουμε μείωση στον χρόνο εκτέλεσης του σεναρίου της τάξης του 12% μέχρι και το 20% της συνολικής διαθέσιμης μνήμης του Η/Υ (όπως και πριν). Όμως όταν αυξάνεται η μνήμη που διανέμεται στις ακμές πάνω του 20% έχουμε μία μικρή αύξηση στον χρόνο εκτέλεσης περίπου 10% όμως τελικά ισορροπεί και έχει μία σταθερή κατάληξη μέχρι να διατεθεί όλη η συνολική διαθέσιμη μνήμη.

Πάντως σε γενικές γραμμές και στα δύο πειράματα που έγιναν για το ίδιο σενάριο με διαφορετικούς παράγοντες κλιμάκωσης παρατηρούμε ότι υπάρχουν μικρές διαφορές στους χρόνους εκτέλεσης του κάθε πειράματος για διαφορετικό ποσοστό διανομής της μνήμης ανάμεσα στις ακμές του γράφου.

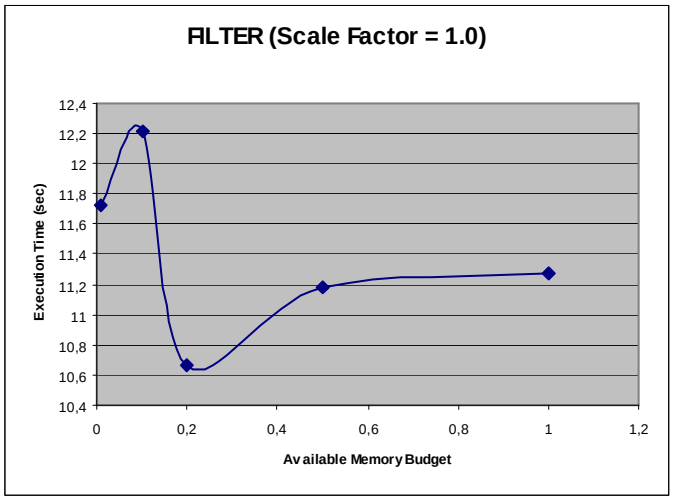


Σχήμα 4.11 Στοιχειώδες Σενάριο Φίλτρου (sf=0.5)

4.4.2. Συνένωση

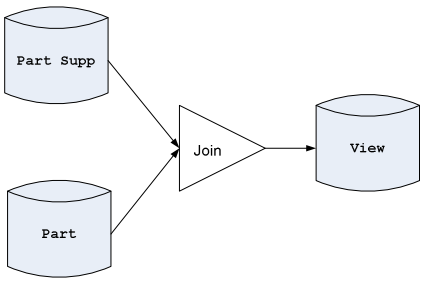
Για την δραστηριότητα της συνένωσης εκτελέστηκαν πειράματα με παράγοντες κλιμάκωσης 0.5 και 1.0, τα οποία χρησιμοποίησαν σαν ETL σενάριο αυτό του

Σχήματος 4.13. Στο σενάριο που έχουμε σε αυτή την περίπτωση υπάρχει μία δραστηριότητα συνένωσης δύο πινάκων και η έξοδος της συνένωσης γράφει τα δεδομένα σε μία όψη. Όμοια με τα προηγούμενα πειράματα για την δραστηριότητα του φίλτρου, η μνήμη διανέμεται δίκαια ανάμεσα στις ακμές του γράφου.



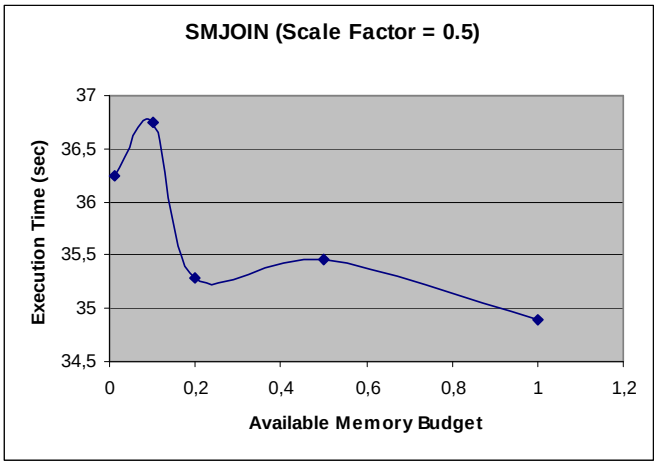
Σχήμα 4.12 Στοιχειώδες Σενάριο Φίλτρου (sf=0.1)

Παρατηρούμε ότι για τα πειράματα που έγιναν με παράγοντα κλιμάκωσης ίσο με 0.5 (Σχήμα 4.14) έχουμε μία ομαλή πτώση στον χρόνο εκτέλεσης του σεναρίου όσο αυξάνεται το ποσοστό της μνήμης που διανέμεται ανάμεσα στις ακμές του γράφου. Εξαίρεση αποτελεί μία μικρή αύξηση όταν το ποσοστό της μνήμης που διανέμεται είναι ίσο με 10% της συνολικής διαθέσιμης μνήμης του Η/Υ.

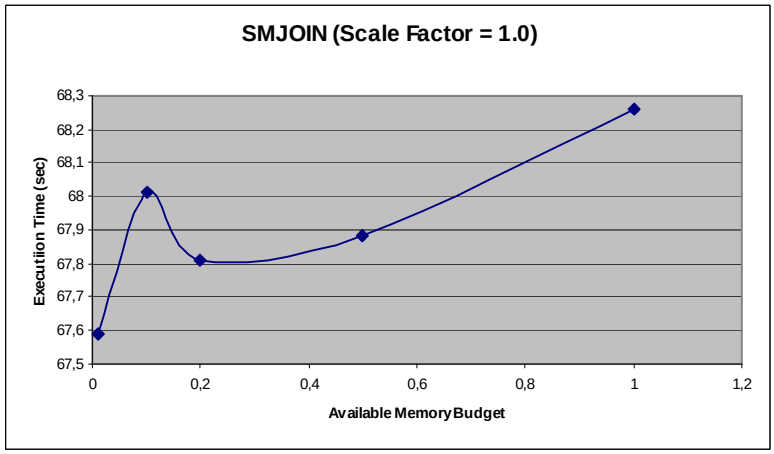


Σχήμα 4.13 Παράδειγμα Στοιχειώδους Σεναρίου Συνένωσης

Σε αντίθεση με ότι συμβαίνει για τον παράγοντα κλιμάκωσης ίσο με 0.5, όταν αυξάνονται τα δεδομένα εισόδου του σεναρίου στην περίπτωσή μας έχουμε παράγοντα κλιμάκωσης ίσο με 1.0 (Σχήμα 4.15) εμφανίζονται λίγο αλλοπρόσαλλα αποτελέσματα. Σε αυτή την περίπτωση όπως φαίνεται και στο Σχήμα 4.15 έχουμε συνεχόμενη αύξηση του χρόνου εκτέλεσης του πειράματος. Εξαίρεση αποτελεί ένα μικρό κομμάτι ανάμεσα στο 10-20% της μνήμης που διανέμεται, όπου εκεί παρατηρούμε μία μικρή μείωση στον χρόνο εκτέλεσης.



Σχήμα 4.14 Στοιχειώδες Σενάριο Συνένωσης (sf=0.5)

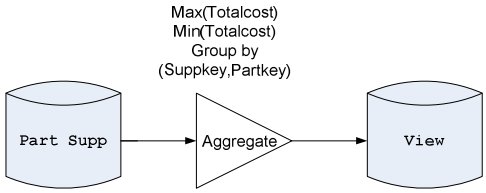


Σχήμα 4.15 Στοιχειώδες Σενάριο Συνένωσης (sf=1.0)

Έτσι παρατηρούμε ότι στη περίπτωση που έχουμε παράγοντα κλιμάκωσης ίσο με 0.5 όσο δίνεται περισσότερη μνήμη μειώνεται ο χρόνος εκτέλεσης του σεναρίου, ενώ στο σενάριο με παράγοντα κλιμάκωσης 1.0 όσο περισσότερη μνήμη δίνουμε τόσο αυξάνεται ο χρόνος εκτέλεσης του σεναρίου.

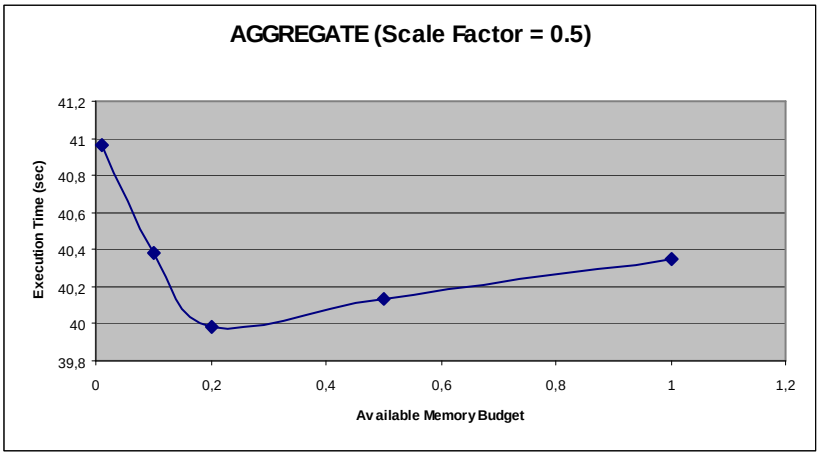
4.4.3. Συνάθροιση

Στο σενάριο που δόθηκε σαν είσοδο για να αξιολογήσουμε την δραστηριότητα της συνάθροισης ήταν αυτό του Σχήματος 4.16. Στο οποίο υπάρχει μία δραστηριότητα συνάθροισης, η οποία συνδέεται στην είσοδό της με ένα πίνακα και στην έξοδο της με μία όψη και απλώς υπολογίζει ένα μέγιστο και ένα ελάχιστο κάποιου γνωρίσματος του πίνακα εισόδου ομαδοποιημένα με τα κλειδιά του.



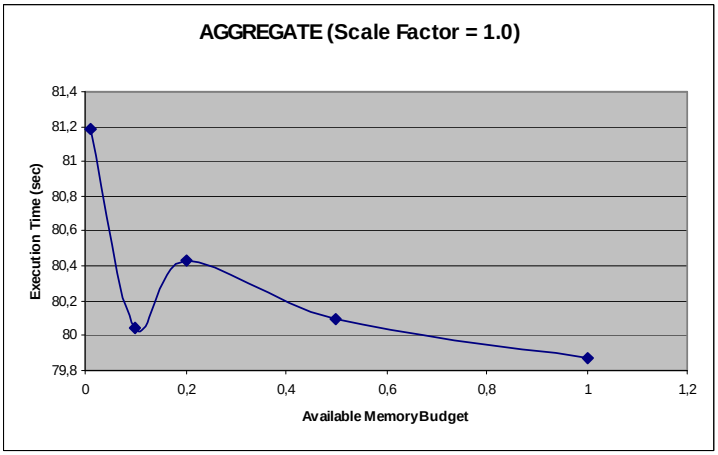
Σχήμα 4.16 Παράδειγμα Στοιχειώδους Σεναρίου Συνάθροισης

Στα πειράματα που έγιναν για παράγοντα κλιμάκωσης ίσο με 0.5 (Σχήμα 4.17) παρατηρούμε μείωση του χρόνου εκτέλεσης του σεναρίου της τάξης του 1-2% μέχρι και το ποσοστό του 20% της συνολικής διαθέσιμης μνήμης που διατίθεται στις ακμές του γράφου. Από εκεί και μετά μέχρι και το 100% της συνολικής μνήμης παρατηρούμε μία μικρή αύξηση στον χρόνο εκτέλεσης.



Σχήμα 4.17 Στοιχειώδες Σενάριο Συνάθροισης (sf=0.5)

Τέλος, στα πειράματα του σεναρίου της συνάθροισης (Σχήμα 4.16) για παράγοντα κλιμάκωσης ίσο με 1.0 (Σχήμα 4.18) έχουμε μία συνεχόμενη μείωση του χρόνου εκτέλεσης του σεναρίου με εξαίρεση το διάστημα από 10-20% όπου παρατηρείται μία μικρή άνοδο στο χρόνο.

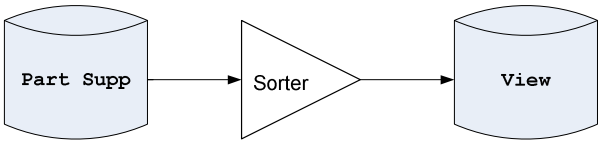


Σχήμα 4.18 Στοιχειώδες Σενάριο Συνάθροισης (sf=1.0)

4.4.4. Ταξινομητής

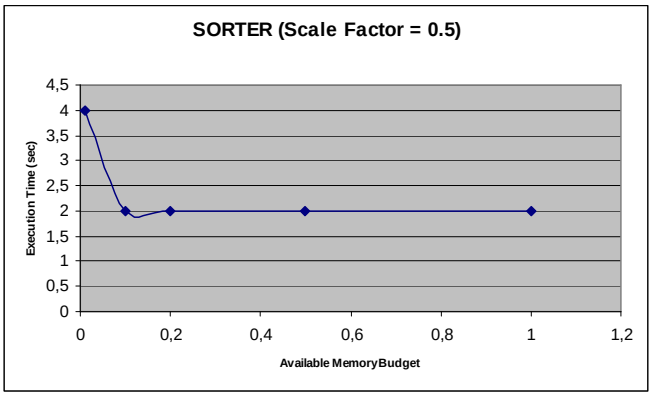
Τέλος, το σενάριο που δόθηκε σαν είσοδο για να αξιολογήσουμε την δραστηριότητα της συνάθροισης ήταν αυτό του Σχήματος 4.19. Στο οποίο υπάρχει μία

δραστηριότητα ταξινόμησης, η οποία συνδέεται στην είσοδό της με ένα πίνακα και στην έξοδο της με μία όψη και απλώς ταξινομεί τα δεδομένα του πίνακα εισόδου με βάση κάποιο πεδίο του πίνακα αυτού.

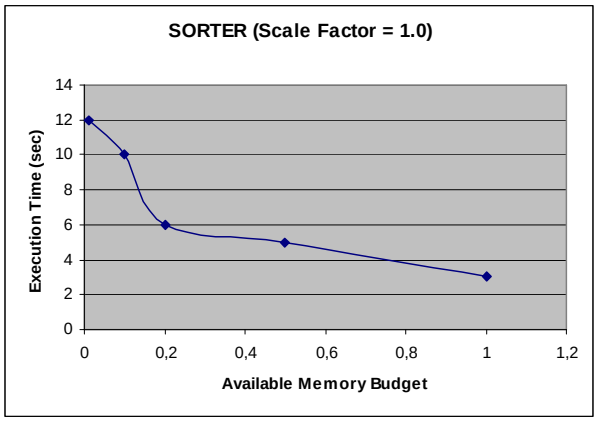


Σχήμα 4.19 Στοιχειώδες Σενάριο Ταξινόμησης

Όπως παρατηρούμε στα πειράματα που έγιναν για παράγοντα κλιμάκωσης 0.5 (Σχήμα 4.20) και 1.0 (Σχήμα 4.21) όσο περισσότερη μνήμη δίνουμε στο σενάριο της ταξινόμησης τόσο γρηγορότερα εκτελείται αυτό.



Σχήμα 4.20 Στοιχειώδες Σενάριο Ταξινόμησης (sf=0.5)



Σχήμα 4.21 Στοιχειώδες Σενάριο Ταξινόμησης (sf=1.0)

4.5. Δίκαιη Διανομή Μνήμης

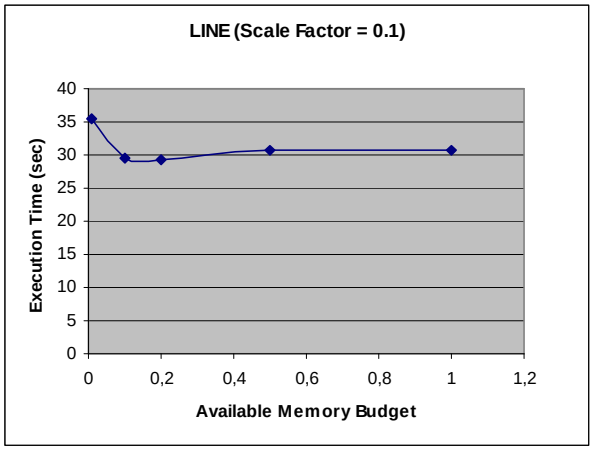
Αφού παρατηρήσαμε την συμπεριφορά των δραστηριοτήτων του φίλτρου, της συνένωσης και της συνάθροισης ξεχωριστά, τώρα θέλουμε να δούμε την συμπεριφορά τους μέσα σε ολοκληρωμένα σενάρια. Έτσι πλέον θέλουμε να παρατηρήσουμε τον χρόνο εκτέλεσης ενός ολοκληρωμένου σεναρίου με σκοπό να υπολογιστεί το ακριβές ποσοστό της συνολικής μνήμης που θα διανέμεται σε κάθε δραστηριότητα του γράφου.

Τα πειράματα διεξήχθησαν με διάφορες παραλλαγές των σεναρίων του γράφου αλλά και του μεγέθους εισόδου των δεδομένων. Τα σενάρια που χρησιμοποιήθηκαν ήταν το Γραμμικό Σενάριο (Σχήμα 4.4), το Σενάριο Δένδρου (Σχήμα 4.8) και το Σενάριο Πολλών Εκροών (Σχήμα 4.9) και το Σενάριο Πεταλούδας (Σχήμα 4.7). Επίσης το μέγεθος εισόδου που χρησιμοποιήθηκε στο κάθε πείραμα ήταν με παράγοντα κλιμάκωσης (scale factor) ίσο με 0.1, 0.5 και 1.0. Ο παράγοντας κλιμάκωσης χρησιμοποιείται από την γεννήτρια δεδομένων που χρησιμοποιεί το TPC-H [TPCH09] και ορίζει το μέγεθος των δεδομένων που θα παραχθούν. Τέλος η επιλεκτικότητα που χρησιμοποιείται στα δεδομένα μας είναι ίση με 0.8.

Εδώ παρουσιάζονται οι γραφικές παραστάσεις στα πειράματα που διεξήχθησαν μοιράζοντας την μνήμη δίκαια ανάμεσα στις λειτουργίες του γράφου. Δηλαδή, η

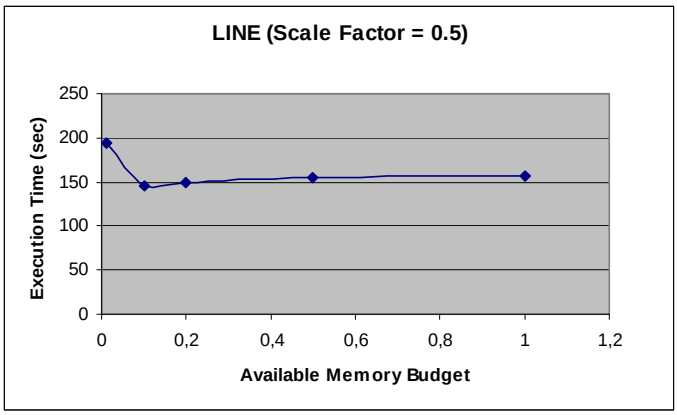
μνήμη που δίνεται σαν είσοδο στο πείραμα μοιράζεται εξίσου σε κάθε δραστηριότητα του γράφου.

Στο Σχήμα 4.22 παρουσιάζονται τα αποτελέσματα της εκτέλεσης του πειράματος για το Γραμμικό Σενάριο με παράγοντα κλιμάκωσης ίσο με 0.1. Παρατηρούμε από την γραφική παράσταση ότι για πολύ μικρό ποσοστό εκχώρησης μνήμης στο γράφημα το πείραμα έχει πολύ μεγάλο χρόνο εκτέλεσης. Όμως από το 20% περίπου της συνολικής μνήμης ο χρόνος εκτέλεσης μειώνεται κατά 15% και από κει πέρα παραμένει σχετικά σταθερός όση μνήμη και να εκχωρήσουμε στο πειραμά μας.



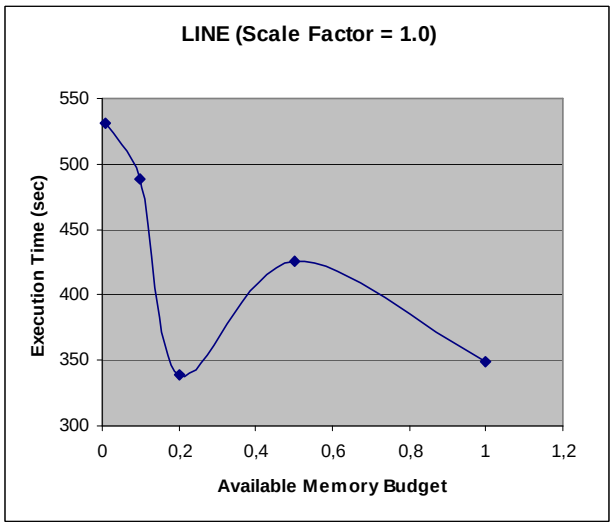
Σχήμα 4.22 Δίκαιος Αλγόριθμος σε Γραμμικό Σενάριο (sf=0.1)

Το Σχήμα 4.23 παρουσιάζει πάλι την εκτέλεση ενός Γραμμικού Σεναρίου αλλά αυτή την φορά με παράγοντα κλιμάκωσης ίσο με 0.5. Παρατηρούμε ότι πλέον το πείραμα εκτελείται πέντε φορές πιο αργά, λογικό γιατί πλέον τα δεδομένα που μπαίνουν σαν είσοδο είναι και πέντε φορές μεγαλύτερα σε μέγεθος (βλέπε παράγοντα κλιμάκωσης). Παρόλα αυτά το πείραμα έχει ακριβώς την ίδια συμπεριφορά μόνο που τώρα έχουμε μείωση του χρόνου εκτέλεσης στο διάστημα 1 έως 20% κατά 25% και μετά το 20% της συνολικής μνήμης έρχεται σε σχετική ισορροπία και δεν αλλάζει ο χρόνος εκτέλεσης του.



Σχήμα 4.23 Δίκαιος Αλγόριθμος σε Γραμμικό Σενάριο (sf=0.5)

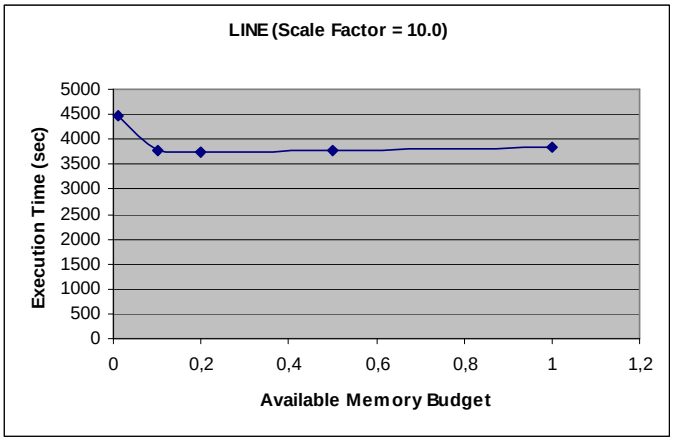
Στο Σχήμα 4.24 παρουσιάζεται το Γραμμικό Σενάριο με παράγοντα κλιμάκωσης ίσο με 1.0. Παρατηρούμε μία πτώση στον χρόνο εκτέλεσης του σεναρίου μέχρι να δοθεί το 20% της συνολικής μνήμης, από εκεί και μέχρι το 50% της διαθέσιμης μνήμης έχουμε μία άνοδο και τέλος στο διάστημα 50-100% της μνήμης έχουμε εκ' νέου πτώση στο χρόνο εκτέλεσης του σεναρίου.



Σχήμα 4.24 Δίκαιος Αλγόριθμος σε Γραμμικό Σενάριο (sf=1.0)

Επίσης, στο Σχήμα 4.25 όπου αναπαρίσταται γραφικά το Γραμμικό Σενάριο με παράγοντα κλιμάκωσης 10.0 παρατηρούνται ακριβώς τα ίδια συμπεράσματα με τα

προηγούμενα με την λογική διαφορά στον χρόνο εκτέλεσης του πειράματος και πλέον η μείωση του χρόνου εκτέλεσης στο διάστημα 1% έως 20% φτάνει έως και 36%.



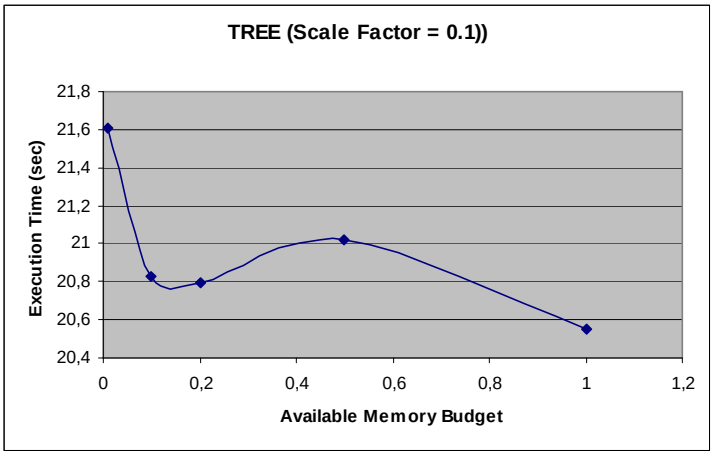
Σχήμα 4.25 Δίκαιος Αλγόριθμος σε Γραμμικό Σενάριο (sf=10.0)

Έτσι, βλέπουμε ότι το Γραμμικό Σενάριο για όλες τους παράγοντες κλιμάκωσης έχει την ίδια συμπεριφορά, δηλαδή μέχρι το 20% της μνήμης έχουμε συνεχόμενη μείωση στο χρόνο εκτέλεσης των πειραμάτων και έπειτα όση μνήμη και να δώσουμε διατηρείται σταθερός ο χρόνος εκτέλεσης των πειραμάτων.

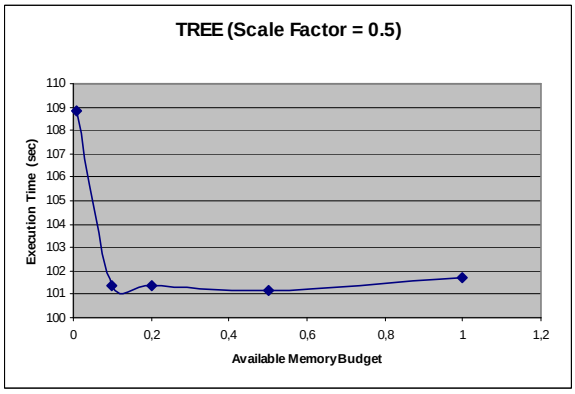
Στην συνέχεια, παρουσιάζονται γραφικές παραστάσεις με το Σενάριο του Δένδρου με δύο διαφορετικούς παράγοντες κλιμάκωσης 0.1 και 0.5. Η γραφική παράσταση του Σχήματος 4.26 αναπαριστά το Σενάριο Δένδρου με παράγοντα κλιμάκωσης ίσο με 0.1. Στο σχήμα αυτό παρατηρούμε ότι πάλι σε ποσοστό περίπου 20% της συνολικής μνήμης αρχίζει να ισορροπεί η γραφική παράσταση. Όμως, σε ποσοστό κοντά στο 50% έχει μία μικρή αύξηση και τελικά έχει μία δραματική πτώση του χρόνου όσο δίνουμε περισσότερη μνήμη από το 50% της συνολικής με αποκορύφωμα το 100% όπου έχουμε δώσει όλη την διαθέσιμη μνήμη και ο χρόνος εκτέλεσης έχει φτάσει στο ελάχιστο του σημείο.

Έπειτα, στο Σχήμα 4.27 έχουμε πάλι την γραφική παράσταση του Σεναρίου Δένδρου αλλά αυτήν την φορά με παράγοντα κλιμάκωσης ίσο με 0.5. Σε αυτή την περίπτωση έχουμε παρατηρούμε διαφορετικά αποτελέσματα από ότι είχαμε προηγουμένως για

παράγοντα κλιμάκωσης 0.1. Τώρα βλέπουμε από την γραφική παράσταση ότι μετά το ποσοστό του 20% περίπου αρχίζει και ισορροπεί το σύστημά μας και από εκεί και πλέον όση μνήμη και να δώσουμε δεν επηρεάζεται σχεδόν καθόλου, εκτός από μία ανεπαίσθητη άνοδο στον χρόνο εκτέλεσης του πειράματος μετά το 60% περίπου.



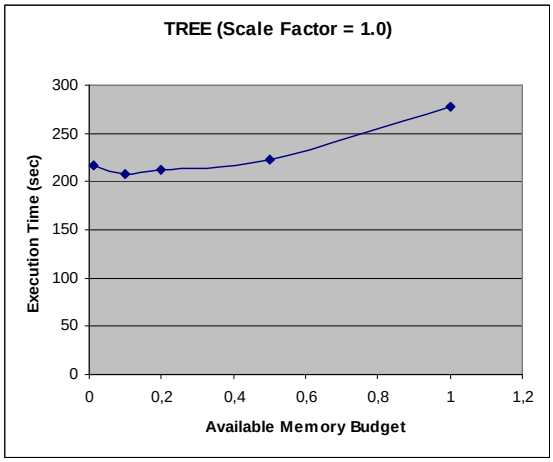
Σχήμα 4.26 Δίκαιος Αλγόριθμος σε Σενάριο Δένδρου (sf=0.1)



Σχήμα 4.27 Δίκαιος Αλγόριθμος σε Σενάριο Δένδρου (sf=0.5)

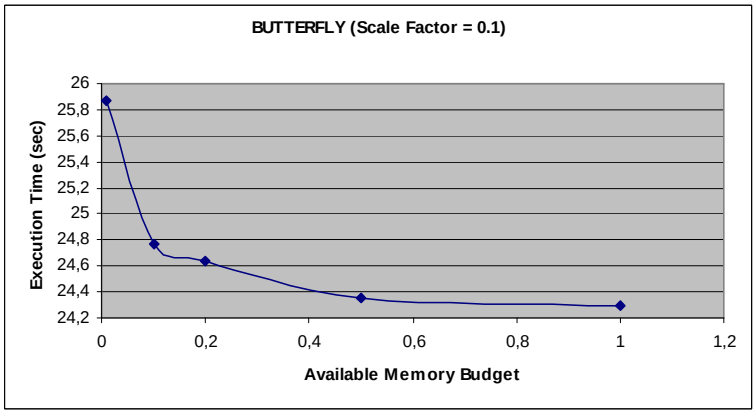
Σε αντίθεση με ότι συμβαίνει στα πειράματα με παράγοντα κλιμάκωσης 0.1 και 0.5 στα πειράματα με παράγοντα κλιμάκωσης 1.0 έχουμε διαφορετικά αποτελέσματα. Εδώ όσο περισσότερη μνήμη δίνουμε στις δραστηριότητες του σεναρίου τόσο πιο

πολύ αυξάνεται ο χρόνος εκτέλεσης του και φτάνει μέχρι και στο 25% αύξησης όταν έχει δοθεί όλη η διαθέσιμη συνολικά μνήμη.

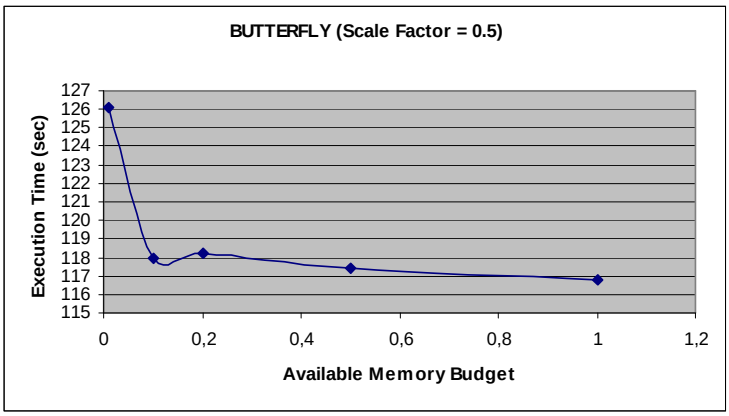


Σχήμα 4.28 Δίκαιος Αλγόριθμος σε Σενάριο Δένδρου (sf=1.0)

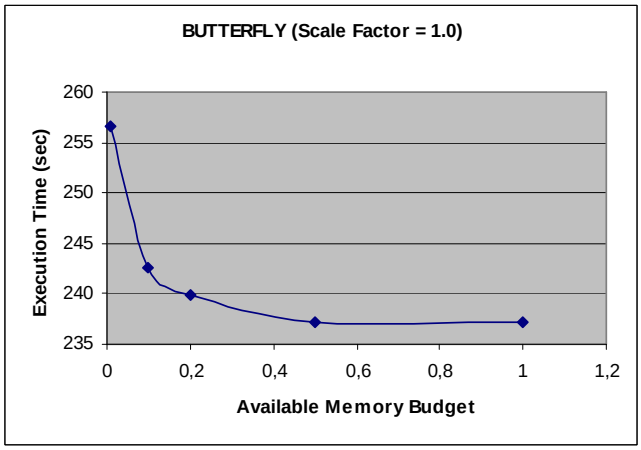
Στα Σχήματα 4.29, 4.30, 4.31 παρουσιάζονται οι γραφικές παραστάσεις των Σεναρίων της Πεταλούδας με τον Δίκαιο Αλγόριθμο για παράγοντες κλιμάκωσης 0.1, .05 και 1.0 αντίστοιχα. Και στις τρεις παρατηρείται συνεχόμενη μείωση στο χρόνο εκτέλεσης του σεναρίου που φτάνει στο ποσοστό του 7% για τον παράγοντα κλιμάκωσης 0.1 και 8% για παράγοντες κλιμάκωσης ίσους με 0.5 και 1.0.



Σχήμα 4.29 Δίκαιος Αλγόριθμος σε Σενάριο Πεταλούδας (sf=0.1)



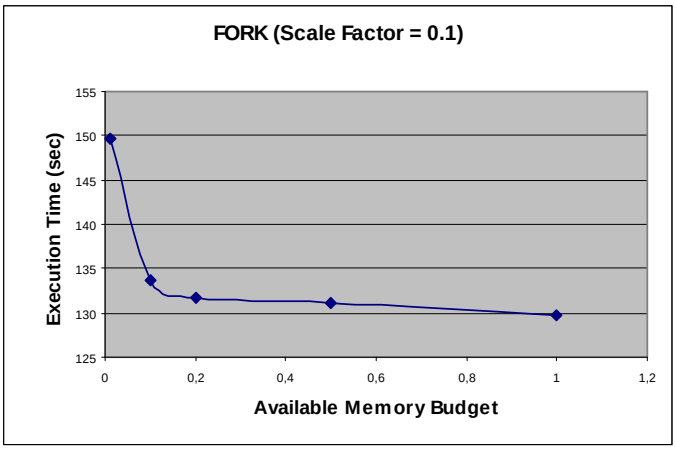
Σχήμα 4.30 Δίκαιος Αλγόριθμος σε Σενάριο Πεταλούδας (sf=0.5)



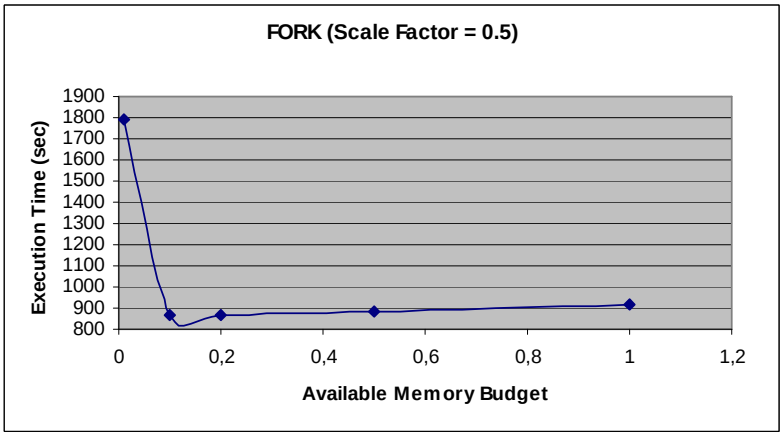
Σχήμα 4.31 Δίκαιος Αλγόριθμος σε Σενάριο Πεταλούδας (sf=1.0)

Τέλος, παρουσιάζονται γραφικές παραστάσεις με το Σενάριο Πολλών Εκροών με παράγοντες κλιμάκωσης με τιμή ίση με 0.1 (Σχήμα 4.32), 0.5 (Σχήμα 4.33) και 1.0 (Σχήμα 3.34). Και στις τρεις γραφικές παραστάσεις παρατηρούμε το ίδιο φαινόμενο. Έχουμε αρχικά την πτώση στον χρόνο εκτέλεσης (12% μείωση του χρόνου για παράγοντα κλιμάκωσης 0.1, 50% μείωση του χρόνου για παράγοντα κλιμάκωσης ίσο με 0.5 και 50% μείωση του χρόνου για 1.0 παράγοντα κλιμάκωσης) μέχρι την στιγμή που φτάνει στο 20% της διαθέσιμης μνήμης όπου εκεί αρχίζει και ισορροπεί. Από εκείνο το σημείο και μέχρι το 100% της διαθέσιμης μνήμης το σύστημα μας έχει

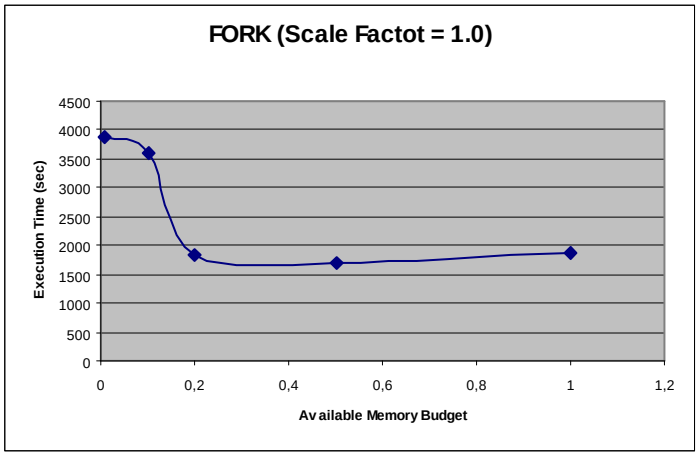
έρθει σε ισορροπία και όση διαθέσιμη μνήμη και να δώσουμε δεν παίζει μεγάλο ρόλο στον τελικό χρόνο εκτέλεσης του πειράματος.



Σχήμα 4.32 Δίκαιος Αλγόριθμος σε Σενάριο Πολλών Εκροών (sf=0.1)



Σχήμα 4.33 Δίκαιος Αλγόριθμος σε Σενάριο Πολλών Εκροών (sf=0.5)



Σχήμα 4.34 Δίκαιος Αλγόριθμος σε Σενάριο Πολλών Εκροών (sf=1.0)

Παρατηρήθηκε ότι τα πειράματα που έγιναν με τον *Δίκαιο Αλγόριθμο* έχουν όλα την ίδια συμπεριφορά (εξαίρεση το Σενάριο Δένδρου για παράγοντα κλιμάκωσης ίσο με 0.1) ότι δηλαδή μέχρι να δοθεί το 20% της συνολικής διαθέσιμης μνήμης ο χρόνος εκτέλεσης μειώνεται με αυξητικό ρυθμό. Από εκεί και πέρα και μέχρι να δοθεί το 100% της συνολικής διαθέσιμης μνήμης ο χρόνος εκτέλεσης παραμένει σταθερός.

Συμπερασματικά, το 20% της διαθέσιμης μνήμης είναι το κρίσιμο σημείο κάθε πειράματος. Έτσι όσο μνήμη παραπάνω και να δώσουμε στο σενάριο μας δεν επηρεάζεται καθόλου ο χρόνος εκτέλεσης. Γι' αυτό τον λόγω στα πειράματα που θα ακολουθήσουν με του υπόλοιπους αλγορίθμους εκχώρησης μνήμης θα δίνεται αρχικά το 20% της συνολικής μνήμης σε όλους τους κόμβους δίκαια και το υπόλοιπο θα διανέμεται στις εκάστοτε ευνοούμενες δραστηριότητες.

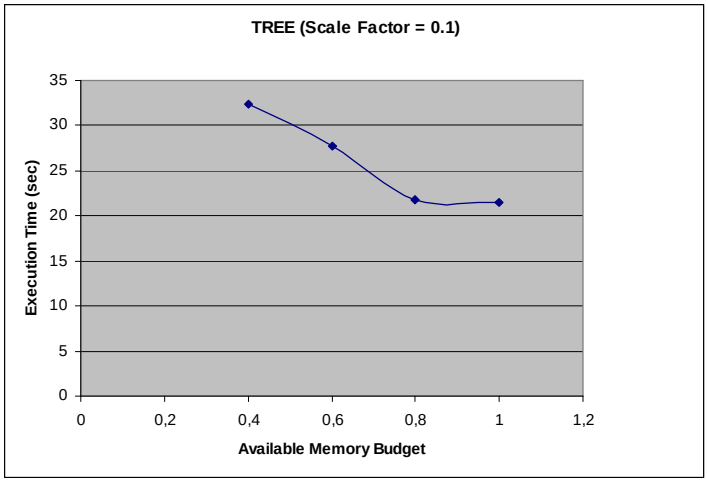
4.6. Εκχώρηση Μνήμης με Εύνοια των Δραστηριοτήτων Συνένωσης

Στην παράγραφο αυτή παρουσιάζονται πειράματα που έγιναν εκτελώντας τον *Αλγόριθμο Προτεραιότητας Συνενώσεων* πάνω στο Σενάριο του Δένδρου (Σχήμα 4.8) και στο Σενάριο της Πεταλούδας (Σχήμα 4.7). Τέλος, τα παρακάτω πειράματα έγιναν με παράγοντες κλιμάκωσης ίσους με 0.1, 0.5 και 1.0.

Όπως παρατηρήθηκε στην προηγούμενη παράγραφο με τον *Δίκαιο* Αλγόριθμο διανομής της μνήμης ανάμεσα στις ακμές του γράφου υπάρχει ένα κρίσιμο σημείο που από εκεί και μετά όση μνήμη και να δοθεί στα σενάριά μας ο χρόνος εκτέλεσης σε γενικές γραμμές παραμένει σταθερός μέχρι δοθεί το 100% της συνολικής διαθέσιμης μνήμης. Αυτό το κρίσιμο σημείο είναι το 20% της συνολικής διαθέσιμης μνήμης. Έτσι τα πειράματα που έγιναν με τον *Αλγόριθμο Προτεραιότητας Συνενώσεων* αρχικά διανέμουν τον 20% της διαθέσιμης μνήμης στις ακμές του γράφου και στην συνέχεια αυξανόμενα δίνουν το υπόλοιπο ποσοστό στις δραστηριότητες συνένωσης που υπάρχουν στον γράφο μας.

4.6.1. Δενδρικό Σενάριο

Όπως παρατηρούμε για παράγοντα κλιμάκωσης ίσο με 0.1 (Σχήμα 4.35) ο χρόνος εκτέλεσης του πειράματος στο Δενδρικό Σενάριο μειώνεται όσο αυξάνεται το ποσοστό της διαθέσιμης μνήμης που δίνουμε στις δραστηριότητες συνένωσης. Έτσι έχει σαν αποτέλεσμα να φτάσουμε μέχρι και στην μείωση του χρόνου εκτέλεσης του πειράματος κατά 34% περίπου για το 80-100% της διαθέσιμης συνολικής μνήμης.

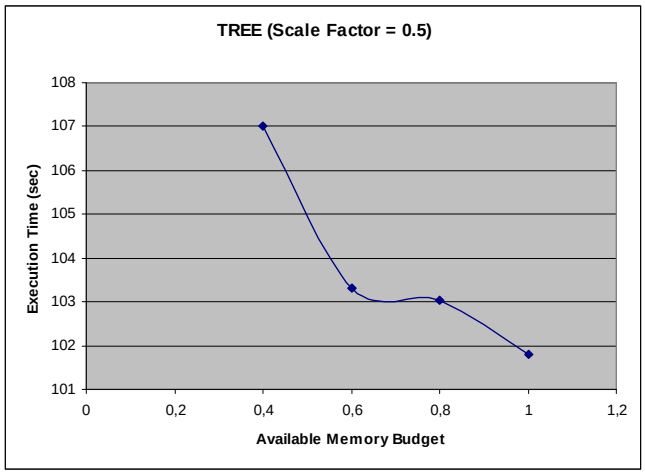


Σχήμα 4.35 Αλγόριθμος Προτεραιότητας Συνενώσεων σε Δενδρικό Σενάριο (sf=0.1)

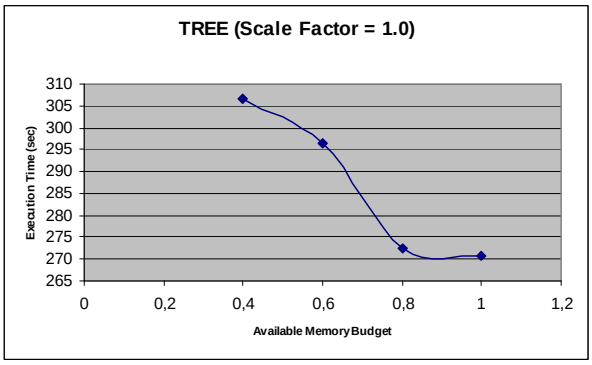
Σε όμοια συμπεράσματα οδηγούμαστε και όταν μεγαλώνουμε το μέγεθος των δεδομένων εισόδου, τώρα έχουμε παράγοντα κλιμάκωσης ίσο με 0.5 (Σχήμα 4.36). Μόνο που σε αυτήν την περίπτωση δεν φτάνουμε σε τόσο υψηλό ποσοστό μείωσης του χρόνου εκτέλεσης του πειράματος. Το ύψος της μείωσης είναι της τάξης αυτή την φορά του 10% και συναντάτε όταν πλέον έχει δοθεί όλη η διαθέσιμη μνήμη στις δραστηριότητες συνένωσης του γράφου. Επίσης σε αυτή την περίπτωση έχουμε διαφοροποίηση στο διάστημα 60-80% της συνολικής διαθέσιμης μνήμης όπου εδώ παραμένει σταθερός ο χρόνος εκτέλεσης του πειράματος.

Επίσης και στα πειράματα που έγιναν με παράγοντα κλιμάκωσης ίσο με 1.0 (Σχήμα 4.37) οδηγούμαστε στα ίδια συμπεράσματα με τα προηγούμενα, δηλαδή έχουμε συνεχόμενη πτώση του χρόνου εκτέλεσης του σεναρίου που φτάνει μέχρι και το 12% όταν έχει δοθεί όλη η διαθέσιμη μνήμη στις δραστηριότητες συνένωσης.

Συμπερασματικά, στην περίπτωση του Δενδρικού Σεναρίου όταν ευνοούμε τις δραστηριότητες των συνενώσεων όσο περισσότερη μνήμη δίνεται στις συνενώσεις τόσο γρηγορότερα εκτελείται το σενάριό μας.



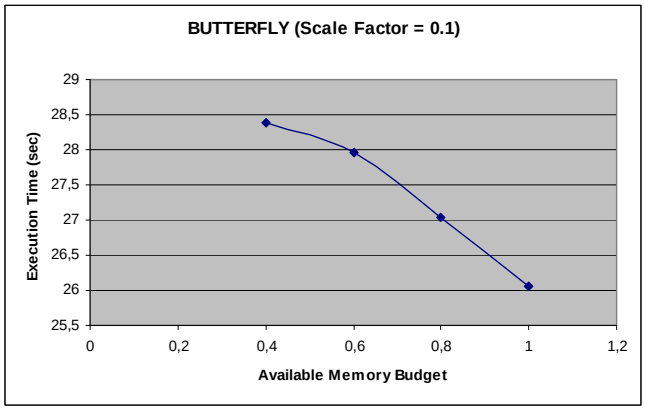
Σχήμα 4.36 Αλγόριθμος Προτεραιότητας Συνενώσεων σε Δενδρικό Σενάριο (sf=0.5)



Σχήμα 4.37 Αλγόριθμος Προτεραιότητας Συνενώσεων σε Δενδρικό Σενάριο (sf=1.0)

4.6.2. Σενάριο Πεταλούδας

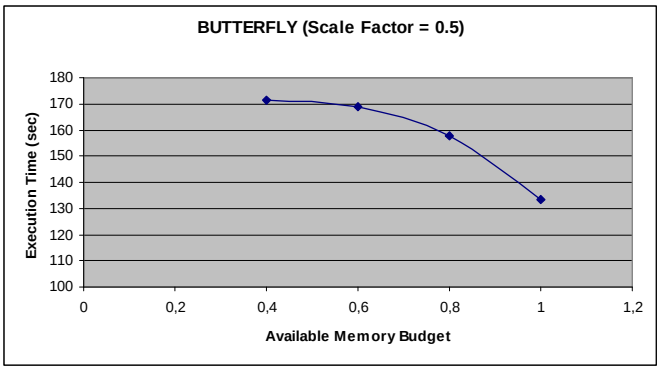
Στην περίπτωση που έχουμε τώρα το Σενάριο της Πεταλούδας χρησιμοποιώντας τον *Αλγόριθμο Προτεραιότητας Συνενώσεων* με παράγοντα κλιμάκωσης ίσο με 0.1 (Σχήμα 4.38) παρατηρούμε συνεχείς μείωση του χρόνου εκτέλεσης του πειράματος όσο δίνουμε περισσότερη μνήμη στις ακμές των συνενώσεων του σεναρίου. Η μείωση που παρατηρείται φθάνει μέχρι και την τάξη του 10% όταν έχει δοθεί πλέον όλη η συνολική διαθέσιμη μνήμη στις ακμές του γράφου.



Σχήμα 4.38 Αλγόριθμος Προτεραιότητας Συνενώσεων σε Σενάριο Πεταλούδας (sf=0.1)

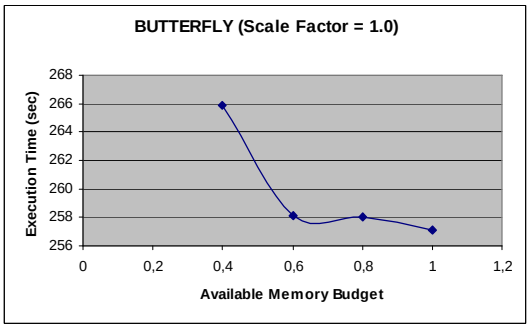
Όμοια και στην περίπτωση όπου έχουμε παράγοντα κλιμάκωσης ίσο με 0.5 (Σχήμα 4.39) παρατηρούμε μείωση στον χρόνο εκτέλεσης των πειραμάτων όσο δίνουμε

περισσότερη μνήμη στις ακμές των δραστηριοτήτων της συνένωσης. Μόνο που σε αυτή την περίπτωση έχουμε πολύ μεγαλύτερη μείωση στον χρόνο. Ξεκινάει με 5% περίπου όταν έχει δοθεί το 60% της συνολικής μνήμης, αυξάνεται στο 10% περίπου όταν έχει διανεμηθεί το 80% της μνήμης και καταλήγει σε 25% μείωση του συνολικού χρόνου όταν πλέον έχει διανεμηθεί όλη η διαθέσιμη μνήμη του Η/Υ.



Σχήμα 4.39 Αλγόριθμος Προτεραιότητας Συνενώσεων σε Σενάριο Πεταλούδας (sf=0.5)

Όμοια με τα προηγούμενα πειράματα με παράγοντες κλιμάκωσης 0.1 και 0.5 και το σενάριο με παράγοντα κλιμάκωσης 1.0 (Σχήμα 4.40) εκτελείται γρηγορότερα όσο δίνουμε περισσότερη μνήμη στις δραστηριότητες των συνενώσεων μόνο που στην προκειμένη περίπτωση δεν έχει τόσο μεγάλη πτώση ο χρόνος εκτέλεσης του σεναρίου, η οποία φτάνει μέχρι και το 4% όταν έχει δοθεί πλέον όλη η διαθέσιμη μνήμη.



Σχήμα 4.40 Αλγόριθμος Προτεραιότητας Συνενώσεων σε Σενάριο Πεταλούδας (sf=1.0)

Σαν συμπέρασμα, παρατηρούμε ότι και στο Σενάριο Πεταλούδας όση περισσότερη μνήμη δίνουμε στις δραστηριότητες των συνενώσεων τόσο γρηγορότερα εκτελείται στο συγκεκριμένο σενάριο ανεξαρτήτως μεγέθους των δεδομένων εισόδου.

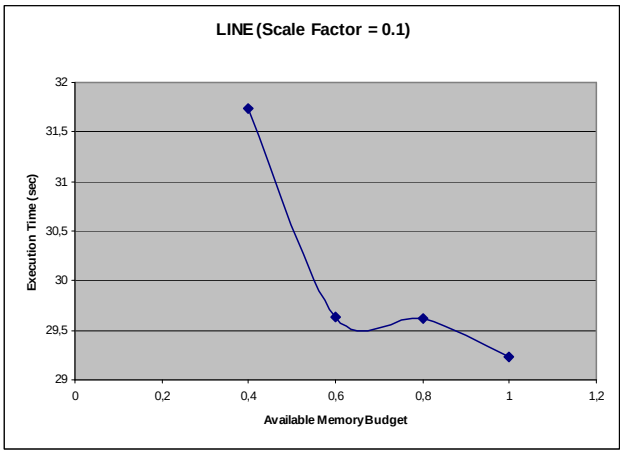
4.7. Εκχώρηση Μνήμης με Εύνοια των Δραστηριοτήτων Εγγραφής

Σε αυτό το σημείο παρουσιάζονται η συμπεριφορά του *Αλγορίθμου Προτεραιότητας Δραστηριοτήτων Εγγραφής* μέσα από μία σειρά πειραμάτων χρησιμοποιώντας ως είσοδο το Γραμμικό Σενάριο (Σχήμα 4.4), το Δενδρικό Σενάριο (Σχήμα 4.8), το Σενάριο Πολλών Εκροών (Σχήμα 4.9) και το Σενάριο Πεταλούδας (Σχήμα 4.7).

Όπως έχει αναφερθεί στην προηγούμενη παράγραφο μέσω των πειραμάτων με τον *Δίκαιο Αλγόριθμο* βρέθηκε το κρίσιμο σημείο διανομής μνήμης που εκεί και πέρα ο χρόνος παραμένει σταθερός όσο μνήμη και να δοθεί. Αυτό το σημείο είναι το 20% της συνολικής διαθέσιμης μνήμης. Επομένως στα πειράματα που θα παρουσιαστούν παρακάτω με τον *Αλγόριθμο Προτεραιότητας Δραστηριοτήτων Εγγραφής* διανέμεται αρχικά το 20% της μνήμης σε όλες τις ακμές του γράφου και έπειτα το υπόλοιπο ποσοστό σταδιακά διανέμεται στις ακμές που ενώνονται με μία όψη ή μία Αποθήκη Δεδομένων.

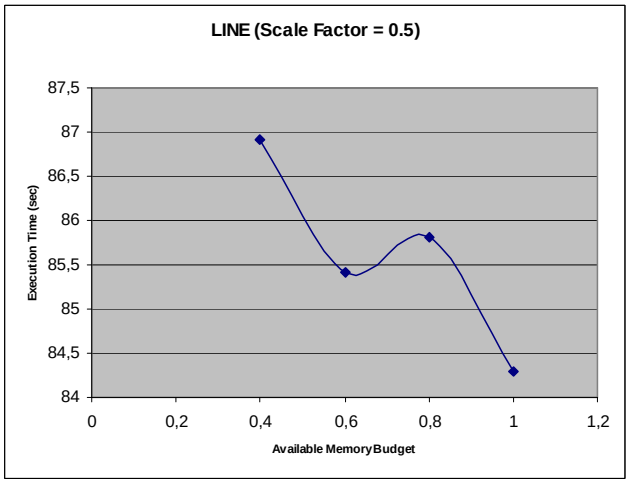
4.7.1. Γραμμικό Σενάριο

Στα πειράματα που έγιναν με το Γραμμικό Σενάριο σαν είσοδο και παράγοντα κλιμάκωσης ίσο με 0.1 (Σχήμα 4.41) παρατηρείται μείωση στον χρόνο εκτέλεσης της τάξης του 7% κατά το διάστημα που δίνεται στις δραστηριότητες των εγγραφών του σεναρίου το 40-60% της συνολικής διαθέσιμης. Στην συνέχεια, στο διάστημα 60-80% έχουμε μία πάρα πολύ μικρή αύξηση (σε σημείο να χαρακτηριστεί σταθερός) του χρόνου εκτέλεσης και στο τελευταίο διάστημα 80-100% της συνολικής μνήμης παρατηρείται εκ’ νέου νέα μείωση του χρόνου της τάξης του 2-3%.



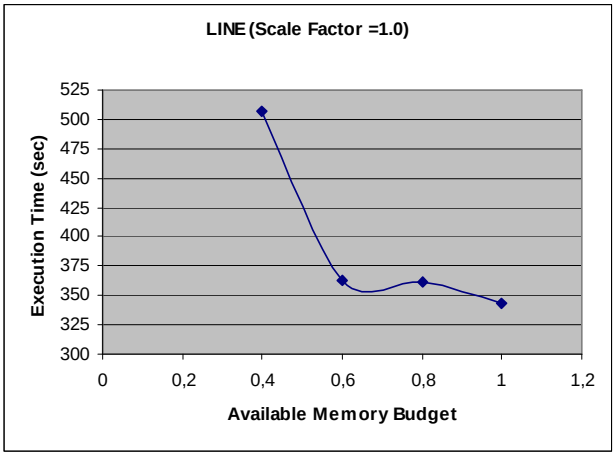
Σχήμα 4.41 Αλγόριθμος Προτεραιότητας Δραστηριοτήτων Εγγραφών σε Γραμμικό Σενάριο (sf=0.1)

Όμοια αποτελέσματα έχουμε και για παράγοντα κλιμάκωσης ίσο με 0.5 (Σχήμα 4.42) με την λογική διαφορά ότι τώρα ο χρόνος εκτέλεσης των πειραμάτων θα είναι πέντε φορές μεγαλύτερος. Έτσι, αρχικά έχουμε πάλι μείωση του χρόνου εκτέλεσης σε πολύ μικρό ποσοστό του 2% στο διάστημα 40-60% της συνολικής μνήμη, έπειτα παρατηρείται μία μικρή αύξηση από το 60-80% και τέλος, στο διάστημα 80-100% της συνολικής διαθέσιμης μνήμης έχουμε πάλι μείωση του χρόνου εκτέλεσης σε ποσοστό πάλι 2%.



Σχήμα 4.42 Αλγόριθμος Προτεραιότητας Δραστηριοτήτων Εγγραφής σε Γραμμικό Σενάριο (sf=0.5)

Επίσης, και στα πειράματα που έγιναν με το Γραμμικό Σενάριο χρησιμοποιώντας τον Αλγόριθμο Προτεραιότητας Δραστηριοτήτων Εγγραφής για παράγοντα κλιμάκωσης ίσο με 1.0 (Σχήμα 4.42) παρατηρούμε συνεχόμενη μείωση του χρόνου εκτέλεσης του σεναρίου. Αυτή η πτώση στον χρόνο φτάνει στο 29% για ποσοστά μνήμης 60-80% και στο 32% για το 100% της συνολικής διαθέσιμης μνήμης.

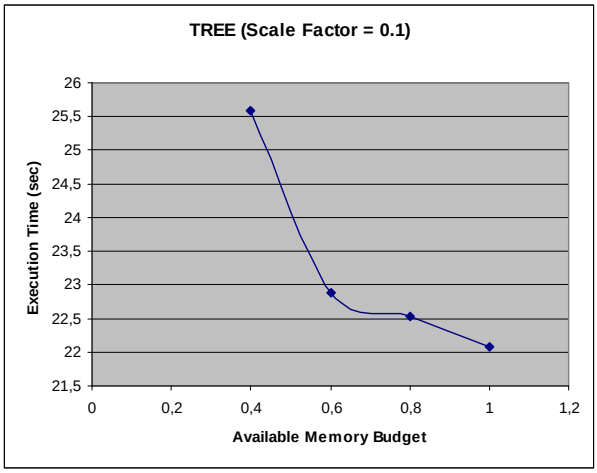


Σχήμα 4.43 Αλγόριθμος Προτεραιότητας Δραστηριοτήτων Εγγραφής σε Γραμμικό Σενάριο (sf=1.0)

Σαν συμπέρασμα, στο Γραμμικό Σενάριο με Εύνοια στις δραστηριότητες εγγραφών παρατηρούμε συνεχόμενη μείωση στο χρόνο εκτέλεσης των πειραμάτων όσο περισσότερη μνήμη δίνουμε σε αυτά. Μία εξαίρεση παρατηρείται για παράγοντα κλιμάκωσης 0.5 όπου έχουμε μία άνοδο του χρόνου εκτέλεσης στο διάστημα 20-40% της διαθέσιμης μνήμης.

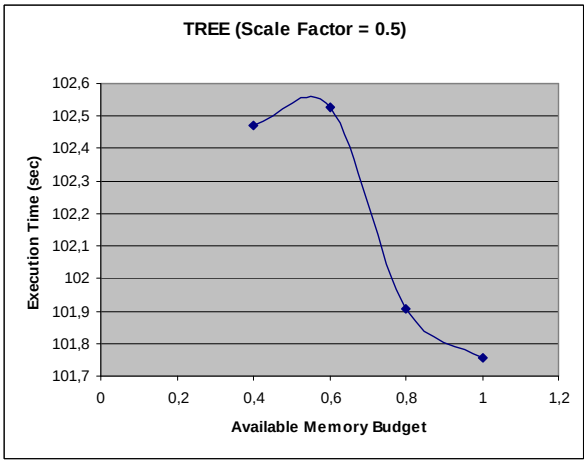
4.7.2. Δενδρικό Σενάριο

Στα πειράματα που χρησιμοποιούν το Δενδρικό Σενάριο ως είσοδο και παράγοντα κλιμάκωσης ίσο με 0.1 (Σχήμα 4.40) υπάρχει μία μικρή μείωση στον χρόνο εκτέλεσης των πειραμάτων που φτάνει μέχρι το 15% του αρχικού χρόνου όταν πλέον έχει διανεμηθεί το 100% της διαθέσιμης μνήμης.



Σχήμα 4.44 Αλγόριθμος Προτεραιότητας Δραστηριοτήτων Εγγραφής σε Δενδρικό Σενάριο (sf=0.1)

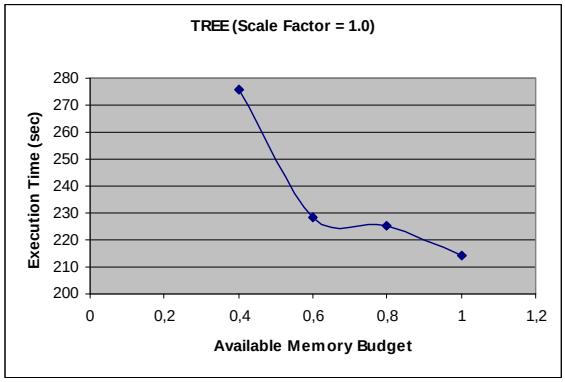
Σε αντίθεση με το προηγούμενο πείραμα, το πείραμα για παράγοντα κλιμάκωσης ίσο με 0.5 (Σχήμα 4.41) έχει πάλι μείωση στο χρόνο εκτέλεσης μικρότερη αυτή την φορά, της τάξης του 1% αλλά επίσης παρουσιάζει μία μικρή και ανεπαίσθητη αύξηση στον χρόνο ανάμεσα στο 40 και 60% της συνολικής διαθέσιμης μνήμης.



Σχήμα 4.45 Αλγόριθμος Προτεραιότητας Δραστηριοτήτων Εγγραφής σε Δενδρικό Σενάριο (sf=0.5)

Όσο μεγαλώνει το μέγεθος των δεδομένων εισόδου (παράγοντας κλιμάκωσης 1.0) για το Σενάριο Δένδρου με την χρήση του Αλγορίθμου Προτεραιότητας Δραστηριοτήτων

Εγγραφής (Σχήμα 4.42) παρατηρούνται πάλι τα ίδια φαινόμενα συνεχής μείωσης του χρόνου εκτέλεσης του σεναρίου αλλά με όλο και μεγαλύτερα ποσοστά μείωσης. Σε αυτή την περίπτωση ο συνολικός χρόνος εκτέλεσης των πειραμάτων φτάνει μέχρι το 22% σε σχέση με την αρχική εκτέλεση.



Σχήμα 4.46 Αλγόριθμος Προτεραιότητας Δραστηριοτήτων Εγγραφής σε Δενδρικό Σενάριο (sf=1.0)

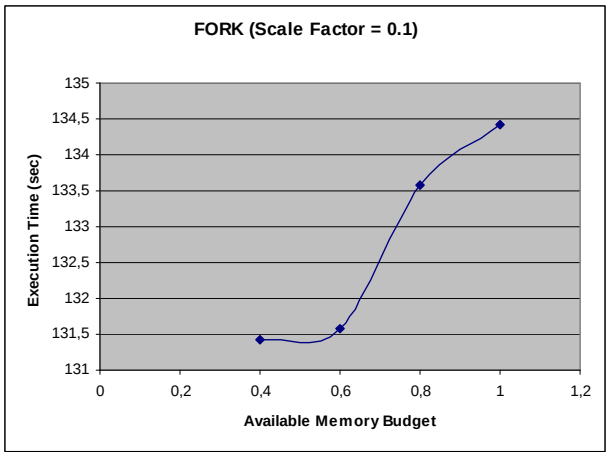
Συμπερασματικά, για το Σενάριο Δένδρου στην περίπτωση όπου ευνοούμε τις δραστηριότητες εγγραφής όσο πιο πολύ μνήμη δίνουμε σε αυτές τόσο γρηγορότερα εκτελείται το σενάριο.

4.7.3. Σενάριο Πολλών Εκροών

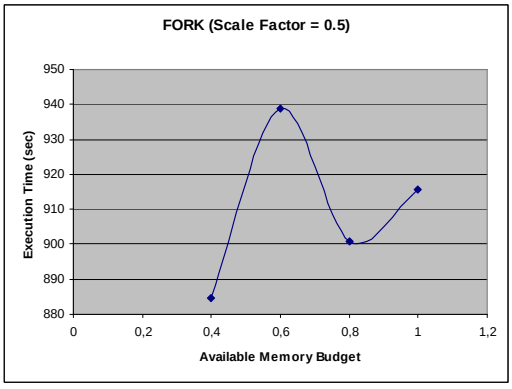
Σε αντίθεση με ότι έχουμε δει μέχρι στιγμής το Σενάριο Πολλών Εκροών παρουσιάζει μία περίεργη συμπεριφορά όταν δίνεται περισσότερη μνήμη στις ακμές που συνδέονται με όψεις ή Αποθήκες Δεδομένων.

Στα πειράματα με παράγοντα κλιμάκωσης ίσο με 0.1 (Σχήμα 4.47) ο χρόνος εκτέλεσης αυξάνεται εκθετικά όσο αυξάνεται και το ποσό της μνήμης που διανέμεται στις ακμές του γράφου που ενώνονται με όψεις ή Αποθήκες Δεδομένων. Τα ίδια περίπου αποτελέσματα συναντώνται και στην περίπτωση για παράγοντα κλιμάκωσης ίσο με 0.5 (Σχήμα 4.48) με την διαφορά ότι σε αυτή την περίπτωση έχουμε και

αρκετά μεγάλη μείωση του χρόνου στο διάστημα 60-80% της συνολικής διαθέσιμης μνήμης. Κατά τα άλλα στα υπόλοιπα διαστήματα η γραφική μας παράσταση παρουσιάζει συνεχής αύξηση.

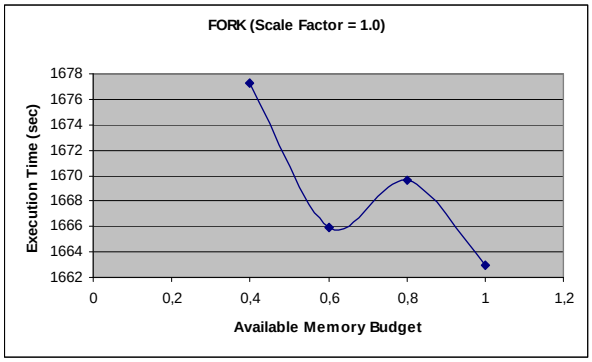


Σχήμα 4.47 Αλγόριθμος Προτεραιότητας Δραστηριοτήτων Εγγραφής σε Σενάριο Πολλών Εκροών (sf=0.1)



Σχήμα 4.48 Αλγόριθμος Προτεραιότητας Δραστηριοτήτων Εγγραφής σε Σενάριο Πολλών Εκροών (sf=0.5)

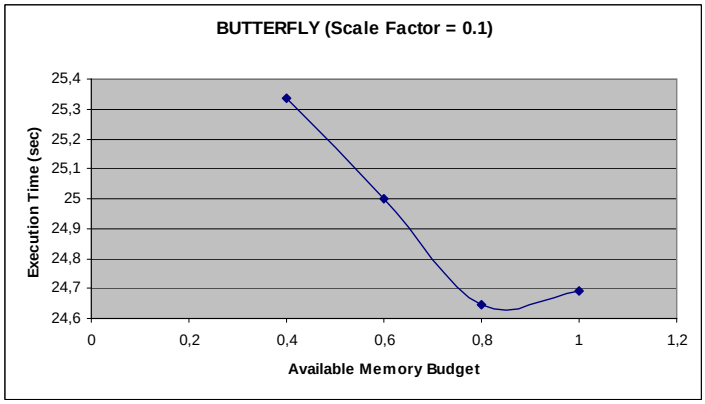
Αντίθετα σε ότι συμβαίνει για παράγοντες κλιμάκωσης 0.1 και 0.5, για παράγοντα κλιμάκωσης ίσο με 1.0 έχουμε πτώση του χρόνου εκτέλεσης του σεναρίου εξαίρεση αποτελεί το διάστημα 60-80% της διαθέσιμης μνήμης.



Σχήμα 4.49 Αλγόριθμος Προτεραιότητας Δραστηριοτήτων Εγγραφής σε Σενάριο Πολλών Εκροών (sf=1.0)

4.7.4. Σενάριο Πεταλούδας

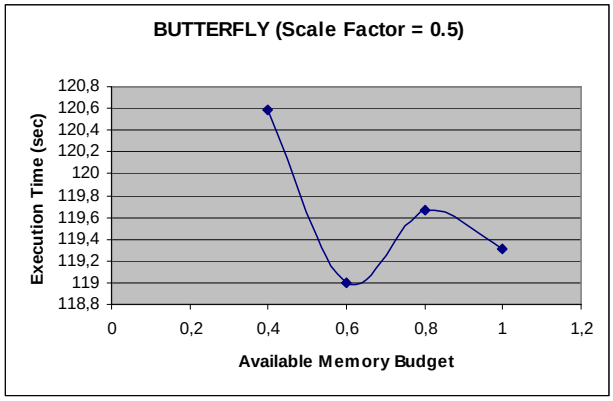
Στην περίπτωση του Σεναρίου της Πεταλούδας για παράγοντα κλιμάκωσης 0.1 (Σχήμα 4.50) παρατηρούμε συνεχόμενη μείωση στο χρόνο εκτέλεσης του σεναρίου όσο περισσότερη μνήμη δίνουμε στις δραστηριότητες εγγραφών. Η μείωση φθάνει έως το 3% του αρχικού χρόνου.



Σχήμα 4.50 Αλγόριθμος Προτεραιότητας Δραστηριοτήτων Εγγραφής σε Σενάριο Πεταλούδας (sf=0.1)

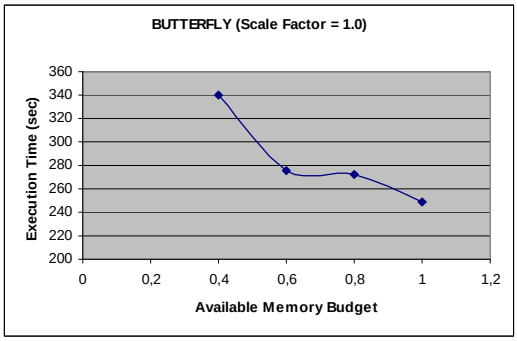
Σε αντίθεση με την προηγούμενη περίπτωση, τώρα για παράγοντα κλιμάκωσης 0.5 (Σχήμα 4.51)) έχουμε αρχικά μία πτώσης της τάξης του 2% μέχρι το 60% της μνήμης,

έπειτα έχουμε μία άνοδο της τάξης του 1% και όταν δίνουμε όλη την διαθέσιμη μνήμη έχει μία πολύ μικρή πτώση της τάξης του 0.5%.



Σχήμα 4.51 Αλγόριθμος Προτεραιότητας Δραστηριοτήτων Εγγραφής σε Σενάριο Πεταλούδας (sf=0.5)

Τέλος, για παράγοντα κλιμάκωσης ίσο με 1.0 (Σχήμα 4.52) έχουμε συνεχόμενη μείωση στο χρόνο εκτέλεσης του σεναρίου και η πτώση φθάνει το 27% όταν έχει δοθεί τελικά όλη η μνήμη στο σενάριο.



Σχήμα 4.52 Αλγόριθμος Προτεραιότητας Δραστηριοτήτων Εγγραφής σε Σενάριο Πεταλούδας (sf=1.0)

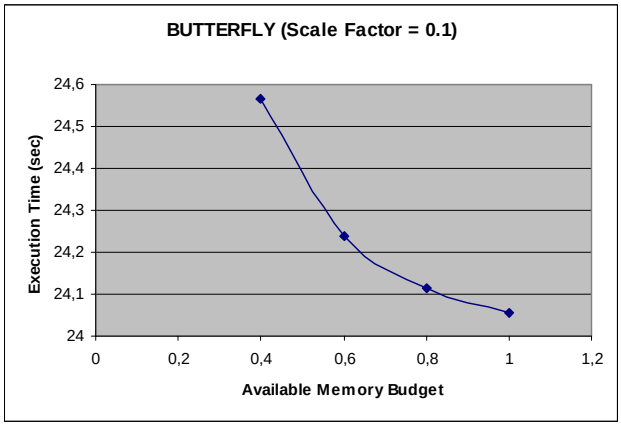
Συμπερασματικά, για παράγοντες κλιμάκωσης 0.1 και 1.0 έχουμε την ίδια ακριβώς συμπεριφορά μείωση του χρόνου εκτέλεσης όσο δίνουμε περισσότερη μνήμη στις δραστηριότητες των εγγραφών ενώ για παράγοντα κλιμάκωσης 0.5 έχουμε κάποια σκαμπανεβάσματα.

4.8. Εκχώρηση Μνήμης με Εύνοια των Δραστηριοτήτων Συνάθροισης

Στην παράγραφο αυτή παρουσιάζονται πειράματα που έγιναν εκτελώντας τον *Αλγόριθμο Προτεραιότητας Συναθροίσεων* χρησιμοποιώντας σαν είσοδο το Σενάριο της Πεταλούδας (Σχήμα 4.7), το Γραμμικό Σενάριο (Σχήμα 4.8) και το Σενάριο Δένδρου (Σχήμα 4.8) και τα δεδομένα εισόδου παρήχθησαν με παράγοντες κλιμάκωσης ίσους με 0.1, 0.5 και 1.0.

Όπως ειπώθηκε και πριν με βάση τα πειράματα που έγιναν χρησιμοποιώντας τον *Δίκαιο Αλγόριθμο* διανομή μνήμης, βρέθηκε ότι το κρίσιμο σημείο κατά την όλη διαδικασία διανομής είναι το 20% της συνολικής διαθέσιμης μνήμης. Γι' αυτό το λόγο τα πειράματα που διεξήχθησαν σε αυτή την παράγραφο διανέμουν το 20% της συνολικής μνήμης σε όλες τις ακμές και έπειτα διανέμουν την υπόλοιπη στις ακμές που ενώνονται στην είσοδό τους με μία δραστηριότητα συνάθροισης.

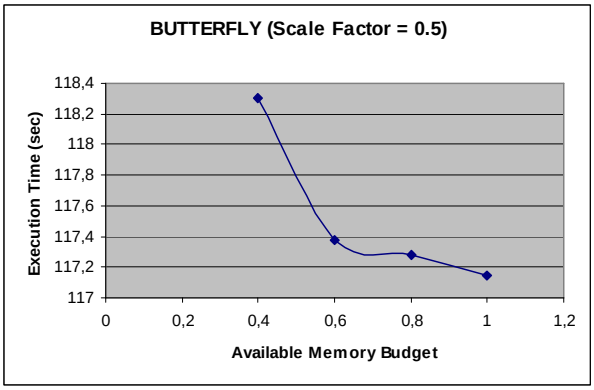
4.8.1. Σενάριο Πεταλούδας



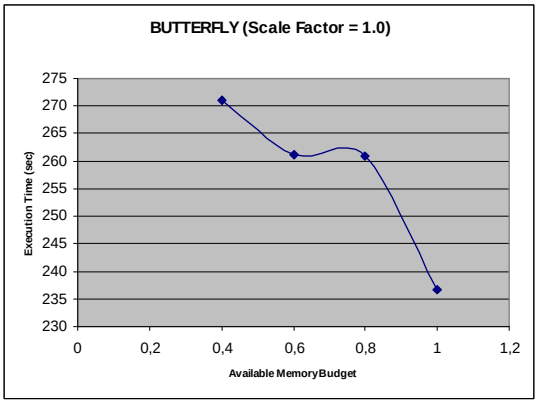
Σχήμα 4.53 Αλγόριθμος Προτεραιότητας Συναθροίσεων σε Σενάριο Πεταλούδας (sf=0.1)

Όπως φαίνεται από το Σχήμα 4.53 για παράγοντα κλιμάκωσης ίσο με 0.1 ο χρόνος εκτέλεσης των πειραμάτων μειώνεται αλλά με πολύ αργό ρυθμό όσο δίνουμε περισσότερη μνήμη στις ακμές που ενώνονται με μία δραστηριότητα συνάθροισης.

Το ίδιο φαινόμενο παρατηρείται και για τα πειράματα που έγιναν με δεδομένα εισόδου με παράγοντα κλιμάκωσης ίσο με 0.5 (Σχήμα 4.54). Επίσης, και στα πειράματα με παράγοντα κλιμάκωσης 1.0 (Σχήμα 4.55) παρατηρείται συνεχόμενη μείωση στον χρόνο εκτέλεσης του σεναρίου που φτάνει μέχρι και την τάξη του 13% όταν πλέον έχει δοθεί όλη διαθέσιμη μνήμη.



Σχήμα 4.54 Αλγόριθμος Προτεραιότητας Συναθροίσεων σε Σενάριο Πεταλούδας (sf=0.5)



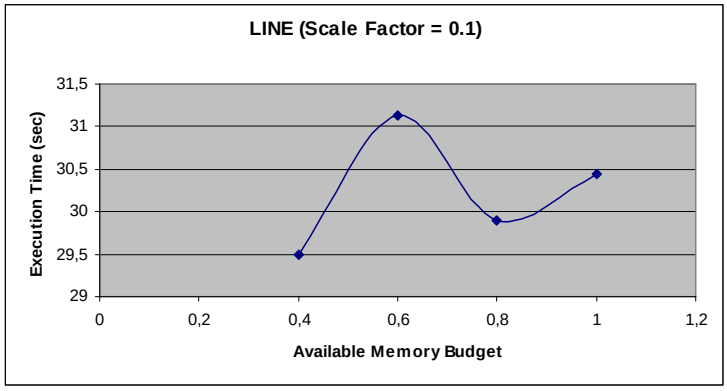
Σχήμα 4.55 Αλγόριθμος Προτεραιότητας Συναθροίσεων σε Σενάριο Πεταλούδας (sf=1.0)

Συμπερασματικά, στο Σενάριο της Πεταλούδας εάν ευνοήσουμε τις δραστηριότητες των συναθροίσεων χρόνος εκτέλεσης του σεναρίου μειώνεται συνεχώς αλλά με αργό ρυθμό. Και αυτό είναι λογικό γιατί το Σενάριο της Πεταλούδας στο δεξί φτερό έχει

πολλές δραστηριότητες συνάθροισης και δίνοντας σε αυτές επιπλέον μνήμη διευκολύνεται και επιταχύνεται η εκτέλεση του σεναρίου και έτσι δεν δημιουργείται συμφόρηση στο σενάριο.

4.8.2. Γραμμικό Σενάριο

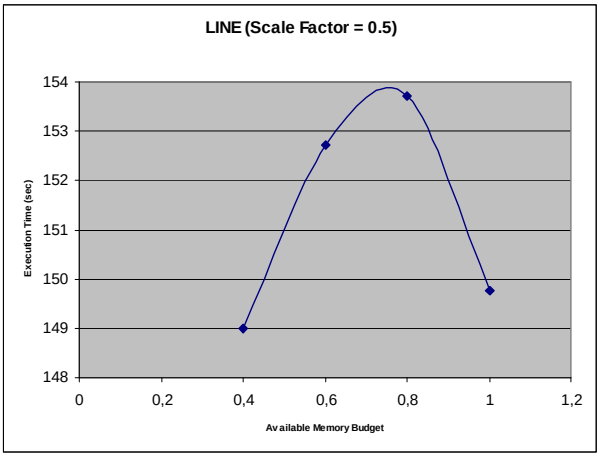
Στην περίπτωση του εκτελείται ο Αλγόριθμος Προτεραιότητας Συναθροίσεων με είσοδο το Γραμμικό Σενάριο και για παράγοντα κλιμάκωσης ίσο με 0.1 (Σχήμα 4.56) παρατηρούμε λίγο περίεργη συμπεριφορά. Αρχικά ο χρόνος εκτέλεσης παρουσιάζει αύξηση (διάστημα 40-60% της διαθέσιμης μνήμης), στην συνέχεια εμφανίζει μείωση (διάστημα 60-80% της μνήμης) και τέλος έχει μία μικρή άνοδο μέχρι να δοθεί όλη η διαθέσιμη μνήμη. Πάντως σε γενικές γραμμές οι διαφορές στους χρόνους για διαφορετικές τιμές εκχώρησης μνήμης είναι πολύ μικρές και για αυτό δεν μπορούν να βγουν ασφαλή συμπεράσματα.



Σχήμα 4.56 Αλγόριθμος Προτεραιότητας Συναθροίσεων σε Γραμμικό Σενάριο (sf=0.1)

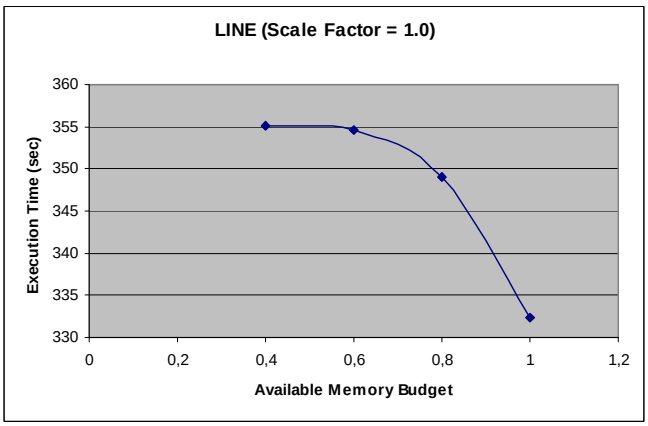
Περίεργη συμπεριφορά εμφανίζουν και τα πειράματα που έγιναν με παράγοντα κλιμάκωσης 0.5 (Σχήμα 4.57). Εκεί, παρατηρούμε μία συνεχή αύξηση του χρόνου μέχρι να δοθεί το 80% της διαθέσιμης μνήμης και από εκεί και πέρα μία ραγδαία πτώση μέχρι να δοθεί όλη η διαθέσιμη μνήμη. Αλλά και πάλι οι χρόνοι εκτέλεσης

παρουσιάζουν πολύ μικρές διαφορές και για αυτό δεν μπορούμε να βγάλουμε ακριβή συμπεράσματα.



Σχήμα 4.57 Αλγόριθμος Προτεραιότητας Συναθροίσεων σε Γραμμικό Σενάριο (sf=0.5)

Τέλος, για παράγοντα κλιμάκωσης ίσο με 1.0 (Σχήμα 4.58) παρατηρούμε μία συνεχόμενη πτώση στου χρόνους εκτέλεσης των πειραμάτων όσο αυξάνεται το ποσοστό εκχώρησης μνήμης στις δραστηριότητες συνάθροισης του σεναρίου.

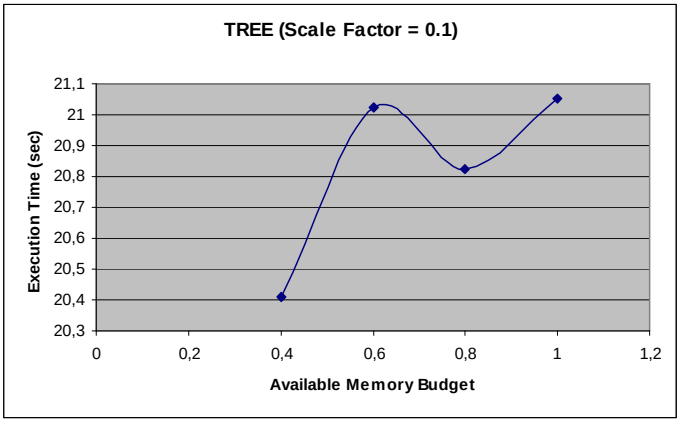


Σχήμα 4.58 Αλγόριθμος Προτεραιότητας Συναθροίσεων σε Γραμμικό Σενάριο (sf=1.0)

Σαν συμπέρασμα για το Γραμμικό Σενάριο με προτεραιότητα στις συναθροίσεις βλέπουμε διαφορετικά πράγματα όσο αλλάζει ο παράγοντας κλιμάκωσης. Για 0.1 έχουμε σκαμπανεβάσματα, για 0.5 έχουμε αρχικά άνοδο του χρόνου εκτέλεσης και μετά μείωση και τέλος για 1.0 έχουμε συνεχόμενη μείωση στο χρόνο εκτέλεσης του σεναρίου.

4.8.3. Σενάριο Δένδρου

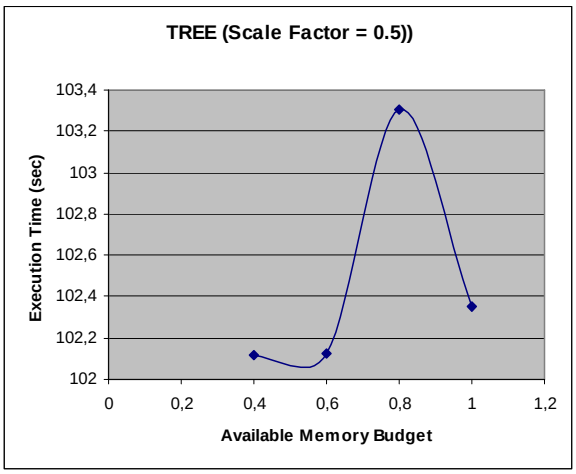
Στην περίπτωση που εφαρμόζεται ο Αλγόριθμος Προτεραιότητας Συναθροίσεων σε Σενάριο Δένδρου παρατηρούμε τις εξής συμπεριφορές: για παράγοντα κλιμάκωσης ίσο με 0.1 (Σχήμα 4.51) έχουμε άνοδο του χρόνου εκτέλεσης των πειραμάτων όσο δίνεται περισσότερο μνήμη στις δραστηριότητες του σεναρίου που είναι συναθροίσεις. Πάντως, ανάμεσα στις διαφορετικές εκχωρήσεις μνήμης παρατηρούνται πολύ μικρές αυξομειώσεις στους χρόνους εκτέλεσης των πειραμάτων.



Σχήμα 4.59 Αλγόριθμος Προτεραιότητας Συναθροίσεων σε Σενάριο Δένδρου (sf=0.1)

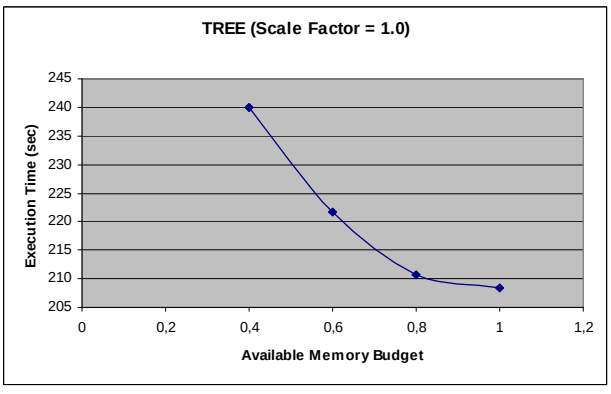
Για παράγοντα κλιμάκωσης ίσο με 0.5 (Σχήμα 4.52) παρατηρούνται κάποιες αυξομειώσεις στου χρόνους εκτέλεσης των πειραμάτων με διαφορετικά ποσοστά εκχώρησης μνήμης κάθε φορά. Αρχικά (διάστημα 40-60%) παρατηρείται μία φυσιολογική μείωση στους χρόνους εκτέλεσης. Στην συνέχει εμφανίζεται μία αύξησης το διάστημα 60-80% της διανομής της μνήμης και καταλήγει σε μείωση

όταν έχει δοθεί πλέον όλη η διαθέσιμη μνήμη. Πάντως δεν μπορούν να βγουν ασφαλή συμπεράσματα γιατί και οι μειώσεις αλλά και οι αυξήσεις που παρατηρούνται είναι της τάξης του 1%.



Σχήμα 4.60 Αλγόριθμος Προτεραιότητας Συναθροίσεων σε Σενάριο Δένδρου (sf=0.5)

Τέλος, για παράγοντα κλιμάκωσης ίσο με 1.0 (Σχήμα 4.53) παρατηρείται συνεχόμενη μείωση στους χρόνους εκτέλεσης του Γραμμικού Σεναρίου όσο δίνεται περισσότερη μνήμη στις δραστηριότητες συνάθροισης που περιέχει. Η μείωση του χρόνου αυτού φτάνει μέχρι και το 15% σε σχέση με τον αρχικό χρόνο εκτέλεσης και συναντάτε όταν πλέον έχει δοθεί στο σενάριο όλη η διαθέσιμη μνήμη.

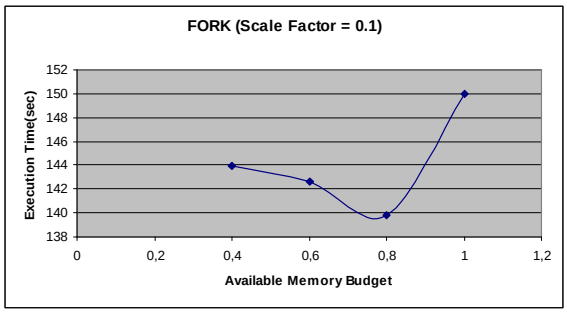


Σχήμα 4.61 Αλγόριθμος Προτεραιότητας Συναθροίσεων σε Σενάριο Δένδρου (sf=1.0)

Συμπερασματικά, για το Σενάριο Δένδρου για παράγοντα κλιμάκωσης 0.1 έχουμε άνοδο του χρόνου εκτέλεσης των πειραμάτων, για παράγοντα κλιμάκωσης 0.5 έχουμε άνοδο αρχικά και μετά μείωση στο χρόνο εκτέλεσης και τέλος για παράγοντα κλιμάκωσης 1.0 έχουμε συνεχόμενη μείωση.

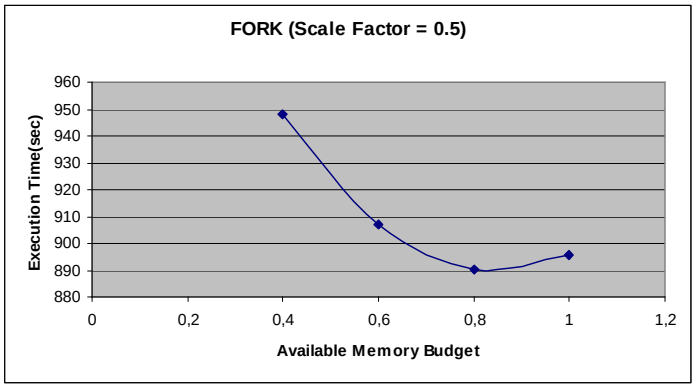
4.8.4. Σενάριο Πολλών Εκροών

Για την περίπτωση του Σεναρίου Πολλών Εκροών για παράγοντα κλιμάκωσης 0.1 (Σχήμα 4.62) παρατηρούμε μείωση στο χρόνο εκτέλεσης της τάξης του 3% μέχρι το 80% της συνολικής μνήμη και από εκεί και μετά άνοδο της τάξης του 7%.



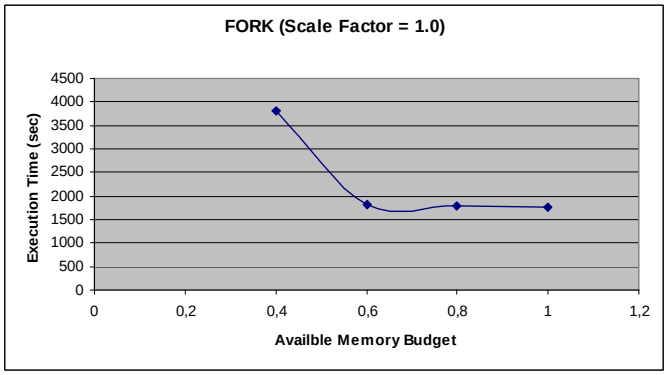
Σχήμα 4.62 Αλγόριθμος Προτεραιότητας Συναθροίσεων σε Σενάριο Πολλών Εκροών (sf=0.1)

Για παράγοντα κλιμάκωσης 0.5 (Σχήμα 4.63) έχουμε πτώση του χρόνου εκτέλεσης του σεναρίου μέχρι το 80% της συνολικής διαθέσιμης μνήμης. Η πτώση αυτή φθάνει στο 7% του αρχικού χρόνου. Έπειτα μέχρι να δοθεί όλη η μνήμη έχουμε μία άνοδο στον χρόνο της τάξης του 1%.



Σχήμα 4.63 Αλγόριθμος Προτεραιότητας Συναθροίσεων σε Σενάριο Πολλών Εκροών (sf=0.5)

Ενώ για παράγοντα κλιμάκωσης 1.0 (4.64) έχουμε συνεχόμενη μείωση του χρόνου εκτέλεσης του σεναρίου όσο δίνουμε μνήμη στις συναθροίσεις και αυτή η μείωση φθάνει μέχρι και το 54% του αρχικού χρόνου.



Σχήμα 4.64 Αλγόριθμος Προτεραιότητας Συναθροίσεων σε Σενάριο Πολλών Εκροών (sf=1.0)

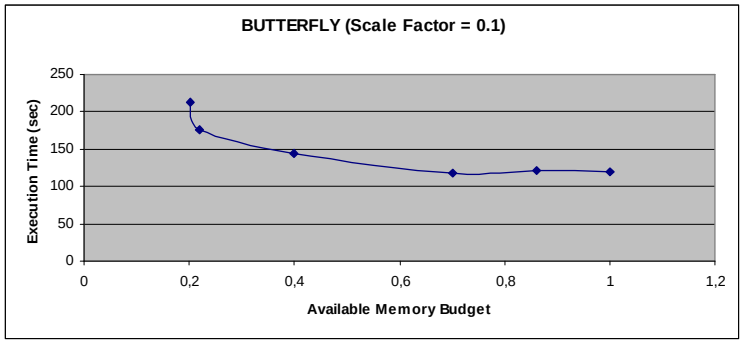
4.9. Εκχώρηση Μνήμης με Εύνοια των Δραστηριοτήτων Ταξινόμησης

Στην παράγραφο αυτή παρουσιάζονται πειράματα που έγιναν εκτελώντας τον *Αλγόριθμο Προτεραιότητας Ταξινόμησης* χρησιμοποιώντας σαν είσοδο το Σενάριο της Πεταλούδας (Σχήμα 4.7) και το Σενάριο Δένδρου (Σχήμα 4.8) και τα δεδομένα εισόδου παρήχθησαν με παράγοντες κλιμάκωσης ίσους με 0.1, 0.5 και 1.0.

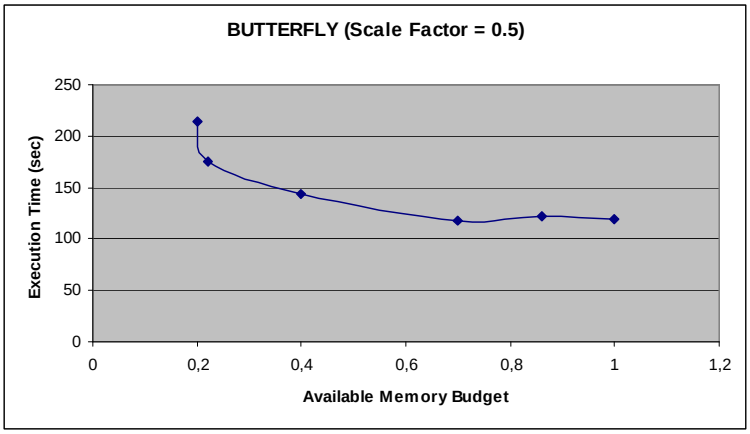
Όπως ειπώθηκε και πριν με βάση τα πειράματα που έγιναν χρησιμοποιώντας τον *Δίκαιο Αλγόριθμο* διανομή μνήμης, βρέθηκε ότι το κρίσιμο σημείο κατά την όλη διαδικασία διανομής είναι το 20% της συνολικής διαθέσιμης μνήμης. Γι' αυτό το λόγω τα πειράματα που διεξήχθησαν σε αυτή την παράγραφο διανέμουν το 20% της συνολικής μνήμης σε όλες τις ακμές και έπειτα διανέμουν την υπόλοιπη στις ακμές που ενώνονται στην είσοδό τους με μία δραστηριότητα ταξινόμησης ή μία εκ των δραστηριοτήτων συνάθροισης ή συνένωσης που κρύβουν εσωτερικά την δραστηριότητα ταξινόμησης.

4.9.1. Σενάριο Πεταλούδας

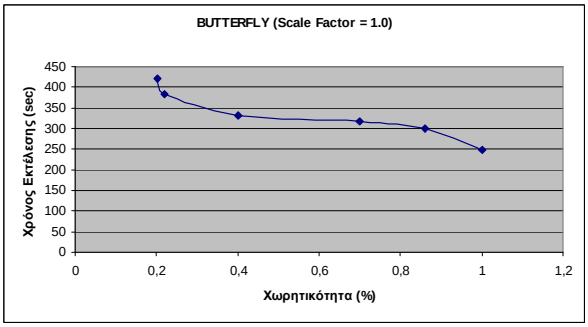
Στην περίπτωση του Σεναρίου της Πεταλούδας παρατηρούμε στην συμπεριφορά του όταν χρησιμοποιείται ο Αλγόριθμος Προτεραιότητας Ταξινομήσεων μία μεγάλη πτώση στους χρόνους εκτέλεσης των πειραμάτων και για του τρεις διαφορετικούς παράγοντες κλιμάκωσης (0.1, 1.0, 0.5). Χαρακτηριστικά για τον παράγοντα κλιμάκωσης 0.1 (Σχήμα 4.65) φτάνει μέχρι και 60% η μείωση του χρόνου εκτέλεσης, για τον παράγοντα κλιμάκωσης 0.5 (Σχήμα 4.66) φτάνει η μείωση το 47% όταν χρησιμοποιείται το 70% της συνολικής μνήμης και για παράγοντα κλιμάκωσης 1.0 (Σχήμα 4.67) φτάνει μέχρι το 41% η πτώση του χρόνου όταν πλέον έχει χρησιμοποιηθεί όλη η διαθέσιμη μνήμη.



Σχήμα 4.65 Αλγόριθμος Προτεραιότητας Ταξινομήσεων σε Σενάριο Πεταλούδας (sf=0.1)



Σχήμα 4.66 Αλγόριθμος Προτεραιότητας Ταξινομήσεων σε Σενάριο Πεταλούδας (sf=0.5)

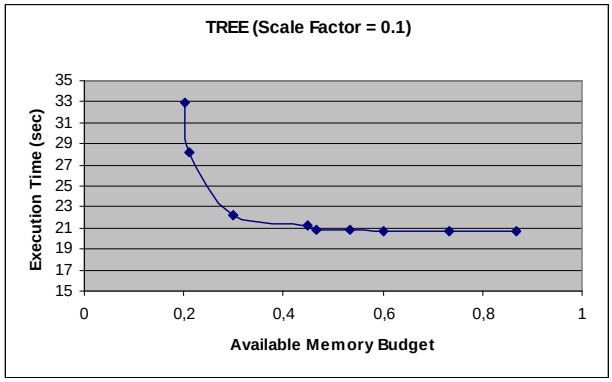


Σχήμα 4.67 Αλγόριθμος Προτεραιότητας Ταξινομήσεων σε Σενάριο Πεταλούδας (sf=1.0)

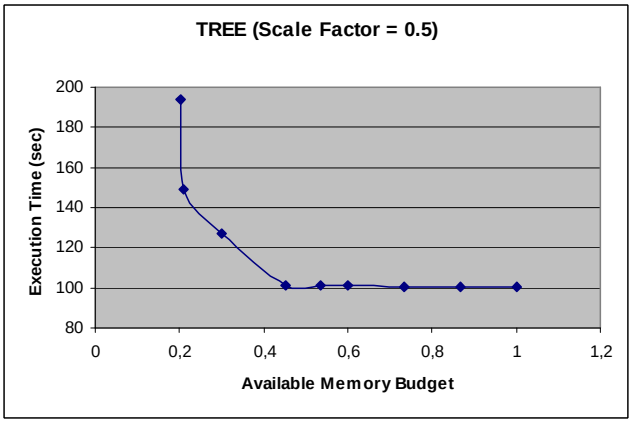
4.9.2. Σενάριο Δένδρου

Όμοια συμπεριφορά με την περίπτωση του Σεναρίου της Πεταλούδας παρατηρούμε και στην συμπεριφορά του Δενδρικού Σεναρίου όταν χρησιμοποιείται ο Αλγόριθμος Προτεραιότητας Ταξινομήσεων. Έτσι, παρατηρείται μεγάλη πτώση στους χρόνους εκτέλεσης των πειραμάτων και για του τρεις διαφορετικούς παράγοντες κλιμάκωσης (0.1, 1.0, 0.5). Χαρακτηριστικά για τον παράγοντα κλιμάκωσης 0.1 (Σχήμα 4.68) φτάνει μέχρι και 37% η μείωση του χρόνου εκτέλεσης, για τον παράγοντα

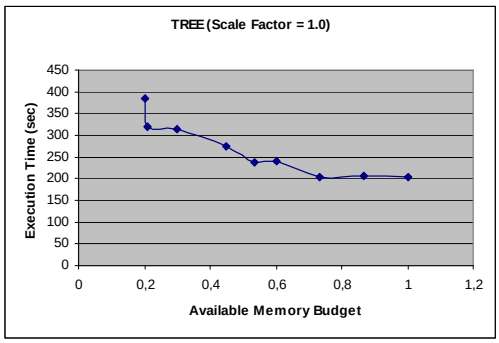
κλιμάκωσης 0.5 (Σχήμα 4.69) φτάνει η μείωση το 48% και για παράγοντα κλιμάκωσης 1.0 (Σχήμα 4.70) φτάνει μέχρι το 48% η πτώση του χρόνου όταν πλέον έχει χρησιμοποιηθεί όλη η διαθέσιμη μνήμη.



Σχήμα 4.68 Αλγόριθμος Προτεραιότητας Ταξινομήσεων σε Σενάριο Δένδρου (sf=0.1)



Σχήμα 4.69 Αλγόριθμος Προτεραιότητας Ταξινομήσεων σε Σενάριο Δένδρου (sf=0.5)

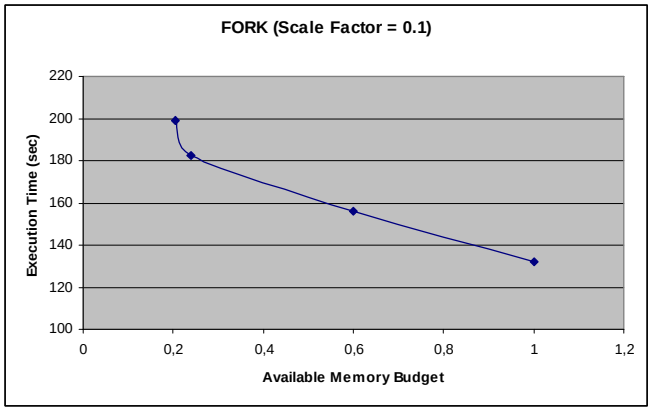


Σχήμα 4.70 Αλγόριθμος Προτεραιότητας Ταξινομήσεων σε Σενάριο Δένδρου (sf=1.0)

4.9.3. Σενάριο Πολλών Εκροών

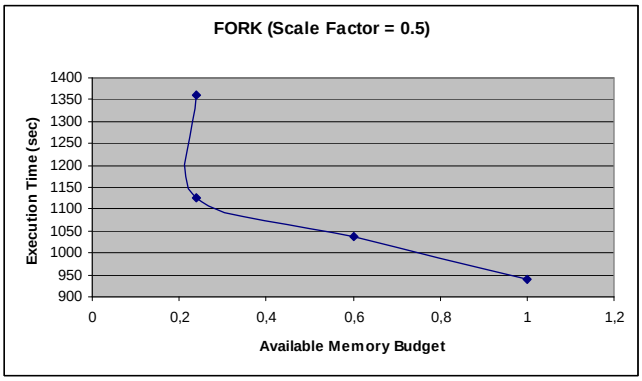
Σε όλες τις περιπτώσεις για διαφορετικούς παράγοντες κλιμάκωσης το Σενάριο Πολλών Εκροών όταν δίνουμε εύνοια στις δραστηριότητες ταξινόμησης εμφανίζει την ίδια συμπεριφορά. Παρατηρούμε ότι έχει συνεχόμενη μείωση του χρόνου εκτέλεσης το σενάριο όσο δίνουμε περισσότερη μνήμη στις δραστηριότητες ταξινομήσεων.

Για παράγοντα κλιμάκωσης 0.1 (Σχήμα 4.71) η μείωση φθάνει στο 38% του αρχικού χρόνου.

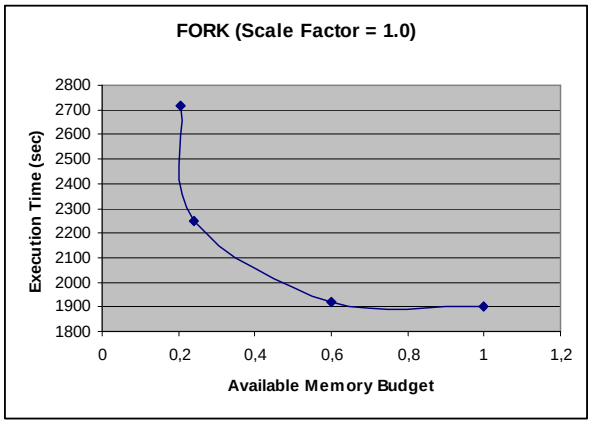


Σχήμα 4.71 Αλγόριθμος Προτεραιότητας Ταξινομήσεων σε Σενάριο Πολλών Εκροών (sf=0.1)

Για παράγοντα κλιμάκωσης 0.5 (Σχήμα 4.72) η μείωση φθάνει μέχρι και το 30% του αρχικού χρόνου, ενώ για παράγοντα κλιμάκωσης 1.0 (Σχήμα 4.73) φθάνει και αυτό στο 30% του αρχικού χρόνου εκτέλεσης.



Σχήμα 4.72 Αλγόριθμος Προτεραιότητας Ταξινομήσεων σε Σενάριο Πολλών Εκροών (sf=0.5)



Σχήμα 4.73 Αλγόριθμος Προτεραιότητας Ταξινομήσεων σε Σενάριο Πολλών Εκροών (sf=1.0)

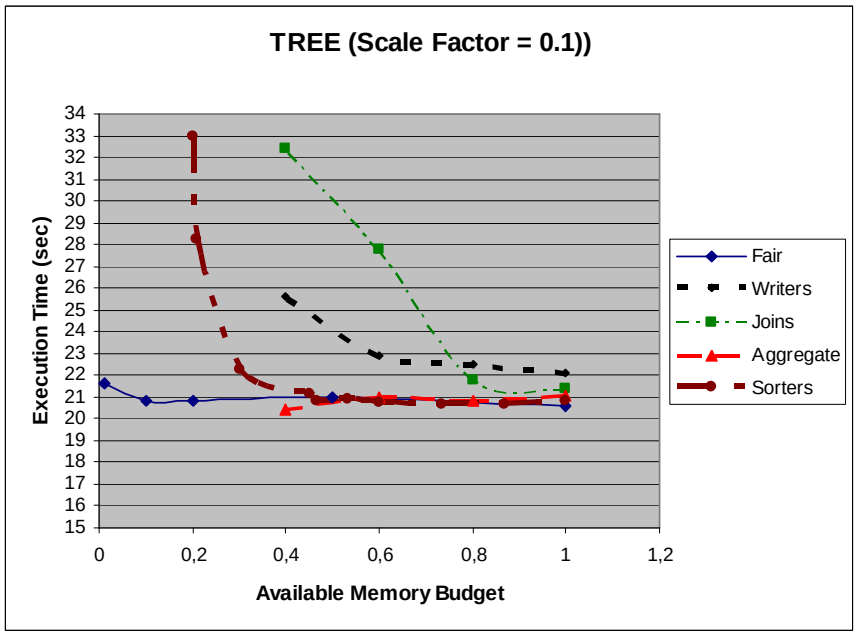
4.10. Συγκριτική Αξιολόγηση των Διαφορετικών Αλγορίθμων

Σε αυτή την ενότητα παρατίθεται μία συγκριτική αξιολόγηση των διαφορετικών αλγορίθμων σε τέσσερα διαφορετικά σενάρια. Τα σενάρια στα οποία συγκρίνονται οι αλγόριθμοι είναι τα εξής: Σενάριο Δένδρου, Σενάριο Πεταλούδας, Σενάριο Πολλών

Εκροών και Γραμμικό Σενάριο. Τα δεδομένα που χρησιμοποιήθηκαν σαν είσοδο στα σενάρια παρήχθησαν με τρεις διαφορετικούς παράγοντες κλιμάκωσης 0.1, 0.5 και 1.0.

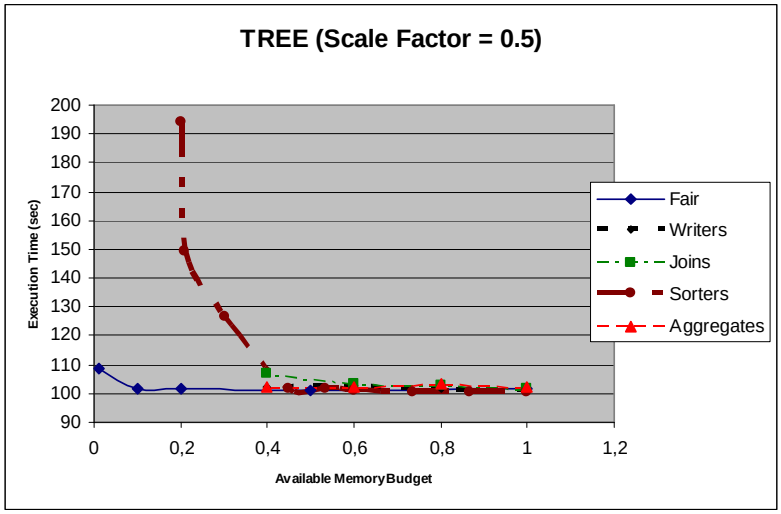
4.10.1. Σενάριο Δένδρου

Στη περίπτωση του Σεναρίου Δένδρου για παράγοντα κλιμάκωσης ίσο με 0.1 παρατηρούμε ότι οι Αλγόριθμοι Προτεραιότητας Ταξινομήσεων και Συνάθροισης δίνουν σχεδόν τους ίδιους χρόνους με τον Δίκαιο Αλγόριθμο από το 50% και πέρα της χρήσης της διαθέσιμης μνήμης (Σχήμα 4.74).



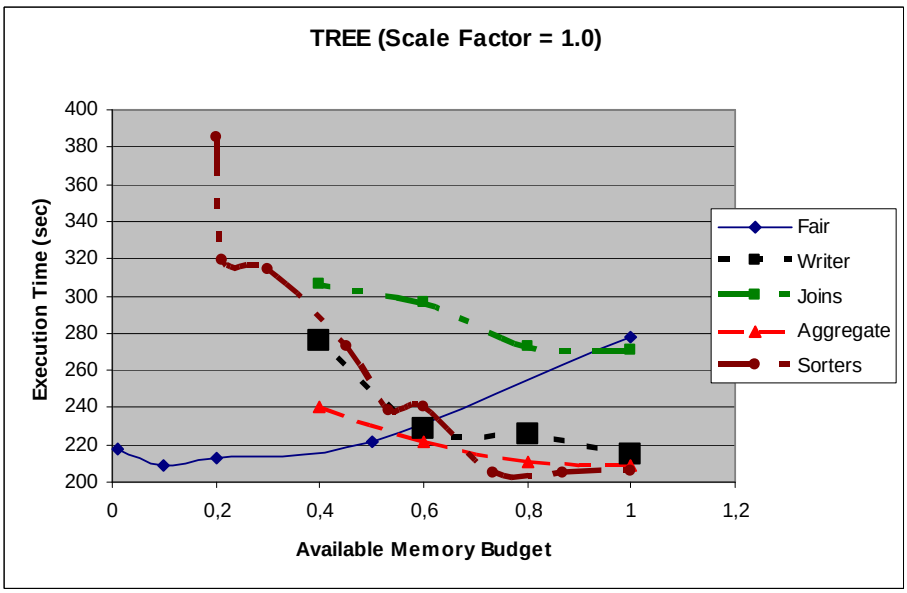
Σχήμα 4.74 Αξιολόγηση Σεναρίου Δένδρου με sf=0.1

Τώρα στην περίπτωση που έχουμε παράγοντα κλιμάκωσης ίσο με 0.5 έχουμε όλους τους αλγορίθμους να έχουν τους ίδιους χρόνους εκτέλεσης από το 40% και πέρα της διαθέσιμης μνήμης (Σχήμα 4.75).



Σχήμα 4.75 Αξιολόγηση Σεναρίου Δένδρου με sf=0.5

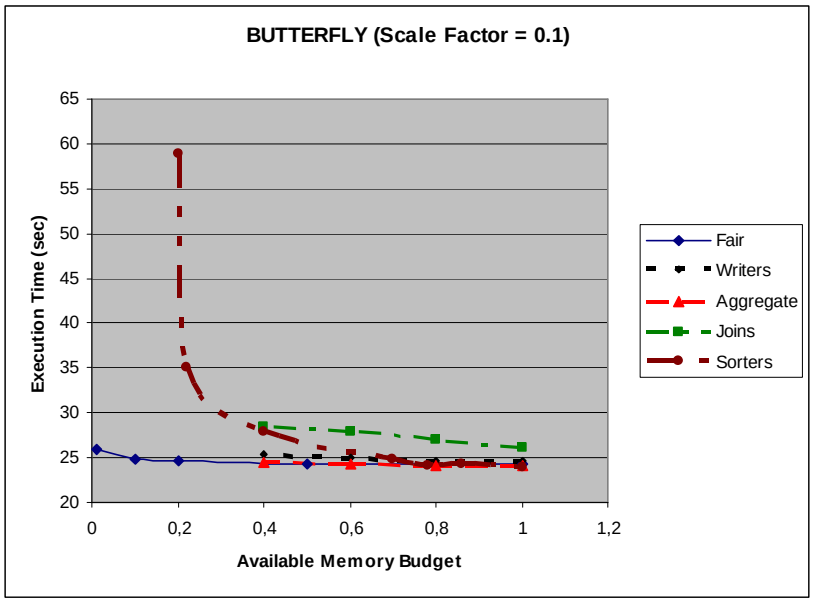
Σε αντίθεση με ότι παρατηρήσαμε στους δύο προηγούμενους παράγοντες κλιμάκωσης, για παράγοντα κλιμάκωσης ίσο με 1.0 (Σχήμα 4.76) έχουμε με διαφορά τρεις αλγορίθμους να ξεχωρίζουν από τους υπολοίπους (Αλγόριθμος Προτεραιότητας Ταξινομήσεων, Συναθροίσεων και Δραστηριοτήτων Εγγραφής) και τελικά να κερδίζει ο Αλγόριθμος Προτεραιότητας Ταξινομήσεων έστω και με μικρή διαφορά όταν πλέον έχει δοθεί όλη η διαθέσιμη μνήμη.



Σχήμα 4.76 Αξιολόγηση Σεναρίου Δένδρου με sf=1.0

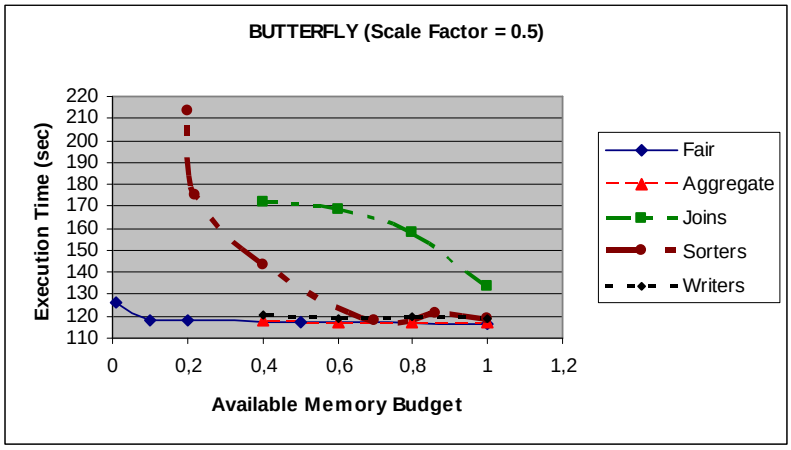
4.10.2. Σενάριο Πεταλούδας

Στο Σενάριο της Πεταλούδας για παράγοντα κλιμάκωσης ίσο με 0.1 (Σχήμα 4.77) παρατηρούμε ότι μετά το 50% της διανομής της διαθέσιμης μνήμης όλοι οι αλγόριθμοι εκτός του Αλγόριθμο Προτεραιότητας Συνενώσεων έχουν του ίδιους χρόνους.



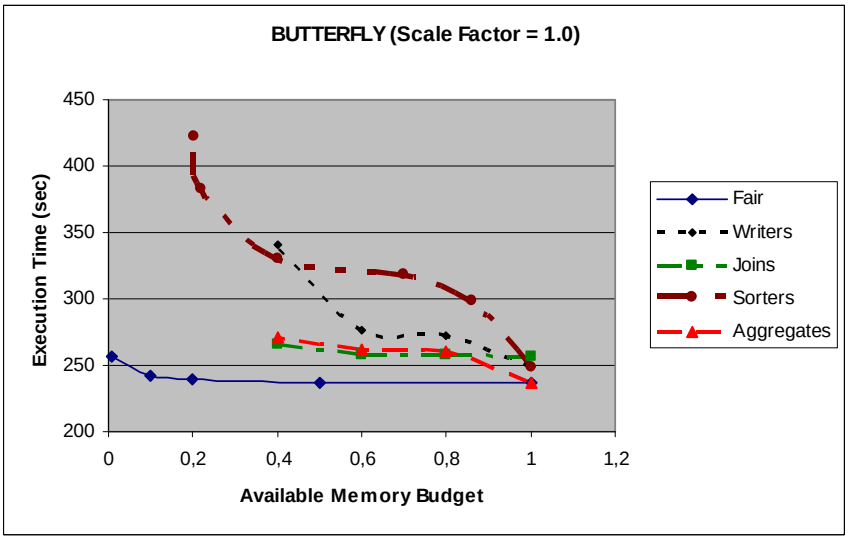
Σχήμα 4.77 Αξιολόγηση Σεναρίου Πεταλούδας με sf=0.1

Τώρα για παράγοντα κλιμάκωσης ίσο με 0.5 (Σχήμα 4.78) βλέπουμε ότι κερδίζει έστω και μικρή διαφορά ο Αλγόριθμος Προτεραιότητας Συναθροίσεων και από κοντά του είναι ο Δίκαιος Αλγόριθμος. Πάντως ο Αλγόριθμος Προτεραιότητας Συνενώσεων συνεχίζει να έχει την χειρότερη συμπεριφορά και κατ' επέκταση χειρότερους χρόνους.



Σχήμα 4.78 Αξιολόγηση Σεναρίου Πεταλούδας με sf=0.5

Τέλος, για παράγοντα κλιμάκωσης ίσο με 1.0 (Σχήμα 4.79) γίνονται πιο ευδιάκριτα τα πράγματα. Ο Δίκαιος μαζί με τον Αλγόριθμο Προτεραιότητας Συναθροίσεων παρατηρούμε ότι έχουν να έχουν τους καλύτερους χρόνους.

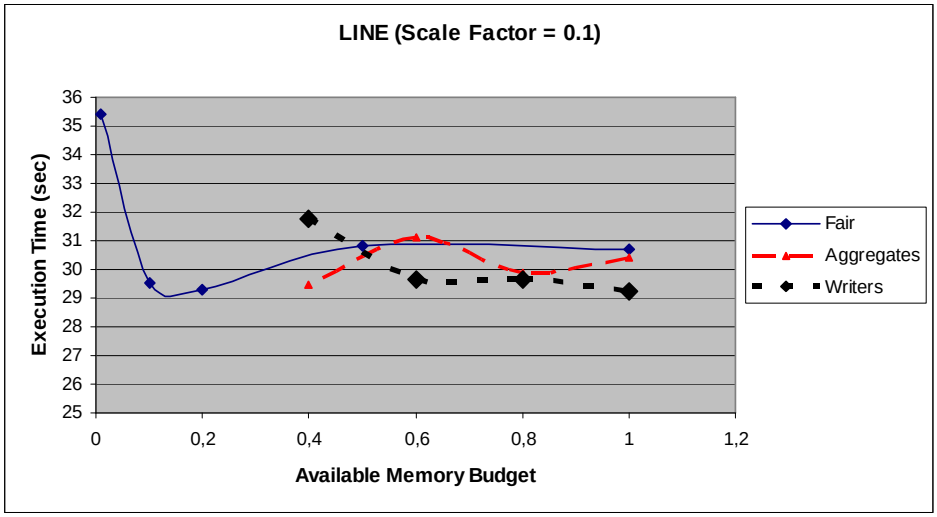


Σχήμα 4.79 Αξιολόγηση Σεναρίου Πεταλούδας με sf=1.0

4.10.3. Γραμμικό Σενάριο

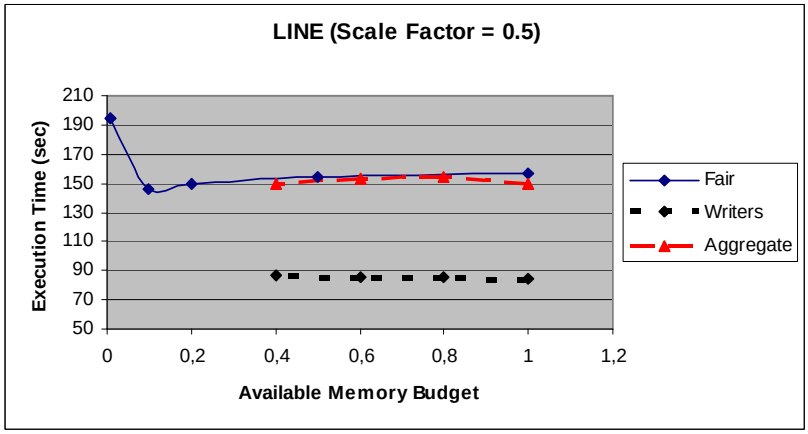
Στην περίπτωση του Γραμμικού Σεναρίου για παράγοντα κλιμάκωσης ίσο με 0.1 (Σχήμα 4.80) παρατηρούμε ότι μικρότερους χρόνους εκτέλεσης έχει ο Αλγόριθμος

Προτεραιότητας Δραστηριοτήτων Εγγραφής και γενικά από το 60% της διαθέσιμης μνήμης και μετά αρχίζει να ξεχωρίζει έναντι των άλλων δύο αλγορίθμων (Δίκαιος και Προτεραιότητας Συναθροίσεων).



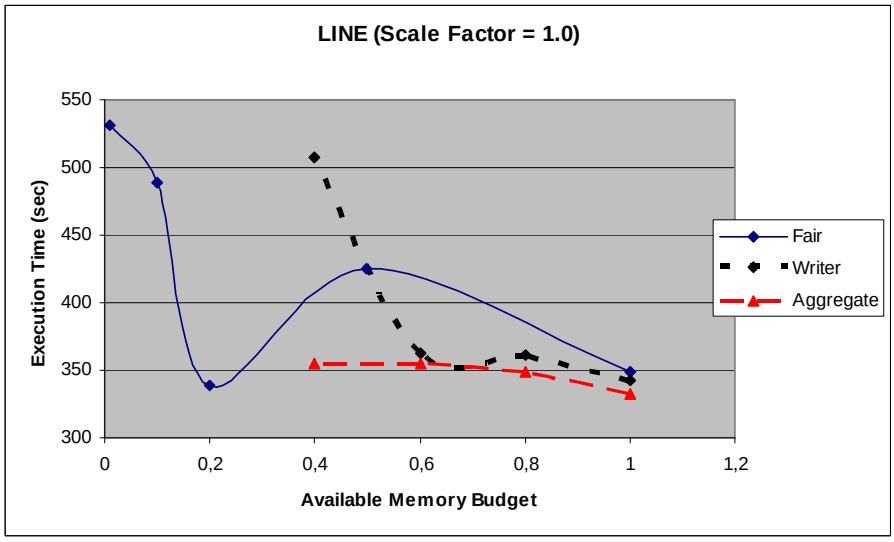
Σχήμα 4.80 Αξιολόγηση Γραμμικού Σεναρίου με sf=0.1

Η ίδια συμπεριφορά παρατηρείται και για παράγοντα κλιμάκωσης ίσο με 0.5 (Σχήμα 4.81) όπου και πάλι μικρότερους χρόνους εκτέλεσης του σεναρίου εμφανίζει ο Αλγόριθμος Προτεραιότητας των Δραστηριοτήτων Εγγραφής και αυτή την φορά με πολύ μεγάλη διαφορά από τους άλλους δύο της τάξης του 50% μείωσης του χρόνου εκτέλεσης.



Σχήμα 4.81 Αξιολόγηση Γραμμικού Σεναρίου με sf=0.5

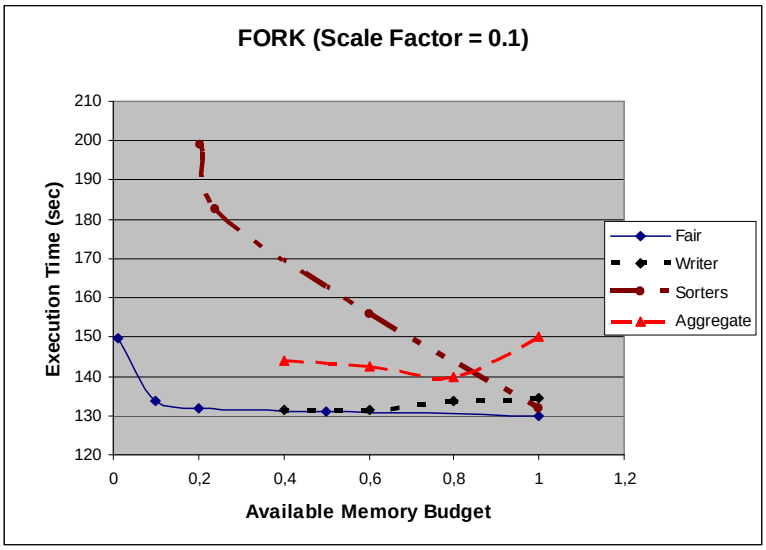
Τέλος, για τα πειράματα που έγιναν με παράγοντα κλιμάκωσης ίσο 1.0 (Σχήμα 4.82) παρατηρούμε μικρότερους χρόνους στον Αλγόριθμος Προτεραιότητας Συναθροίσεων που σε όλη την διάρκεια διανομής μνήμης είχε σταθερούς χρόνους σε σχέση με τους άλλους δύο που για μικρά ποσοστά μνήμης έχουν πολύ μεγάλους χρόνους και τελικά όταν δίνεται το 100% της διαθέσιμης μνήμης πλησιάζουν κατά πολύ τον Αλγόριθμο Προτεραιότητας των Συναθροίσεων.



Σχήμα 4.82 Αξιολόγηση Γραμμικού Σεναρίου με sf=1.0

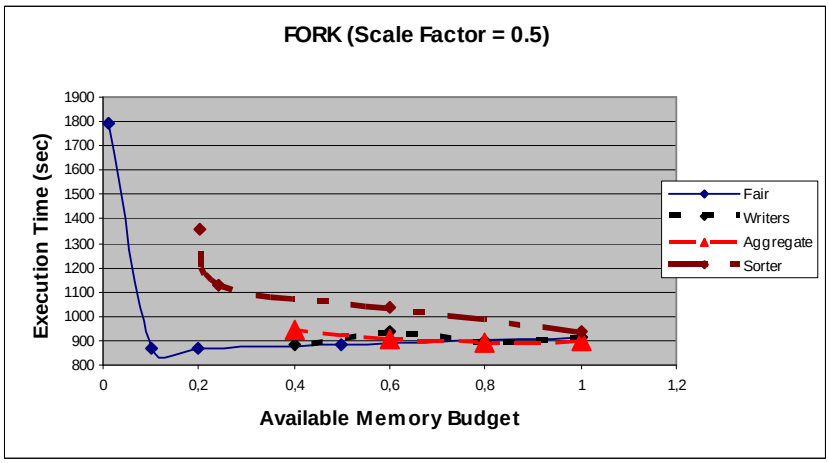
4.10.4. Σενάριο Πολλών Εκροών

Για το Σενάριο Πολλών Εκροών για παράγοντα κλιμάκωσης ίσο με 0.1 (Σχήμα 4.83) παρατηρούμε ότι ο Δίκαιος Αλγόριθμος είναι ο καλύτερος από όλους τους άλλους. Ο Αλγόριθμος Προτεραιότητας των Ταξινομήσεων ξεκινάει με πολύ μεγάλους χρόνους αλλά όταν του δίνουμε όλη τη διαθέσιμη μνήμη φθάνει τους χρόνους του Δίκαιου. Ο Αλγόριθμος Προτεραιότητας των Δραστηριοτήτων Εγγραφής έχει λίγο χειρότερους χρόνους της τάξης του 5%, ενώ ο Αλγόριθμος Προτεραιότητας των Συναθροίσεων είναι ο χειρότερος από όλους.



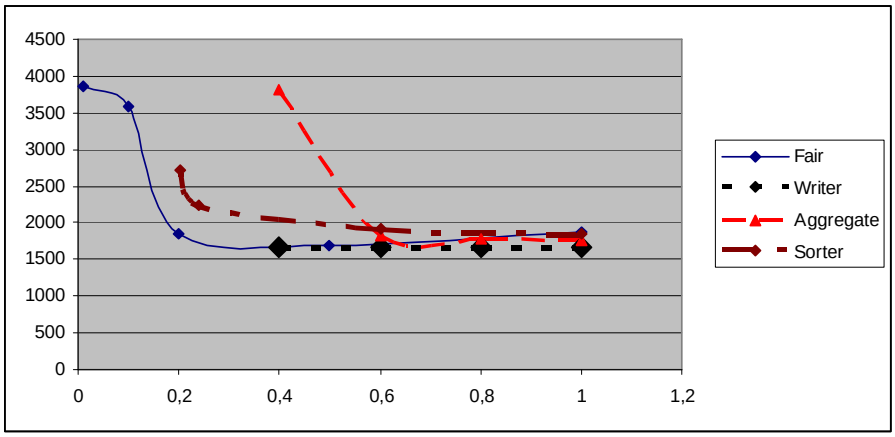
Σχήμα 4.83 Αξιολόγηση Σεναρίου Πολλών Εκροών με sf=0.1

Για παράγοντα κλιμάκωσης 0.5 όλοι οι Αλγόριθμοι ισορροπούν μαζί στο τέλος που δίνουμε όλη την μνήμη του συστήματος. Λίγο καλύτερος εμφανίζεται ο Αλγόριθμος Προτεραιότητας των Συναθροίσεων που κερδίζει στο τέλος τους υπολοίπους.



Σχήμα 4.84 Αξιολόγηση Σεναρίου Πολλών Εκροών με sf=0.5

Τώρα στην περίπτωση για παράγοντα κλιμάκωσης ίσο με 1.0 βλέπου ότι ο καλύτερος είναι ο Αλγόριθμος Προτεραιότητας Δραστηριοτήτων Εγγραφής και από πολύ κοντά όλοι οι υπόλοιποι αλγόριθμοι.



Σχήμα 4.85 Αξιολόγηση Σεναρίου Πολλών Εκροών με sf=1.0

Συμπερασματικά, σε όλες τις περιπτώσεις όταν δίνουμε πολύ λίγη μνήμη έχουμε μεγάλους χρόνους εκτέλεσης των σεναρίων και διαφοροποιήσεις ανάλογα τον αλγόριθμο. Ενώ, όσο δίνουμε την διαθέσιμη μνήμη τόσο πέφτουν οι χρόνοι εκτέλεσης των πειραμάτων και επί το πλείστον ισορροπούν οι περισσότεροι αλγόριθμοι στα ίδιο σημείο. Για το Σενάριο Δένδρου καλύτεροι είναι οι: Δίκαιος Αλγόριθμος, Αλγόριθμος Προτεραιότητας Ταξινομήσεων και Αλγόριθμος Προτεραιότητας Συναθροίσεων. Για το Σενάριο Πεταλούδας πάλι καλύτεροι είναι ο Δίκαιος Αλγόριθμος, ο Αλγόριθμος Προτεραιότητας Ταξινομήσεων και ο Αλγόριθμος Προτεραιότητας Συναθροίσεων. Για το Γραμμικό Σενάριο κερδίζει ο Αλγόριθμος Προτεραιότητας Δραστηριοτήτων Εγγραφής για παράγοντες κλιμάκωσης 0.1 και 0.5, ενώ για παράγοντα κλιμάκωσης 1.0 κερδίζει ο Αλγόριθμος Προτεραιότητας Συναθροίσεων. Τέλος, για το Σενάριο Πολλών Εκροών ο Δίκαιος Αλγόριθμος έχει την καλύτερη συμπεριφορά από όλους τους άλλους.

ΚΕΦΑΛΑΙΟ 5. ΕΠΙΛΟΓΟΣ

5.1. Ανακεφαλαίωση

Όπως έχει ειπωθεί προηγουμένως, οι Αποθήκες Δεδομένων είναι συλλογές δεδομένων που προέρχονται από διαφορετικές πηγές και χρησιμοποιούνται κυρίως για τη λήψη αποφάσεων σε ένα οργανισμό. Για να τροφοδοτηθεί μια αποθήκη με νέα δεδομένα, όπως αυτά παράγονται στις πηγές, χρησιμοποιούνται εργαλεία Εξαγωγής – Μετασχηματισμού – Φόρτωσης δεδομένων (Extract – Transform – Load tools, ETL), τα οποία οργανώνουν τα επί μέρους βήματα της όλης διαδικασίας σαν μία ροή εργασίας. Μία ροή εργασίας ETL μπορεί να θεωρηθεί ως ένας κατευθυνόμενο ακυκλικό γράφημα που χρησιμοποιείται για να αναπαραστήσει τη ροή δεδομένων από τις πηγές δεδομένων προς την αποθήκη δεδομένων. Οι κόμβοι του γράφου είναι διαδικασίες καθαρισμού/ μετασχηματισμού δεδομένων ή σύνολα εγγραφών και οι ακμές σχέσεις εισόδου/εξόδου μεταξύ των κόμβων. Η ροή εργασίας είναι ένα αφηρημένο σχήμα σε λογικό επίπεδο, το οποίο πρέπει να υλοποιηθεί σε φυσικό επίπεδο, δηλαδή να αντιστοιχηθεί σε ένα συνδυασμό από εκτελέσιμα προγράμματα που εκτελούν την ETL ροή εργασίας.

Σε επίπεδο λογισμικού, μεταβίβαση των δεδομένων μεταξύ των κόμβων της ροής εργασίας υλοποιείται με ουρές που επικοινωνούν μεταξύ τους. Στην μέχρι τώρα βιβλιογραφία δεν έχει προταθεί κάποια αποδοτική μέθοδος για την ρύθμιση των

πόρων σε ροές εργασίας ETL. Όλες οι μέθοδοι που έχουν προταθεί μέχρι σήμερα αφορούν πλάνα εκτέλεσης κυρίως για αριστεροβαθή [HuSS00] δένδρα και δεν λαμβάνουν υπόψη τους τις ιδιαιτερότητες των ροών εργασίας ETL ή αφορούν πολιτικές για τον καλύτερο χρονοπρογραμματισμό αυτών [Kara07]. Έτσι, στην εργασία αυτή μελετάται η όσο το δυνατό αποδοτικότερη ρύθμιση της απόδοσης πόρων σε ροές εργασίας ETL, δηλαδή η αποδοτικότερη διανομή της διαθέσιμης μνήμης ανάμεσα στις διάφορες δραστηριότητες. Η διανομή της διαθέσιμης μνήμης γίνεται με διάφορες τεχνικές όπως: εύνοια δραστηριοτήτων συνένωσης, εύνοια δραστηριοτήτων συνάθροισης, εύνοια δραστηριοτήτων εγγραφής και εύνοια δραστηριοτήτων ταξινόμησης.

Η εργασία αυτή συνεισφέρει στον τομέα των ETL εργαλείων την μοντελοποίηση ενός θεωρητικού προβλήματος που επιλύθηκε αναπτύσσοντας αρχικά κάποιες μεθόδους ρύθμιση των πόρων με μικρο-ελέγχους. Στην συνέχεια εξετάστηκαν εναλλακτικές στρατηγικές απόδοσης της μνήμης όπως: δίκαιη διαμοίραση μνήμης, εύνοια δραστηριοτήτων συνένωσης, εύνοια δραστηριοτήτων συνάθροισης, εύνοια δραστηριοτήτων εγγραφής και εύνοια δραστηριοτήτων ταξινόμησης. Στην δίκαιη διαμοίραση μνήμης σε όλες οι δραστηριότητες μίας ροής εργασιών ETL διανέμεται το ίδιο ποσοστό μνήμης. Στην περίπτωση της εύνοιας δραστηριοτήτων συνένωσης δίνεται περισσότερη μνήμη σε όλες τις δραστηριότητες της ροής εργασιών ETL που είναι δραστηριότητες συνένωσης. Δηλαδή, οι ουρές που συνδέονται με αυτού του τύπου τις δραστηριότητες θα πάρουν περισσότερη μνήμη από ότι θα πάρουν οι υπόλοιπες μέσα στο γράφημα. Στην στρατηγική εύνοιας δραστηριοτήτων συνάθροισης δίνεται εύνοια σε όσες από τις δραστηριότητες του γραφήματος που κάνουν κάποιου είδους συναθροίσεις, δηλαδή δίνεται περισσότερη μνήμη στις ουρές που συνδέονται με δραστηριότητες συνάθροισης. Στην εύνοια δραστηριοτήτων εγγραφής ευνοούνται οι δραστηριότητες που γράφουν τα δεδομένα τους σε μία Αποθήκη Δεδομένων. Αυτό έχει σαν αποτέλεσμα να παίρνουν μεγαλύτερο ποσοστό μνήμης από ότι οι υπόλοιπες του γραφήματος. Τέλος στην εύνοια δραστηριοτήτων ταξινόμησης ευνοούνται οι δραστηριότητες ταξινόμησης, είτε αυτές λειτουργούν μόνες τους, είτε ως μέρος άλλων δραστηριοτήτων (όπως στην περίπτωση των δραστηριοτήτων συνάθροισης ή συνένωσης). Έτσι, σε αυτή την περίπτωση

διανέμεται μεγαλύτερο ποσοστό μνήμης στις ουρές που συνδέονται με αυτού του τύπου τις δραστηριότητες από ότι στις υπόλοιπες.

Τέλος, με βάση τα πειράματα που έγιναν βγήκαν τα εξής συμπεράσματα: σε όλες τις περιπτώσεις όταν δίνουμε πολύ λίγη μνήμη έχουμε μεγάλους χρόνους εκτέλεσης των σεναρίων και διαφοροποιήσεις ανάλογα τον αλγόριθμο. Ενώ, όσο δίνουμε την διαθέσιμη μνήμη τόσο πέφτουν οι χρόνοι εκτέλεσης των πειραμάτων και επί το πλείστον ισορροπούν οι περισσότεροι αλγόριθμοι στα ίδιο σημείο. Για το Σενάριο Δένδρου καλύτεροι είναι οι: Δίκαιος Αλγόριθμος, Αλγόριθμος Προτεραιότητας Ταξινομήσεων και Αλγόριθμος Προτεραιότητας Συναθροίσεων. Για το Σενάριο Πεταλούδας πάλι καλύτεροι είναι ο Δίκαιος Αλγόριθμος, ο Αλγόριθμος Προτεραιότητας Ταξινομήσεων και ο Αλγόριθμος Προτεραιότητας Συναθροίσεων. Για το Γραμμικό Σενάριο κερδίζει ο Αλγόριθμος Προτεραιότητας Δραστηριοτήτων Εγγραφής για παράγοντες κλιμάκωσης 0.1 και 0.5, ενώ για παράγοντα κλιμάκωσης 1.0 κερδίζει ο Αλγόριθμος Προτεραιότητας Συναθροίσεων. Τέλος, για το Σενάριο Πολλών Εκροών ο Δίκαιος Αλγόριθμος έχει την καλύτερη συμπεριφορά από όλους τους άλλους.

5.2. Μελλοντικές Επεκτάσεις

Η συγκεκριμένη εργασία ασχολήθηκε με την διαχείριση της μνήμης σε ETL διεργασίες. Οι ETL διεργασίες που χρησιμοποιήθηκαν ήταν το Γραμμικό Σενάριο, το Σενάριο Πεταλούδας, το Σενάριο Δένδρου και το Σενάριο Πολλών Εκροών. Μία επέκταση που θα μπορούσε να γίνει είναι να γίνεται διαχείριση της μνήμης σε πιο πολύπλοκα σενάρια τα οποία μέσα τους μπορούν να περιέχουν πολλά από τα παραπάνω σενάρια. Έτσι για να γίνει η διαχείριση της μνήμης σε αυτού του τύπου τα σενάρια θα μπορούσαμε να τα σπάσουμε σε μικρότερα ήδη μελετημένα σενάρια και να αναθέταμε την μνήμη σε κάθε ένα σενάριο ξεχωριστά με βάση την εργασία που έγινε.

ΑΝΑΦΟΡΕΣ

[Asce09] Ascential DataStage Integration. Διαθέσιμο εδώ:
http://www.iwaysoftware.com/products/ascential.html

[BoKV98] L. Bougnim, O. Kapitskaia, P. Valduriez. Memory-adaptive scheduling for large query execution. In Proc. of the 7th CIKM Conf., pages 105-115, Bethesda, U.S.A., October, 1998.

[BuPf95] Rainer E. Burkard, Ulrich Pferscy, The inverse-parametric knapsack problem, European Journal of Operational Research, pages 376-393, 1995.

[GFSS00] H. Galhardas, D. Florescu, D. Shasha, E. Simon. Ajax: An Extensible Data Cleaning Tool. In Proceedings ACM SIGMOD International Conference on the Management of Data, pages 590-595, Dallas, Texas, May, 2000.

[FaNS95] C. Faloutsos, R. Ng, T. K. Sellis. Flexible and Adaptable Buffer Management Techniques for Database Management Systems. IEEE Trans. Computers, Volume 44, Number 4, pages 546-560, April, 1995.

[HuSS00] A. Hulgeri, S. Seshadri, S. Sudarshan. Memory Cognizant Query Optimization. In Proc. Of COMAD, pages 181-193, Pune, India, 2000.

[IBM09] IBM InfoSphere DataStage. Διαθέσιμο εδώ:
http://www-01.ibm.com/software/data/infosphere/datastage/

[Infrm09] Informatica Power Center. Διαθέσιμο εδώ:
http://www.informatica.com/products_services/powercenter/Pages/index.aspx

[Inmo02] W. Inmon, *Building the Data Warehouse*, John Wiley & Sons, Inc. 2002.

[Kara07] Α. Καραγιάννης, Πολιτικές Ρύθμισης της Διαχείρισης της Ενημέρωσης Αποθηκών Δεδομένων. Μεταπτυχιακή Διατριβή, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Ελλάδα, Ιούλιος, 2007

[KRRT98] R. Kimball, L. Reeves, M. Ross, W. Thornthwaite. The Data Warehouse Lifecycle Toolkit, Wiley, 1998

[MeDe93] M. Mehta, D. J. DeWitt. Dynamic memory allocation for multiple query workload. In Proc. of the Int'l Conf on VLDB, pages 354-367, Dublin, Ireland, August,1993

[Orac09] Oracle Warehouse Builder 11g. Διαθέσιμο εδώ: http://www.oracle.com/technology/products/warehouse/index.html

[RaHe01] V. Raman, J. Hellerstein, Potter's Wheel: An Interactive Data Cleaning System. In Procceedings of 27th International Conference on Very Large Data Bases (VLDB), pages 381-390, Roma, Italy, September, 2001

[SiVS05a] A. Simitsis, T. Sellis. Extraction-Transformation-Loading (ETL) Processes in Data Warehouse Environments. In Encyclopedia of Database Technologies and Applications, Idea Group, ISBN: 1-59140-560-2, Pub. Date: August 9, 2005.

[SiVS05b] A. Simitis, P. Vassiliadis, T. K. Sellis. Optimizing ET Processes in Data Warehouses. In Proc. 21st Int. Conference on Data Engineering (ICDE), pages 564-575, Tokyo, Japan, April, 2005.

[SiVS05c] A. Simitsis, P. Vassiliadis, T. K. Sellis. State-Space Optimization of ETL Workflows. IEEE Transactions on Knowledge and Data Engineering (IEEE TKDE), volume 17, number 10, pages 1404-1419, October 2005.

[SSIS09] Microsoft. SQL Server 2005 Integration Services (SSIS). Διαθέσιμο εδώ: http://www.microsoft.com/sqlserver/2005/en/us/integration-services.aspx

[SVSS06] A. Simitsis, P. Vassiliadis, S. Skiadopoulos, T. Sellis. Data Warehouse Refreshment. In Data Warehouses and OLAP: Concepts, Architectures and Solutions, IGI Global, ISBN-10: 1599043645, ISBN-13: 978-1599043647, December 2006.

[Tpch09] TPC-H Benchmark. Διαθέσιμο εδώ: http://www.tpc.org/tpch/

[Tzio06] V. Tziovara, Order Aware Workflows. MSc Thesis, Computer Science, University of Ioannina, Hellas, October, 2006

[TzVS07] V. Tziovara, P. Vassiliadis, A. Simitsis. Deciding the Physical Implementation of ETL Workflows. In Proc. ACM 10th International Workshop on Data Warehousing and OLAP (DOLAP), pages 49-56, Lisbon, Portugal, November 2007.

[VaSi09] P. Vassiliadis, A. Simitsis. EXTRACTION-TRANSFORMATION-LOADING. Encyclopedia of Database Systems, Editors-in-chief: Liu, Ling; Özsu, M. Tamer, Springer, 2009.

[VaSS02] P. Vassiliadis, A. Simitsis, S. Skiadopoulos. Modeling ETL Activities as Graphs. In Proc. Of 4th International Workshop of the Design and Management of

Data Warehouses (DMDW) in conjuction with CAISE, pages 52-61, Toronto, Canada, May, 2002.

[VKTS07] P. Vassiliadis, A. Karagiannis, V. Tziovara, A. Simitsis. Towards a Benchmark for ETL Workflows. 5th International Workshop on Quality in Databases (QDB), held in conjuction with VLDB, pages 49-60,Vienna, Austria, September 2007.

[VSGT03] P. Vassiliadis, A. Simitsis, P. Georgantas, M. Terrovitis. A Framework for the Design of ETL Scenarios. In 15th Conference on Andanced Information Systems Engineering (CAISE), 520-535, Klagenfurt, Austria, June, 2003.

[YuCo93] P. S. Yu, D. W. Cornell. Buffer management based on return on consumption in a multi-query environment. VLDB Journal, Volume 2, Number1, pages 1-37, 1993

ΣΥΝΤΟΜΟ ΒΙΟΓΡΑΦΙΚΟ

Ο Παναγιώτης Δομουχτσίδης γεννήθηκε στα Γιαννιτσά το 1985 και τελείωσε το Λύκειο το 2002. Το έτος 2002 εισήχθη στο Τμήμα Πληροφορικής του Πανεπιστημίου Ιωαννίνων από όπου αποφοίτησε το έτος 2006. Το έτος 2006 εισήχθη στο Μεταπτυχιακό Πρόγραμμα Σπουδών του Τμήματος Πληροφορικής του Πανεπιστημίου Ιωαννίνων. Τα ερευνητικά του ενδιαφέροντα είναι στις Βάσεις Δεδομένων, στις Αποθήκες Δεδομένων και στα ETL εργαλεία.