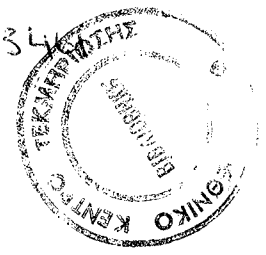
ΠΟΚΤ 03506
ηλ: 34



Ελένη Ν. Παπαθανασίου

ΤΜΗΜΑΤΟΠΟΙΗΜΕΝΕΣ ΒΑΣΕΙΣ ΓΝΩΣΗΣ ΣΕ ΤΟΠΙΚΑ ΔΙΚΤΥΑ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ

Διδακτορική Διατριβή

Πειραιάς 1994

ΤΜΗΜΑΤΟΠΟΙΗΜΕΝΕΣ ΒΑΣΕΙΣ ΓΝΩΣΗΣ ΣΕ ΤΟΠΙΚΑ ΔΙΚΤΥΑ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ

Ελένη Ν. Παπαθανασίου

ΤΜΗΜΑΤΟΠΟΙΗΜΕΝΕΣ ΒΑΣΕΙΣ ΓΝΩΣΗΣ ΣΕ ΤΟΠΙΚΑ ΔΙΚΤΥΑ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ

Διδακτορική Διατριβή
Υποβληθείσα στο Τμήμα Πληροφορικής
του Πανεπιστημίου Πειραιώς

Επταμελής Επιτροπή:

Τριμελής Συμβουλευτική Επιτροπή:
Ι.-Χ. Παναγιωτόπουλος (Καθηγητής - επιβλέπων)
Ν. Αλεξανδρής (Καθηγητής)
Β. Χρυσικόπουλος (Αν. Καθηγητής)
Πανεπιστήμιο Πειραιώς, Τμήμα Πληροφορικής

Μαρίτσας Δ. (Καθηγητής) *Πολυτεχνείο Πάτρας, Τμήμα Μηχανικών Η/Υ & Πληροφορικής,*
Σιδερίδης Α. (Καθηγητής) *Γεωργικό Πανεπιστήμιο, Τμήμα Γενικό*
Γεωργιάδης Π. (Αν. Καθηγητής) *Πανεπιστήμιο Αθηνών, Τμήμα Πληροφορικής*
Λορέντζος Ν. (Επ. Καθηγητής) *Γεωργικό Πανεπιστήμιο, Τμήμα Γενικό*

Πειραιάς, 1994

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Η παρούσα Διατριβή με τίτλο "Τμηματοποιημένες Βάσεις Γνώσης σε τοπικά δίκτυα μικροϋπολογιστών" προτείνει μία νέα δομή που μπορεί να εφαρμοστεί ακόμη και σε σχετικά μεγάλες Βάσεις Γνώσης. Στόχος είναι η δυνατότητα διαχείρισής τους σε μικροϋπολογιστές και σε τοπικά δίκτυα αυτών. Με την προτεινόμενη δομή εξουδετερώνονται οι περιορισμοί σε κύρια μνήμη τους οποίους συνεπάγεται η χρήση μικροϋπολογιστών. Η ελαχιστοποίηση απαιτήσεων σε κύρια μνήμη επιτυγχάνεται με την συνδιαστική τμηματοποίηση Γνώσης και συνεργασία αυτής με κλασσικές Βάσεις Δεδομένων (ΒΔ) σε δίσκους. Αυτή η έννοια του συνεργαζόμενου συστήματος (cooperative system) σε συνδυασμό με τον Πραγματικό Χρόνο προσπέλασης, διακρίνει την παρούσα Διατριβή και αποτελεί τον κύριο σκοπό της.

Εξ ορισμού Βάση Γνώσης (ΒΓ) είναι το σύνολο γεγονότων και κανόνων που αφορούν ένα γνωσιολογικό χώρο. Στο χώρο της Πέμπτης Γενιάς Πληροφορικής,όπου χρησιμοποιούνται Βάσεις Γνώσης στα πλαίσια εμπείρων συστημάτων και κυρίως όταν χρησιμοποιείται Prolog, υπάρχει αφ' ενός ο περιορισμός κύριας μνήμης μικροϋπολογιστών και αφ' ετέρου το πρόβλημα άμεσης απόκρισης

μεταξύ σταθμών ενός δικτύου μικροϋπολογιστών, όταν διαχειρίζονται το ίδιο τμήμα Βάσης Γνώσης ταυτόχρονα.

Η παρούσα εργασία αναφέρεται στις νέες τάσεις που προτείνονται από την Πέμπτη γενιά πληροφορικής όσον αφορά τη χρήση Βάσεων Γνώσης. Θίγει το θέμα μετάβασης από κλασσικές ΒΔ σε ΒΓ, προτείνοντας δομή και μέθοδο και εστιάζεται ιδιαίτερα στη διαχείριση Τμηματοποιημένων Βάσεων Γνώσης σε μικροϋπολογιστές και τοπικά δίκτυα αυτών, με βασικό εργαλείο του Λογικού Προγραμματισμού την γλώσσα Prolog.

Αποτελείται από πέντε κεφάλαια και από ένα παράρτημα:

Το πρώτο κεφάλαιο "Βασικές έννοιες και ορισμοί" περιέχει έννοιες και ορισμούς Δεδομένων, Γνώσης και τις βασικές τεχνικές παράστασής της. Πραγματοποιείται μία σύγκριση των εννοιών "δεδομένα" και "γνώση" και αναφέρονται ερευνητικές εργασίες στο χώρο συνύπαρξης των δύο εννοιών.

Στο δεύτερο κεφάλαιο "Από τα Δεδομένα στη Γνώση" προτείνεται μία νέα μέθοδος μετατροπής μίας κλασσικής ΒΔ σε ΒΓ. Για την παραγωγή κανόνων χρησιμοποιεί ήδη υπάρχουσες μεθόδους της βιβλιογραφίας [Thompson, 1986], [Lin, 1989], [Piatetsky - Shapiro,Frawley,1991].Χρησιμοποιούνται δομές δηλωτικής παράστασης Γνώσης (Prolog). Γίνεται αναφορά σε σχέσεις μεταξύ Prolog και ΒΔ, σε μεγάλες ΒΓ, σε περιβάλλον τοπικών δικτύων μικροϋπολογιστών.Εισάγεται η έννοια της Τμηματοποιημένης Βάσης

Γνώσης Περιορισμένου Μεγέθους Μνήμης και των Τμημάτων ΒΓ Περιορισμένου Μεγέθους Μνήμης (Λ) για δίκτυα μικροϋπολογιστών και δίνεται ο αντίστοιχος ορισμός. Ακολουθεί περιγραφή των βασικών δομών του συστήματος που θα προκύψει μετά τη μετατροπή ΒΔ σε ΒΓ. Δίνεται η μέθοδος μετατροπής και ακολουθούν παραδείγματα και εφαρμογές.

Στο τρίτο κεφάλαιο "Τμήματα Βάσης Γνώσης Περιορισμένου μεγέθους Μνήμης (Λ) με κανόνες" γίνεται αναφορά σε εργασίες τμηματοποίησης και ταξινόμησης Γνώσης κανόνων, αναπτύσσονται οι δομές παράστασης και υποστήριξης της Βάσης κανόνων, του συστήματος Τμηματοποιημένων ΒΓ. Δίνεται η έννοια και ο ορισμός της Τμηματοποιημένης Βάσης Γνώσης Κανόνων Περιορισμένου Μεγέθους Μνήμης (Λ κανόνων) για δίκτυα μικροϋπολογιστών. Αναπτύσσονται μέθοδοι Τμηματοποίησης του συνόλου των κανόνων σε Τμήματα ΒΓ κανόνων Περιορισμένου μεγέθους μνήμης (Λ κανόνων) και δίνονται οι αντίστοιχες πράξεις προσπέλασης.

Στο τέταρτο κεφάλαιο, "Αρχιτεκτονική και Διαχείριση", παρατίθεται η διαχείριση του παραγόμενου συστήματος ΒΓ γεγονότων και οι βασικές πράξεις προσπέλασης, συνοδευόμενες από παραδείγματα.

Το πέμπτο κεφάλαιο "Διαχείριση Γνώσης σε εφαρμογές Πραγματικού Χρόνου και αντιμετώπιση μεγάλων Βάσεων

Γνώσης", αναφέρεται αποκλειστικά στο πρόβλημα "πραγματικού χρόνου κύριας μνήμης" - RAM-Real-Time. Αναφέρονται υπάρχουσες προσεγγίσεις στο πρόβλημα του πραγματικού χρόνου στο χώρο των εμπείρων συστημάτων. Προσεγγίζεται το θέμα πραγματικού χρόνου μεταξύ σταθμών του δικτύου μικροϋπολογιστών για μεγάλες Βάσεις Γνώσης Κατανεμημένης αρχιτεκτονικής. Τέλος, προτείνεται ένας σχεδιασμός αντιμετώπισης μεγάλων ΒΓ με σύγχρονο προγραμματισμό (concurrent programming) σε επίπεδο RAM κάτω από μία ομάδα διαθέσιμων επεξεργαστών όπου ένας επεξεργαστής εκτελεί χρέη συντονιστού του όλου περιγραφέντος συστήματος.

Τέλος ακολουθεί ένα παράρτημα με μετρήσεις στο βασικό μέτρο χρόνου προσπέλασης και δίνονται ορισμένα πρωτότυπα βασικά κατηγορήματα (predicates) που έχουν υλοποιηθεί σε Prolog, για την κάλυψη βασικών θεμάτων της παρούσης εργασίας.

ΚΕΦΑΛΑΙΟ 1

ΒΑΣΙΚΕΣ ΕΝΝΟΙΕΣ ΚΑΙ ΟΡΙΣΜΟΙ

1.1. ΕΙΣΑΓΩΓΗ

Στην Ιστορία της Πληροφορικής διακρίνονται οι γενιές του Υλικού (Hardware) και του Λογισμικού (Software), δηλαδή οι πέντε γενιές Πληροφορικής. Προς το τέλος της δεκαετίας του '70 εμφανίζεται η τέταρτη γενιά που χαρακτηρίζεται από νέες ταχύτητες και δυνατότητες στον προγραμματισμό. Τα προϊόντα του Λογισμικού είναι πλέον πιό προσιτά στον τελικό Χρήστη και πολλές εφαρμογές εμπεριέχουν ορισμούς δεδομένων, λογική επεξεργασία και σχήματα δεδομένων (data schemas) που αφορούν εφαρμογές όπου υπεισέρχεται η έννοια της αβεβαιότητας. Εφαρμόζεται το Σχεσιακό μοντέλο δεδομένων (Relational data model) και η γλώσσα χειρισμού δεδομένων SQL (Structured Query anguage), καθώς και τα Ιεραρχικά (Hierarchical) και Δικτυωτά (Networked) μοντέλα [Βασιλακόπουλος, Χρυσικόπουλος, 1990]. Εφαρμόζονται τα Σχεσιακά, Ιεραρχικά και Δικτυωτά Συστήματα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ) [Clarke, 1991].

Ωστόσο, το ζήτημα της ανάκτησης/προσπέλασης δεδομένων στη γενική μορφή τους, δεν επιλύεται ακριβώς (άριστες λύσεις), μιά

που είναι γνωστό πως για να επιλυθεί χρειάζεται η χρήση NPC αλγορίθμων (Non-Polynomial Complete Algorithms).

Κατά το διάστημα 1975-1990 τα προϊόντα της 4ης γενιάς έχουν κάνει σημαντικές προόδους, ξεπερνώντας τις βασικές αδυναμίες των προηγούμενων γενεών. Τις Βάσεις Δεδομένων (δομημένες ολοκληρωμένες συλλογές από αρχεία - συλλογές εγγραφών αποθηκευμέννη σε δίσκους) τις διαχειρίζονται ΣΔΒΔ τα οποία αποτελούν ένα αναπόσπαστο τμήμα της ολότητας του Πληροφοριακού Συστήματος.

Στη δεκαετία του '70 ήταν κοινή διαπίστωση ότι τα Πληροφοριακά Συστήματα ή έδιναν λύσεις που δεν αντιμετώπιζαν σε ικανοποιητικό βαθμό τις απαιτήσεις του πραγματικού προβλήματος ή παρουσίαζαν δυσκαμψία στην προσαρμογή τους σε δυναμικές αλλαγές ("προγραμματιστική σκληρότητα"). Δηλαδή είτε περιόριζαν την εξέλιξη του πραγματικού χώρου μέσα στους περιορισμένους βαθμούς ελευθερίας του Πληροφοριακού Συστήματος ή αδυνατούσαν στην εξαγωγή αποφάσεων σε θέματα των οποίων οι ταχύτητες επίλυσης έτρεχαν σε εκθετικούς χρόνους (π.χ. NPC προβλήματα). Επιπλέον, πολλά Πληροφοριακά Συστήματα είχαν απαιτήσεις που ήταν εκ των πραγμάτων κατανεμημένες (distributed) και λογικά και φυσικά. Τούτο είχε σαν συνέπεια να αναπτυχθούν τα Κατανεμημένα Συστήματα Διαχείρισης Βάσεων Δεδομένων - ΚΣΔΒΔ (Disributed Data base Management Systems - DDBMSs) [Ceri, Pelagati, 1985].

Στα πλαίσια της Πέμπτης γενιάς Πληροφορικής και της Τεχνητής Νοημοσύνης - ΤΝ (Artificial Intelligence - AI), οι Βάσεις Γνώσης (Knowledge Bases - KBs) διαχειριζόμενες την Γνώση στα πλαίσια Συστημάτων Διαχείρισης Βάσεων Γνώσης - ΣΔΒΓ (Knowledge Base Management Systems - KBMS) έρχονται να καλύψουν τα κενά που αφήνουν οι Βάσεις Δεδομένων όχι τόσο σε θέματα ταχύτητας και χώρου περιφεριακής μνήμης [Brodie, Mylopoulos, 1985], [Su, Shi, Whang, Hu, Wang, 1990], [Hodil, Butler, Richardson, 1986], αλλά περισσότερο σε προβλήματα όπου τον κύριο λόγο τον έχει η Γνώση, παρά τα δεδομένα και γίνεται χρήση ευρετικών τεχνικών (Heuristics).

1.2. ΔΕΔΟΜΕΝΑ

Ο όρος "δεδομένα" είναι έννοια αυτοοριζόμενη, η οποία χρησιμοποιείται χωρίς να της αποδίδεται κάποια άλλη έννοια ή μετάφραση [Barsalou, Wiederhold, 1989]. Κάθε πραγματικό πρόβλημα έχει τα δεδομένα του τα οποία δύνανται να αποθηκευτούν σε αρχεία τών οποίων η δομή πρέπει να επιτρέπει τουλάχιστον την εφαρμογή των τεσσάρων βασικών πράξεων:

1. Προσθήκη Νέας Εγγραφής,
2. Αναζήτηση Υπάρχουσας Εγγραφής,
3. Μεταβολή Στοιχείων Υπάρχουσας Εγγραφής,
4. Διαγραφή Υπάρχουσας Εγγραφής.

Οπωσδήποτε, οι Δομές Δεδομένων είναι ο θεμέλιος λίθος για το κτίσιμο των ΒΔ και των Πληροφοριακών Συστημάτων.

Οταν ένα πρόβλημα του πραγματικού κόσμου πρόκειται να επιλυθεί με χρήση Η/Υ, είναι απαραίτητη η κατάλληλη αναπαράσταση των Δεδομένων που αφορούν το πρόβλημα. Ετσι, σε κάθε σύγχρονη γλώσσα προγραμματισμού υπάρχουν δυνατότητες ορισμού των Δεδομένων με προσδιορισμό των τύπων και πεδίων τιμών αυτών. Οι τύποι δεδομένων είτε υποστηρίζονται άμεσα από τη γλώσσα είτε ορίζονται από τον προγραμματιστή, ανάλογα με τις ανάγκες του προβλήματος.

Η μελέτη των Δεδομένων από πλευράς Υλικού, Γλωσσών Προγραμματισμού, Δομής Δεδομένων και Ανάλυσης Δεδομένων δίνεται σαν ορισμός της Πληροφορικής από τον Κόλλια [Κόλλιας, 1986]. Μία Δομή Δεδομένων ορίζεται ως ένα ζεύγος (Σ, Π) όπου Σ είναι ένα σύνολο Δεδομένων και Π είναι το σύνολο των επιτρεπτών πράξεων πάνω στα Δεδομένα αυτά. Κάθε Δομή Δεδομένων παρίσταται από κόμβους (nodes). Κάθε κόμβος αποτελείται από δύο τμήματα. Το πρώτο τμήμα είναι το αυτούσιο Δεδομένο. Το δεύτερο είναι δείκτης (pointer) εσωτερικής προσπέλασης και σε πολλές περιπτώσεις δεν εμφανίζεται, διότι εννοείται. Για παράδειγμα, κάθε κόμβος σε μία μήτρα αποτελείται μόνο από Δεδομένα (οι δείκτες εννοούνται), ενώ σε μία λίστα κάθε κόμβος αποτελείται και από Δεδομένα και από δείκτη, ο οποίος στην περίπτωση αυτή είναι αναγκαίος και δηλώνει την "επόμενη εγγραφή" κατά τάξη προσπέλασης. Δομές στις οποίες ο πληθικός αριθμός, δηλαδή το πλήθος των κόμβων τους, είναι σταθερός, ονομάζονται Στατικές (π.χ. μία μήτρα στην Pascal), ενώ δομές με

μεταβαλλόμενο πληθικό αριθμό χαρακτηρίζονται Δυναμικές (π.χ. λίστα, αρχείο). Δομές που αποτελούνται από στοιχεία του αυτού τύπου (ομοιογενείς) είναι απλές (π.χ. λίστα, μήτρα), σε αντίθεση με εκείνες που αποτελούνται από στοιχεία διαφόρων τύπων (ανομοιογενείς, π.χ. record).

"Ο τρόπος καταγραφής και αλληλοσυσχέτισης των δεδομένων με στόχο την παράσταση της γνώσης γεγονότων του πραγματικού κόσμου, οι τεχνολογίες των ΒΔ και της αναπαράστασης της Γνώσης είναι ένας κλάδος μελέτης των δεδομένων από πλευράς 'ανάλυσης δεδομένων' " [Κόλλιας, 1986].

1.3. ΓΝΩΣΗ

Στη δεκαετία του '70 αρχίζει ουσιαστικά η Πέμπτη γενιά Πληροφορικής, με κύριο στόχο την ανάπτυξη επαγωγικών μηχανισμών οι οποίοι βρίσκουν κατ' ανάγκη εφαρμογή στα Εμπειρα Συστήματα - ΕΣ (Expert Systems - ES), η προϊστορία των οποίων ξεκινάει από τη δεκαετία του '50. Η Γνώση, οργανωμένη σε Βάσεις Γνώσης, είναι ένα από τα βασικά μέρη από τα οποία αποτελείται ένα ΕΣ (Βάση Γνώσης, Λογική, Υποσύστημα αυτοδιδασκαλίας, Υποσύστημα επαφής με τον Χρήστη) [Παναγιωτόπουλος Ι.-Χ, 1988].

Κάθε Γνωσιολογικός Χώρος περιλαμβάνει γεγονότα (facts) και κανόνες (rules) που τον αφορούν. Η Γνώση προέρχεται από τους

Ειδικούς ή/και είναι προϊόν εμπειρίας, είτε αυτή προέρχεται από άνθρωπο, είτε από επεξεργασία υπάρχουσας Γνώσης ή/και δεδομένων.

"Η έννοια "Γνώση" εδώ θεωρείται ξένη από διφορούμενες καταστάσεις που πηγάζουν από διαίσθηση, αισθητική, συμβολική, μεταφυσική. Η έννοια αυτή θεωρείται μέσα σε πλαίσια αβεβαιότητας, όπως συχνά εμφανίζεται σε διάφορους κλάδους πολλών επιστημών" [Παναγιωτόπουλος Ι-Χ., 1988]. Εμπεριέχει αφαιρέσεις, γενικότητες, θεωρίες και στρατηγικές χρήσης όλων αυτών των εννοιών (μεταγνώση - metaknowledge) [Widerhold, 1985], [Nait Abdallah, 1991], [Barsalou, Wiederhold, 1989].

Βάση Γνώσης είναι ένα οργανωμένο σύνολο των γεγονότων και των κανόνων που αφορούν ένα συγκεκριμένο γνωσιολογικό χώρο. Σύστημα Βάσεων Γνώσης - ΣΒΓ (Knowledge-Based System - KBS) είναι κάθε σύστημα που χησιμοποιεί Βάση Γνώσης. Για την απόκτηση της Γνώσης χρησιμοποιούνται μέθοδοι εκμαίευσής της από τους Ειδικούς αλλά και γενικότερα από κάθε πηγή Γνώσης. Η απόκτηση της Γνώσης (Knowledge Acquisition) [Kidd, 1987] είναι ένας κύριος κλάδος της τεχνολογίας της Γνώσης (Knowledge Engineering) [Diaper, 1988] και αναφέρεται στην απόκτηση της Γνώσης που απαιτείται από το Σύστημα. Στην διαδικασία απόκτησης Γνώσης διακρίνονται τα ακόλουθα στάδια:

_ Η απόφαση ποιά Γνώση χρειάζεται (definition stage or initial analysis).

_ Η εκμαίευσης (elicitation) της Γνώσης από τους ειδικούς (human experts) ή άλλες πηγές.

_ Η αναπαράσταση της Γνώσης (knowledge representation) δηλαδή απόδοσή της με μορφή συμβατή με κάποια ειδική γλώσσα προγραμματισμού [Cordingley, 1989], [Gammack, 1987], [Lee, 1983].

1.4. ΔΟΜΕΣ ΓΝΩΣΗΣ

Η παράσταση της Γνώσης μπορεί να ακολουθεί ένα Δηλωτικό (Declarative) ή Διαδικαστικό (Procedural) τρόπο. Η Δηλωτική Παράσταση Γνώσης περιγράφει τη Γνώση χωρίς να εμπεριέχει και τον τρόπο χρησιμοποίησης της. Αντίθετα, η Διαδικαστική Παράσταση Γνώσης χρησιμοποιεί διαδικασίες οι οποίες εκτός από τη Γνώση περιέχουν και τους τρόπους χρησιμοποίησής της.

Παράδειγμα δηλωτικής παράστασης:
Μία μήτρα (array) Α με στοιχεία 0, 1, με m γραμμές και n στήλες. Η γνώση απλά δηλώνεται. Ο Χρήστης εννοεί ότι πιθανόν οι γραμμές να είναι ασθένειες και οι στήλες συμπτώματα.

Παράδειγμα διαδικαστικής παράστασης:

```
είναι_η_ασθένεια(x1):-
                είναι_το_σύμπτωμα(x10),
                είναι_το_σύμπτωμα(x11),
```

βεβαιώσου(έχει,x8),
βεβαιώσου(έχει,x9).

είναι_το_σύμπτωμα(x10):-
βεβαιώσου(έχει,x100);
βεβαιώσου(έχει,x300).

είναι_το_σύμπτωμα(x11):-
βεβαιώσου(έχει,x110),
βεβαιώσου(έχει,x200).

Δηλαδή ισχύει ότι η ασθένεια είναι x1 αν ισχύει ότι έχει τα συμπτώματα x10, x11 και επιπλέον ότι έχει x8 και x9. Το σύμπτωμα x10 ισχύει άν έχει x100 ή x300 (εδώ το ";" έχει την έννοια του εγκλητικού "ή"). Το σύμπτωμα x11 ισχύει άν έχει x110 και x300 (εδώ το "," έχει την έννοια του "και").

1.4.1 ΠΑΡΑΣΤΑΣΗ ΓΝΩΣΗΣ (Knowledge representation)

Από τα πιό σημαντικά σημεία της τεχνολογίας της Γνώσης είναι η επιλογή του τρόπου παράστασης αυτής. Η παράσταση της Γνώσης αποβλέπει σε πλήρη περιγραφή πραγματικών προβλημάτων. Εμπεριέχει τις ικανότητες της ανάκτησης και της επαναχρησιμοποίησης της Γνώσης.

Η Παράσταση της Γνώσης είναι από τα δυσκολότερα θέματα, διότι εξαρτάται σε μεγάλο βαθμό από τον τρόπο χρησιμοποίησης αυτής της Γνώσης. Από τον Minsky το 1975 χρησιμοποιήθηκε ο όρος "Interaction Problem" ο οποίος αναφέρεται στα προβλήματα που προκύπτουν από την αλληλεπίδραση του τρόπου χρήσης της Γνώσης και την Αναπαράστασή της [Minsky, 1975]. Ο Chandrasekaran [Chandrasekaran, 1988] χρησιμοποιεί την έννοια του προβλήματος αλληλεπίδρασης (Interaction Problem), έτσι ώστε τα πλεονεκτήματα του τρόπου παράστασης της Γνώσης να ενισχύουν τον τρόπο χρήσης της.

Με την ανάπτυξη της Τεχνητής Νοημοσύνης, έγινε φανερή η υπεροχή των γλωσσών Λογικού Προγραμματισμού στην παράσταση της Γνώσης έναντι των αλγοριθμικών γλωσσών Προγραμματισμού [Allen, 1987], [Brule', 1986], [Guenther, Lehman, 1986], [Garner, 1987], [Leitheiser, 1986], [Leung, Wong, Lam, 1989], [MacKellar, Maryanski, 1984], [Randall, Duglas, 1982], [Ringland, Duce, 1988], [Sowa, 1990], [Vedder, 1989], [Hardzband, Maryansky, 1985].

1.4.2 ΜΕΘΟΔΟΙ - ΓΛΩΣΣΕΣ ΠΑΡΑΣΤΑΣΗΣ ΓΝΩΣΗΣ

Μία σύντομη ματιά στο πεδίο της Παράστασης της Γνώσης μας δίνει διάφορους τρόπους που έχουν χρησιμοποιηθεί άλλοι

λιγότερο και άλλοι περισσότερο. Συχνότερα απαντόμενοι στη βιβλιογραφία είναι οι ακόλουθοι:

.Λογική (Logic)
.Κανόνες (Rules-Rule Based Systems, Production Rules)
.Σημασιολογικά Δίκτυα (Semantic Networks)
.Πλαίσια (Frames)
.Σενάρια (Scripts)
.Εννοιολογική Εξάρτηση (Conceptual Dependency)
.Γλώσσες Παράστασης Γνώσης

1.4.2.1 Λογική (Logic)

Η Λογική βρίσκει εφαρμογή στην Παράσταση της Γνώσης, παρουσιαζόμενη σε διάφορες μορφές της. Η παράσταση της Γνώσης με λογική ακολουθεί τους γνωστούς κανόνες λογικής και χρησιμοποιεί τον ακόλουθο συντακτικό τύπο:

C if F1 and F2 and ... and Fn με:

C: συμπέρασμα

Fi, i=1,...,n : συνθήκες

Παράδειγμα:

(ο Χ έχει την ασθένεια Υ) αν (έχει το σύμπτωμα S1 και το σύμπτωμα S2)

Ειδικότερα αν χρησιμοποιήσουμε Κατηγορική Λογική (Predicate Logic) με εκφράσεις τύπου Prolog, το παράδειγμα θα έχει ως εξής:

disease(X,Y) if symptom(X,S1) and symptom(X,S2).

Εδώ τα disease και symptom είναι συμβολικά κατηγορήματα (symbol predicates). Το disease παριστά τη σχέση (Relation) η οποία συνδέει τις έννοιες που εκφράζουν οι όροι (Terms) X και Y.

1.4.2.2 Σημασιολογικά δίκτυα (Semantic Networks)

Ενα τέτοιο δίκτυο είναι ένα γράφημα το οποίο αποτελείται από κόμβους (nodes) που αναπαριστούν έννοιες και από γραμμές (links) προσανατολισμένες, κατευθυνόμενες και χαρακτηρισμένες που αναπαριστούν τις σχέσεις (relations) μεταξύ των εννοιών. Κάθε κατηγόρημα (predicate) εκφράζει μία σχέση μεταξύ δύο εννοιών ή περιγράφει μία έννοια [Deliyanni, Kowalsky, 1979]

Παράδειγμα:

(i) βιβλίο(A), αγοράζει(X,A), πουλάει(Y,A)

(ii) ' ο X αγοράζει το βιβλίο A '

' ο Y πουλάει το βιβλίο A'

κόμβοι (*) : X, Y, 'βιβλίο A'

συνδέσεις (-->) : 'αγοράζει ', 'πουλάει'

αγοράζει πουλάει

* ----------> * <---------- *

X βιβλίο Α Y

1.4.2.3 Πλαίσια (Frames)

Τα Πλαίσια, σαν τρόπος Παράστασης της Γνώσης, παρουσιάσθηκαν από τον Minsky (1975). Κάθε Πλαίσιο εκτός από το ότι εκπροσωπεί μία έννοια, εμπεριέχει και τα ιδιαίτερα χαρακτηριστικά αυτής της έννοιας.

Οπως στα Σημασιολογικά δίκτυα έχουμε τους κόμβους, έτσι και στα Πλαίσια υπάρχουν οι κόμβοι που παριστούν έννοιες, αλλά επιπλέον κάθε κόμβος αποτελείται από υποδοχές (slots) και γεμίσματα των υποδοχών (fillers). Πράγματι, η έννοια του πλαισίου του πωλητή (με όλα τα χαρακτηριστικά του) στο προηγούμενο παράδειγμα δύναται να παρασταθεί ως εξής:

Ονομα Πλαισίου	Πωλητής
slot1	Ονομα: "Νέο Βιβλιοπωλείο"
slot2	Διεύθυνση:Πειραιώς 123
slot3	Α.Φ.Μ. :123456

1.4.2.4 Κανόνες (Rules)

Ενας ευρείας εφαρμογής τρόπος παράστασης Γνώσης είναι με κανόνες, με συντακτικό τύπο τον ακόλουθο:

IF συνθήκη THEN συμπέρασμα

Οι κανόνες απεικονίζουν καταστάσεις (situations) σε ενέργειες (actions), όπου κεντρικό ρόλο παίζει ο καθορισμός των συνθηκών (conditions) που χαρακτηρίζουν τις καταστάσεις και τα συμπεράσματα (conclusions) που χαρακτηρίζουν τις ενέργειες.

Τα βασιζόμενα σε κανόνες συστήματα (rule-based systems) απαρτίζονται από τρία μέρη [Williams, Bainbridge, 1988]:

α) Μνήμη εργασίας (working memory), όπου περιλαμβάνονται αντικείμενα (objects) που αναπαριστούν γεγονότα του πραγματικού κόσμου. Εδώ προστίθενται τα νέα γεγονότα που παράγονται όταν η Δυναμική Βάση Γνώσης χρησιμοποιεί προϋπάρχοντα γεγονότα.

β) Μνήμη Κανόνων (rule memory), που περιλαμβάνει κανόνες της παραπάνω μορφής, και

γ) Μηχανισμός εξαγωγής συμπεράσματος (inference engine).

Η διατύπωση εκφράσεων υπό μορφή κανόνων είναι ιδιαίτερα απλή.

Χρησιμοποιούνται σε πολλά Έμπειρα Συστήματα (Expert Systems). Επιπλέον, έχουν σχέση με τη Λογική, αφού η φύση της εμπεριέχει εφαρμογή κανόνων.

Πολλές φορές η Γνώση που χρησιμοποιείται είναι ασαφής ή/και ατελής, με αποτέλεσμα τα συμπεράσματα να μην είναι απόλυτα και να βγαίνουν κάτω από συνθήκες αβεβαιότητας [Παναγιωτόπουλος Ι-Χ., 1988]. Ετσι χρησιμοποιείται Ασταθής Λογική (Fuzzy Logic) στο βαθμό βεβαιότητας ενός κανόνα, και τελεστές βεβαιότητας στο να ισχύει ένα γεγονός. Μία επέκταση της Ασταθούς Λογικής, η Θεωρία των Αποδεικτικών Στοιχείων (Theory of Evidence) εφαρμόζεται στα Έμπειρα Συστήματα, τα οποία χρησιμοποιούν παράγοντες βεβαιότητας, και μετρά την εμπιστοσύνη του συμπεράσματος.

Παράδειγμα:

rule1: if X1 and Y1 then C

rule2: if X2 and Y2 then C

με μέτρα αξιοπιστίας (MB):

MB[X1]=0.8, MB[Y1]=0.7, MB[X2]=0.9, MB[Y2]=0.6

Σύμφωνα με την θεωρία της Ασαφούς Λογικής τα μέτρα αξιοπιστίας του συμπεράσματος, με βάση τον εκάστοτε κανόνα είναι:

MB[C:rule1]=0.7, MB[C:rule2]=0.6, άρα:

MB[C:rule1,rule2]=MB[C:rule1]+MB[C:rule2]*(1-MB[C:rule1])=0.88

Σύμφωνα με τον Shortliffe χρησιμοποιείται η αξιοπιστία (CR) για την εύρεση του πραγματικού MB:

Αν CR[rule1]=0.7, CR[rule2]=0.8, ισχύει:

REAL{MB[C:rule1]}=0.7*0.7=0.49,

REAL{MB[C:rule2]}=0.6*0.8=0.48

Μέτρο Αξιοπιστίας :

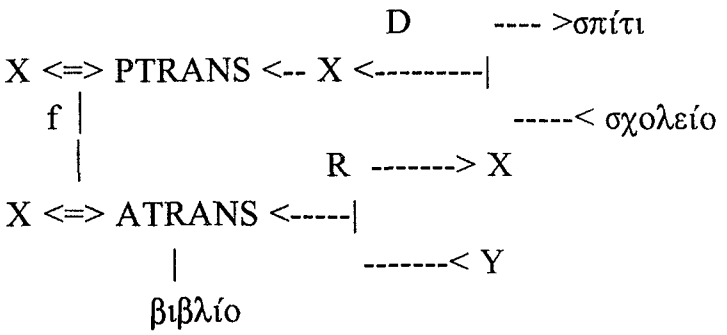
MB[C:rule1,rule2]=0.49+0.48*(1-0.49)=0.735.

1.4.2.5 Εννοιολογική εξάρτηση

Εφαρμόζεται για αναπαράσταση γεγονότων. Χρησιμοποιείται στην επεξεργασία φυσικής γλώσσας. Χρησιμοποιεί πρωτογενείς εκφράσεις και σύμβολα για ομαδοποίηση εννοιών (ATRANS, PTRANS, MTRANS για αφηρημένη, φυσική, διανοητική μεταφορά π.χ. "παίρνω", "έρχομαι", "λέω"). Απεικονίζεται με γράφημα (όπου κάθε κόμβος παριστά ρήμα, ουσιαστικό, επίθετο, επίρρημα) [Ringland, Duce, 1988].

Παράδειγμα:

"Ο Χ πηγαίνοντας από το σχολείο στο σπίτι θα αγοράσει ένα βιβλίο από το βιβλιοπωλείο Υ".



D : από που, που
R : ποιός, από που
o : έννοια αντικειμένου
f : μέλλον
<=>
|
<=> :συμβαίνει κάτι ταυτόχρονα με κάτι άλλο.

1.4.2.6 Σενάρια (Scripts)

Τα σενάρια αναπαριστούν ομαδοποιημένα στερεότυπα γεγονότα που έχουν μία διαδοχή. Προτάθηκαν από τους Schank, Abelson (1977). Για την κατανόηση μή στερεότυπων γεγονότων χρειάζεται πρόσθετη Γνώση. Ενα σενάριο αποτελείται από συνιστώσες (π.χ.ρόλους, σκηνές, αποτελέσματα) [Ringland,Duce, 1988].

1.4.2.7 Συστήματα και Γλώσσες Παράστασης Γνώσης (Knowledge Representation Systems and Languages)

Υπάρχουν Γλώσσες Παράστασης της Γνώσης και αρκετά εργαλεία τεχνολογίας Γνώσης για την ανάπτυξη Εμπείρων

Συστημάτων, όπως είναι για παράδειγμα τα έξυπνα κελύφη (Expert Shels). Στο χώρο της Τεχνητής Νοημοσύνης κυρίαρχη θέση κατέχει η Prolog. Τα ακολούθως αναφερόμενα επελέγησαν με τα εξής κριτήρια:
(a) Η Γλώσσα Υλοποίησης τους είναι η Prolog και
(b). Εφαρμόζονται σε συμβατούς μικροϋπολογιστές (PC) [Bodkin, Graham, 1989].

_ APES με χρήση Λογικής και με αβέβαια συλλογιστική, σε micro Prolog.
_ ARITY ESDP με Πλαίσια και κανόνες.
_ ESP Advisor με κανόνες και Λογική.
_ MI4 με Λογική, αντικείμενα και μεταβλητή σύνταξη κανόνων, σε Prolog-VM.
_ PROMETHEUS με Πλαίσια και Λογική .
_ TWAICE με κανόνες, Λογική, Πλαίσια.
_ XI USER με κανόνες και Λογική.

1.5. ΔΕΔΟΜΕΝΑ - ΓΝΩΣΗ: ΟΜΟΙΟΤΗΤΕΣ ΚΑΙ ΔΙΑΦΟΡΕΣ

Η έννοια των Δεδομένων είναι απλή και αυτοοριζόμενη χωρίς την ανάγκη ή τη δυνατότητα να αναλυθεί σε απλούστερες έννοιες. Αντίθετα, η Γνώση εμπεριέχει, όπως ήδη έχει αναφερθεί, αφαιρέσεις, γενικότητες, θεωρίες και στρατηγικές χρήσης όλων αυτών των εννοιών (μεταγνώση). Ετσι η Γνώση, σε αντίθεση με τα δεδομένα, θεωρείται μέσα σε πλαίσια αβεβαιότητας (uncertainty).

"Αν μπορούμε να εμπιστευθούμε μία αυτόματη διαδικασία σε ένα άτομο για τη συλλογή υλικού, τότε αναφερόμαστε σε δεδομένα. Αν αναζητήσουμε έναν ειδικό (expert) για την εξασφάλιση υλικού, τότε αναφερόμαστε σε Γνώση." [Wiederhold, 1986].

Οι δομές των δεδομένων είναι σχετικά απλές με σκοπό να φιλοξενήσουν συχνές ενημερώσεις (updates), ενώ η ενημέρωση της Γνώσης, η οποία γίνεται από ειδικούς (experts) ή συστήματα μάθησης (learning systems), χρειάζεται να έχει πιό σύνθετες αναπαραστάσεις (representations). [Wiederhold, 1985].

Επιπλέον, ενώ η διαχείριση των Δεδομένων γίνεται σε επίπεδο περιφεριακής μνήμης (δίσκους), η διαχείριση της Γνώσης στην Τεχνητή Νοημοσύνη και πιό συγκεκριμένα στο λογικό προγραμματισμό (π.χ. σε PROLOG) γίνεται περισσότερο σε επίπεδο κύριας μνήμης. Αυτό συνεπάγεται τεράστιες ταχύτητες μεν, περιορισμό χωρητικότητας δε, στο εκάστοτε μέγεθος της κύριας μνήμης, ειδικά όταν πρόκειται για μικροϋπολογιστές. Ωστόσο, αυτός ο περιορισμός φαίνεται πως μπορεί να μετριασθεί με τη δυνατότητα τμηματοποίησης της Γνώσης.

Πολλές φορές ένα σύστημα είναι κατανεμημένο και λογικά και φυσικά, και οι απαιτήσεις των Χρηστών έχουν κατανεμημένη (distributed) μορφή. Στην περίπτωση των Δεδομένων -- ΒΔ, χρησιμοποιούνται Κατανεμημένες Βάσεις Δεδομένων - ΚΒΔ, των

οποίων η διαχείριση γίνεται στα πλαίσια των Κατανεμημένων Συστημάτων Διαχείρισης ΒΔ - ΚΣΔΒΔ (Distributed Data Base management systems - DDBMS), [Ceri, Pelagati, 1985], [Apers,1988]. Αντίστοιχα, στην περίπτωση που το κατανεμημένο σύστημα αφορά Γνώση, ΒΓ, πρέπει να χρησιμοποιηθεί Κατανεμημένο Σύστημα Διαχείρισης ΒΓ (Distributed Knowledge Base Management System DKBMS) [Charlson, Sudha, 1991], [Randall, Douglas, 1982] για τη διαχείριση της κατανεμημένης γνώσης, μεταξύ των σταθμών ενός δικτύου.

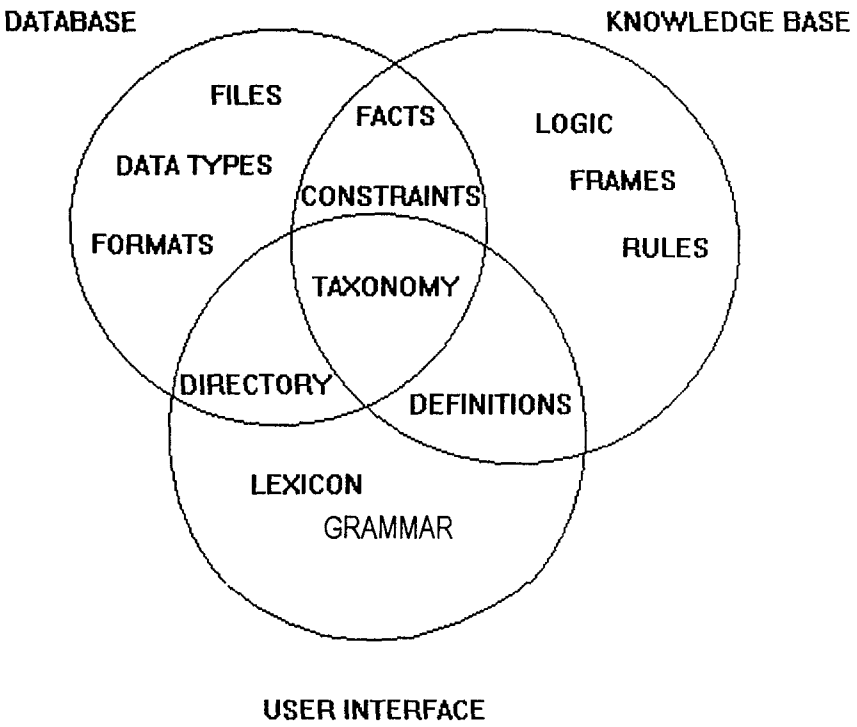
Είναι προφανές λοιπόν ότι τα Δεδομένα και η Γνώση πρέπει να συνδιαστούν με κύριο σκοπό την δημιουργία Συστημάτων με ελαχιστοποίηση των μειονεκτημάτων και μεγιστοποίηση των πλεονεκτημάτων που περικλείουν αυτές οι δύο έννοιες. Για παράδειγμα, το κύριο μειονέκτημα των Βάσεων Γνώσης είναι η υπερβολική απαίτηση σε κύρια μνήμη. Αυτό έχει σαν αποτέλεσμα τον περιορισμό δυνατότητας αξιοποίησης μεγάλων Βάσεων Γνώσης με τη βοήθεια μικροϋπολογιστών, εξαιτίας του περιορισμού κύριας μνήμης αυτών. Ο περιορισμός μπορεί να ελαχιστοποιηθεί με την συνεργασία Βάσεων Δεδομένων και Βάσεων Γνώσης.

1.6. ΔΕΔΟΜΕΝΑ ΚΑΙ ΓΝΩΣΗ: ΣΥΝΥΠΑΡΞΗ

Η πορεία από την Τέταρτη προς την Πέμπτη Γενιά Πληροφορικής, από τα πρώτα της βήματα, έκανε εμφανή την ανάγκη συνεργασίας Δεδομένων και Γνώσης, την ανάγκη μεταφοράς από

Δεδομένα σε Γνώση, από ΒΔ σε ΒΓ [Berghel,1985], [Deen, Thomas, 1990], [Farris and Singleton, 1989], [Ip, Holden, 1989], [Kellog, 1984], [Li Deyi, 1984], [Lizhu, Deyuan, Zhenping, Liping, 1988], [Obretenov, Angelov, Mihaylov, Dishlieva, Kirova, 1988], [Παναγιωτόπουλος Ι.-Χ., Παπαθανασίου Ε., 1991], [Rundensteiner, 1988].

Το ακόλουθο σχήμα με τρείς επικαλυπτόμενους κύκλους παριστά τις έννοιες "Knowledge base", "user interface" και "Database" [Sowa, 1990].



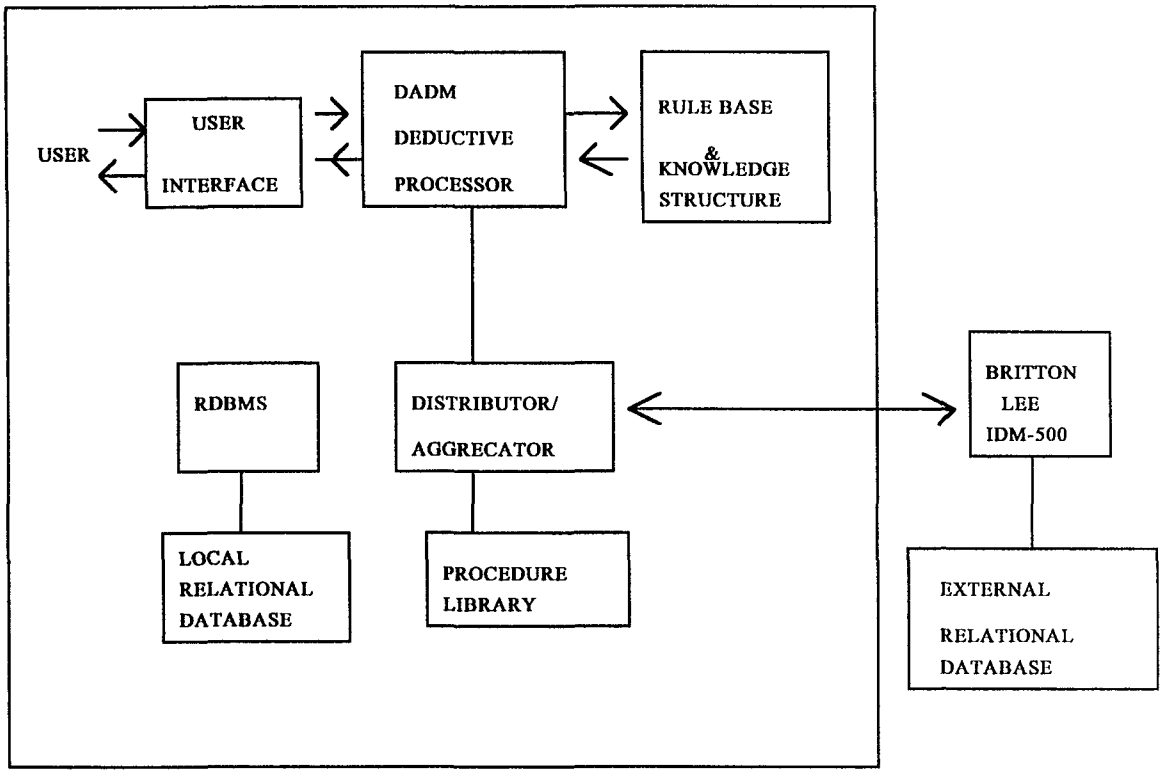
Σύμφωνα με τον Sowa, ένα πλήρες σύστημα Γνώσης πρέπει να έχει και τους τρείς κύκλους. Αλλά ανά δύο αποτελούν χρήσιμα συστήματα: μία Βάση Δεδομένων συνδυασμένη με ένα υποσύστημα επικοινωνίας με Χρήστη (user interface) δίνει ένα σύστημα απαιτήσεων από τη Βάση Δεδομένων (database query system), ένα σύστημα Γνώσης (Knowledge system) συνδιασμένο με ένα υποσύστημα επικοινωνίας με Χρήστη δίνει ένα εκπαιδευτικό υποσύστημα (consultant of advisor), ο δε συνδιασμός Βάσης Δεδομένων με σύστημα Γνώσης δίνει ένα Έμπειρο σύστημα Βάσεων Δεδομένων (Expert Database System).

Ακολουθεί αναφορά σε χαρακτηριστικές εργασίες που έχουν γίνει σ' αυτά τα πλαίσια, στο χώρο του Λογικού προγραμματισμού. Γίνεται κυρίως χρήση της PROLOG αφού είναι γλώσσα προγραμματισμού υψηλού επιπέδου σχεδιασμένη να απεικονίζει τα χαρακτηριστικά first-order predicate logic [Bergel, 1985].

Ο Berghel (1985) παρουσιάζει ένα συνδιασμό της PROLOG με ένα Σχεσιακό Σύστημα Βάσης Δεδομένων. Η εργασία αναφέρεται σε μία διαδικασία ανταλλαγής πληροφορίας μεταξύ συστημάτων. Σκοπός είναι να αποκομίζει κέρδος ο Χρήστης από την PROLOG και από το Σχεσιακό Σύστημα Βάσης δεδομένων. Εκμεταλλεύεται τα πλεονεκτήματα της PROLOG σε σχέση με τον Χρήστη μιας Βάσης δεδομένων. Χρησιμοποιεί δύο διαφορετικά προγραμματιστικά τμήματα (software packages) σε δύο διαφορετικές βάσεις δεδομένων,

με αυτόματη μετατροπή (conversion) μεταξύ τους. Γίνεται μία ολοκλήρωση (integration) ΒΔ παραγόμενης από DBase II, με μία ΒΓ δημιουργημένη από micro-PROLOG v.3.0, ως εξής: κάθε φορά που ο χρήστης δίνει μία απαίτηση, κάθε DBMS σχέση μετατρέπεται σε Prolog πρόταση.

Ο Kellogg (1984) παρουσιάζει ένα διαχειριστή Γνώσης, το KM-1 που πραγματοποιείται σε μία Xerox 1100 Lisp machine, που χρησιμεύει σαν σταθμός εργασίας (user workstation) και μηχανισμός συλλογιστικής (reasoning engine), και μία Britton-Lee IDM-500 μηχανή σχεσιακών ΒΔ (relational database machine), που πραγματοποιεί αναζητήσεις σε μεγάλες εξωτερικές ΒΔ. Οι απαιτήσεις του Χρήστη αποσυντίθενται σε υποπροβλήματα (ακολουθεί το αντίστοιχο σχήμα):



"Συστατικά του KM-1" [Kellog, 1984, σελ. 469]

Οι Farris και Singleton, (1989) παρουσιάζουν ένα πειραματικό συνδυασμό Prolog με τεχνολογία σχεσιακής ΒΔ για να χρησιμοποιηθεί στη διαχείριση βιβλιοθηκών προγραμμάτων και διαμόρφωσης (configuration).

Ο Deyi Li, (1984) παρουσιάζει ένα λογικό ενδιάμεσο (interface) μεταξύ PROLOG και RDBM .

Οι Zhou Lizhu, Yang Deyuan, Fan Zhengping, and Zhu Liping, (1988) ακολουθούν τη μεθοδολογία "build by integration",

δηλαδή κατασκευή του Συστήματος Διαχείρισης Βάσης Γνώσης συνδυάζοντας ένα σύστημα διαχείρισης σχεσιακών ΒΔ και μία γλώσσα Λογικού Προγραμματισμού (BPU-PROLOG). Το προκύπτον KBMS παρέχει στους Χρήστες μία γλώσσα Λογικού Προγραμματισμού για την ανάπτυξη εφαρμογών ΒΓ, υποστηρίζει μοίρασμα της γνώσης (knowledge sharing) μεταξύ ομάδας Χρηστών, και την ικανοποιητική πρόσβαση σε μεγάλες Βάσεις Γνώσης.

Οι Bogon Su, Chunyi Shi, Kehong Wang, Peng Hu, Hao Shi, and Jian Wang, (1988) παρουσιάζουν μία αρχιτεκτονική για ένα Κατανεμημένο Σύστημα Βάσης Γνώσης, όπου γίνεται ανάλυση του προβλήματος, τοπική διαχείριση, επίλυση των επιμέρους τμημάτων και τέλος σύνθεση της λύσης με χρήση της τεχνικής μαυροπίνακα (blackboard).

ΣΥΜΠΕΡΑΣΜΑ

Οι εξελιγμένες γλώσσες ΒΔ Τέταρτης Γενιάς (π.χ. Ingres), χρησιμοποιούν λεξικά (dictionaries) (δηλαδή ειδικές Βάσεις Γνώσης), ενώ παράλληλα οι ΒΓ Πέμπτης Γενιάς Πληροφορικής τείνουν στη χρήση μεικτών δομών Γνώσης (Συνεργαζόμενα Συστήματα), όπου πολλές φορές η προσπέλαση Γνώσης σε κάποιο βαθμό ανάγεται σε προσπέλαση Βάσεων Δεδομένων. Στόχος είναι η αριστοποίηση του χρόνου προσπέλασης με επάρκεια κύριας και περιφεριακής μνήμης.

Η παρούσα Διατριβή σκοπό έχει αρχικά να δώσει μέθοδο μετάβασης από κλασσικές ΒΔ σε ισοδύναμες ΒΓ, εφαρμόζοντας τις σύγχρονες έννοιες της τμηματοποίησης και ταξινόμησης της Γνώσης (knowledge clustering). Προσδιορίζονται και ορίζονται οι έννοιες του Πραγματικού χρόνου σε επίπεδο κύριας μνήμης, καθώς και η έννοια των Τμημάτων Βάσης Γνώσης Περιορισμένου Μεγέθους Μνήμης (Λ). Τέλος, προτείνεται μία μέθοδος επίλυσης του προβλήματος Πραγματικού Χρόνου μεγάλων ΒΓ σε επίπεδο Κύριας Μνήμης (RAM-Real-Time / RRT).

ΚΕΦΑΛΑΙΟ 2

ΑΠΟ ΤΑ ΔΕΔΟΜΕΝΑ ΣΤΗ ΓΝΩΣΗ

2.1. ΕΙΣΑΓΩΓΗ

Στη δεκαετία του '90 τα Πληροφοριακά Συστήματα βρίσκονται στο στάδιο μετάβασης προς Γνωσιολογικές (Knowledge-based) αρχιτεκτονικές [Garner, 1987], [Brodie, Mylopoulos, 1985]. Η καρδιά της Πέμπτης γενιάς σήμερα είναι τα Έμπειρα Συστήματα, τα οποία όμως λειτουργούν κυρίως σε επίπεδο Γνώσης.

Είναι γνωστό, πως η ΒΓ θεωρεί κάθε παράμετρο που εμπλέκεται σ' αυτήν, ισχυρά δομημένη σε σχέση με όλες τις άλλες παραμέτρους, με αποτέλεσμα και η δυσκολότερη απαίτηση Χρήστη να είναι δυνατόν να υλοποιηθεί. Ωστόσο, τα προϊόντα παραγωγής και διαχείρισης Γνώσης σε επίπεδο μικροϋπολογιστών, λειτουργούν αποκλειστικά σε επίπεδο κύριας μνήμης και έτσι υπάρχει αδυναμία στην περίπτωση μεγάλων ΒΓ [Παναγιωτόπουλος Ι.-Χ., 1988].

Η παραγωγή Γνώσης από Δεδομένα παίζει σημαντικό ρόλο στη μετάβαση από ΒΔ σε ΒΓ. Το ίδιο σημαντική είναι και η ανάγκη συνεργασίας ΒΔ και ΒΓ για την υλοποίηση Εμπείρων Συστημάτων. Από τη μία πλευρά διαφαίνεται η ανάγκη δυνατότητας μετάβασης των υπαρχόντων Συστημάτων που βασίζονται σε ΒΔ, σε Συστήματα

ΒΓ (Knowledge based systems), και από την άλλη πλευρά η ανάγκη συνεργασίας δύο τέτοιων συστημάτων, (Συνεργαζόμενα Συστήματα, Cooperative Systems).

Στην παρούσα Διατριβή, πραγματοποιείται ένας συνδυασμός των δύο αυτών απόψεων:

1) Παρουσιάζεται μία μέθοδος μετατροπής τυχαίας ΒΔ σε ΒΓ ειδικής δομής,
2) Η παραγόμενη δομή ΒΓ διατηρεί κάποια χαρακτηριστικά ΒΔ και η διαχείρισή της γίνεται από ένα Διαχειριστή ΒΓ (Knowledge base Management system) που εκμεταλλεύεται τις δυνατότητες του λογικού προγραμματισμού και ειδικότερα της Prolog,
3) Προτείνεται μία τεχνική τμηματοποίησης της παραγόμενης Γνώσης σε Τμήματα Βάσης Γνώσης Περιορισμένου Μεγέθους Μνήμης (δομή Λ), ώστε να είναι εφικτή η χρησημοποίησή της σε δίκτυο μικροϋπολογιστών.

Στην εργασία [Παναγιωτόπουλος Ι.-Χ., Τσιρόπουλος, 1988] είχε αναπτυχθεί μιά δομή ΒΓ, όπου το κύριο βάρος είχε δοθεί στο να υπάρχει στην κύρια μνήμη όλη η ΒΓ σε κωδική μορφή, ενώ όλα τα υπόλοιπα αρχεία λειτουργούσαν σε επίπεδο δίσκου. Στην εργασία [Γιαλούρης, 1993] προτείνεται ο χειρισμός της ΒΓ σε δίσκους αποκλειστικά, με στόχο την ελαχιστοποίηση απαιτήσεων σε κύρια μνήμη. Στην παρούσα εργασία γίνεται χρήση κωδικοποιημένης ΒΓ, η οποία δεν βρίσκεται όλη στην κύρια μνήμη, παρά κάθε φορά ένα

απαραίτητο μέρος της. Με τον όρο Κωδικοποιημένη Γνώση εννοείται η αναπαράσταση της Γνώσης έτσι ώστε, ένα τμήμα της να είναι στη μνήμη και ειδικοί δείκτες να δείχνουν το υπόλοιπο τμήμα της σε αρχεία στους δίσκους. Αυτή η δομή επιτρέπει άμεσες προσπελάσεις σε πολύ μεγάλες ΒΓ σε επίπεδο κύριας μνήμης, ακόμη και σε απλούς μικροϋπολογιστές με απλό περιβάλλον DOS.

Στη συνέχεια θα προταθεί μία μέθοδος μετατροπής μιάς κλασσικής ΒΔ (οποιασδήποτε γνωστής δομής) σε ΒΓ. Για την υλοποίηση χρησιμοποιήθηκαν δομές όπως εκείνες που χρησιμοποιεί η Prolog (δηλωτική παράσταση Γνώσης) [Clocksin, Mellish, 1987] [Townsend, 1986]. Σε επίπεδο κύριας μνήμης θα υπάρχει αρχικά μιά Δυναμική, σχετικά περιεκτική, ΒΓ που θα παίζει το ρόλο ενός ταχύτατου ευρετηρίου περιοχών Γνώσης, οι οποίες, όπως θα αναλυθεί, θα συγκροτούν τους Τμήματα Βάσης Γνώσης Περιορισμένου Μεγέθους Μνήμης (Λ). Επίσης στην κύρια μνήμη θα έρχεται κάθε φορά, ένα κωδικοποιημένο τμήμα της Βάσης Γνώσης. Τελικά, η κωδικοποιημένη ΒΓ θα είναι χωρισμένη σε επιμέρους κωδικοποιημένες ΒΓ, μεταβλητού μεγέθους, οι οποίες θα μπορούν να βρίσκονται σε διάφορους δίσκους ενός Τοπικού Δικτύου Υπολογιστών.

Η ελαχιστοποίηση απαιτήσεων περιφεριακής μνήμης επιτυγχάνεται με αυτοοργάνωση, ώστε να μη μένουν φυσικοί χώροι κενοί στους δίσκους μετά από αλλεπάλληλες διαγραφές. Επιπλέον, κάθε αλλαγή που εκτελείται σε επίπεδο κύριας μνήμης πραγματοποιείται και στους δίσκους. Η προσπέλαση θα

επιτυγχάνεται με χρήση ειδικών τεχνικών προσπέλασης, και με ενημέρωση πραγματικού χρόνου σε επίπεδο Χρήστη.

2.2. PROLOG ΚΑΙ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

Το πρώτο Σύστημα Λογικού Προγραμματισμού (Logic Programming System) υλοποιήθηκε στη Μασσαλία από τον Colmerauer και τους συνεργάτες του. Από τότε υπήρξαν πολλές εφαρμογές της Prolog που είναι γλώσσα υψηλού επιπέδου Πέμπτης Γενιάς στην οποία υλοποιούνται οι αρχές του Λογικού Προγραμματισμού, ώστε να είναι εύχρηστη ακόμη και σε πρακτικά προβλήματα [Conery, 1987].

Χάρις στην αυστηρή μαθηματική δομή της, η PROLOG αναδείχτηκε η επικρατέστερη γλώσσα λογικού προγραμματισμού [Irani, Shih, 1984]. Τα κύρια χαρακτηριστικά της είναι τα ακόλουθα:

- Είναι κατάλληλη για υλοποίηση ΕΣ.
- Υποστηρίζει Στατικές Δυναμικές και Μεικτές ΒΓ.
- Διαχειρίζεται ΒΓ σε επίπεδο κύριας μνήμης, με ικανοποιητικούς χρόνους προσπέλασης
- Είναι κατάλληλη για υλοποίηση εκπαιδευτικών συστημάτων ικανών να εξηγήσουν τη συλλογιστική τους.

. Λόγω της αναζήτησης λύσεων με την τεχνική της οπισθοδρόμησης (backtraking), κάθε παράμετρος που λαμβάνει μέρος σε οποιαδήποτε διαδικασία είναι ισχυρά δομημένη (δευτερεύοντα κλειδιά).
. Είναι κατάλληλη για επεξεργασία φυσικής γλώσσας.

Αρκετές ερευνητικές εργασίες ανέδειξαν σχέσεις μεταξύ Σχεσιακού μοντέλου και Prolog, καθώς και μεταξύ Δικτυωτού μοντέλου και Prolog [Takizawa, Katsumata, 1989]. Αρκετές εργασίες υλοποιούν συνεργασία PROLOG με ΒΔ σε θέματα προσπέλασης και γενικότερα μέσων επικοινωνίας μεταξύ Χρήστη και ΒΔ, καθώς και σε θέματα συνύπαρξης ΒΔ και ΒΓ [Li, 1984], [Bergel, 1985], [Farris, Singleton,1989], [Takizawa,1989], [Deen,Thomas, 1990], [Meersman, Shi, Kung, 1990], [Chen, Kambayashi, 1993].

2.3. ΜΕΓΑΛΕΣ ΒΑΣΕΙΣ ΓΝΩΣΗΣ

Σε αντίθεση με τα αρχικά Συστήματα Βάσεων Γνώσης, στη δεκαετία του '90 τέτοια Συστήματα θα έχουν τεράστια έκταση (large Knowledge bases), ισχυρίζονταν οι F.Manola, M.Brodie [Brodie, Mylopoulos, 1985]. Μέχρι σήμερα είναι γεγονός ότι έχουν γίνει βήματα προς αυτή την κατεύθυνση. Οι λόγοι που ανέφεραν είναι:

a. Η πιθανή συνεργασία των Βάσεων Γνώσης με μεγάλες Βάσεις Δεδομένων (large databases),

b. Οι Βάσεις Γνώσης δεν είναι δομημένες συνοπτικά όσον

αφορά ιδιαίτερα χαρακτηριστικά που περιγράφονται με σενάρια.

c. Τα Συστήματα Διαχείρισης Βάσεων Γνώσης θα αφορούν όλες τις Βάσεις Γνώσης για όλα τα συστατικά Γνώσης (Knowledge based components). Ετσι το αποτέλεσμα θα είναι μεγάλες συγκεντρωτικές Βάσεις Γνώσης (large central knowledge bases).

Στην πορεία της Πληροφορικής η ανάπτυξη του Λογισμικού και του Υλικού έχει ως αποτέλεσμα την ευρεία χρήση μικροϋπολογιστών υψηλών δυνατοτήτων σε όλους τους τομείς. Σήμερα ένας μικροϋπολογιστής (PC) είναι σε θέση να χρησιμοποιήσει μεγάλη κύρια μνήμη σε σχέση με παλιότερα, ιδιαίτερα με DOS.x, x>=6 ή UNIX.

Παρ' όλα αυτά όμως, το μέγεθος μίας Μεγάλης ΒΓ είναι δυνατόν να ξεπερνά κατά πολύ το μέγεθος της διαθέσιμης κύριας μνήμης του μικροϋπολογιστή (και ειδικά στην περίπτωση εκείνη που ταυτόχρονα υπάρχουν στη μνήμη και άλλα προγράμματα όπως π.χ. Λογισμικό δικτύου, Windows). Γι αυτό, απαιτείται τρόπος οργάνωσης της Γνώσης, με σκοπό και να ξεπεραστεί το πρόβλημα χωρητικότητας μνήμης και να ικανοποιείται το θέμα της διαχείρισης της Βάσης Γνώσης.

Οσο για το πρόβλημα της απώλειας των μεταβολών που τυχόν έγιναν σε επίπεδο κύριας μνήμης, αυτό θα πρέπει να διευθετηθεί με έγκαιρες ενημερώσεις των δίσκων και μάλιστα ταυτόχρονα με τις μεταβολές που λαμβάνουν χώρα σε επίπεδο κύριας μνήμης.

2.4. ΤΟΠΙΚΑ ΔΙΚΤΥΑ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ ΚΑΙ ΓΝΩΣΗ

Μετά την ανάπτυξη των μεγάλων και mini συστημάτων και ταυτόχρονα με τη υλοποίηση ειδικών μηχανών για υποστήριξη θεμάτων Πέμπτης Γενιάς, όπως Lisp Machines, αντίστοιχων με τις Relational Data Base Machines, παρατηρείται η δυναμική εμφάνιση και επικράτηση των δικτύων μικροϋπολογιστών για κάλυψη αναγκών που προηγουμένως κάλυπταν μεγαλύτερα συστήματα ή συστήματα πολυεπεξεργαστών (multiprocessors).

Οσον αφορά τις ΒΔ σε δίκτυα μικροϋπολογιστών, βλέπουμε ότι ήδη έχουν αναπτυχθεί Συστήματα Κατανεμημένων ΒΔ (Distributed Data Base Systems). Επίσης τα περισσότερα προϊόντα Λογισμικού έχουν πλέον δυνατότητες εφαρμογής τους σε δίκτυα. Για την περίπτωση των ΒΓ, έχουμε αντίστοιχα τις Κατανεμημένες ΒΓ (Distributed Knowledge Bases) των οποίων η διαχείριση γίνεται στα πλαίσια ειδικών Συστημάτων Διαχείρισης Κατανεμημένων ΒΓ και βρίσκονται στο επίκεντρο της έρευνας τα τελευταία χρόνια [Mohania, Sarda, 1994], [Schwelger, 1994].

Στην περίπτωση όμως των Κατανεμημένων ΒΓ, τα ζητήματα που ανακύπτουν και θεωρούνται ανοικτά προβλήματα, είναι το πρόβλημα του πραγματικού χρόνου σε LAN και το πρόβλημα της πολυχρησίας των αρχείων δεδομένων.

2.5. ΤΜΗΜΑΤΑ ΒΑΣΗΣ ΓΝΩΣΗΣ ΠΕΡΙΟΡΙΣΜΕΝΟΥ ΜΕΓΕΘΟΥΣ ΜΝΗΜΗΣ

2.5.1 Ορισμός :

Εστω RAM_SIZE(Pj), j=1,2,...,m είναι η ελεύθερη μνήμη που αντιστοιχεί στον επεξεργαστή Pj που υπάρχει σε ένα Τοπικό Δίκτυο Υπολογιστών. Αν Hi , i=1,...,k, είναι διαμέριση της ΒΓ ΚΒ, και size(Hi) το μέγεθος του Hi, τότε και μόνο τότε τα Hi καλούνται "Τμήματα ΒΓ Περιορισμένου Μεγέθους Μνήμης" (Λ) αν ισχύει η ιδιότητα:

min(RAM_SIZE(Pj)) >= max(SIZE(Hi))

[Panayiotopoulos J.-C., Papathanassiou E., 1994]

2.5.2. Ο βασικός στόχος

Ο βασικός στόχος της παρούσης εργασίας είναι η μετάβαση από συστήματα ΒΔ σε συστήματα Κατανεμημένων ΒΓ και η δυνατότητα διαχείρισής τους σε δίκτυα μικροϋπολογιστών.

Για να αντιμετωπιστούν οι περιορισμοί μεγέθους κύριας μνήμης, είναι απαραίτητο να θεωρηθεί μία κατανεμημένη τοπολογία Γνώσης, που θα απαρτίζει λογικά μία και μοναδική Βάση Γνώσης την ΚΒ:

KB = H1∪ H2∪... ∪Hi∪...∪Hk,

όπου Hi είναι Λ, που ορίζουν μία διαμέριση στην KB, όπου κάθε κόμβος του δικτύου μπορεί να χρησιμοποιήσει οποιονδήποτε Λ, σε κάθε δεδομένη στιγμή.

2.6. ΒΑΣΙΚΕΣ ΔΟΜΕΣ

Οι δομές που ακολουθούν αφορούν την περίπτωση για έναν υπολογιστή - ένα Χρήστη, ενώ στο 5ο κεφάλαιο της εργασίας παρουσιάζεται η επέκταση των δομών αυτών για την εφαρμογή τους σε δίκτυο μικροϋπολογιστών.

Στη συνέχεια, σε κάθε περίπτωση που γίνεται προσπέλαση σε συγκεκριμένη θέση μέσα σε ένα ASCII αρχείο (με σταθερό μήκος εγγραφής), χρησιμοποιείται η σχέση [Παναγιωτόπουλος Ι.-Χ., 1990]:

(1) Filepos = (Record_size + 2) * (Record_num - 1)

Filepos : θέση της αρχής μιάς εγγραφής μέσα στο αρχείο, σε σχέση με την αρχή αυτού, εκφρασμένη σε πλήθος bytes.

Record_size : το μήκος της εγγραφής (σταθερό) σε bytes.

Record_num : ο αριθμός εγγραφής (record number).

Ακολούθως αναλύονται τα βασικά αρχεία της προτεινόμενης Δομής.

(i) Αρχείο F

Είναι μιά μικρή Βάση Γνώσης Πιλότος, ένα ευρετήριο περιοχών Γνώσης. Η δομή της εγγραφής του είναι:

h(Fromi,Untili,NameHi)

όπου:

Fromi (Untili) είναι η ελάχιστη (μέγιστη) λογική εγγραφή του Hi που αντιστοιχεί σε ένα πρωτεύον κλειδί.

NameHi είναι το πλήρες όνομα του Λ Hi (full_path name).

(ii) αρχείο namesKB

Ευρετήριο πλήρων ονομάτων (full-path, full-name) για τα f1i, f2i, piloti που αντιστοιχούν στο Hi. Η δομή του:

names(nameHi, f1i, f2i, piloti)

(iii) Αρχεία f1i , i=1,...,k

Κωδικοποιημένες ΒΓ εγγραφής σταθερού μήκους. Η κάθε μία έχει τέτοιο μέγεθος ώστε να καλύπτει τις απαιτήσεις των Λ Γνώσης. Υπάρχουν τόσα f1i, όσες οι λογικές εγγραφές του F. Η δομή εγγραφής κάθε f1i είναι:

p(Key, P0, P1, P01) (*)

όπου:

Key : πρωτεύον κλειδί

P0 :δείκτης προσπέλασης του αρχείου f1i. Δείχνει τη θέση της εγγραφής μέσα στο αρχείο f1i αν πρόκειται για υπαρκτή εγγραφή, ενώ δείχνει τη θέση της προηγουμένως διαγραμμένης, αν πρόκειται για διαγραμμένη εγγραφή (δομή διπλού ρόλου).

P1 :Ενδειξη 1 , -1 , 0 , 2 (κ.λ.π ,με ελεύθερο τρόπο) αν πρόκειται για υπαρκτή ή διαγραμμένη εγγραφή ή για εγγραφή όπου απαγορεύεται η διαγραφή ή αλλαγή (κ.λ.π.) αντίστοιχα.

P01 :δείκτης προσπέλασης στο αρχείο f2i. Δείχνει τη θέση όπου υπάρχουν όλα τα υπόλοιπα πεδία (Πραγματική Βάση Γνώσης). Κάθε κωδικοποιημένη Βάση Γνώσης (αρχείο f1i) αντιστοιχεί σε ένα αρχείο f2i.

(*) Εκτός από το key, στο f1i είναι δυνατόν να υπάρχουν οσαδήποτε πεδία, ανάλογα με τις απαιτήσεις του συστήματος.

(iv) αρχεία Piloti , i=1,...,k

Είναι απλά αρχεία ASCII, σε αμφιμονοσήμαντη αντιστοιχία με τα f1i. Η δομή είναι:

Next

All

όπου:

Next: Δείκτης προσπέλασης στο f1i, όπου υπάρχει λογικά

διαγραμμένη εγγραφή. Αν Next=-1, τότε δεν υπάρχει λογικά διαγραμμένη εγγραφή.

All : θέση πιθανής νέας εγγραφής στο τέλος του f1i.

Αν Next=-1, η νέα εγγραφή γίνεται στη θέση All (αν και μόνο αν All+(μήκος νέας εγγραφής)≤ μέγιστο επιτρεπτό μέγεθος f1i).

(v) Αρχεία f2i , i=1,...,k

Αρχεία αναλυτικής έκφρασης όλων των υπολοίπων πεδίων, σταθερού μήκους εγγραφής. Μεταξύ των f2i και f1i υπάρχει αμφιμονοσήμαντη αντιστοιχία. Η πρώτη εγγραφή του αρχείου αυτού έχει τη δομή:

Nextp, Bottomp

όπου:

Nextp: Δείκτης προσπέλασης μέσα στο αρχείο f2i. Δείχνει την αρχή της τελευταία διαγραμμένης εγγραφής που είναι έτοιμη να δεχθεί την επόμενη νέα εγγραφή. Αν Next=-1, δεν υπάρχει διαγραμμένη εγγραφή.

Bottomp :το συνολικό μέγεθος του αρχείου f2i συν 1. Αν Next=-1, τότε η νέα εγγραφή γίνεται στη θέση Bottomp.

Οι υπόλοιπες εγγραφές του f2i έχουν την εξής δομή:

field1,field2,.....,cont

όπου:

field1,field2,... είναι όλα τα υπόλοιπα πεδία Γνώσης, εκτός από τα

πεδία που υπάρχουν στο fli.

cont : η θέση όπου πιθανόν συνεχίζεται η εγγραφή. Αν δεν συνεχίζεται είναι cont=-1. Στην περίπτωση που μία εγγραφή του αρχείου f2i είναι διαθέσιμη για νέα εγγραφή (διαγραμμένη) τότε είναι αδιάφορες οι τιμές fieldi, ενώ ο δείκτης cont δείχνει το επόμενο διαθέσιμο record.

(vi) Το αρχείο inform:

Περιέχει όλες τις βασικές πληροφορίες για τη δομή της αρχικής Βάσης Δεδομένων (λεξικό - dictionary). Χρησιμοποιείται μόνο στην μετατροπή της αρχικής ΒΔ σε ΒΓ. Η δομή του είναι η ακόλουθη:

N (πλήθος εγγραφών της DB)

M (πλήθος πεδίων κάθε εγγραφής)

Start_db (θέση αρχής εγγραφών της DB)

i, ai, bi, di, Li, pi (i :αύξων αριθμός πεδίου)

όπου:

ai :μήκος πεδίου

bi :είδος πεδίου

di :1 ή 0 αν είναι κλειδί ή όχι

Li :πεδίο τιμών του i πεδίου

pi :θέση του i πεδίου

Η έννοια του inform είναι δυνατόν να ενυπάρχει σε μία ΒΔ, όπως π.χ. στις ΒΔ που κατασκευάζονται με DBaseIV.

(vii) Το αρχείο inform_kb :

Περιέχει όλες τις βασικές πληροφορίες για τη δομή της KB. Χρησιμοποιείται μόνο στην μετατροπή. Η δομή του είναι η ακόλουθη:

SP (symbol predicate (είναι το p στο p(_,_,_,_)))
NF (το όνομα του αρχείου F)
NF1 (το όνομα του αρχείου f1i)
NF2 (το όνομα του αρχείου f2i)
Y (μέγεθος ελάχιστης διατιθέμενης μνήμης.
Size_SP (μήκος του symbol predicate που χρησιμοποιείται στο f1i)
Size_P0 (μήκος του P0 και P01)
Size_Next (μήκος του Next, του All, του Nextp, του Bottomp)
i, a1i, b1i, d1i, L1i
όπου:
a1i :μήκος i πεδίου
b1i :είδος i πεδίου
d1i :1 ή 0 αν είναι κλειδί ή όχι
L1i :πεδίο τιμών του i πεδίου
Οι τιμές αυτές διαβάζονται είτε μαζικά στην αρχή είτε όπου κρίνεται σκόπιμο.

2.7. ΜΕΤΑΤΡΟΠΗ

Βασική παραδοχή: Χωρίς βλάβη της γενικότητας, μπορεί

να θεωρηθεί ότι η ΒΔ έχει τη μορφή αρχείου κειμένου (text file), αφού κάθε σύστημα έχει τη δυνατότητα παραγωγής τέτοιου αρχείου, οπότε και καλύπτονται όλες οι περιπτώσεις ΒΔ, οποιασδήποτε δομής.

Εστω η ΒΔ DB.

Εστω N το πλήθος των εγγραφών της DB,

M το πλήθος των πεδίων της εγγραφής της DB.

R={r1,r2,...,rn} το σύνολο των κανόνων από την αρχική DB.

Ας θεωρηθούν οι ακολουθίες:

a=(ai), i=1,...,M,

b=(bi), i=1,...,M,

όπου:

ai : το μήκος του i-πεδίου

bi : ο τύπος του i-πεδίου με τιμές :1,0,2,3,-1 κ.λ.π., αντίστοιχα ακέραιος, λογικός, πραγματικός, παράθεσης χαρακτήρων και -1 αν δεν αντιστοιχεί σε ουσιαστική μεταβλητή αλλά προέρχεται από κάποιο δείκτη εσωτερικής οργάνωσης ή απλά, πρόκειται για παράμετρο που δεν επιθυμούμε την ύπαρξη της στη Βάση Γνώσης.

Επιπλέον αν :

Maxi : η μέγιστη δυνατή τιμή του αριθμητικού πεδίου-i της DB,

δ2 = [-Maxint1, Maxint1] το διάστημα ακεραίων στην KB, ορίζεται η συνάρτηση:

g: a x b -> NxN

N: σύνολο φυσικών Ακεραίων

έτσι ώστε:

g(I1,1)=(I1,1) για Maxi $\in \delta 2$

g(I1,1)=(R1,2) για Maxi $\notin \delta 2$

g(R1,2)=(R1,2)

g(ai,3)=(ai,3)

...

g(ai,0)=(ai,0)

g(ai,-1)=(0,-1)

όπου:

I1 :μήκος του ακέραιου στην KB,

R1 :μήκος του πραγματικού στην KB.

Ετσι ορίζονται δύο νέες ακολουθίες που αφορούν την KB:

a1=(a1i), i=1,...,M,

b1=(b1i), i=1,...,M,

με :

a1i: μήκος του i-πεδίου στην KB με τιμές I, R, ai, 0

b1i :τύπος του πεδίου, με τιμές 1,0,2,3,-1 (ακέραιος, λογικός, πραγματικός, παράθεσης χαρακτήρων και -1 αν δεν αντιστοιχεί σε ουσιαστική μεταβλητή αλλά προέρχεται από κάποιο δείκτη και δεν υπάρχει σαν πεδίο για την KB).

2.8. Η ΜΕΤΑΦΟΡΑ

Στη συνέχεια δίνεται μία πρωτότυπη διαδικασία

μεταφοράς από Βάση Δεδομένων οιασδήποτε δομής, σε ισοδύναμη Βάση Γνώσης. Η βασική ιδέα είναι η μετατροπή των δομών της Βάσης Δεδομένων σε δομές ικανές να διαχειρίζεται η Prolog, επεκταμένες με τους κανόνες που διέπουν την αρχική Βάση Δεδομένων και με κανόνες που παράγονται από αυτήν. Οι προκύπτουσες δομές είναι εφαρμόσιμες όχι μόνο σε επίπεδο ενός μικροϋπολογιστή, αλλά και σε δίκτυα μικροϋπολογιστών.

Για τη Μεταφορά από τη Βάση Δεδομένων DB σε Βάση Γνώσης KB ακολουθούνται τρείς βασικές διαδικασίες:

Α. Προετοιμασία των βασικών αρχείων.

Β. Κυρίως μετατροπή.

Γ. Παραγωγή κανόνων.

Α. Προετοιμασία των βασικών αρχείων.

Από το αρχείο inform, σε επίπεδο κύριας μνήμης λαμβάνονται οι πληροφορίες για την DB:

N

M

Start_db

για κάθε πεδίο i=1,...,M, προσδιορίζονται τα

Ai, Bi, Di, Li, Pi

Από το inform_kb λαμβάνονται οι πληροφορίες για την KB:

Sp, NF, NF1, NF2, Y

Για i=1,...,M

A1i, B1i, D1i, L1i, καθώς και τα:

Size_SP, Size_P0, Size_P1, Size_P01, Size_Next, Size_All, Size_Nextp, Size_Bottomp

Αναζητείται το πεδίο κλειδί (key_num), το είδος του (B_{key_num}), το μήκος του (A_{key_num}) και το πεδίο ορισμού του (L_{key_num}):

Η διαμέριση των κλειδιών καθορίζεται με κάποιο τρόπο (π.χ. ευρετικά) με σκοπό τον προσδιορισμό των Λ.

$$Hi <--> [Fromi, Untili],\ i=1,..,k,$$
$$\cup\ Hi=KB,\ i=1,..,k$$
$$\cap\ Hi = \{\}$$

Το ελάχιστο πλήθος k των διαμερίσεων υπολογίζεται ως εξής:

N_f1 = int[Y/Size_rec_f1]

k= N/N_f1

με:

k :ελάχιστο πλήθος Λ

N_f1 :μέγιστο πλήθος εγγραφών f1i

Size_rec_f1:μήκος εγγραφής f1i)

Δημιουργούνται τα αρχεία F, namesKB με βάση το πεδίο ορισμού του κλειδιού, έτσι ώστε για κάθε διαμέριση κλειδιών

i=1,...,k , στο F και το namesKb γίνονται αντίστοιχα οι εγγραφές:

h(Fromi, Untili, NameHi)

names(f1i, f2i, piloti).

Δημιουργούνται τα f1i, f2i, piloti γιά κάθε i, αφού γίνει υπολογισμός του μήκους κάθε εγγραφής του f1i και f2i:

Size_rec_f1 = Len_SP + a_{key_num} + Len_P0 + Len_P1 + Len_P01+5,

Size_rec_f2 = Σa_i + (m-1) + 2 , όπου i=1,...,m και m πλήθος πεδίων f2i και Len_SP μήκος του symbol predicate

Β. Διαδικασία μετατροπής.

Για κάθε λ-εγγραφή της DB με λ=1,...,N

. Από DB, στη θέση

Pos=(λ-1)*(Size_rec_db+2)+Start_db+ Pkey_num,

διαβάζεται το κλειδί KEY1, τύπου b_{key_num}, μήκους a_{key_num}.

. Από F επιστρέφεται το Hi:

h(Fromi, Untili, nameHi),

Fromi≤Key<Untili

. Από το namesKB λαμβάνονται τα ονόματα των f1i, f2i, piloti:

f(nameHi, f1i, Piloti,F2i)

. Από piloti στη θέση Pos_p=(Size_rec_pilot+2) διαβάζεται το P0.

. Από το f2i στη θέση Pos_2=(Size_Nextp+2), διαβάζεται το P01.

. Στο f1i στη θέση P0, γίνεται η εγγραφή p(KEY1,P0,+1,P01).

. Στο piloti, στη θέση Pos_p=(Size_p+2), εγγράφεται η τιμή
All=P0+Size_rec_f1+2.

. Για κάθε j τέτοιο ώστε dj<>1 (όχι κλειδί):

..Από DB, στη θέση
Pos=(Size_rec_db+2)*(λ-1)+Start_db+Pj,
διαβάζεται το πεδίο Fieldj τύπου bj, μήκους aj.
Αν aj<>-1, στο f2H, στη θέση P01 εγγράφεται το Fieldj.

. εγγράφεται (-1) στο τέλος της λ-εγγραφής.

. Το f2i ενημερώνεται για τις τιμές:
Nextp=-1,
Bottomp=P01+Size_rec_f2 +2.

2.9. ΠΑΡΑΔΕΙΓΜΑ

Ακολουθεί ένα παράδειγμα όπου από μία δοθείσα ΒΔ παράγεται η αντίστοιχη δομή Γνώσης σύμφωνα με τη μέθοδο που αναλύθηκε στην προηγούμενη παράγραφο.

Εστω η Βάση δεδομένων DB που αφορά προϊόντα κάποιων εταιρειών. Τα προϊόντα είναι δύο τύπων, Α και Β και χαρακτηρίζονται από τον κωδικό, την ηλικία (παλαιό, μέσο, νέο), το κέρδος (υψηλό, μέτριο, χαμηλό), τον ανταγωνισμό (ναί,όχι), κόστος (χαμηλό, μέσο, υψηλό). Ας θεωρηθεί το πεδίο Κωδικός σαν πρωτεύον κλειδί.

ΚΩΔΙΚΟΣ	ΕΙΔΟΣ	ΗΛΙΚΙΑ	ΚΟΣΤΟΣ	ΚΕΡΔΟΣ	ΑΝΤΑΓΩΝ.
100	A	νέο	υψηλό	υψηλό	ναί
105	A	νέο	υψηλό	υψηλό	ναί
170	A	νέο	υψηλό	μέτριο	όχι
120	A	παλαιό	χαμηλό	χαμηλό	όχι
150	A	μέσο	χαμηλό	μέσο	όχι
190	A	νέο	χαμηλό	υψηλό	ναί
200	B	νέο	υψηλό	υψηλό	όχι
205	B	νέο	μέτριο	υψηλό	ναί
270	B	μέσο	υψηλό	μέτριο	όχι
220	B	παλαιό	χαμηλό	χαμηλό	όχι
250	B	μέσο	χαμηλό	μέτριο	ναί
290	B	νέο	χαμηλό	υψηλό	ναί

Με την εφαρμογή της μεθόδου μετατροπής παράγεται η Βάση Γνώσης KB, δομής όπως αυτή που αναλύθηκε προηγουμένως.

1) Από DB προκύπτει η KB ως εξής:

Ευρετήριο περιοχών Γνώσης(F):

---

h(100,199,1)
h(200,299,2)

---

namesKB1:

---

f(1, d:\new\KB1 , c:\new\pilot1 , e:\dir1\DKB1)
f(2, d:\mild\KB2 , c:\mild\pilot2 , c:\dir2\DKB2)

```
---     ---------------------------------------------
Pos     KB1 (f11)
---     ---------------------------------------------
0       p(          100,              0, 1,  23)
40      p(          105,             40, 1,  63)
80      p(          120,             80, 1, 103)
120     p(          150,            120, 1, 143)
160     p(          170,            160, 1, 183)
200     p(          190,            200, 1, 213)
---     ---------------------------------------------
Pos     KB2 (f12)
---     ---------------------------------------------
0       p(          200,              0, 1,  23)
40      p(          205,             40, 1,  63)
80      p(          220,             80, 1, 103)
120     p(          250,            120, 1, 143)
160     p(          270,            160, 1, 183)
200     p(          290,            200, 1, 213)
---------------------------------------
  PILOT11          PILOT12
-----------------  -------------------
-1                   -1
240                  240
--- ------------------------------------------------
 Pos    DBK11 (f21)
--- ------------------------------------------------
 0          -1    253
23      A    νέο      υψηλό      υψηλό   ναί
63      A    νέο      υψηλό      υψηλό   ναί
103     A    νέο      υψηλό      μέτριο  όχι
143     A παλαιό      χαμηλό     χαμηλό  όχι
183     A   μέσο      χαμηλό       μέσο  όχι
213     A    νέο      χαμηλό     υψηλό   ναί
```

--- ----------

Pos DKB12 (f22)

--- ----------

0		-1	253		
23	B	νέο	υψηλό	υψηλό	όχι
63	B	νέο	μέτριο	υψηλό	ναί
103	B	μέσο	υψηλό	μέτριο	όχι
143	B	παλαιό	χαμηλό	χαμηλό	όχι
183	B	μέσο	χαμηλό	μέτριο	ναί
213	B	νέο	χαμηλό	υψηλό	ναί

2.10. Παρατηρήσεις - Συμπεράσματα

Πολλά εργαλεία λογισμικού είναι προσανατολισμένα στην παραγωγή Γνώσης από Δεδομένα, με την έννοια της παραγωγής Γνώσης ή και του συστήματος αυτοδιδασκαλίας (machine learning). Επίσης, έχουν δοθεί προσεγγιστικοί αλγόριθμοι για την παραγωγή κανόνων από μία Βάση Δεδομένων (π.χ. Classification trees, ID3 Algorithm), [Quinlan, 1986], [Lin,1989], [Thomson, Thomson, 1986], [Quinlan, 1990], [Piatetsky-Shapiro, 1991].

Για τον όρο "ανακάλυψη της Γνώσης" (knowledge discovery) μιας ΒΔ, δίνεται ο ακόλουθος ορισμός από τους W. Frawley, G. Piatetsky-Shapiro και C. Matheus [Piatetsky-Shapiro, 1991 σελ 1-27]:

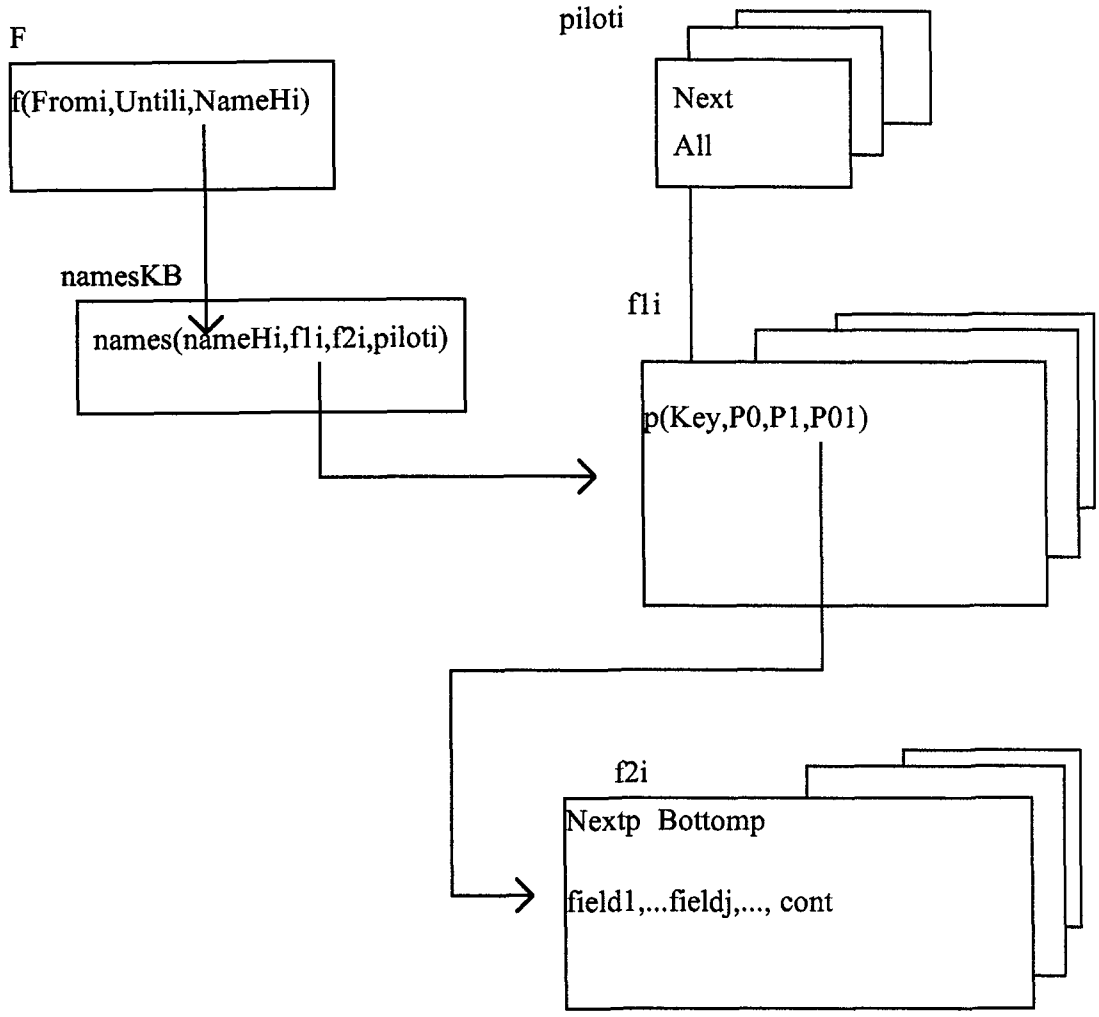
"Ανακάλυψη της Γνώσης είναι η μή τετριμμένη (nontrivial)

εξαγωγή της υπονοούμενης, προηγουμένως άγνωστης και δυναμικά χρήσιμης πληροφορίας από Δεδομένα. Αν F είναι ένα σύνολο από γεγονότα (facts), L μία γλώσσα και C κάποιο μέτρο βεβαιότητας, ορίζεται ένας σχεδιασμός (pattern) σαν μία πρόταση (statement) S στο L η οποία περιγράφει σχέσεις μεταξύ ενός υποσυνόλου Fs του S με μία βεβαιότητα C έτσι ώστε το S είναι απλούστερο (με κάποια έννοια) παρά η απαρίθμιση όλων των δεδομένων στο Fs. Ενας σχεδιασμός που είναι ενδιαφέρων (με ένα μέτρο ενδιαφέροντος καθορισμένο από τον Χρήστη) και αρκετά βέβαιο (σύμφωνα με κρητήρια του Χρήστη) καλείται Γνώση."

Για την παραγωγή κανόνων από την αρχική ΒΔ είναι δυνατή η εφαρμογή κάποιου αλγορίθμου από τους ήδη υπάρχοντες, για το σκοπό αυτό, και ειδικά για την περίπτωση μεγάλων Βάσεων Γνώσης.

Στο παρόν κεφάλαιο προτάθηκε μία μέθοδος μετατροπής Βάσης Δεδομένων σε ισοδύναμη Βάση Γνώσης Γεγονότων. Με τις παραγόμενες δομές Γνώσης επιτυγχάνεται ταχύτητα προσπέλασης και ελαχιστοποίηση της απαιτουμένης κύριας μνήμης, με αποτέλεσμα μεγάλες Βάσεις να μπορούν να φιλοξενηθούν σε απλό περιβάλλον μικροεπεξεργαστών.

Ακολούθως δίνονται σχηματικά οι δομές που έχουν ήδη περιγραφεί:


F
f(Fromi,Untili,NameHi)
namesKB
names(nameHi,f1i,f2i,piloti)
piloti
Next
All
f1i
p(Key,P0,P1,P01)
f2i
Nextp Bottomp
field1,...fieldj,..., cont

ΚΕΦΑΛΑΙΟ 3

ΤΜΗΜΑΤΑ ΒΑΣΗΣ ΓΝΩΣΗΣ ΠΕΡΙΟΡΙΣΜΕΝΟΥ ΜΕΓΕΘΟΥΣ ΜΝΗΜΗΣ (Λ) ΜΕ ΚΑΝΟΝΕΣ

3.1. ΕΙΣΑΓΩΓΗ

Στο παρόν κεφάλαιο θα αναπτυχθούν οι δομές αναπαράστασης και υποστήριξης της Βάσης Κανόνων (ΒΚ). Η ΒΚ είναι ένα από τα συστατικά της Βάσης Γνώσης του Συστήματος. Στη συνέχεια, θεωρείται ότι για την παραγωγή της ΒΚ έχει χρησιμοποιηθεί κάποια από τις γνωστές μεθόδους παραγωγής τους ή και οποιαδήποτε αντίστοιχη αλγοριθμική τεχνική [Piatetsky-Shapiro, Frawley, 1991].

Οι δομές που θα χρησιμοποιηθούν για την αναπαράσταση της Γνώσης κανόνων είναι Λ κανόνων (Τμήματα Βάσης Γνώσης κανόνων Περιορισμένου Μεγέθους Μνήμης) με σκοπό τη δυνατότητα εφαρμογής τους σε δίκτυα με n μικροϋπολογιστές και στην τετριμμένη περίπτωση, για n=1, σε ένα μεμονωμένο σύστημα μικροϋπολογιστή. Βέβαια, κάθε ένας από τους Λ, βάσει του ορισμού αυτών, θα πρέπει να έχει τη δυνατότητα ανόδου στην κύρια μνήμη οποιουδήποτε σταθμού του δικτύου.

Η τμηματοποίηση της Βάσης Γνώσης (Knowledge Base

partitioning) είναι από τα σημαντικότερα στάδια σχεδιασμού μιάς Βάσης Γνώσης αφού την καθιστά αρκετά πιό εύχρηστη [Raz, Botten, 1992]. Η ταξινόμιση της Γνώσης σε ομάδες (knowledge base clustering) είναι μία από τις προτεινόμενες προσεγγίσεις τμηματοποίησης της Βάσης Γνώσης [Botten, Kusiak, Raz, 1989], [Cheng, Fu, 1985], [Gomez, Chandrasekaran, 1981], [Jacob, Froscher, 1985], [Sridhar, Murty, 1991], [Rasmussen, 1992], [Kim, Novic, 1993], [Sridhar, Murty, 1994] . Οπωσδήποτε, το θέμα αυτό ανήκει στην ομάδα των ανοιχτών εκθετικών προβλημάτων (NP-Complete problems).

Οι Jacob και Froscher (1985) παρουσίασαν έναν αλγόριθμο ομαδοποίησης Βάσης Κανόνων χρησιμοποιώντας ως μέτρο ομοιότητας μεταξύ κανόνων το αθροιστικό βάρος (weighted sum) των κοινών μελών δύο κανόνων. Επίσης, οι Cheng και Fu (1985) έδωσαν μία διαδικασία ομαδοποίησης Βάσης κανόνων. Το μέτρο ομοιότητας μεταξύ κανόνων που πρότειναν βασίζεται στα μή κοινά μέλη δύο κανόνων. Ο προτεινόμενος αλγόριθμος δημιουργεί αστέρες κοντινών κανόνων με αποτέλεσμα τη δημιουργία αντιπροσωπευτικών κανόνων (representative rules). Οι Raz και Botten (1992) έδωσαν έναν αλγόριθμο κατανομής των κανόνων σε τμήματα περιορισμένου μεγέθους. Βασικός στόχος αυτού του αλγορίθμου είναι η ελαχιστοποίηση του αθροίσματος των επαφών (connections) μεταξύ διαμερίσεων, με απώτερο σκοπό την συντόμευση του χρόνου που απαιτείται για μεταγλωττισμό και εκτέλεση ενός Εμπείρου Συστήματος.

Το παρόν κεφάλαιο ασχολείται με την ομαδοποίηση Μεγάλων Βάσεων Γνώσης με στόχο τη χρήση τους σε τοπικά δίκτυα Ethernet με μικροϋπολογιστές οι οποίοι έχουν περιορισμό σε κύρια μνήμη και διαφορετικό μέγεθος από κεντρική μονάδα σε κεντρική μονάδα. Στη συνέχεια, προτείνεται μία μέθοδος για την δημιουργία των δομών Λ και την κατανομή των κανόνων σε αυτές.

3.2. Τμήματα Γνώσης κανόνων Περιορισμένου Μεγέθους Μνήμης (Λ κανόνων)

3.2.1 Ορισμός:

Εστω RAM_SIZE(Pj), j=1,2,...,m είναι η ελεύθερη μνήμη που αντιστοιχεί στον επεξεργαστή Pj που υπάρχει σε ένα Τοπικό Δίκτυο Υπολογιστών. Αν Hi , i=1,...,k, είναι διαμέριση της Βάσης Γνώσης κανόνων KB και size(Hi) το μέγεθος του Hi, τότε και μόνο τότε τα Hi καλούνται "Τμήματα Γνώσης κανόνων Περιορισμένου Μεγέθους Μνήμης" (Λ κανόνων) αν ισχύει η ιδιότητα:

min(RAM_SIZE(Pj)) >= max(SIZE(Hi))

[Panayiotopoulos J.–C., Papathanassiou E., 1994]

Ο ορισμός Λ κανόνων είναι ο ίδιος με εκείνον που δόθηκε για ΒΓ γεγονότων (Κεφάλαιο 2).

Αξιώματα:

_ Στους Λ κανόνων όλα τα στοιχεία του Λ έχουν κοινό πρόγονο που δεν ανήκει στο Λ.

_ Η τάξη 0 (ρίζα) του όλου δέντρου κανόνων θεωρείται ότι από μόνη της αποτελεί τον αρχικό μηδενικό Λ (κορυφή του όλου συγκροτήματος).

Το ζητούμενο είναι να γίνει σχηματισμός των Λ κανόνων έτσι ώστε ο μέσος χρόνος τελικής προσπέλασης (εντοπισμός διάγνωσης ή μη ύπαρξής της) να είναι ο ελάχιστος δυνατός. Αυτό σημαίνει να έχουμε το ελάχιστο πλήθος εναλλαγής Λ στη μνήμη μέχρι την τελική απόφαση, ενώ να υπάρχει το πολύ μία φορά ανάγκη φορτώματος ενός Λ σε μνήμη. Επιπρόσθετα, πρέπει η όλη δομή του διαχωρισμού σε Λ να επιτρέπει την κωδικοποίηση σύμφωνα με την τυποποίηση του Λογικού προγραμματισμού.

Προφανώς άριστη λύση για άριστη διαμέριση Λόφων κάτω από τέτοια πολυσύνθετα κριτήρια δεν είναι δυνατόν να βρεθεί και επομένως στη συνέχεια θα προταθεί μία ευρετική μέθοδος που να δίνει σχεδόν παντού μία καλή λύση στο πρόβλημα. Αλλωστε η έννοια της ευρετικής ανάλυσης (heuristics) υπάρχει σαν βασική τεχνική μέσα στα Έμπειρα Συστήματα, τα οποία είναι και το κεντρικό σημείο της Τεχνολογίας Γνώσης.

3.2.2. Βασικές δομές

Θεωρείται ότι κάθε κανόνας έχει την ακόλουθη δομή:

r(List_Of_Conclusions, List_of_Conditions),

όπου:

List_Of_Conditions: λίστα χαρακτηριστικών αριθμών των λεκτικών των συνθηκών του κανόνα. Συνδέονται με AND.

List_Of_Conclusions: λίστα χαρακτηριστικών αριθμών των λεκτικών των συμπερασμάτων του κανόνα. Η μορφή της είναι [a,b], όπου a,b χαρακτηριστικοί αριθμοί λεκτικών. Μεταξύ των a,b υπάρχει μία σχέση, κάποιο ρήμα, π.χ. 'είναι', 'like' που θα συμβολίζεται με G(a,b) και προκύπτει από το συγκεκριμένο δένδρο κανόνων.

Στη συνέχεια θα αναλυθούν οι βασικές δομές των αρχείων που συγκροτούν το ΣΔΒΓ.

(i) Αρχείο FR

Ευρετήριο περιοχών Γνώσης. Η δομή της εγγραφής του είναι:

L(a, NameHi, List_Of_Next(a))

όπου:

a	:όπως στον ορισμό του κανόνα.
NameHi	:το συμβολικό όνομα του Λ όπου περιέχεται ο κανόνας

List_Of_Next(a) :λίστα που περιέχει τα συμπεράσματα b όπως στον ορισμό του κανόνα, με την έννοια του G(a,b), π.χ. αν ri,rj είναι κανόνες:

ri=r([ai,bi],[ci1,...,cin]),

rj=r([ai,bj],[cj1,...,cjm])

τότε:

L(ai,NameHi,[bi,bj]).

(ii) Αρχείο namesR

Ευρετήριο τοπολογίας για τα Hi, σε σχέση με το χρησιμοποιούμενο δίκτυο:

names(nameHi, fRi, f2Ri)

όπου:

NameHi :το συμβολικό όνομα Λ που περιέχεται ο κανόνας.

fRi :πλήρη ονόματα Λ κανόνων.

f2Ri :πλήρη ονόματα Λ Λεκτικών συνθηκών-συμπερασμάτων

Βέβαια στην απλή περίπτωση ενός μοναδικού Χρήστη το αρχείο namesR θα μπορούσε να ενσωματωθεί σαν αντικείμενο μέσα στο αρχείο FR. Ωστόσο, μέσα στα πλαίσια τοπικών δικτύων με πολλούς Χρήστες, για να υπάρχει δυνατότητα κάθε Χρήστης να συνδέεται με δικό του ελεύθερο τοπικό τρόπο με τα υπόλοιπα βασικά αρχεία του

όλου γνωσιολογικού συστήματος, πρέπει να υπάρχει χωριστά η έννοια του αρχείου namesR.

Π.χ. ο Χρήστης τοπικά αποφασίζει να διαθέσει το Λογικό οδηγό M: για σύνδεση με κάποιο εξυπηρετητή (server) του δικτύου που περιέχει τα αρχεία FR, ενώ ο Υ Χρήστης μπορεί να διαθέσει το Λογικό οδηγό G: για τον ίδιο σκοπό. Με τον τρόπο αυτό η παρούσα δομή προσφέρει βαθμούς ελευθερίας στους Χρήστες δικτύων για να τρέχουν παράλληλα και άλλες εφαρμογές, οι οποίες βέβαια κατά κανόνα είναι ισχυρά τυποποιημένες και δεν προσφέρουν τέτοιες ελευθερίες.

(iii) Οικογένεια αρχείων fRi , i=1,...,k

Τα αρχεία FRi συγκροτούν τη Βάση Γνώσης κανόνων. Το μέγεθός τους είναι τέτοιο ώστε να είναι εφικτή η άνοδός τους στην κύρια μνήμη, με διαχείριση από Λογικό προγραμματισμό (Prolog). Υπάρχουν τόσα αρχεία fRi, όσοι είναι οι Λ. Η δομή είναι η ακόλουθη:

r(List_of_Conclusions, List_of_Conditions).

(V) Οικογένεια αρχείων f2Ri

Είναι οι Λ λεκτικών των συνθηκών και συμπερασμάτων, με δομή εγγραφής:

diction_No diction [r1,...,rn]

όπου:

diction_No	:ο χαρακτηριστικός αριθμός λεκτικού (όπως τα a,b,ci)
diction	:είναι το λεκτικό
[r1,..,rn]	:λίστα κανόνων που χρησιμοποιούν το λεκτικό

(vi) αρχείο F.tmp

Με εγγραφή T(a, List_Of_Nex(a)), χρησιμοποιείται για την καταχώριση της δομής δέντρου που σχηματίζουν τα λεκτικά που συμμετέχουν στα συμπεράσματα των κανόνων, όπου List_Of_Nex(a) η λίστα των b: r(_,[a,b],_).

(vii) αρχείο Index

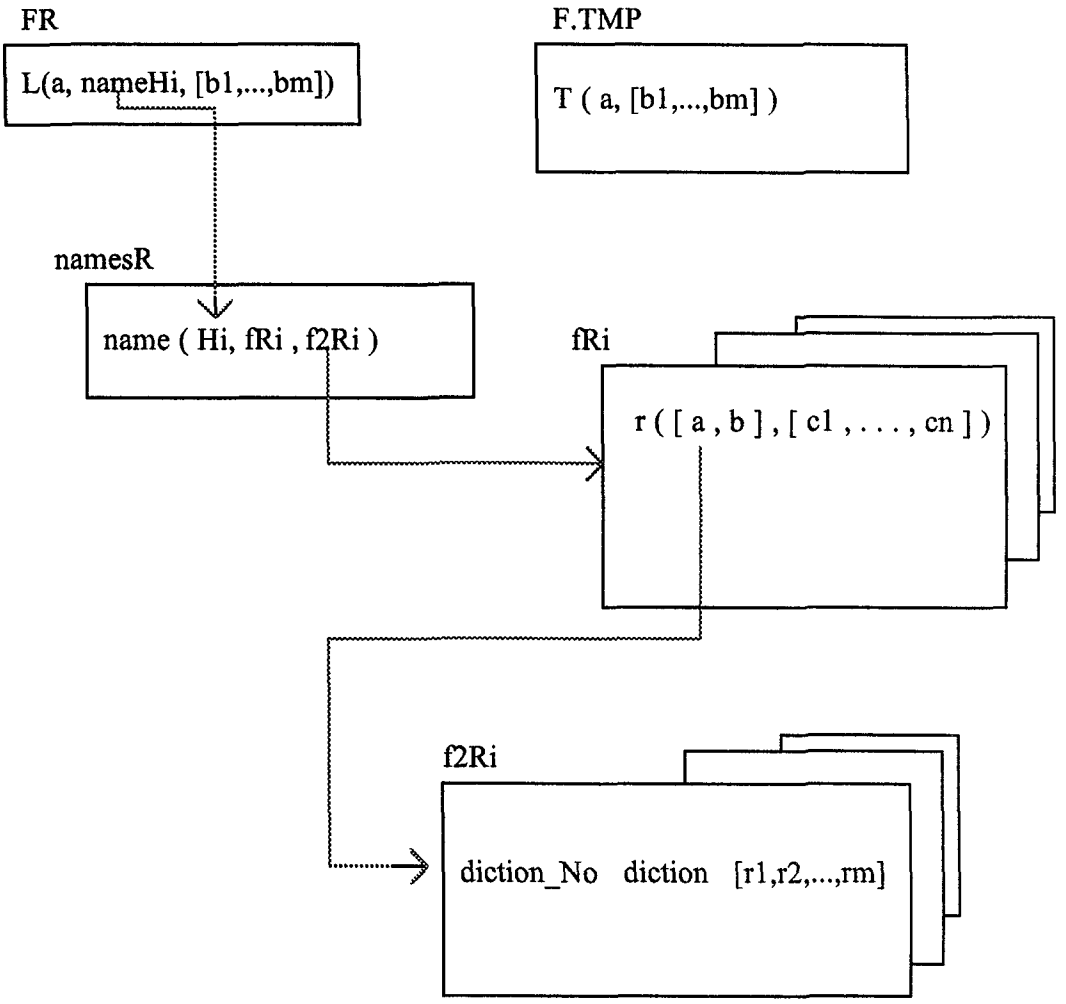
Το Index έχει την ακόλουθη δομή:

indexr(Node,List_Of_Hills)

όπου:

Node	:είναι κόμβος του δέντρου κανόνων (κάποιο root), πρόγονος του List_Of_Hills
List_Of_Hills	:η λίστα των Λ με κοινό πρόγονο το Node

Ακολουθεί σχηματική περιγραφή της δομής:



3.3. Τμηματοποίηση της Βάσης Γνώσης κανόνων σε Λ

Εστω Υ το μέγεθος του μεγαλύτερου Λ.

Η διαδικασία τμηματοποίησης θα ακολουθεί τα εξής στάδια:

α. Εύρεση της λίστας λεκτικών - ρίζα (List_of_Roots)

β. Με ρίζα το κάθε λεκτικό-ρίζα (Root) δημιουργείται το δέντρο

(tree) λεκτικών από τους κλάδους του οποίου προκύπτουν οι Λ κανόνων.

γ. Προσδιορισμός των Λ από τα δέντρα κανόνων.

Ακολούθως, θα γίνει χρήση των εξής συμβολισμών:

r([a,b],List_Of_Conditions): κανόνας.

[a,b] :συμπέρασμα με την έννοια G(a,b), π.χ."a 'είναι' b". Το a είναι ρίζα και b είναι 'επόμενό' του. b∈List_of_Next(a), a=parent(b).

List_Of_Roots: Λίστα που περιλαμβάνει έννοιες-ρίζα.

Root :είναι στοιχείο της λίστας List_of_Roots.

List_Of_Next(Root): είναι λίστα που περιλαμβάνει έννοιες του αμέσως επομένου επιπέδου του 'root'.

Parent(b):είναι το λεκτικό του αμέσως προηγουμένου του b επιπέδου. b∈ List_of_Next (Parent (b)).

List_Of_Brothers(Root): αν root ∈ list_of_roots = [Root|Rest], τότε List_of_brothers(Root)=Rest

size(r) : μήκος του κανόνα r σε bytes

assert(H,r): προσθήκη του r στο H (σε επίπεδο κύριας μνήμης)

α. Εύρεση αρχικής λίστας ριζών

_ List_Of_Roots=[]

_ Για κάθε i=1,...,N :

Διαβάζεται ο κανόνας ri=r(ai,bi,_) με Root=ai

Για κάθε κανόνα rj=r(aj,bj,_) με j<>i,

αν bj=ai τότε Root=aj,

append (List_of_Roots, Root)

β. Προσδιορισμός δένδρου για κάθε ρίζα

Προσδιορισμός:- While List_Of_Roots <> [] do

List_Of_Roots=[Root|Rest]

Εύρεση δέντρου για το Root

List_Of_Roots = Rest

Εύρεση δέντρου για το Root :-

_List_Of_Next(Root)=[]

_T(Root, List_Of_Next(Root))

_Αν δεν υπάρχει κανόνας rj=r(Root, b, _), Τέλος διαδικασίας,

διαφορετικά

∀ rj=r(Root, b, _) : append (List_Of_Next(Root), b),

Επανάληψη της διαδικασίας με :

List_Of_Roots=List_Of_Next(Root).

γ. Προσδιορισμός Λ κανόνων

Τίθεται size=0

Εκτελείται το υποπρόγραμμα εύρεσης αρχικής λίστας ριζών :

List_Of_Roots (βλ. (α)).

Repeat

List_of_Roots = [Root|Brothers(Root)]

Εύρεση List_of_Next(Root) από το T(Root,List_Of_Next(Root))

Προσδιορισμός του parent(Root): T(parent,[...,Root,...])

Αν List_Of_Next(Root) ≠ [] τότε:

List_Of_Next(Root) = [Next1|Brothers(Next1)],

parent(Next1) ← Root.

newRoot ← Next1,

εύρεση List_Of_Next(Next1): (Next1,List_Of_Next(Next1),

newList_Of_Roots ← List_Of_Next(Root),

ΕΛΕΓΧΟΣ

else

newRoot ← parent(Root)

List_Of_Next(newRoot) ← Brothers(Root)

newList_Of_Roots ← Brothers(parent(Root))

Until List_of_Roots = []

ΕΛΕΓΧΟΣ:- r=r(Root,Next1,_),

size=size+size(r),

αν size < Y , τότε

assert(Hi, r),

assert(Fr,L(Root,nameHi,List_Of_Next(Root)))

διαφορετικά

ΕΝΗΜΕΡΩΣΗ INDEX

size=0

assert(H_{i+1}, r),

assert(Fr,L(Root,nameH_{i+1},List_Of_Next(Root)))

ΕΝΗΜΕΡΩΣΗ INDEX:- Node← parent(Root),

append(Hills_of_Node,Hi),

assert(index,indexr(Node,Hills_of_Node)

δ. Συνένωση Λ

Στην πορεία των εργασιών (πράξεων), είτε λόγω διαγραφών, είτε λόγω υπερχείλισης Λ, οπότε αναγκαστικά δημιουργούνται και άλλοι μικρότεροι Λ, είναι δυνατόν να υπάρχουν και Λ μικρού πληθυσμού. Στο σημείο αυτό πρέπει να αντιμετωπιστεί το ενδεχόμενο ύπαρξης αρκετών Λ μικρού μεγέθους, γεγονός το οποίο θα επιβάρυνε το χρονικό κόστος σε κάθε πολλαπλή απαίτηση.

Για κάθε σύνολο Λ {H_i, i=1,...,n} με κοινό Root προηγούμενου επιπέδου: indexr(Root,List_Of_Hills) με Hi∈ List_Of_Hills, είναι δυνατόν να εφαρμοστεί ο αλγόριθμος συνένωσης ο προτεινόμενος από τους Raz, Botten [Raz, Botten, 1992], αλλά με τις εξής προτεινόμενες τροποποιήσεις:

. Σαν "περιοχή" (area) με την έννοια που δίνουν οι Raz, Botten [Raz, Botten, 1992] στον όρο "area", είναι δυνατόν να θεωρηθεί κάθε Λ.

. Ο αριθμός που χαρακτηρίζει την κάθε 'περιοχή' δεν είναι το μέγεθος

κανόνα αλλά το μέγεθος size(H_i) του κάθε Λ H_i.

. Ο περιορισμός μεγέθους μνήμης Υ είναι πλέον καθορισμένος: Υ=min(max_size(pj)), όπως ήδη έχει οριστεί στον ορισμό του Λ.

ε. Διάσπαση Λ κανόνων

Η Διάσπαση ενός Λ κανόνων επιβάλλεται στην περίπτωση που το αναμενόμενο μέγεθος του Λ είναι μεγαλύτερο από την τιμή Υ:

_ Αν Size(FRi)>Υ τότε για το σύνολο κανόνων FRi εκτελούνται τα ακόλουθα:

(α) Εύρεση αρχικής λίστας ριζών

(β) Προσδιορισμός δέντρου για κάθε ρίζα

(γ) Προσδιορισμός Λ κανόνων

Επομένως, για να γίνει δυνατή η διάσπαση Λ κανόνων θα πρέπει στην ουσία να γίνει εξ αρχής πλήρης αναδιοργάνωση προσδιορισμού Λ. Αρα είναι προτιμότερο να γίνεται δημιουργία νέου Λ όταν αυτό απαιτηθεί, παρά η πράξη διάσπασης Λ.

3.4. ΒΑΣΙΚΕΣ ΠΡΑΞΕΙΣ με κανόνες

3.4.1 Αναζήτηση κανόνα (πράξη INQR)

_Δίνεται το a και αναζητείται ο κανόνας r(a,b, [c1,c2,...,cn])

Από το αρχείο FR προσδιορίζεται η εγγραφή L(a,NameHi,)

_Από το namesR προσδιορίζεται το πλήρες όνομα του Λ κανόνων fRi, όπου ανήκει ο κανόνας: names(nameHi,fRi)

_Από το Λ κανόνων fRi, σε επίπεδο κύριας μνήμης προσδιορίζεται ο κανόνας και η λίστα υποθέσεων του.

_Αν ισχύει η λίστα υποθέσεων (δηλ. αν ιχύει ο κανόνας), τότε για κάθε b∈List_Of_Next επαναλαμβάνεται η διαδικασία από το πρώτο βήμα, ενώ παράλληλα τα γεγονότα που τον ικανοποιούν προστίθενται στη λίστα dynamik_yes, [γ1,γ2,...,γθ], ενώ αυτά που δεν τον ικανοποιούν στη λίστα dynamik_not, [ο1,ο2,..,ολ].

Οι dynamik_yes,dynamik_not είναι λίστες οι οποίες περιέχουν γεγονότα που έχουν ελεγχθεί και αντίστοιχα ανήκουν ή όχι στη λίστα υποθέσεων του κανόνα.

3.4.2 Προσθήκη νέου κανόνα (πράξη ADDR)

_Δίνεται ο κανόνας που πρόκειται να προστεθεί r(a,bθ,[ψ1,ψ2,...,ψν]).

_Εκτελείται η πράξη INQR

_Αν ο κανόνας δεν υπάρχει, γίνεται έλεγχος αν τα επιπρόσθετα γεγονότα (αυτά που υπάγονται στον κανόνα) και το bθ ανήκουν στη λίστα dynamik_yes. Αν S_new είναι τα επιπρόσθετα γεγονότα που δεν ανήκουν στην dynamik_yes, και το bθ, τότε:

$$\text{dynamik\_yes} = [\gamma 1,\gamma 2,...\gamma\theta] \cup \text{S\_new}$$

_Αν το bθ δεν υπάρχει γίνεται πρόσθεση του λεκτικού του.

_Αν κάποιο από τα S_new δεν υπάρχει, το λεκτικού του προστίθεται στο f2Ri

Προσδιορισμός Λ: L(a,Hi,)

_Αν size(Hi)+size(r)<Y τότε:

assert(Hi,r), append(List_of_Next(a), b)

διαφορετικά

assert(Hnew,r), assert(T,b,[])

(δηλαδή δημιουργείται νέο Λ: Hnew)

3.4.3 Διαγραφή κανόνα (πράξη DELR)

_Δίνεται ο κανόνας r(a,b,conditions)

Εκτελείται η πράξη INQR

_Αν ο κανόνας υπάρχει, και fRi ο Λ στον οποίο ανήκει:

Στο fRi : retract(r),

save(fRi).

_Διαγράφεται το b από List_of_Next(a)

3.4.4 Αλλαγή κανόνα (πράξη CHGR)

_Δίνεται ο κανόνας που θα υποστεί μεταβολές.

_Εκτελείται η πράξη INQR.

_Αν ο κανόνας υπάρχει, εκτελείται η πράξη DELR

_Εκτελείται η πράξη ADDR για την νέα μορφή του κανόνα.

3.5. Παρατηρήσεις - Συμπεράσματα

Με την τμηματοποίηση της Γνώσης κανόνων σε Λ καθίσταται δυνατή η χρήση της σε επίπεδο κύριας μνήμης ανεξάρτητα από το μέγεθος και της Βάσης Γνώσης Κανόνων και της κύριας μνήμης σε μικροϋπολογιστές και τοπικά δίκτυα.

Ενα λεπτό σημείο που πρέπει οπωσδήποτε να προσεχθεί ιδιαίτερα είναι η σχέση των κανόνων με τα απλά γεγονότα στην περίπτωση της παραγωγικής Γνώσης, όπου ένας κανόνας αντικαθιστά γνωσιολογικά δύο ή και περισσότερα απλά γεγονότα. Επομένως, στην περίπτωση διαγραφής κανόνα (DELR), αυτό ισοδυναμεί με την αυτόματη διαγραφή ενός συνόλου γεγονότων. Αρα, είναι πιθανό να διαγραφούν αυτομάτως ορισμένα γεγονότα που πιθανώς είναι χρήσιμα. Προτείνεται λοιπόν σαν πρώτο στάδιο η εφαρμογή του κανόνα σε πράξη αναζήτησης γεγονότων, στη συνέχεια η απομόνωση αυτών που δεν αναφέρονται στη βάση γεγονότων και τέλος η προσεκτική παρατήρηση της χρησιμότητας των γεγονότων που απέμειναν. Παρατηρούμε λοιπόν ότι τελικά επαφίεται στο Χρήστη η απόφαση διαγραφής κανόνα.

Η πράξη ADDR είναι επίσης πιθανό να απορροφά αυτόματα γνωσιολογικά ένα ή και περισσότερα γεγονότα από τη

βάση γεγονότων. Στην περίπτωση αυτή πριν προστεθεί ο νέος κανόνας θα πρέπει να εφαρμοστεί σε επίπεδο αναζήτησης γεγονότων, με την έννοια της παραγωγικής γνώσης (γραμμή διαταγής στόχου- goal command line), οπότε θα εμφανιστούν και τα γεγονότα που περικλείει γνωσιολογικά ο κανόνας αυτός. Στη συνέχεια θα πρέπει κανονικά ο Χρήστης να διαγράψει από τα γεγονότα που εμφανίστηκαν, όσα ήδη υπάρχουν στη βάση γεγονότων, διαφορετικά υπάρχει ο κίνδυνος διπλοεγγραφών με τις γνωστές δυσάρεστες συνέπειες.

Συμπερασματικά λοιπόν, η διαχείριση κανόνων εμπεριέχει από τη φύση της δύσκολα ανοικτά προβλήματα, ιδιαίτερα στα θέματα εκείνα που συνδέουν τους κανόνες με τα απλά γεγονότα.

ΚΕΦΑΛΑΙΟ 4

ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΚΑΙ ΔΙΑΧΕΙΡΙΣΗ

4.1. ΕΙΣΑΓΩΓΗ

Στο παρόν κεφάλαιο παρατίθεται η διαχείριση του παραγόμενου συστήματος σε επίπεδο μικροϋπολογιστή.

Ενα σύστημα Βάσεων Γνώσης (knowledge base system) αποτελείται από Βάσεις Γνώσης, από ένα μηχανισμό εξαγωγής συμπερασμάτων (Inference Engine IE), ενδεχομένως από Βάσεις δεδομένων (Συνεργαζόμενα Συστήματα, Cooperative Systems) και ένα Σύστημα επικοινωνίας με τον Χρήστη (User Interface UI), (Human Window).

Σύμφωνα με τις δομές που έχουν αναλυθεί στο 2ο κεφάλαιο το σύνολο των τριάδων:

{<f1i,f2i,Rj> i=1,...,k, j>=1 }

έχει την έννοια του Κατανεμημένου συστήματος Βάσης Γνώσης, αφού οι δομές είναι κατανεμημένες από φυσική και λογική άποψη. :

. Τα f1i αποτελούν την βάση γεγονότων του συστήματος. Η προσπέλαση τους γίνεται σε επίπεδο κύριας μνήμης.
. Τα Rj περιέχουν κανόνες και αποτελούν την βάση κανόνων

(rule base) του συστήματος, η διαχείριση της οποίας γίνεται σε επίπεδο κύριας μνήμης.

. Τα f2i προσπελαύνονται σε επίπεδο δίσκων και διατηρούν τη δομή της κλασσικής Βάσης Δεδομένων.

. Ο μηχανισμός εξαγωγής συμπερασμάτων (IE) αναλύει τις απαιτήσεις του Χρήστη, παράγει τα επί μέρους συμπεράσματα και τα συνθέτει, με αποτέλεσμα να προκύψει το τελικό συμπέρασμα που απαντά στην ερώτηση του Χρήστη.

. Το υποσύστημα επικοινωνίας με τον Χρήστη (UI) δέχεται την απαίτηση (query), την αναθέτει στον μηχανισμό εξαγωγής συμπερασμάτων και επιστρέφει τα συμπεράσματα.

. Τέλος, το υποσύστημα μάθησης (L) έχει στόχο την παραγωγή νέων κανόνων (Παραγωγική Γνώση), την τήρηση στατιστικών στοιχείων και την αυτο-αναδιοργάνωση.

Η διαχείριση γίνεται στα πλαίσια ενός Συστήματος Διαχείρισης Τμηματοποιημένων Βάσεων Γνώσης (ΣΔΚΒΓ), που εφαρμόζεται σε επίπεδο ενός Χρήστη, καθώς και σε επίπεδο δικτύου ικανοποιώντας ορισμένες προϋποθέσεις πραγματικού χρόνου και λογισμικής ασφάλειας αρχείων.

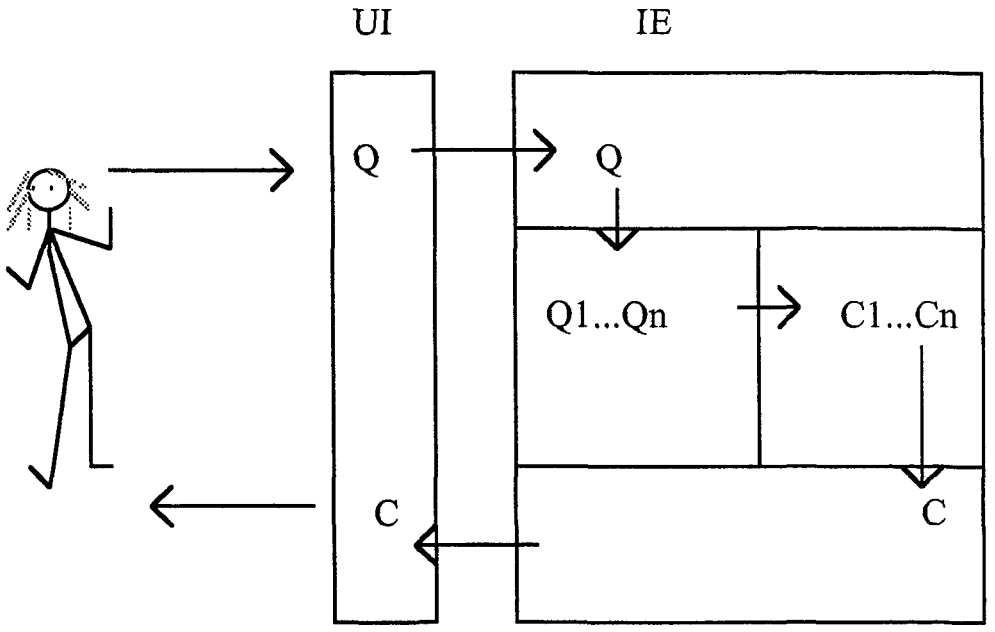
4.2. Η ΠΟΡΕΙΑ ΑΠΟ ΤΗΝ ΑΠΑΙΤΗΣΗ ΣΤΟ ΣΥΜΠΕΡΑΣΜΑ

Τα βήματα που ακολουθούνται για να παραχθεί συμπέρασμα-απάντηση σε κάποια απαίτηση του Χρήστη είναι τα

ακόλουθα:

1. Λήψη της απαίτησης Q του Χρήστη από το υποσύστημα UI.
2. Το UI στέλνει την Q στο υποσύστημα IE.
3. Το IE αναλύει την Q σε απλές απαιτήσεις q1,q2,...,qn.
4. Παράγονται, σε συνεργασία με τις υπόλοιπες δομές τα επιμέρους συμπεράσματα c1,c2,...,cn .
5. Συντίθενται τα c1,c2,...,cn στο τελικό συμπέρασμα C.
6. Το C στέλνεται στο υποσύστημα UI.
8. Το UI επιστρέφει το C στον Χρήστη.

Παρατίθεται σχηματικά η όλη διαδικασία:



4.3. ΟΙ ΒΑΣΙΚΕΣ ΠΡΑΞΕΙΣ

4.3.1 ΑΠΛΗ ΑΝΑΖΗΤΗΣΗ: ΠΡΑΞΗ INQ

_ Εστω ο u-Χρήστης σε επίπεδο πράξης αναζήτησης (INQ).

_ Εισαγωγή απαίτησης Q(Key1).

_ Αν Key1 εκτός πεδίου ορισμού, τότε

Εξοδος [1]: "Ανύπαρκτο περιεχόμενο εξόδου"

_ Προσπέλαση αρχείου F για προσδιορισμό του H:

h(Fromk,Untilk,H), Fromk<=Key1<Untilk

_ Από το namesKB λαμβάνεται το όνομα F1H :

f(H,F1H,PilotH,F2H)

_ Αν το αρχείο F1H δεν είναι στη μνήμη (RAM), τότε

. Φόρτωμα F1H στη μνήμη.

_ Εντοπισμός τιμών P1, P01 από F1H:

p(Key1,P0,P1,P01).

_ Αν P1<>-1, τότε

. Επιστρέφονται οι τιμές cont, FieldHj, (όλα τα πεδία στο F2H) από τη θέση P01 του αρχείουF2H.

. Αν cont<>-1, τότε στη θέση cont γίνεται προσπέλαση του υπολοίπου της εγγραφής.

Διαφορετικά

.. Εξοδος [2]:"Ολοκληρώθηκε η προσπέλαση της εγγραφής".

_Τέλος Υποδιαδικασίας.

Παράδειγμα 3.1: Θα ληφθεί υπ' όψην το παράδειγμα που αναπτύχθηκε στο 2ο κεφάλαιο. Αν δοθεί από τον Χρήστη η απαίτηση Q1: "Να βρεθούν όλα τα χαρακτηριστικά για το προϊόν με κωδικό 205", η αναζήτηση θα έχει ως εξής (Τα F, namesKb βρίσκονται διαρκώς στην κύρια μνήμη):

_Ο u-Χρήστης είναι σε επίπεδο πράξης INQ.

_Εισαγωγή απαίτησης Q(205).

_205 εντός πεδίου ορισμού.

_Προσπέλαση αρχείου f0 για προσδιορισμό του H:

h(Fromk,UntilK,H), Fromk<=205<Untilk,

ισχύει h(200,299,2), άρα H=2.

_Από το namesKb λαμβάνεται το όνομα F1H

f(2,F1H, PilotH, F2H)

ισχύει

f(2,"d:\mild\KB2","c:\new\pilot2","c:\dir2\DKB2"),

άρα :

F1H="d:\mild\KB2",

PilotH="c:\new\pilot2",

F2H="c:\dir2\DKB2".

_Αν το αρχείο δεν είναι σε RAM, τότε

. Φόρτωμα F1H σε RAM.

_Εντοπισμός τιμών P1,P01 από F1H:

p(205,40,1,63)

άρα P1=1, P01=63.

_ Ισχύει P1<>-1, οπότε

. Επιστρέφονται οι τιμές cont, FieldHj,
από τη θέση 63 του αρχείου F2H:
cont=-1,
τύπος="Α", ηλικία="νέο",κόστος="υψηλό",
κέρδος="υψηλό", ανταγωνισμός="ναι".

_Εξοδος: "Η έξοδος των αποτελεσμάτων ολοκληρώθηκε".

_Τέλος.

4.3.2 ΑΠΛΗ ΔΙΑΓΡΑΦΗ: ΠΡΑΞΗ DEL

_Εστω ο u-Χρήστης σε επίπεδο πράξης DEL.

_Εισαγωγή της αντίστοιχης απαίτησης διαγραφής Q(Key1).

_ Εκτέλεση της πράξης INQ.

_Αν η πράξη INQ είχε επιτυχία εκτέλεσης, τότε

. Από pilotH διαβάζονται οι τιμές Next, All.

. Τίθεται Next_new=P0

P0_new =Next

P1_new =-1

. Στο αρχείο f1i στη θέση P0 γίνεται η εγγραφή

p(Key1,P0_new,P1_new,P01)

. Στο αρχείο pilotH στη θέση του Next εγγράφεται η τιμή Next_new.

. Στο αρχείο f2i εγγράφεται Nextp_new=P01

Αν cont=-1 τότε cont_new=Nextp

Διαφορετικά Cont_new=Nextp, Next_new=Cont.

Εξοδος [1]: "Αποτυχία εκτέλεσης πράξης DEL"

_ Εξοδος [2]: "Επιτυχία εκτέλεσης πράξης DEL".

_ Τέλος Υποδιαδικασίας.

Παράδειγμα 3.2.α: Αν δοθεί από τον Χρήστη η απαίτηση Q1:"Να διαγραφεί το προϊόν με κωδικό 100", η διαδικασία διαγραφής θα είναι η ακόλουθη (θεωρώντας ότι F, namesKb, βρίσκονται διαρκώς στη μνήμη):

_Ο u-Χρήστης σε επίπεδο πράξης DEL.

_Εισαγωγή της απαίτησης Q(100).

_Εκτέλεση της πράξης INQ.

_Η πράξη INQ είχε επιτυχία.

(Από το F βρέθηκε ο Λ όπου ανήκει το κλειδί 100:

h(Fromi,Untili,H), Fromi<=100<Untili,

ισχύει για h(100,199,1), άρα H=1.

Από το namesKb βρέθηκαν τα πλήρη ονόματα των:

F1H="d:\new\KB1",

PilotH="c:\new\pilot1" και F2H="c:\dir1\DKB1".

Από το F1H="d:\new\KB1":

p(100,P0,1,P01),

που ισχύει για:

p(100,0,1,23),

είναι P0=0, P01=23, οπότε

. Τίθεται Next_new=P0 =0

P0_new=Next =-1

P1_new =-1

. Στο αρχείο KB1 στη θέση 0 γίνεται η εγγραφή

p(100,-1,-1,23)

. Στο αρχείο PilotH εγγράφεται η τιμή

Next_new=-1.

. Στο αρχείο DKB1 εγγράφεται

Nextp_new=P01=23

είναι cont=-1, άρα cont_new=Nextp=-1.

_Επιτυχία της πράξης DEL.

_Τέλος.

Οι νέες λοιπόν μορφές των δομών που υπέστησαν μεταβολή έχουν ως εξής:

Pos	KB1 (f11)
0	p(100, -1,-1, 23)
40	p(105, 40, 1, 63)
80	p(120, 80, 1, 103)
120	p(150, 120, 1, 143)
160	p(170, 160, 1, 183)
200	p(190, 200, 1, 213)

PILOT1
0
200

Pos DBK1 (f21)

0	23	253				
23	"A"	"νέο"	"υψηλό"	"υψηλό"	"ναί"	-1
63	"A"	"νέο"	"υψηλό"	"υψηλό"	"ναί"	-1
103	"A"	"νέο"	"υψηλό"	"μέτριο"	"όχι"	-1
143	"A"	"παλαιό"	"χαμηλό"	"χαμηλό"	"όχι"	-1
183	"A"	"μέσο"	"χαμηλό"	"μέσο"	"όχι"	-1
213	"A"	"νέο"	"χαμηλό"	"υψηλό"	"ναί"	-1

Παράδειγμα 3.2.β: Σε συνέχεια του Παραδείγματος 3.2.α, δίνεται η απαίτηση Q:"Να διαγραφεί το προϊόν με κωδικό 170"

_Εισαγωγή της απαίτησης Q(100).

_Εκτέλεση της πράξης INQ.

_Η πράξη INQ είχε επιτυχία.

(Από το f0 βρέθηκε ο Λ όπου ανήκει το κλειδί 170:

h(Fromi,Untili,H), Fromi<=170<Untili,

που ισχύει για h(100,199,1), άρα H=1.

Από το namesKb βρέθηκαν τα πλήρη ονόματα των:

F1H=d:\new\KB1, PilotH=c:\new\pilot1, F2H=c:\dir1\DKB1

Από το F1H=d:\new\KB1: p(170,P0,1,P01),

που ισχύει για:

p(170,160,1,183),

είναι P0=170, P01=183,

οπότε:

. Next_new=P0 =160

P0_new=Next =0

P1_new =-1

. Στο αρχείο KB1 στη θέση 160 γίνεται η εγγραφή

p(170,-1,-1,183)

. Στο αρχείο Pilot1 εγγράφεται η τιμή

Next_new=160

. Στο αρχείο DKB1 (όπου Nextp=23) εγγράφεται

Nextp_new=P01=183

είναι cont=-1, άρα cont_new=Nextp=23.

Επιτυχία της πράξης DEL.

Τέλος.

Οι ακόλουθες δομές θα έχουν ως εξής:

Pos	KB1 (f11)
0	p(100, -1,-1, 23)
40	p(105, 40, 1, 63)
80	p(120, 80, 1, 103)
120	p(150, 120, 1, 143)
160	p(170, 0,-1, 183)
200	p(190, 200, 1, 213)

PILOT1
160
200

```
---- ------------------------------------------
Pos     DBK1 (f21)
--- ------------------------------------------
0           183          253
23      "A" "     νέο" "   υψηλό" "   υψηλό" "ναί" -1
63      "A" "     νέο" "   υψηλό" "   υψηλό" "ναί" -1
103     "A" "     νέο" "   υψηλό" "   μέτριο" "όχι" -1
143     "A" "παλαιό" " χαμηλό" " χαμηλό" "όχι" -1
183     "A" "    μέσο" " χαμηλό" "    μέσο" "όχι" 23
213     "A" "     νέο" " χαμηλό" "  υψηλό" "ναί" -1
```

4.3.3 ΕΙΣΑΓΩΓΗ ΑΠΛΗΣ ΕΓΓΡΑΦΗΣ: ΠΡΑΞΗ ADD

_Εστω ο u-Χρήστης σε επίπεδο πράξης ADD.

_Εισαγωγή της αντίστοιχης απαίτησης νέας εγγραφής

Q(Key1).

_Εκτέλεση της πράξης INQ.

_Αν η πράξη INQ έχει επιτυχία εκτέλεσης, τότε

. Εξοδος [1]: "αποτυχία εκτέλεσης πράξης ADD".

Διαφορετικά

. Εισαγωγή Fieldj, j=1,...,k-1,k+1,...,m

. Διαβάζονται οι τιμές των Next,All,Nextp, Bottomp.

. Αν Next<>-1, τότε

Pos1=Next,

Next_new=P0,

All_new=All,

διαφορετικά

Pos1=All, All_new=All+Size_rec_f1

Next_new=Next.

. Από το F1H στη θέση Pos1 διαβάζεται η

p(Key,P0,P1,P01)

. Αν Nextp<>-1 τότε

.. Pos2=Nextp

.. Στο F2H θέση Pos2 διαβάζεται η cont

.. Nextp_new=cont

διαφορετικά (αν Nextp=-1)

.. Pos2=Bottomp

.. Bottomp_new=Bottomp+Rec_Size_f2

. Τίθεται P0_new =Pos1

P01_new =Pos2

P1_new=1, (ή cont_new=Nextp)

. Στο PilotH εγγράφονται οι τιμές

Next_new, All_new

. Στο F1H στη θέση Pos1 εγγράφεται

p(Key1,P0_new,P1_new,P01_new).

. Στο F2H στη θέση Pos2 εγγράφονται τα

Fieldi, i=1,...,k-1,k+1,...,m, cont_new

_Αν All_new+Size_Rec_f1 > max_Size_f1, τότε

εκτελείται η διαδικασία διάσπασης του F1H.

_Εξοδος [2]: Επιτυχία εκτέλεσης πράξης ADD.

_Τέλος διαδικασίας.

Σημείωση: Για αποφυγή της περίπτωσης το All να έχει τιμή μεγαλύτερη από το μέγιστο επιτρεπτό μέγεθος κάθε αρχείου f1i, την

κατάλληλη στιγμή (πρίν την εγγραφή) γίνεται διάσπαση του αρχείου fli σε αρχείο fli1, αρχείο fli2 και ταυτόχρονα αναδιοργανώνεται το αρχείο F (η αντίστοιχη διαδικασία περιγράφεται αργότερα).

Παράδειγμα 3.3.α: Ας θεωρηθούν οι δομές του παραδείγματος του 2ου κεφαλαίου όπως έχουν προκύψει μετά και από το παράδειγμα 3.2.β. Αν δοθεί από τον u-Χρήστη η απαίτηση Q3: "Να εγγραφεί το προϊόν με κωδικό=102".

_Εισαγωγή της αντίστοιχης απαίτησης Q(102).

_Εκτέλεση της πράξης INQ.

_Η πράξη INQ αποτυγχάνει.

(Αφού έχουν βρεθεί τα πλήρη ονόματα των F1H="d:\new\KB1", PilotH="c:\new\pilot1", F2H="e:\dir1\DKB1", και οι τιμές Next=160, Nextp=183).

_Εισαγωγή των νέων πεδίων

τύπος="A", ηλικία="νέο", κόστος="χαμηλό",

κέρδος="υψηλό", ανταγωνισμός="ναι"

_Είναι Next=160 <>-1, οπότε Pos1=Next

_Από το F1H στη θέση 160 διαβάζεται η p(170,0,-1,183)

_Είναι Nextp<>-1, οπότε

Pos2=Nextp=183

_Στο F2H θέση Pos2=183 διαβάζεται η cont=23

_ Τίθεται: Next_new=P0=0,

P0_new=Next=160,

Nextp_new=cont =23,

cont_new=-1,

_Στο pilotH εγγράφεται η Next_new

_Στο αρχείο F1H="d:\new\KB1" στη θέση 160 εγγράφεται

p(102, 160, 1, 183).

_Στο F2H="e:\dir1\DKB1" θέση 160 εγγράφονται οι τιμές όλων, πλήν του κλειδιού, πεδίων και η νέα τιμή cont=-1.

Κατόπιν αυτών θα είναι:

```
----  ------------------------------------------------
Pos    KB1 (f11)
----  ------------------------------------------------
0      p(100,                  -1,-1,      23)
40     p(105,                  40, 1,      63)
80     p(120,                   80, 1,     103)
120    p(150,                  120, 1,     143)
160    p(102,                  160, 1,     183)
200    p(190,                  200, 1,     213)

 PILOT1
---------------
  0
200
----  ------------------------------------------
Pos         DBK1 (f21)
----  ------------------------------------------
 0         23     253
23         "A" "     νέο" "   υψηλό" "   υψηλό" "ναί" -1
63         "A" "     νέο" "   υψηλό" "   υψηλό" "ναί" -1
103        "A" "     νέο" "   υψηλό" "   μέτριο" "όχι" -1
143        "A" "παλαιό" " χαμηλό" " χαμηλό" "όχι" -1
183        "A" "     νέο" " χαμηλό" "   υψηλό" "ναί" -1
213        "A" "     νέο" " χαμηλό" "   υψηλό" "ναί" -1
```

4.3.4 ΑΠΛΗ ΑΛΛΑΓΗ: ΠΡΑΞΗ CHG

_.Εστω ο u-Χρήστης σε επίπεδο πράξης CHG.

_.Εισαγωγή της αντίστοιχης απαίτησης ενημέρωσης εγγραφής Q(Key1).

_.Εκτέλεση της πράξης INQ.

_.Αν η πράξη INQ αποτύχει, τότε

. Εξοδος [1]: "αποτυχία εκτέλεσης πράξης CHG".

Διαφορετικά

. Εισαγωγή Fieldi, i=1,...,k-1,k+1,...,m

Στο F2H στη θέση P01 εγγράφονται οι νέες τιμές των

Fieldi (i=1,...,k-1,k+1,...,m), cont

_ Εξοδος [2]: "Επιτυχία εκτέλεσης πράξης CHG".

_Τέλος διαδικασίας.

Παράδειγμα 3.4: Θα γίνει χρήση του παραδείγματος του 2ου κεφαλαίου όπως έχει ήδη διαμορφωθεί. Αν ο Χρήστης δόσει την απαίτηση Q:"Να γίνει μεταβολή στα χαρακτηριστικά του προϊόντος με κωδικό 105"

_ Εστω ο u-Χρήστης σε επίπεδο πράξης CHG.

_ Εισαγωγή της απαίτησης ενημέρωσης εγγραφής Q(105).

_ Εκτέλεση της πράξης INQ.

_ Η πράξη INQ επιτυγχάνει

τύπος="Α", ηλικία="νέο", κόστος="υψηλό",

κέρδος="υψηλό", ανταγωνισμός="ναί" και P01=63), οπότε

. Εισαγωγή των νέων χαρακτηριστικών:

τύπος="Α", ηλικία="μέσο", κόστος="μέσο",
κέρδος="υψηλό" ανταγωνισμός="ναί".

. Στο F2H="e:\dir1\DKB1" στη θέση P01=63 εγγράφονται οι νέες τιμές.

_ Τέλος διαδικασίας.

Η μορφή του "e:\dir1\DKB1" θα έχει τώρα ως εξής:

```
----  ----------------------------------------
Pos     DBK1 (f21)
----  ----------------------------------------
0       23      253
23   "A" "      νέο" "  υψηλό" "  υψηλό" "ναί" -1
63   "A" "    μέσο" "   μέσο" "  υψηλό" "ναί" -1
103  "A" "      νέο" "  υψηλό" "  μέτριο" "όχι"  -1
143  "A" "παλαιό" "  χαμηλό" "  χαμηλό" "όχι" -1
183  "A" "      νέο" "  χαμηλό" "  υψηλό" "ναί" -1
213  "A" "      νέο" "  χαμηλό" "  υψηλό" "ναί" -1
```

4.4. ΑΥΤΟΑΝΑΔΙΟΡΓΑΝΩΣΗ ΤΩΝ fli

Οταν το μέγεθος ενός αρχείου fli τείνει να γίνει μεγαλύτερο από την διατιθέμενη σε αυτό κύρια μνήμη (All≥ RAM_SIZE) το fli διασπάται σε δύο αρχεία, τα flil και fli2. Μία βασική παρατήρηση εδώ είναι ότι το όλο Σύστημα δεν υφίσταται καμία μεταβολή, αφού η μόνες μεταβολές που λαμβάνουν χώρα είναι η διάσπαση του fli και οι αντίστοιχες απαραίτητες μεταβολές μέσα

στο F, και namesKB όσον αφορά τους νέους Λ. Υπάρχουν οι δυνατότητες:

α) επέκτασης του fl_i στα fl_{i1} και fl_{i2} (το περιεχόμενο του fl_i ταυτίζεται με εκείνο του fl_{i1}) με χρήση του ίδιου κοινού piloti, και

β) η δυνατότητα διάσπασης fl_i σε fl_{i1} και fl_{i2} χρησιμοποιώντας αντίστοιχα τα piloti1 και piloti2 τα οποία αντικαθιστούν το piloti.

Στην (α) περίπτωση η μόνη μεταβολή στο σύστημα αφορά το fli, καθώς δεν υφίστανται μεταβολές ούτε οι Λ κλειδιών. Στην περίπτωση όμως που πρέπει να γίνει κάποια πράξη που αφορά το fli, η πράξη αυτή εφαρμόζεται αρχικά στο fli1 και σε περίπτωση αποτυχίας εφαρμόζεται στο fli2, με αποτέλεσμα κάποιες απώλειες χρόνου (ειδικά στην περίπτωση εκείνη που από το αρχικό fli έχει προκύψει μετά από διαδοχικές επεκτάσεις, μία ακολουθία fli1,....flin, n>2).

Στην (β) περίπτωση η μεταβολή αφορά μόνο τη διάσπαση του fli σε fli1, fli2, την αντικατάσταση του piloti από piloti1, piloti2, καθώς και την μεταβολή στο F, namesKB όσον αφορά τους Λ κλειδιών.

Ακολούθως δίνεται η διαδικασία αυτοαναδιοργάνωσης-διάσπασης για τα fli (περίπτωση β).

4.4.1 Η διαδικασία διάσπασης

_ Εστω το αρχείο f1H με All ≥ MAX_SIZE.

_ Από το namesKB λαμβάνονται τα H, PilotH, F2H:

f(H,F1H,PilotH,F2H).

_Προσπέλαση του αρχείου F για προσδιορισμό του διαστήματος d=[Fromk,Untilk] που αντιστοιχεί στο H:

h(Fromk,Untilk,H).

_Διαμερισμός του d σε d1, d2:

d1 ∪ d2 = d, d1 ∩ d2 = {}

_Στα PilotHi, i=1,2 τίθεται

Nexti=-1, Alli=0

_Στο αρχείο f1Hi, i=1,2

. Για κάθε j-εγγραφή p(Keyj,P0j,1,P01j)

. Αν Keyj ∈ d_k, k=1,2

. Στο f1Hk θέση P0j εγγράφεται

p(Keyj,P0j,1,P01j)

. Στο PilotHk All_newk=Allk+size_rec_f1

. Αν Nextk=-1 τότε

Pos=Allk

διαφορετικά Pos=Nextk

. Στο f1Hk θέση Pos εγγράφεται

p(Keyj,Pos,-1,P01j)

. Στο PilotHk Next_newk=P0j.

Παράδειγμα 4.1.α: Ας θεωρηθεί το fli με MAX_SIZE=240 :

Pos	fli ("d:\new\KB1")			
0	p(	100,	0, 1,	23)
40	p(	105,	40, 1,	63)
80	p(	120,	80, 1,	103)
120	p(	150,	120, 1,	143)
160	p(	170,	160, 1,	183)
200	p(	190,	200, 1,	213)

PILOT1

-1

240

Θα γίνει διάσπαση του fli σε fli1 και fli2:

_ Στο piloti1 εγγράφονται οι τιμές Next1=-1, All1=0
και στο piloti2 οι Next2=-1, All2=0.

_ Από το namesKB είναι:

f(1,"d:\new\KB1","c:\new\pilot1","e:\dir1\DKB1")
απ' όπου προκύπτει H=1.

_ Από το F είναι h(100,199,1).

_ Το διάστημα [100,199] διαμερίζεται σε
d1=[100,149], d2=[150,199].

_ Στο KB1 για j=1,...,6 εγγραφές θα είναι :

Για j=1: Key=100 ε d1.

Στο fli1 στη θέση P0=0 εγγράφεται η p(100,0,1,23).

Στο piloti1 εγγράφεται η All_new1=0+40=40 (Next1=-1).

Στο fli2 στη θέση 0 (P0=All2 αφού Next2=-1) εγγράφεται το p(100,0,-1,23).

Στο piloti2 εγγράφεται η Next_new2=0, All2=40.

Για j=2i: Key=105 ε d1.

Στο fli1 στη θέση P0=All1=40 εγγράφεται p(105,40,1,63).

Στο piloti1 εγγράφεται η All_new1=40+40=80 (Next1=-1).

Στο fli2 στη θέση 40 εγγράφεται το p(105,0,-1,63).

Στο piloti2 εγγράφεται η Next_new2=40, All2=80.

Για j=3: Key=120 ε d1.

Στο fli1 στη θέση P0=All=80 εγγράφεται p(120,80,1,103).

Στο piloti1 εγγράφεται η All_new1=80+40=120 (Next1=-1).

Στο fli2 στη θέση 40 εγγράφεται το p(120,0,-1,103).

Στο piloti2 εγγράφεται η Next_new2=80, All=120.

Για j=4: Key=150 ε d2.

Στο fli2 στη θέση P0=120 εγγράφεται το p(150,120,1,143).

Στο piloti2 εγγράφεται η All2=160 (Next2=80).

Στο fli1 στη θέση 120 εγγράφεται το p(150,-1,-1,143).

Στο piloti1 εγγράφεται η Next1=80, All1=160.

Για j=5: Key=170 ε d2.

Στο fli2 στη θέση P0=160 εγγράφεται p(170,160,1,183).

Στο piloti2 εγγράφεται η All2=200 (Next2=80).

Στο fli1 στη θέση 170 εγγράφεται το p(170,80,-1,143).

Στο pilotil εγγράφεται η Next1=160, All1=200.

Για j=6: Key=190 ε d2.

Στο fli2 στη θέση P0=200 εγγράφεται p(190,200,1,213).

Στο piloti2 εγγράφεται η All2=240 (Next2=80).

Στο flil στη θέση 200 εγγράφεται το p(190,160,-1,213).

Στο pilotil εγγράφεται η Next1=200, All1=240.

Στους πίνακες που ακολουθούν η εξέλιξή των flil, pilotil, fli2, piloti2 κατά τη διάσπαση έχει ως εξής:

Nextil	Allil	Nexti2	Alli2
-1	0	-1	0

j	flil	Next1	All1	fli2	Next2	All2
1	p(100,0,1,23)	-1	40	p(100,-1,1,23)	0	40
2	p(150,40,1,63)	-1	80	p(105,0,-1,63)	40	80
3	p(120,80,1,103)	-1	120	p(120,40,-1,103)	80	120
4	p(150,-1,-1,143)	80	160	p(150,120,1,143)	80	160
5	p(170,80,-1,183)	160	200	p(170,160,1,183)	80	200
6	p(190,160,-1,213)	200	240	p(190,200,1,213)	80	240

ΚΕΦΑΛΑΙΟ 5

ΔΙΑΧΕΙΡΙΣΗ ΓΝΩΣΗΣ ΣΕ ΕΦΑΡΜΟΓΕΣ ΠΡΑΓΜΑΤΙΚΟΥ ΧΡΟΝΟΥ ΚΑΙ ΥΠΟΣΤΗΡΙΞΗ ΜΕΓΑΛΩΝ ΒΑΣΕΩΝ ΓΝΩΣΗΣ

5.1. ΕΙΣΑΓΩΓΗ

Η Κατανεμημένη Τεχνητή Νοημοσύνη - ΚΤΝ (Distributed Artificial Intelligence - DAI) έχει προκαλέσει τα τελευταία χρόνια πολλές ερευνητικές προσπάθειες. Στόχος είναι η ανάπτυξη τεχνικών επίλυσης προβλημάτων Τεχνητής Νοημοσύνης κατανεμημένης μορφής. Σύμφωνα με τον Huns (1987) η ΚΤΝ ασχολείται με συνεργάσιμες λύσεις προβλημάτων Τεχνητής Νοημοσύνης από μία αποκεντρωμένη ομάδα διαδικασιών απλών ή σύνθετων, κατανεμημένων οντοτήτων με μερικές τοπικές και ανεξάρτητες δυνατότητες συλλογιστικής [Schalkoff,1990].

Εξελίξεις έχουν σημειωθεί στον τομέα ανάπτυξης τοπικών δικτύων υπολογιστών, συστημάτων πολυεπεξεργαστών και της ανάπτυξης μεθόδων προγραμματισμού που βασίζονται στην ανάλυση σε ανεξάρτητα τμήματα προγραμμάτων. Παράλληλα παρουσιάζεται ανάγκη υλοποίησης συστημάτων πραγματικού χρόνου (Real-Time-systems) [Laffey, 1988], [Schwelger, 1994], [Mohania, Sarda, 1994].

Οταν ένα πρόγραμμα είναι ένα σύνολο παράλληλων επεξεργασιών οι οποίες, επειδή είναι ανεξάρτητες μεταξύ τους, μπορούν να εκτελεστούν ταυτόχρονα, έχουμε τον όρο παράλληλη επεξεργασία (parallelism) [Raynal,1988]. Στο χώρο της Τεχνητής Νοημοσύνης, ο σχεδιασμός και προγραμματισμός μιάς παράλληλης μηχανής (parallel machine) είναι ακόμη ανοιχτό πρόβλημα (open problem) [Schalkoff, 1990].

Σε θέματα ταχύτητας επεξεργασίας και σε θέματα μεγέθους κύριας μνήμης έχουν σημειωθεί εκπληκτικές πρόοδοι. Στην Τεχνητή Νοημοσύνη όμως, υπάρχουν απαιτήσεις και σε θέματα πραγματικού χρόνου, όπου έχουν γίνει προσπάθειες να προσεγγιστούν, κυρίως υλοποιώντας ειδικές αρχιτεκτονικές Υλικού και Λογισμικού. Σήμερα είναι βεβαιότητα ότι η Τ.Ν. (και γενικότερα η Πέμπτη γενιά Πληροφορικής) βρίσκει εφαρμογή στους καθημερινούς χώρους εργασίας και μπορεί να εξυπηρετήσει σημαντικά επιστημονικά θέματα. Ετσι, είναι προφανής η ανάγκη εφαρμογής της Τ.Ν. σε μικροϋπολογιστές και ειδιαίτερα να προσεγγιστεί το θέμα της παράλληλης επεξεργασίας καθώς και του πραγματικού χρόνου σε επίπεδο τοπικών, ομότιμων δικτύων απλών μικροϋπολογιστών.

Τα προγράμματα Τεχνητής Νοημοσύνης που χρησιμοποιούν Βάσεις Γνώσης έχουν απαιτήσεις σε κύρια μνήμη (Random Access Memory - RAM). Επίσης η χρήση περιφεριακής μνήμης (disk

memory) για τη διαχείριση Γνώσης είναι αργή από άποψη χρόνου. Εξ άλλου οι μικροϋπολογιστές (PCs) και τα ομότιμα δίκτυα (peer to peer networks) περιορίζονται από πλευράς μεγέθους κύριας μνήμης αφού οι μεγάλες Βάσεις Γνώσης απαιτούν μεγάλα μεγέθη RAM.

Εκτός από το πρόβλημα περιορισμού της κύριας μνήμης, υπάρχει και το πρόβλημα του πώς είναι δυνατόν δύο Χρήστες στο δίκτυο την ίδια χρονική στιγμή, να γνωρίζουν την ίδια πληροφορία. Ετσι απορρέει το ακόλουθο ερώτημα:

"Δοθέντων δύο κόμβων A και B με το ίδιο τμήμα Γνώσης στη RAM, πώς είναι δυνατόν ο A να γνωρίζει κάθε αλλαγή που πραγματοποιείται από τον B, μέσα σε ένα χρονικό τμήμα Δt, όπου:

$\Delta t = t2 - t1$

t2: χρονική στιγμή που ο A δίνει μία απαίτηση

t1: χρονική στιγμή που ο B πραγματοποιεί αλλαγές".

Το παραπάνω ερώτημα εκφράζει το πρόβλημα του Πραγματικού Χρόνου σε επίπεδο Κύριας Μνήμης (RAM-Real-Time). Σε μερικές εφαρμογές το Δt είναι πολύ κοντά στο μηδέν ($\Delta t \rightarrow 0$) όπως Στρατιωτικές, Τραπεζικές εφαρμογές κ.λ.π.

Ο ορισμός αυτός του προβλήματος πραγματικού χρόνου κύριας μνήμης (RAM-Real-Time) [Panayiotopoulos J.-C., Papathanassiou E., 1994], ανταποκρίνεται περισσότερο στην έννοια Πραγματικού Χρόνου μεταξύ σταθμών δικτύου, από τους ήδη

αναφερόμενους στην βιβλιογραφία [Laffey, 1988]. Γενικά, το χαρακτηριστικό που καθορίζει ένα σύστημα πραγματικού χρόνου είναι η ικανότητα απόκρισης μετά την παρέλευση αυστηρά ορισμένου χρονικού διαστήματος, ίσως και σταθερού (με την έννοια του γρήγορου) [Laffey, 1988], [Chan, 1991]. Στην παρούσα περίπτωση, όπου έχουμε μικροϋπολογιστές που πρέπει να διαχειριστούν μεγάλες κατανεμημένες Βάσεις Γνώσης, ο χρόνος απόκρισης όχι μόνο δεν θεωρείται σταθερός, αλλά εξαρτάται από τον χρόνο απαίτησης του κάθε κόμβου του δικτύου.

Ακολούθως προτείνεται μία αυθεντική οργάνωση της Γνώσης, παρουσιάζεται η τοπολογία του απαιτουμένου δικτύου, δίνονται οι αλγόριθμοι οργάνωσης και η τεχνική που καλύπτει το πρόβλημα RAM-Real-Time. Χρησιμοποιούνται τεχνικές Λογικού Προγραμματισμού, όροι και δομές όπως της PROLOG για την αναπαράσταση της Γνώσης, και γνώσεις τοπικών δικτύων μικροϋπολογιστών.

Θα γίνει αναφορά σε δύο θέματα που αφορούν το Σύστημα που αναπτύχθηκε στα προηγούμενα κεφάλαια, τα οποία είναι:

1. Η άμεση ενημέρωση των αρχείων Hi,i=1,...,m στους δίσκους την στιγμή που υφίστανται κάποια μεταβολή σε επίπεδο RAM, με απότερο σκοπό (α) την εξασφάλιση του Συστήματος σε περιπτώσεις πτώσης του Υλικού και (β) την όσο το δυνατόν αμεσότερη ενημέρωση των δίσκων, σχεδόν ταυτόχρονα με την ενημέρωση σε επίπεδο RAM (συνεργαζόμενα/cooperative Συστήματα).

2. Η ενημέρωση μεταξύ σταθμών του δικτύου οι οποίοι έχουν στην κύρια μνήμη τους το ίδιο αρχείο Hi (βάση γνώσης, βάση γεγονότων ή/και κανόνων).

5.2. ΥΠΑΡΧΟΥΣΕΣ ΠΡΟΣΕΓΓΙΣΕΙΣ ΕΠΙ ΤΟΥ ΘΕΜΑΤΟΣ ΤΟΥ ΠΡΑΓΜΑΤΙΚΟΥ ΧΡΟΝΟΥ

Οι Moore, Hawkinson, Knickerbocker, Churchman (1984), παρουσιάζουν ένα Εμπειρο Σύστημα σε περιβάλλον Πραγματικού Χρόνου και παράλληλης επεξεργασίας, που υλοποιείται σε μία "LISP machine" προσαρμοσμένη σε ένα συμβατικό κατανεμημένο σύστημα ελέγχου, με δυνατότητα προσπέλασης σε μεγάλο αριθμό δεδομένων. Το πρόβλημα της επικοινωνίας αντιμετωπίζεται με μία LAMBDA machine με δύο επεξεργαστές παράλληλης αρχιτεκτονικής (Lisp processor για expert system, 68010 processor για real-time data access).

Οι Grammond και Miller (1984) προτείνουν μία αρχιτεκτονική πολυεπεξεργασίας που υλοποιείται με ένα πεπερασμένο σύνολο επεξεργαστών, ο καθένας από τους οποίους έχει πρόσβαση σε μία μνήμη κοινή για όλους (global). Η μνήμη αυτή χωρίζεται σε τρία τμήματα: τη Στατική Μνήμη (static memory) η οποία περιέχει τον μεταφρασμένο κώδικα του προγράμματος, τη Δυναμική Μνήμη (dynamic memory) που περιλαμβάνει τα ανόμοια περιβάλλοντα bindings (environement bindings) και τη μνήμη Επεξεργασίας

(process memory) που περιέχει πληροφορίες για κάθε επεξεργαστή.

Οι Ouri Wolfson και Aya Ozeri (1990) θεωρούν ένα σύνολο επεξεργαστών που έχουν πρόσβαση σε μία επεκτάσιμη Βάση Δεδομένων. Κάθε επεξεργαστής ασχολείται με ένα σύνολο κανόνων και προσθέτει τα παραγόμενα της επεξεργασίας σε μία κοινή Βάση Δεδομένων. Διακρίνονται δύο περιπτώσεις: α) η Βάση μπορεί να βρίσκεται σε μία κοινή μνήμη όπου θα έχουν πρόσβαση όλοι οι επεξεργαστές ή μπορεί να γίνει δεκτή από ένα επεξεργαστή που θα την "κατέχει" τοπικά και σφαιρική για τους υπόλοιπους και, β) υπάρχει μία τοπική βάση για κάθε επεξεργαστή.

5.3. ΠΡΟΣΕΓΓΙΣΗ ΣΤΟ ΘΕΜΑ ΤΟΥ ΠΡΑΓΜΑΤΙΚΟΥ ΧΡΟΝΟΥ ΓΙΑ ΤΟΠΙΚΑ ΟΜΟΤΙΜΑ ΔΙΚΤΥΑ ΜΙΚΡΟΫΠΟΛΟΓΙΣΤΩΝ ΜΕ ΜΕΓΑΛΕΣ ΒΑΣΕΙΣ ΓΝΩΣΗΣ ΚΑΤΑΝΕΜΗΜΕΝΗΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ

Το πρόβλημα που στη συνέχεια θα αντιμετωπιστεί είναι το θέμα το Πραγματικού Χρόνου σε επίπεδο RAM, στο περιβάλλον ενός Ομότιμου τοπικού δικτύου μικροϋπολογιστών τύπου Ethernet.

Απαιτούμενο Υλικό (Hardware): Το Υλικό που θα χρησιμοποιηθεί είναι Ν μικροϋπολογιστές συνδεδεμένοι σε τοπικό ομότιμο (peer-to-peer) δίκτυο τύπου Ethernet, κάτω από κοινό λειτουργικό σύστημα (π.χ.DOS). Περιφεριακή μνήμη (δίσκοι) υπάρχει

σε διάφορα σημεία του δικτύου, όχι απαραίτητα σε κάθε σταθμό, και υπάρχει δυνατότητα να χρησιμοποιείται σφαιρικά από περισσότερους του ενός σταθμούς. Κάθε σταθμός έχει έναν επεξεργαστή και μία κύρια μνήμη RAM. Ετσι σε κάθε επεξεργαστή αντιστοιχεί μία τοπική (local) RAM.

Απαιτούμενο Λογισμικό (Software): Το απαιτούμενο Λογισμικό είναι εκείνο του ETHERNET δικτύου σε κοινό περιβάλλον (π.χ.DOS). Επίσης γίνεται χρήση της PROLOG σαν γλώσσας Λογικού Προγραμματισμού για την υποστήριξη Βάσεων Γνώσης τύπου PROLOG.

Στην περιφεριακή μνήμη (δίσκους) του δικτύου υπάρχουν μεγάλες Βάσεις Γνώσης, των οποίων η προσπέλαση θα γίνεται σε επίπεδο κύριας μνήμης και δίσκων. Η προσέγγιση που θα αναπτυχθεί ακολούθως μπορεί να εφαρμοστεί γενικά και ειδικότερα ακόμη στις δομές όπως έχουν περιγραφεί στο 2ο κεφάλαιο, όπου σαν KB θεωρείται όλη η Βάση Γνώσης, της μορφής:

$$KB = H1 \cup H2 \cup ... \cup Hi \cup ... \cup Hk$$

όπου τα Hi είναι Λ γνώσης

Ας θεωρηθεί ότι STj, Pj, j=1,...,m είναι οι σταθμοί με τους αντίστοιχους επεξεργαστές που λαμβάνουν μέρος. Αν σε κάποια δεδομένη στιγμή t ισχύει:

Pj -> Hi(j) με i=1,2,...,k, j=1,2,...,m

k το πλήθος των Λ γνώσης Hi

σημαίνει ότι ο σταθμός STj, με τον Pj επεξεργαστή, έχει το Hi, δηλαδή έχει σε επίπεδο RAM το Hi(j) αντίγραφο του Hi. Ετσι ο προτασιακός τύπος του προβλήματος του Πραγματικού χρόνου σε επίπεδο RAM είναι ο ακόλουθος:

" Την χρονική στιγμή t:
Pi -> Hd(i)
Pj -> Hd(j) με i,j=1,...,k, i<>j, d=1,...,m.
Τυχούσα μεταβολή του Hd(i) μπορεί να υπάρξει
και στο Hd(j) μέσα στο δεδομένο χρονικό διάστημα Δt".

Το μέγεθος του Δt εξαρτάται από την εφαρμογή και μπορεί να είναι κοντά στο μηδέν σε στρατιωτικές ή και τραπεζικές εφαρμογές, ενδέχεται όμως να είναι και ένα ικανό διάστημα χρόνου, που περνά μέχρι ο Pj να απαιτήσει κάτι εκ νέου από τη βάση Hd, σε άλλες λιγότερο απαιτητικές εφαρμογές.

5.3.1 ΤΟ ΠΡΟΒΛΗΜΑ ΤΟΥ ΠΡΑΓΜΑΤΙΚΟΥ ΧΡΟΝΟΥ ΣΕ ΕΠΙΠΕΔΟ RAM - ΕΠΙΚΟΙΝΩΝΙΑ ΜΕΤΑΞΥ ΤΩΝ ΣΤΑΘΜΩΝ

Εστω την χρονική στιγμή t:
Pi ->Hd(i)
Pj ->Hd(j) με i,j=1,...,k, i<>j, d=1,...,m.
Εστω ότι ο σταθμός STi επιθυμεί να πραγματοποιήσει τη μεταβολή μ

στο Hd. Τότε:

_ ενημερώνει τη λίστα μεταβολών του Pj

_ πραγματοποιεί την μεταβολή μ στο Hi σε επίπεδο RAM

_Οταν ο Pj λάβει το μήνυμα μεταβολής από τη λίστα των μηνυμάτων του, καλεί το Hj εκ νέου στη RAM.

_Διαγράφει το μύνημα.

_Τέλος διαδικασίας.

5.3.2. ΤΟ ΠΡΟΒΛΗΜΑ ΤΟΥ ΠΡΑΓΜΑΤΙΚΟΥ ΧΡΟΝΟΥ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΓΝΩΣΙΟΛΟΓΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Το απαιτούμενο Ethernet δίκτυο διαθέτει δύο servers: server_1 (P1), με προτεραιότητα εκκίνησης, και server_2 (P2), με προαπαιτούμενο την εκκίνηση του server_1.

Στον Server_1 βρίσκονται :

_ τα αρχεία H1,H2,...,Hn επί των οποίων τα δικαιώματα όλων των άλλων σταθμών είναι ανάγνωσης και εγγραφής (Read/Write [R/W]) ενώ ο Server_1 έχει ανάγνωσης, εγγραφής, δημιουργίας (Read/ Write/ Create [RWC]) δικαιώματα.

Στον Server_2 υπάρχουν:

_ το F, αρχείο-Βάση Γνώσης περιεχομένων της μορφής:

f(Fromi, Untili, NameHi)

Stations(List_of_nodes_On)

List_of_nodes_on : είναι μία λίστα τύπου Prolog που περιέχει

τους κόμβους που βρίσκονται εν ενεργία στο LAN.

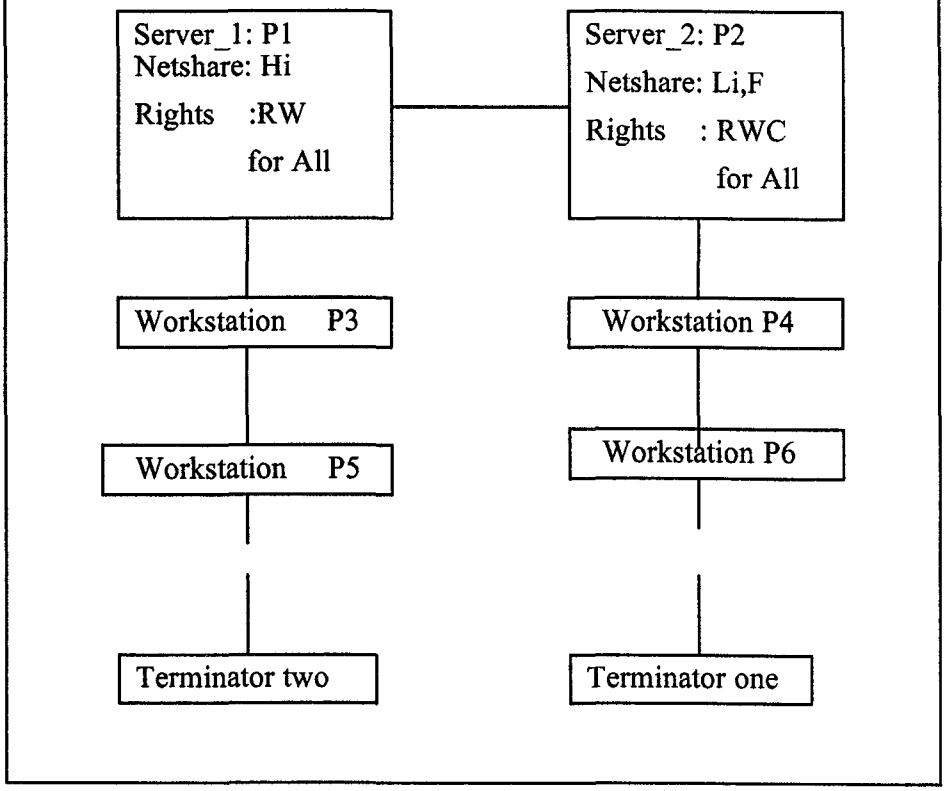
Fromi (Untili) : δίνει το minimum (maximum) για τις λογικές εγγραφές στο Hi, που αντιστοιχούν σε ένα πρωτεύον κλειδί.

NameHi : πλήρες όνομα (full-path file-name) του Λ Hi.

_ τα αρχεία L1,L2,...,Lk ημερολόγια ενημέρωσης των H1,H2,...,Hk αντίστοιχα. Τα δικαιώματα των άλλων σταθμών επί των Li είναι [RWC] .

Η Τοπολογία του Ethernet δικτύου απεικονίζεται στο σχήμα που παρατίθεται.

ΣΧΗΜΑ : Τοπολογία Δικτύου και Γνώσης με δύο Servers



Terminator one, two: Περατωτές Ethernet

F : αρχείο Βάσης Γνώσης περιεχομένων

Hi:αρχεία Βάσεων Γνώσης - Λ

Li:αρχείο ενημέρωσης Λ

Workstation Pj: σταθμοί εργασίας του Γνωσιολογικού Πληροφοριακού Συστήματος.

Βασικοί όροι για τη λειτουργία του συστήματος είναι:

(1) Κάθε σταθμός του δικτύου, καθώς και ο Server_2, κατά την εκκίνησή του παίρνει ημερομηνία και ώρα από τον Server_1 (clock

sychronized).

(2) Σε κάθε μεταβολή ενημερώνονται τα αντίστοιχα αρχεία Hi, Li.

Αρχικές Διαδικασίες

_ Αρχικά

_ Δημιουργείται το Knowledge path στο server_1 (P1 κόμβος) όπου εγγράφονται οι Λ Hi.

_ Αν υπάρχει ο server_2 (P2 κόμβος) δημιουργεί ένα κοινό working path (διαφορετικά αυτό δημιουργείται στο P1).

_ Εκκινώντας το Δίκτυο

_ Γίνεται εκκίνηση του P1 ο οποίος "δίνει" το knowledge path (τα Hi) με δικαιώματα Read/Write (RW) για όλους.

_ Ο P1 συγχρονίζει το δίκτυο δίνοντας Date και Time. Είναι βασική προϋπόθεση να υπάρχει ίδιος χρόνος (Date/Time) σε όλους τους κόμβους του δικτύου.

_ Ο P2 εκκινεί και "δίνει" το working path με δικαιώματα Read/Write/Create (RWC) για κάθε κόμβο.

_ Ο P2 συγχρονίζεται με Date/Time του δικτύου.

_ Ο P1 συνδέεται με το working path του P2.

_ Ο P1 παράγει τη Βάση Γνώσης F με την αρχική μορφή:

stations([1,2]).

_ Τίθεται σε λειτουργία ο τυχών κόμβος STj. Κάθε κόμβος συνδέεται με το knowledge path και working path P1 και P2.

_ Κάθε Pj, j=3,4,...,m συγχρονίζεται με την ημερομηνία και ώρα του δικτύου (clock synchronized) και προστίθεται στη λίστα των εν ενεργία κόμβων List_of_nodes_On το j.

_ Κάθε Pj, j=1,2,...,m δημιουργεί σε επίπεδο RAM τη βάση γνώσης:

me(Date, Time, 0).

Λειτουργικές Διαδικασίες

_ Απαίτηση (query)

_ Εστω ότι ο Pj πρόκειται να ικανοποιήσει μία απαίτηση επί της KB. Ο Pj συμβουλεύεται την F: consult(F), για να βρεί Hi σύμφωνα με το nameHi.

_ Ο Pj συμβουλεύεται τη βάση γνώσης Li: consult(Li).

_ Αν me(Datek,Timek,k), k<>i, τότε

consult(Hi),

retract(me(_,_,_)),

assert(me(Date, Time, i)),

και για κάθε p=j:

retract(updates(p,_,_,_)),

save(Li).

_ Αν me(Datek,Timek,k), k=i τότε:

για κάθε λογική εγγραφή

updates(j, Dateμ, Timeμ, μ) (αν υπάρχει) με

Datek/Timek < Dateμ/Timeμ

στο Hi προστίθεται το μ.

retract(updates(j,_,_,_)),

save(Li),

retract(me(_,_,_)),

assert(me(Date, Time, i)).

_ Διαδικασία update

_ Εστω ότι ο Pj πρόκειται να πραγματοποιήσει την μεταβολή μ στην KB.

_ Η μεταβολή μ πραγματοποιείται σε επίπεδο RAM.

_ Η μεταβολή μ πραγματοποιείται στο αρχείο Hi στο δίσκο.

_ Για κάθε p ε List_of_nodes_On, p<>j :

assert(updates(p,Date,Time,μ)).

_ save(Li).

_ Διαδικασία τερματισμού

_ Εστω ότι ο Pj πρόκειται να διακόψει τη σύνδεσή του με το

δίκτυο ή να τερματίσει την τρέχουσα εφαρμογή.

_ Διαγράφεται από τη λίστα των εν ενεργεία κόμβων του δικτύου και

retract(stations(_)),

assert(stations(New_LIST_of_nodes)),

save(F).

_ Για κάθε Li με p=j εκτελούνται τα ακόλουθα:

retract(updates(p(_,_,_))

και save(Li).

_ Αποσύνδεση από το δίκτυο ή τερματισμός της εφαρμογής.

5.3.3. ΓΕΝΙΚΕΥΣΗ ΓΙΑ ΤΑ ΑΡΧΕΙΑ Hi

Η ενημέρωση μόνο από τις κρατημένες αλλαγές που ήδη έχουν γίνει παρουσιάζει το πλεονέκτημα της ταχύτητας μόνο στην περίπτωση που ο χρόνος ολικού φορτώματος είναι μεγαλύτερος από εκείνον της επιμέρους ενημέρωσης από τις κρατημένες αλλαγές. Και Αυτό συμβαίνει όταν το πλήθος των αλλαγών είναι μικρότερο ή ίσο από το πλήθος των λογικών εγγραφών στο Hi.

Π Α Ρ Α Δ Ε Ι Γ Μ Α :

Ας θεωρήσουμε ένα σύστημα με Server_1, Server_2, και m σταθμούς, που περιέχει το σύνολο των αρχείων Βάσεων Γνώσης - Λ H1, H2, ..., Hk στον Server_1, με δικαιώματα RW για όλους τους σταθμούς, τα αντίστοιχα αρχεία ημερολόγια L1, L2, ..., Ln, με

δικαιώματα RWC για τον Server_1 και RW για τους σταθμούς, στον Server_2.

Επίπεδο 1

Ας θεωρήσουμε ότι οι σταθμοί j,k,l,m ε List_of_nodes_On=[j,k,l,m,q] και έχουν σε επίπεδο RAM αντίστοιχα αντίγραφα του Hi (π.χ. λόγω μίας απλής απαίτησης INQ), με χρόνους loading αντίστοιχα:
me(0, 0:0:1:0, i), me(0, 0:0:2:0, i),
me(0, 0:0:3:0, i), me(0, 0:0:4:0, i).

Η μορφή του αρχείου Li έχει αρχικά ως εξής:
Li:{}

Επίπεδο 2

Εστω ότι ο σταθμός j πρόκειται να πραγματοποιήσει τις μεταβολές m1,m2 που αφορούν τα κλειδιά key1, key2, αντίστοιχα.

_ Διαβάζεται το F και βρίσκεται το Hi.

_ O Pj consult(Li)

_ me(0,0:0:1:0,i), δηλ. Hi είναι στη RAM. Τότε:

Δεν υπάρχει εγγραφή updates(j, Datem,Timem,μ)

άρα δεν υπάρχει τίποτα για ενημέρωση του j.

_ m1, m2 εκτελούνται στη RAM.

_ m1,m2 εκτελούνται στο δίσκο στο Hi.

_ Για κάθε p ε List_of_nodes_On, p<>j εκτελείται:
assert(updates(p,0,0:0:5:1,m1)
assert(updates(p,0,0:0:5:2,m2)

Η μορφή του αρχείου Li έχει ως εξής:

updates(k,0,0:0:5:1,m1)
updates(l,0,0:0:5:1,m1)
updates(m,0,0:0:5:1,m1)
updates(q,0,0:0:5:1,m1)
updates(k,0,0:0:5,2,m2)
updates(l,0,0:0:5,2,m2)
updates(m,0,0:0:5,2,m2)
updates(q,0,0:0:5,2,m2)

Επίπεδο 3

Εστω ότι ο κόμβος k πρόκειται να πραγματοποιήσει την μεταβολή mk που αφορά το κλειδί keyk.

_ Διαβάζεται το F και βρίσκεται το Hi.

_ consult(Li)

_ me(0,0:0:2:0,i), άρα Hi στη RAM. Τότε για

updates(k,0,0:0:5:1,m1) updates(k,0,0:0:5:2,m2)

με 0:0:2:0<0:0:5:1<0:0:5:2

_ Οι μεταβολές m1,m2 προστίθενται στο Hi στη RAM.

_ retract(updates(k,_,_,_)),

_ assert(me(0,1:0:0:0,i)).

_ Η μεταβολή mk πραγματοποιείται στο Hi στη RAM.

_ Η mk πραγματοποιείται στο Hi στο δίσκο.

_ Για κάθε p ε List_of_nodes_On, p<>k εκτελείται:

assert(updates(p,0,0:6:0:1,mk)

Η μορφή του αρχείου Li έχει τώρα ως εξής: updates(l,0,0:0:5:1,m1)

updates(m,0,0:0:5:1,m1)
updates(q,0,0:0:5:1,m1)
updates(l,0,0:0:5,2,m2)
updates(m,0,0:0:5,2,m2)
updates(q,0,0:0:5,2,m2)
updates(j,0,0:6:0:1,mk)
updates(l,0,0:6:0:1,mk)
updates(m,0,0:6:0:1,mk)
updates(q,0,0:6:0:1,mk)

Επίπεδο 4

Εστω ότι ο σταθμός q πρόκειται να πραγματοποιήσει την μεταβολή mq που αφορά το κλειδί keyq.

_ Διαβάζεται το F και βρίσκεται το Hi.

_ consult(Li)

_ me(0,0:0:3:0,v), v<>i. Τότε

consult(Hi),

retract(me(_,_,_)), assert(me(0,5:0:0:0,i)

Για κάθε p=q:

retract(updates(p,_,_,_))

save(Li).

_ Η μεταβολή mq πραγματοποιείται στο Hi στη RAM.

_ Η mq πραγματοποιείται στο Hi στο δίσκο.

_ Για κάθε p ε List_of_nodes_On, p<>k εκτελείται:

assert(updates(p,0,6:0:1:0,mk)

Η μορφή του αρχείου Li έχει τώρα ως εξής:

updates(l,0,0:0:5:1,m1)

updates(m,0,0:0:5:1,m1)

updates(l,0,0:0:5,2,m2)

updates(m,0,0:0:5,2,m2)

updates(j,0,0:6:0:1,mk)

updates(l,0,0:6:0:1,mk)

updates(m,0,0:6:0:1,mk

updates(j,0,6:0:1:0,mq)

updates(k,0,6:0:1:0,mq)

updates(l,0,6:0:1:0,mq)

updates(m,0,6:0:1:0,mq)

5.4. ΣΥΓΚΡΙΣΗ ΧΡΟΝΩΝ ΕΝΗΜΕΡΩΣΗΣ ΤΩΝ ΣΤΑΘΜΩΝ

Οι ακόλουθες μετρήσεις έγιναν με χρήση των εξής δομών για Hi, me:

Hi της μορφής p(Key,P0,P1,P01) που έχει αναλυθεί στο δεύτερο κεφάλαιο (συνολικό μήκος 50 χαρακτήρες) με:

Key :string (30 χαρακτήρες)

P0, P01 :integer (10 χαρακτήρες)

P1 :integer (2 χαρακτήρες)

Li της μορφής updates(ST,D,T,p(Key,P0,P1,P01)) όπως έχει πιό πάνω περιγραφεί (μήκος 86 χαρακτήρες), με:

ST :integer (3 χαρακτήρες)

D,T:string (10 και 11 χαρακτήρες αντίστοιχα).

Για τις μετρήσεις χρησιμοποιήθηκε συμβατό PC/386/DX στα 40 MHz με 4 Mbytes RAM. Οι χρόνοι είναι οι μέγιστοι μεταξύ επαναλαμβανόμενων μετρήσεων.

	Hi	Li			
εγγραφές	500	500	400	300	250
bytes	25500	43500	34800	26100	21750
sec	0:38	0:38	0:32	0:21	0:21

	Hi	Li			
Εγγραφές	1000	1000	800	600	500
bytes	51000	87000	69699	52200	43500
sec	0:93	0:82	0:65	0:49	0:38

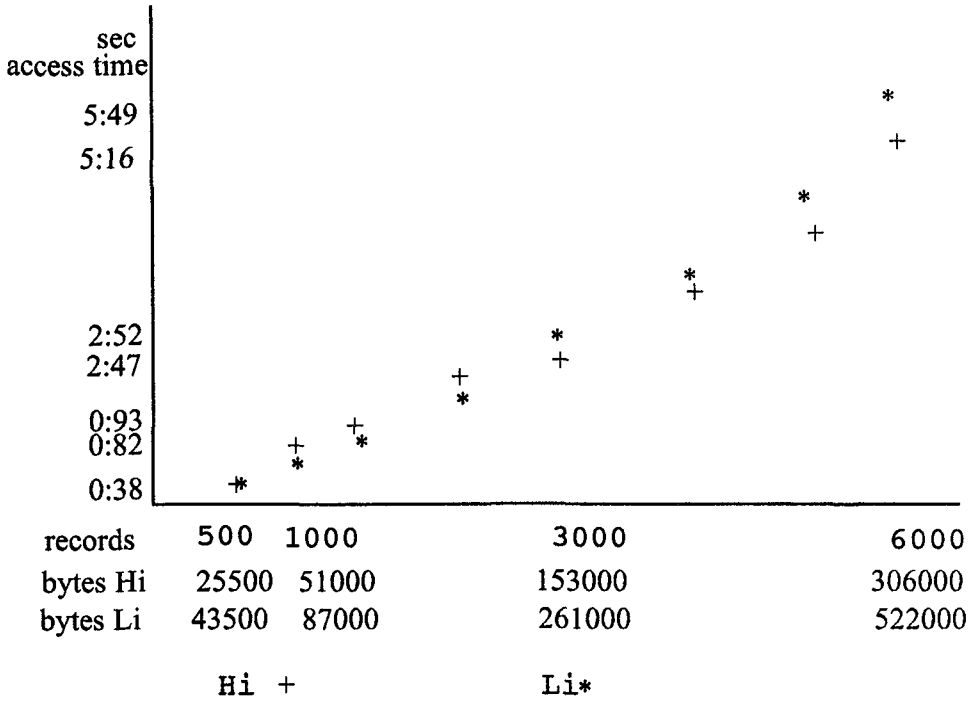
	Hi	Li			
Εγγραφές	3000	3000	2500	2000	1500
bytes	153000	261000	217500	174000	130500
sec	2:47	2:52	2:3	1:64	1:20

	Hi	Li			
Εγγραφές	4000	4000	3000	2500	2000
bytes	204000	348000	261000	217500	174000
sec	3:29	3:35	2:52	2:3	1:64

	Hi	Li			
Εγγραφές	5000	5000	2500		
bytes	255000	435000	217000		
sec	4:11	4:17	2:3		

	Hi	Li			
Εγγραφές	6000	6000	5000	4000	3000
bytes	306000	522000	435000	348000	261000
sec	5:16	5:49	4:17	3:35	2:52

Από τις μετρήσεις χρόνου γίνεται αντιληπτό ότι η ενημέρωση από τις μεταβολές είναι ταχύτερη όταν οι μεταβολές αφορούν το ήμισυ και λιγότερο των εγγραφών του Hi. Σε άλλη περίπτωση η ενημέρωση γίνεται απ' ευθείας από το Hi (consult(Hi)). Ακολουθεί το διάγραμμα:



ΠΑΡΑΤΗΡΗΣΕΙΣ

Από τους παραπάνω συγκριτικούς πίνακες προκύπτει σε ποιές περιπτώσεις είναι προτιμότερο να συμβουλεύεται ο σταθμός όλο το Λ, Li και σε ποιές συμφέρει να συμβουλεύεται μόνο τις μεταβολές που έχουν γίνει.

Για μικρούς Λ συμφέρει η ενημέρωση από όλο το Λ.

Για πολύ μικρό μήκος της εγγραφής του Li και για λίγες εγγραφές είναι ταχύτερη η ενημέρωση από τον Λ.

Για πολύ μικρό μήκος της εγγραφής του Li και για πλήθος μεταβολών αρκετά μικρότερο του ημίσεως των λογικών εγγραφών του Λ ταχύτερη είναι η ενημέρωση από τις μεταβολές.

Οσο μεγαλύτερο είναι το μέγεθος της λογικής εγγραφής του

Li παρατηρείται ότι ο χρόνος είναι σχεδόν ίδιος αν έχουμε μεταβολή σε όλες τις εγγραφές. Και σε αυτή την περίπτωση η ενημέρωση από τις αλλαγές είναι αποδοτικότερη για μεταβολές αρκετά λιγότερες από τις μισές εγγραφές.

Ετσι, γενικά, αν το πλήθος των μεταβολών είναι αρκετά λιγότερο από το ήμισυ του πλήθους εγγραφών του Li, φαίνεται πως είναι συμφερότερο να γίνεται ενημέρωση από τις μεταβολές, ενώ σε κάθε άλλη περίπτωση είναι καλύτερα η ενημέρωση να γίνεται από όλο τον Λ.

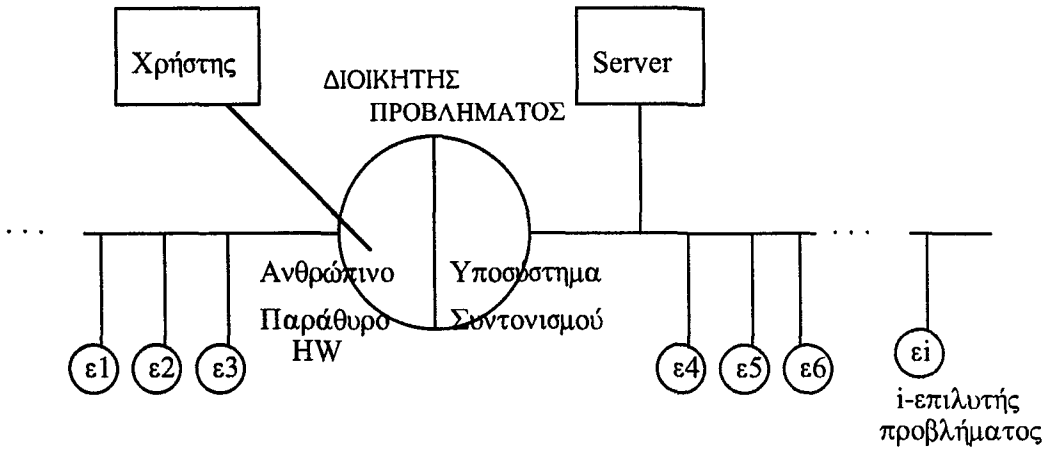
5.5 ΚΑΤΑΝΕΜΗΜΕΝΗ ΔΙΑΧΕΙΡΙΣΗ ΜΕΓΑΛΩΝ ΒΑΣΕΩΝ ΓΝΩΣΗΣ

Στα προηγούμενα ο κύριος στόχος ήταν η ενημέρωση πολλών Χρηστών σε Πραγματικό Χρόνο, πράγμα το οποίο έχει αξία στην περίπτωση εκμετάλλευσης του ίδιου τμήματος γνώσης ταυτόχρονα από δύο τουλάχιστον Χρήστες. Ωστόσο το πρόβλημα της αντιμετώπισης μεγάλων Βάσεων Γνώσης (ΒΓ) παραμένει και σε επίπεδο μεγάλων υπολογιστών και εντονότερα σε περιπτώσεις δικτύων μικροϋπολογιστών.

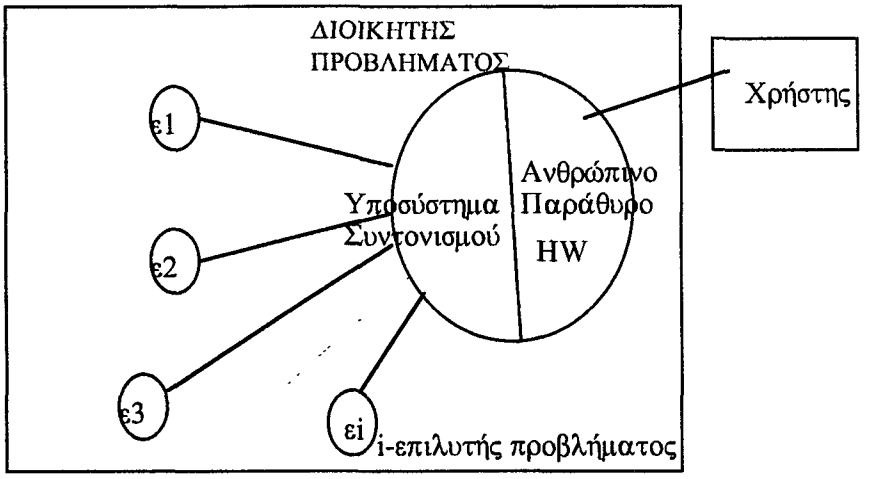
Στη συνέχεια ο κύριος στόχος θα επικεντρωθεί στην αντιμετώπιση μεγάλων ΒΓ με τη μέθοδο του σύνδρομου προγραμματισμού (concurrent programming). Συγκεκριμένα θα τηρηθεί η οργάνωση αντιμετώπισης του προβλήματος στο σχήμα που

παρατίθεται, όπου ένα πρόγραμμα (ή και επεξεργαστής) εκτελεί χρέη Διοικητού προβλήματος (problem manager), ενώ ένα σύνολο άλλων προγραμμάτων (άλλων επεξεργαστών) εκτελούν χρέη Επιλυτών του προβλήματος.

Με τον τρόπο αυτό μία τεράστια ΒΓ μπορεί να τμηματοποιηθεί στη διαχείρισή της από τους επιλυτές του προβλήματος, ενώ ο διοικητής του προβλήματος (Δ) συντονίζει την όλη διαδικασία (υποσύστημα συντονισμού) και διαχειρίζεται βέβαια το ανθρώπινο παράθυρο με το Χρήστη (Human Window - HW).



Οργάνωση αντιμετώπισης Μεγάλων Βάσεων Γνώσης
Περίπτωση: Δίκτυα Ethernet μικροϋπολογιστών



Οργάνωση Αντιμετώπισης Μεγάλων Βάσεων Γνώσης
Περίπτωση: Mainframe

5.6 ΔΟΜΕΣ

Το πρόβλημα Διαχείρισης Μεγάλης Βάσης Γνώσης σε RAM έχει απαίτηση σε μνήμη εκθετικής τάξης μεγέθους. Διότι για κάθε επεξεργαστή υπάρχει πάντοτε ένα μέγεθος k bytes, που για κάθε ΒΓ Δλ, μεγέθους mλ bytes, να ισχύει mλ>k τελικώς για κάθε Δλ.
Εστω το διατεταγμένο σύνολο των διαθεσίμων επεξεργαστών (προγραμμάτων):

$$E=\{\varepsilon_1,...,\varepsilon_i,...,\varepsilon_k,\varepsilon_{k+1},...,\varepsilon_q\},$$

όπου η ταχύτητα του ε_ρ είναι μεγαλύτερη της ταχύτητας του ε_θ, για κάθε $\rho<\theta$, όπου $\rho,\theta=1,2...,k$.

Ορίζεται η απεικόνιση (N^+ σύνολο θετικών ακεραίων):

$$R:E\rightarrow R(\varepsilon_i)\in N^+$$

όπου:

$R(\varepsilon_i)$ διαθέσιμη RAM του ε_i για εκτέλεση προγραμμάτων.

Ας υποθέσουμε ότι σε κάποια διαθέσιμη τοπολογία Υλικού υπάρχουν k επιλυτές. Είναι φανερό ότι κάθε ΒΓ μπορεί να τμηματοποιηθεί, είτε με βάση κάποιους κανόνες τμηματοποίησης, είτε αυθαίρετα, έτσι ώστε να προκύψουν k τμήματα αυτής για τους αντίστοιχους k επιλυτές, αλλά και τα υπόλοιπα τμήματά της k+1,...,w, διαθέσιμα σε επίπεδο περιφερειακής μνήμης. Επομένως, εάν υπάρχει έστω και ένας ακόμα επιλυτής δηλ. ο ε_{k+1} ή και περισσότεροι ακόμη $\varepsilon_{k+1},...,\varepsilon_q$, οι επιλυτές αυτοί είναι δυνατόν να επεξεργάζονται τα υπόλοιπα αυτά τμήματα γνώσης με τις τεχνικές που αναπτύχθηκαν στα προηγούμενα κεφάλαια, αρκεί βέβαια το μέγεθος των υπολοίπων τμημάτων Γνώσης να μην ξεπερνούν τις κατώτερες δυνατότητες σε κύρια μνήμη των υπολοίπων επιλυτών. Αρα ισχύει η ακόλουθη πρόταση:

Πρόταση:

Για κάθε Βάση Γνώσης Δλ, υπάρχει μία Διαμέριση σε Υποβάσεις:

$$\Delta_\lambda = \{\delta_1^\lambda, \delta_2^\lambda, ..., \delta_i^\lambda, ..., \delta_k^\lambda, ..., \delta_w^\lambda\}$$

έτσι ώστε:

$$m_i^\lambda \le R(\varepsilon_i),\ (\forall i = 1,...k)$$

$$\max_{k<\theta\le w}(m_\theta^\lambda) \le \min_{k<\rho<q}(R(\varepsilon_\rho))$$

Κανόνας 1: Οι συχνότερες σε προσπέλαση (από στατιστικά στοιχεία)

εγγραφές σωρεύονται στις διαμερίσεις μνήμης $\delta_1^\lambda, \delta_2^\lambda, ..., \delta_k^\lambda$, ενώ οι υπόλοιπες στις διαμερίσεις $\delta_{k+1}^\lambda, ..., \delta_w^\lambda$. Οι διαμερίσεις δ_φ^λ, φ>k συμφέρει να περιέχουν γνώση ιστορικής σημασίας με ελάχιστο βαθμό συχνότητας προσπέλασης. Εδώ βέβαια θεωρείται πως οι δεδομένες συχνότητες των εγγραφών είναι σε σχέση με κάποια βασική παράμετρο (κλειδί).

Ο κανόνας προϋποθέτει βέβαια την ύπαρξη στατιστικού υποσυστήματος της Βάσης, το οποίο μπορεί να ταυτίζεται με τα κλασσικά υπάρχοντα 4ης γενιάς ή και να ενσωματώνεται σε κάθε λογική εγγραφή με τη μορφή γνωσιολογικής παραμέτρου (παραμέτρων).

Κανόνας 2: Στην τοπολογική οργάνωση που προτάθηκε υπάρχει άριστη λύση όταν το ακόλουθο άθροισμα είναι το ελάχιστο δυνατόν:

$$\sum_{i=1,...,k} (R(\varepsilon_i) - m_i)$$

Επιλυτές

Ορίζεται το σύνολο επιλυτών (προγραμμάτων):

$$E=\{\varepsilon_i \mid i=1,...,k\}$$

όπου k καθορίζεται σε κάθε συγκεκριμένη περίπτωση από τα όρια της διαθέσιμης μνήμης σε mainframe ή από το πλήθος των διαθέσιμων Η/Υ στο δίκτυο, ή από το πλήθος των επεξεργαστών σε περίπτωση

παράλληλων μηχανών. Στην ουσία τα ε_i είναι εκτελέσιμα προγράμματα όχι κατ΄ανάγκη ομοιογενούς περιβάλλοντος (επιτρέπεται ένα υποσύνολο αυτών να είναι κάτω από MS Windows, άλλα κάτω από Unix κ.λ.π.).

Βάση Γνώσης

Ορίζεται η τμηματοποιημένη Βάση Γνώσης Δλ:

$$\Delta_\lambda = \{\delta_1^\lambda, \delta_2^\lambda, ..., \delta_i^\lambda, ..., \delta_k^\lambda, ..., \delta_w^\lambda\}$$

όπου w το πλήθος των διαμερίσεων Δ_i^λ της Δ_λ.

Ισχύουν επίσης οι προϋποθέσεις περιορισμού μνήμης, όπως ακριβώς έχει αναλυθεί στον ορισμό Τμημάτων Βάσης Γνώσης περιορισμένου μεγέθους μνήμης (Λ), για τα τμήματα γνώσης δ_ρ, $\rho>k$. Τέλος, καλό είναι να τηρούνται οι κανόνες 1,2 που αναλύθηκαν.

Διοικητής του Προβλήματος

Ο Διοικητής του Προβλήματος (Δ) είναι ένα πρόγραμμα (επεξεργαστής) που περιέχει δύο βασικά τμήματα (βλ. σχήμα σελ.121): το Υποσύστημα Συντονισμού (ΥΣ) των διαδικασιών επίλυσης του προβλήματος και το Ανθρώπινο Παράθυρο (Human Window - HW) δηλαδή το Υποσύστημα επικοινωνίας με τον Χρήστη.

IndexKnow

Το indexKnow είναι αρχείο ενημέρωσης του Δ, με την ακόλουθη δομή:

ε_i δ_i I_i O_i

όπου:

ε_i :ο επιλυτής,

δ_i :το αντίστοιχο τμήμα ΒΓ,

I_i :αρχείο επικοινωνίας με τον Δ (είσοδος μηνυμάτων από τον Δ)

O_i :αρχείο επικοινωνίας με τον Δ (έξοδος μηνυμάτων προς τον Δ)

Αρχεία επικοινωνίας μεταξύ Επιλυτών και Διοικητού

I_i: εξυπηρετεί την επικοινωνία μεταξύ Δ και ε_i, όπου ο Δ στέλνει τις διαταγές. Εδώ ο Δ εγγράφει και ο ε_i διαβάζει. Η δομή του είναι:

$flag_i$

$query_i$

με:

$flag_i$ ένδειξη με τιμές 0 αν δεν υπάρχει μήνυμα, 1 αν υπάρχει και αφορά ενημέρωση, 2 αν υπάρχει και αφορά αναζήτηση.

$query_i$ η απαίτηση του Δ από τον ε_i, η οποία είναι αναζήτηση αν $flag_i$=1, ενημέρωση (update) αν $flag_i$=2.

O_i: χρησιμοποιείται για την επικοινωνία μεταξύ Δ και ε_i. Εδώ ο ε_i γράφει και ο Δ διαβάζει. Η δομή του είναι:

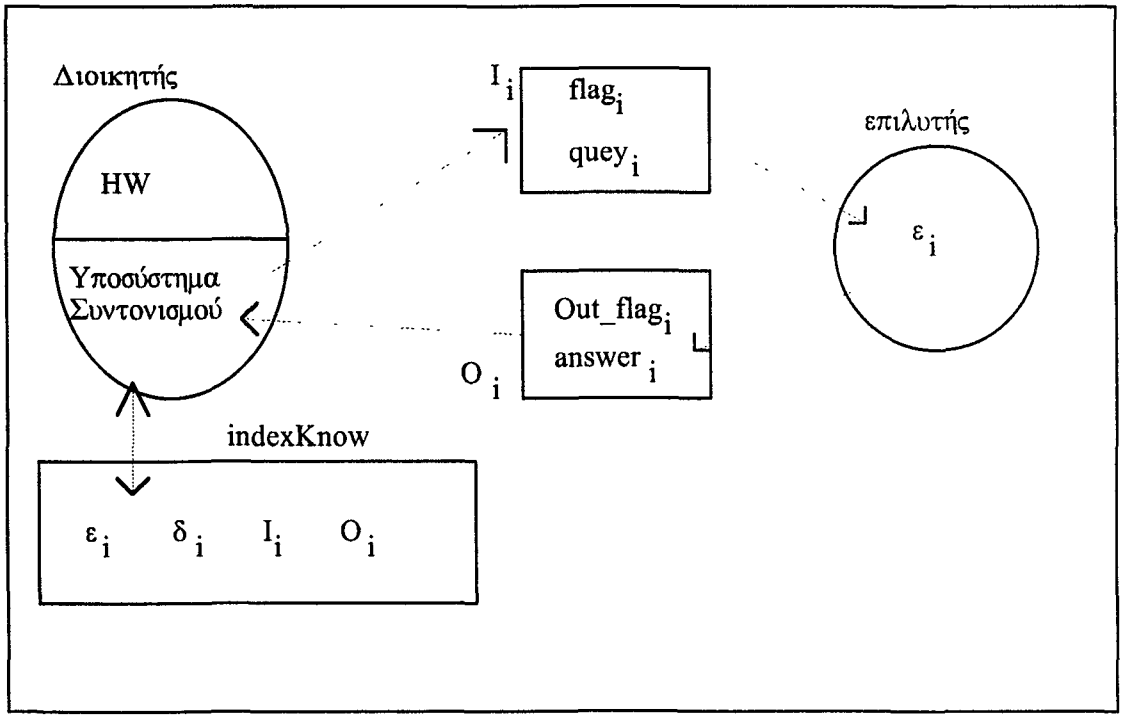
Out_flag$_i$

answer$_i$

με:

Out_flag$_i$ ένδειξη με τιμές, 0 αν ο ε_i δεν έχει ικανοποιήσει την απαίτηση ο Δ , 1 αν έγινε update, 2 αν έγινε η αναζήτηση.

answer$_i$ η απάντηση του ε_i προς τον Δ, η οποία ήταν αναζήτηση (inquery) αν flag=1, ενημέρωση (update) αν flag=2.



Κίνηση δεδομένων μεταξύ Διοικητού και επιλυτών

Εκκίνηση διαδικασίας επίλυσης

Κατά την εκκίνηση του συστήματος πρώτος εκκινά ο Δ και κατόπιν οι ei.

• Ξεκινάει ο Δ ο οποίος ενημερώνεται από το indexKnow

• Για i=1,...,k

Ξεκινάει ο ε_i

Στο I_i τίθεται $flag_i$=0 και στο O_i Out_flag_i=0

Λειτουργικές Διαδικασίες

_ Απαίτηση (query)

Εκκινώντας ο Δ, φορτώνει το ευρετήριο βάση γνώσης indexKnow από την οποία πληροφορείται με ποιό κομμάτι Γνώσης Λ (δ_i) απασχολείται κάθε ε_i, σε σχέση πάντοτε με μία βασική γνωσιολογική παράμετρο ή ένα σύνολο κανόνων.

Το αρχείο indexknow περιέχει την αντιστοιχία μεταξύ ε_i, δ_i, i=1....,k, καθώς και τα πλήρη ονόματα των αρχείων που χρησημοποιεί κάθε ei όπως είναι στο δίκτυο:

ε_i δ_i I_i O_i.

Στη συνέχεια ακολουθεί η διαδικασία:

• Ο Δ λαμβάνει μία απαίτηση από το HW

• Προσδιορίζει τα τμήματα Βάσης Γνώσης δ_i από τα οποία πρέπει να

ληφθεί η απάντηση. Από το indexKnow βρίσκει τα ε_i που είναι αρμόδια για την επίλυση (αντίστοιχα των δ_i).

- Ο Δ ενημερώνει τα αντίστοιχα I_i με $flag_i=2$ και με την απαίτηση που αφορά το κάθε ε_i.
- Κάθε ε_i που εμπλέκεται στην επίλυση ενημερώνεται από το I_i. Αν υπάρχει ερώτηση (αν flag<>0) λαμβάνει την διαταγή, δίνει στο I_i τις αρχικές τιμές $flag_i=0$, query={}.
- Κάθε ε_i εκτελεί τη διαταγή.
- Κάθε ε_i καταγράφει την απάντηση στο O_i, θέτοντας $Out\_flag_i=2$.
- Ο Δ ενημερώνεται διαβάζοντας το περιεχόμενο του O_i. Αν $Out\_flag_i<>0$, λαμβάνει την απάντηση έστω $answer_i$. Θέτει $Out\_flag_i=0$.
- Κάθε ε_i ενημερώνεται αν $Out\_flag_i=0$, στην οποία περίπτωση η διαδικασία τελειώνει.
- Ο Δ συνθέτει την τελική απάντηση από τις $answer_i$ και τη στέλνει στο HW.

_Update

Ο Δ παίρνει μία αναφορά και τη στέλνει για update στον αντίστοιχο επιλυτή (ή στους αντίστοιχους επιλυτές). Ακολουθείται η εξής διαδικασία:

- Ο Δ λαμβάνει την αναφορά από το HW
- Από το indexKnow βρίσκει τους αρμόδιους επιλυτές ε_i
- Ο Δ ενημερώνει τα αντίστοιχα I_i, με $flag_i=1$ και με την αντίστοιχη

διαταγή για ενημέρωση, έστω U_i , που προορίζεται για τον $ε_i$.

- Κάθε αρμόδιος $ε_i$ ενημερώνεται από I_i. Αν $flag_i=1$, τότε λαμβάνει τη διαταγή για να ενημερώσει το αντίστοιχο $δ_i$. Θέτει στο I_i τις αρχικές του τιμές ($flag_i=0$).
- Ο $ε_i$ εκτελεί την ενημέρωση στο $δ_i$ (σε επίπεδο RAM) και δίσκου). Στο O_i θέτει $flag_i=1$.
- Ο Δ ενημερώνεται από το O_i. Αν $flag_i=1$ για κάθε αρμόδιο $ε_i$, ο Δ θεωρεί ότι η ενημέρωση έγινε. Θέτει $Out\_flag^i=0$.
- Κάθε αρμόδιο $ε_i$ ενημερώνεται αν $Out\_flag_i=0$, οπότε η διαδικασία έχει λήξει.

5.7 ΕΦΑΡΜΟΓΕΣ

Από τις προηγούμενες παραγράφους είναι σαφές ότι κάθε επιλυτής αναλαμβάνει στην ουσία έναν υπόχωρο του όλου γνωσιολογικού χώρου. Επομένως η προταθείσα μέθοδος μπορεί να εφαρμοστεί σε κάθε γνωσιολογικό χώρο όπου μπορεί να υπάρξει μία στοιχειώδης διαμέριση (έστω και εμπειρική) σε ένα σύνολο υποχώρων Γνώσης.

Από την πείρα είναι γνωστό πως κάθε γνωσιολογικός χώρος μπορεί να παρασταθεί όπως ένα δένδρο ή (π.χ. εκπαιδευτικά) όπως ένα βιβλίο, όπου διακρίνονται τα βασικά κεφάλαια αυτού με τα αντίστοιχα περιεχόμενά τους. Κατ' ακολουθία η μέθοδος που προτάθηκε μπορεί να εφαρμοστεί σε κάθε επιστημονικό κλάδο.

Τουλάχιστον μέχρι σήμερα κατέστει αδύνατος ο εντοπισμός κάποιου χώρου Γνώσης με αδυναμία διάκρισης αυτού σε υποχώρους. Ωστόσο, στη συνέχεια απλά αναφέρονται τρία συγκεκριμένα παραδείγματα εφαρμογών για περισσότερη κατανόηση της μεθόδου που προτείνεται:

1. Εμπειρο Σύστημα σε Διάγνωση και αγωγή ασθενειών πνευμόνων.
   Η Βάση Γνώσης χωρίζεται σε βασικές ασθένειες πνευμόνων. Ο πρώτος επιλυτής αναλαμβάνει τις πνευμονίες, ο δεύτερος τα βροχικά, ο τρίτος τις ιώσεις πνευμόνων, κ.λ.π., ενώ κάποιος από αυτούς αναλαμβάνει την γενική γνώση αρχικού εντοπισμού της κατηγορίας της ασθένειας.

2. Εμπειρο Σύστημα ανίχνευσης βλάβης Υπολογιστή.
   Ο πρώτος επιλυτής αναλαμβάνει την γνώση για βλάβες motherboard, ο δεύτερος για κάρτες περιφερειακών, ο τρίτος για οθόνη, ο τέταρτος για δίσκο, ο πέμπτος για μονάδα δισκέτας κ.λ.π.

3. Εμπειρο Σύστημα εντοπισμού Νομοθεσίας και σχετικών διατάξεων.
   Ο πρώτος επιλυτής αναλαμβάνει το ποινικό, ο δεύτερος το εμπορικό, ο τρίτος τα εργασιακά θέματα κ.λ.π.

5.8 ΠΛΕΟΝΕΚΤΗΜΑΤΑ - ΕΠΕΚΤΑΣΕΙΣ

- Η δομή αυτή που αναπτύχθηκε για μεγάλες ΒΓ αξιοποιεί τη δομή που περιγράφηκε στα προηγούμενα κεφάλαια, διότι είναι προφανές

ότι τίποτα δεν εμποδίζει την ταυτόχρονη συνύπαρξη και των δύο σχημάτων, αρκεί βέβαια οι επιλυτές εi επιπρόσθετα να τηρούν τις προϋποθέσεις που ισχύουν στα κεφάλαια 2-4. Μάλιστα συνιστάται η ύπαρξη ενός τουλάχιστον επιπλέον επιλυτή για διαχείριση περιοχών γνώσης πέραν του σύνδρομου προγραμματισμού, όπως ακριβώς αναφέρθηκε στα προηγούμενα κεφάλαια.

- Στην περίπτωση ανεξάρτητων επεξεργαστών το ανώτερο όριο μεγέθους της ΒΓ ρυθμίζεται από το ολικό άθροισμα των διαθεσίμων μνημών (RAM) του όλου συστήματος. Αυτό σημαίνει ναι μεν ότι θεωρητικά το πρόβλημα των μεγάλων ΒΓ είναι δυνατόν να αντιμετωπιστεί, αλλά πρακτικά αυξάνεται το κόστος σε υλικό (hardware). Στο σημείο αυτό υπενθυμίζεται πως το πρόβλημα των μεγάλων ΒΓ ανήκει στο σύνολο των NPC προβλημάτων [Garey,Johnson,1979] (Non-polynomial memory complete problems). Αν μάλιστα οι επιλυτές αντί να έχουν όλους τους προτασιακούς γνωσιολογικούς τύπους της Γνώσης (αναλυτική Γνώση) έχουν κωδικοποιημένη μορφή (κεφ. 2), τότε είναι φανερό πως το προταθέν οργανωτικό σχήμα διαχείρισης έχει τη δυνατότητα προσπέλασης τεραστίων ΒΓ (Huge Knowledge Bases).

- Το ίδιο σχήμα που αναφέρθηκε μπορεί να εφαρμοστεί και στην περίπτωση που οι επιλυτές αντί να υποστηρίζουν τμήματα Γνώσης που όλα μαζί αποτελούν διαμερίσεις μιάς μεγάλης ΒΓ, να έχουν υποσύνολα Γνώσης κατανεμημένης μορφής. Σε μία τέτοια περίπτωση δεν έχουμε μόνο κατανεμημένη επεξεργασία αλλά και

κατανεμημένη Γνώση. Το σημείο αυτό θα αποτελέσει ένα από τα αμέσως επόμενα ερευνητικά ενδιαφέροντα της συγγραφέως της παρούσης Διατριβής.

- Η δομή που αναφέρθηκε επιτρέπει αντιμετώπιση μεγάλων ΒΓ χωρίς η παρουσία μεγάλου mainframe να είναι απαραίτητη. Αυτό σημαίνει ότι μπορεί να γίνει υποστήριξη μεγάλης ΒΓ με μερικούς φθηνούς μικροϋπολογιστές ενταγμένους σε ένα απλό peer-to-peer (ομότιμο) δίκτυο. Και μόνο το πλεονέκτημα αυτό αντιμετώπισης σκληρού προβλήματος με φθηνό hardware, δίνει στην προταθείσα μέθοδο μία ιδιαίτερη σημασία, που ο αναγνώστης καλείται ιδιαίτερα να σταθμίσει.

- Είναι αρκετές οι περιπτώσεις που ένα Έμπειρο Σύστημα (ΕΣ) πληροφορεί τελικά το Χρήστη για αδυναμία εξαγωγής συμβουλής (advice) σε κάποια συγκεκριμένη περίπτωση για την οποία ο Χρήστης ενδιαφέρεται για διάγνωση. Μία επέκταση της παρούσης Διατριβής στο θέμα εξαγωγής συμπεράσματος είναι: το ΕΣ να συνεχίζει περεταίρω τη διεργασία χρησιμοποιώντας ειδικούς κανόνες ομοιότητας, με σκοπό τον εντοπισμό της πλησιέστερης περίπτωσης που γνωρίζει προς την παρούσα περίπτωση του Χρήστη. Η επέκταση αυτή έχει μεγάλη πρακτική αξία σε περιπτώσεις όπου ο Χρήστης είναι δυνατόν να δώσει μία ή περισσότερες λανθασμένες απαντήσεις στις ερωτήσεις του ΕΣ . Η επέκταση αυτή θα μπορούσε να χαρακτηριστεί σαν "αυτόματος διορθωτής Χρηστών". Από την παγκόσμια βιβλιογραφία, ως σήμερα τίποτα αντίστοιχο δεν έχει

πέσει στην αντίληψή μας όσον αφορά το θέμα αυτό, που να λειτουργεί σε επίπεδο δικτύου πολλών Χρηστών.

ΠΑΡΑΡΤΗΜΑ

1. ΕΙΣΑΓΩΓΗ

Στο παρόν παράρτημα δίνεται ένας πίνακας μετρήσεων, στο βασικό μέτρο χρόνου προσπέλασης, για τις πράξεις που περιγράφονται στο κεφάλαιο 4 (συνδιασμός προσπέλασης σε επίπεδο κύριας μνήμης με Prolog και άμεσης προσπέλασης στην περιφερειακή μνήμη). Εν συνεχεία δίνονται κάποια predicates Prolog τα οποία εξυπηρετούν ορισμένα βασικά θέματα της παρούσης εργασίας.

2. ΣΤΟΙΧΕΙΑ ΧΡΟΝΟΥ ΠΡΟΣΠΕΛΑΣΗΣ

Ο ακόλουθος πίνακας δείχνει το πλήθος διαβασμάτων, εγγραφών (read's, write's) και μέγιστο πλήθος ψηφιοσυλλαβών (bytes) για κάθε βασική πράξη (Inq, Add, Del, Chg) που έχει περιγραφεί στο κεφάλαιο 4.

Αν B είναι το βασικό μέτρο χρόνου προσπέλασης, τότε ο αναμενόμενος χρόνος προσπέλασης T(i) για την i πράξη είναι:

$$T(i)=B*a*60 \text{ seconds},$$

όπου a το πλήθος προσπελάσεων (read, write) για την συγκεκριμένη πράξη.

ΠΡΑΞΗ	read		write	
	πλήθος	bytes	πλήθος	bytes
INQ	m+2	Σ-ak+c+1	-	-
DEL	m+4	Σ-ak+3*c+1	p+4	p+3*c+ Size_rec_fl
ADD	2*m+7	2*(Σ-ak)+6*c+ Size_rec_fl+ 1	m+p+5	p+4*c+Σ-ak +Size_rec_fl
CHG	2*m+1	2*(Σ-ak) +c+1	p+m	p+Σ-ak+c

όπου:

c: μήκος των Next, Nextp, All, Bottomp, P0,P01,cont

p: πλήθος των fli αρχείων, i=1,...,p

Σ: Σai, i=1,...,m, άθροισμα των μηκών (ai) των m πεδίων κάθε εγγραφής

m: πλήθος πεδίων

ak: μήκος πεδίου κλειδιού

size_rec_fl: μήκος εγγραφής στο fli.

2. ΒΑΣΙΚΑ PREDICATES

• ***list_roots (List_of_roots)*** :

Επιστρέφει τη λίστα λεκτικών - ριζών (των δέντρων που σχηματίζονται από τα λεκτικά συμπεράσματων των κανόνων) List_of_roots που παρουσιάζονται στο σύνολο κανόνων. Το find_root(List_of_roots,[]) λειτουργεί βοηθητικά, αφού ξεκινάει με

αρχική τιμή [] για τη λίστα ριζών. Ομοίως βοηθητικά λειτουργεί το find_root1 το οποίο βρίσκει για κάθε κόμβο Root1 τον πατρικό, κατευθυνόμενο, έτσι, στη ρίζα.

```
list_roots(List_of_roots):- find_root(List_of_roots,[]).

find_root(List_Of_Roots,List0):- r(A,_,_),!,
                                 find_root1(A,Root),
                                 append(List0,[Root],List_Of_Roots).
find_root1(Root,Root1):-   r(A1,B1,_),
                           B1=Root,!,
                           find_root1(A1,Root1).
find_root1(R,R).
```

- ***kid(Root1,Kids)***

Για κάθε κόμβο Root1 του δέντρου κανόνων βρίσκει τη λίστα των επομένων κόμβων-παιδιών kids. Ενεργοποιεί το:

```
Kid1(Root1,Ki,Kids),
```

με Ki=[] αρχική τιμή της λίστας κόμβων-παιδιών.

```
Kid(Root1,Kids):- kid1(Root1,[],Kids).
Kid1(Root1,Ki,Kids):- r(Root1,B,_),
                      not(member(B,Ki)), !, %αν B δεν είναι μέλος της Ki
                      append(Ki,[B],Ki1),
                      kid1(Root1,Ki1,Kids).
kid1(_,K,K).
```

•brothers(Root,Brothers):

Για κάθε κόμβο Root επιστρέφει τη λίστα αδελφών κόμβων Brothers, προς τα δεξιά του δεδομένου κόμβου Root, οι οποίοι είναι οι κόμβοι του αυτού επιπέδου. Γίνεται χρήση των δομών από το T.Tmp

```
brothers(Root,Brothers):- t(_,L),
                member1(Root,L,Brothers),!.
brothers(_,[]).
member1(X,[X|Brothers],Brothers):-!.
member1(X,[_|Tail],Brothers):- member1(X,Tail,Brothers).
```

•parent(Root,ListParent)

Για την εύρεση του πατρικού κόμβου, ο οποίος επιστρέφεται σε λίστα ListParent.

```
parent(Root,ListParent):- t(Parent,List),
                member(Root,List),!,
                ListParent=[Parent].
parent(Root,[]).
```

Ακολουθούν predicates για τη δημιουργία του T.tmp :

•find

```
find(A0,B0) :- r(A,B,_),
               not(t(A,_)), %για τα φύλλα του δέντρου
               assert(t(A,[]),t),
               find(A,B).
fnd(A0,B0):-  r(A,B,_),
               bound(B),
               not(r(B,_,_)),
               not(t(B,[]) ),!,
               assert(t(B,[]),t),
               find(A,B).
find(_,_).
```

•node

```
node(A,B,List0):-r(A,B,_),t(A,List0),
               not(member(B,List0)),
               retract(t(A,List0),t),
               append(List0,[B],List1),
               assert(t(A,List1),t),
               fail.
node(A,B,List0):-t(A,List0), member(B,List0), fail.

node(A,B,List0):-bound(A), assertz(t(A,[B]),t), fail.
node(_,_,_).
```

- ***NextRoot(Root,Next)***

Σύμφωνα με την προδιατεταγμένη διάσχιση ενός δέντρου, (Κόλλιας,1986), γίνεται επίσκεψη των κόμβων από τη ρίζα και από αριστερά προς δεξιά με επίσκεψη κάθε υποδέντρου. Σύμφωνα με το predicate NextRoot, αν Root είναι ο παρών κόμβος, New είναι ο αμέσως επόμενος για επίσκεψη κόμβος.

```
nextroot(Root,NEW):-
        t(Root,[]),
        % αν ο κόβος Root δεν έχει κόμβους παιδιά
        backroot(Root,NEW).
        % backroot για ανάδρομη πορεία στο δέντρο

nextroot(Root,NEW):-
        t(Root,List_Of_Next),
        % List_Of_Next λίστα κόμβων-παιδιών του Root
        not ( List_Of_Roots=[] ), List_Of_Next=[Next1|_],
        % ο νέος κόμβος θα είναι πρώτο στοιχείο της
        %List_Of_Next
        New=Next.
```

- ***backRoot(Root,New)***

Αν Root είναι ο παρών κόμβος, New είναι ο πρώτος κόμβος που

συναντάται κατά την οπισθοδρόμιση πάνω στους κλάδους του δέντρου λεκτικών, με την ιδιότητα ότι η διάσχιση του δέντρου δεν έχει περάσει ακόμη από αυτόν.

```
backroot(Root,NEW):-
        brothers(Root,[Brother1|_]),!,
        % αν το Root έχει προς τα δεξιά κόμβους του ίδιου
        %επιπέδου
        % ο πρώτος εξ αυτών είναι ο ζητούμενος.
        New=Brother1.

backroot(Root,NEW):-
        brothers(Root,[]),
        % αν το Root δεν έχει προς τα δεξιά κόμβους
        % του αυτού επιπέδου
        parent(Root,[ParentRoot]),
        % με κόμβο εκκίνησης τον πατρικό
        % καλείται το backroot
        backroot(Parentroot,NEW).
```

- ***Tree***

Το Tree ξεκινώντας από τη ρίζα, διασχίζει προδιατεταγμένα (in preorder) το όλο δέντρο. Με το βοηθητικό predicate CONTROL(Rule) γίνεται ο απαραίτητος έλεγχος, σε ποιό Λ κανόνων Η θα καταχωρηθεί ο κανόνας Rule.

Τα Root, List_Of_Roots, δεν περνάνε στο Tree σαν ορίσματα για

καλύτερη εκμετάλλευση του πίνακα Heap (ότι απομένει από Stack , Trail), κατά τη διαχείριση της RAM από την Prolog. Ετσι τα Root, List_Of_Roots διαβάζονται από το αρχείο δίσκου "new", αδιαφορώντας για τη σχετική καθυστέριση.

Το Tree τερματίζεται με επιτυχία όταν το New είναι το τελευταίο στοιχείο της αρχικής λίστας ριζών List_Of_ROots.

```
tree :-
        openread(new,"new"),readdevice(new),
        readint(Root),nl,
        readterm(list,List_Of_Roots),closefile(new),
        % εύρεση του "επόμενου κόμβου"
        nextroot(Root,New),!,
        openmodify(new,"new"),writedevice(new),
        filepos(new,0,0),
        writef("%20",New),nl,closefile(new),
        % αν New όχι τελευταίο στοιχείο της λίστας ριζών
        not(lastlist(List_Of_Roots,New)),
        % εύρεση του πατρικού κόμβου
        parent(New,[Parent]),!,
        % αναζήτηση κανόνα στη βάση κανόνων στο δίσκο
        openread(rule,"rulebase"),
        readdevice(rule),filepos(rule,0,0),
        readrec(Parent,New,Rule),
        % αναζήτηση του Rule=r(Parent,New,_),
        closefile(rule),
```

```
                % έλεγχος ένταξης του κανόνα σε κάποιο από τα Hi
                control(Rule),
                tree.
tree.
```

- ***control***

Για τον έλεγχο ένταξης του κανόνα σε κάποιο από τα Hi. Το αρχειο "x" έχει εντελώς βοηθητικό ρόλο, με στόχο τον ταχύ υπολογισμό του μήκους του κανόνα.

```
control(Rule) :- rule_str(Rule,Rulestring),
                % υπολογισμός του μήκους του κανόνα.
                str_len(Rulestring,RuleLenght),
                openmodify(size,"size"),
                readdevice(size),
                % διάβασμα των τιμών Size (μέγεθος του H)
                % και Y (μέγιστο επιτρεπτό μέγεθος κάθε H)
                readreal(Size),readreal(Y),
                % νέο μέγεθος του H μαζί με τον κανόνα
                NewSize=Size+RuleLenght,
                writedevice(size),
                Filepos(size,0,0),
                writef("%20",NewSize),
                closefile(size),
                %  από το αρχείο "whatH" εύρεση τελευταίου H
```

```
WHATH(H),!,
if_size(H,NewSize,Y,Rule).
```

- ***if_size(H,Size,Y,Rule)***

Ανάλογα με το αναμενόμενο μέγεθος του H, μετά από την προσθήκη του κανόνα, αυτός εγγράφεται στο παρόν H ή στο αμέσως επόμενο.

```
if_size(H,Size,Y,Rule) :- Size<Y, !,
        % αν αναμενόμενο H  είναιμικρότερο από το Y
        closefile(h), openappend(h,H),writedevice(h),
        % ο κανόνας εγγράφεται στο H
        write(Rule),nl,closefile(h).
if_size(H,Size,Y,Rule):- Size>=Y, !,
        % αν το αναμενόμενο μέγεθος του H
        % είναι μεγαλύτερο ή ίσο με Y
        % ενημέρωση του "index"
        update_index(H,Root),
        % μέγεθος νέου H = 0
        NewSize=0,
        openmodify(size,"size"),writedevice(size),
        Filepos(size,0,0),writef("%20" ,NewSize),
        closefile(size),% καταχώρηση του κανόνα στο νέο H1
        newH(H,H1),
        openappend(h,H1),writedevice(h),
        write(Rule),nl,closefile(h).
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

[Allen J, 1987] Allen James, *"Natural Language Understanding"*, Benjamin/ Cummings Publishing Company, INC, 1987.

[Apers, 1988] Apers P., "*Data Allocation in Distributed Database Systems*", ACM Transactions on Database Systems, Vol.13, No.3, pp.263-304, Sept. 1988.

[Barsalou T. Wiederhold G., 1989] Barsalou T. Wiederhold G., *"Knowledge-directed Mediation between Application Objects and Base Data"* , Proc. of the working Conf., Keele,1989, eds. Deen, Thomas, 1990.

[Βασιλακόπουλος, Χρυσικόπουλος, 1990] Βασιλακόπουλος Γ., Χρυσικόπουλος Β., *"Πληροφοριακά Συστήματα Διοίκησης"*, εκδ. Σταμούλης ,Πειραιάς, 1990.

[Berghel, 1985] Berghel H. L., *"Simplified Integration of Prolog with RDBMS"*, DATA BASE, pp. 3-12, Spring 1985.

[Bodkin, Graham, 1989] Bodkin T., Graham I., *"Case Studies of Expert Systems Development using microcomputer software packages"*, Expert Systems, Vol.6, No.1, February 1989.

[Botten, Kusiak and Raz,1989], Botten N., Kusiak A, Raz T, *"Knowledge Bases: Integration, verification, and partitioning"*, European Journal of Operation Research, North-Holland, Vol. 42, pp.111-128, 1989.

[Brodie, Mylopoulos, 1985] Brodie M., Mylopoulos J., *"On Knowledge Base Management Systems"*, Spinger- Verlag, 1985.

[Brule' F. James, 1986] Brule' F. James, *"Artificial Intelligence, theory, logic and application"*, U.S.A., 1986.

[Carlson, Sudha, 1991] Carlson D., Sudha R., *"An Architecture For Distributed Knowledge-Based systems"*, Data Base, Winter/Spring, pp. 11-21, 1991.

[Ceri, Pelagati, 1985] Ceri S. , Pelagati G., *"Distributed Data bases"*, McGraw Hill Book corp., 1985.

[Chandrasekaran, 1988], Chandrasekaran B., *"From Numbers to Symbols to Knowledge Structures: Artificial Intelligence Perspectives on the Classification Task"*, IEEE Transactions on Systems, Man and Cybernetics, 1988.

[Chen, Kambayashi, 1993], Chen Q., Kambayaski Y., *"Coordination of data and knowledge base systems under distributed environment"*, IFIP Transaction A: Computer Science and Technology No. A-25, pp.139-154, 1993.

[Cheng, Fu, 1984] Cheng Y., Fu K.S., *"Conceptual clustering in Knowledge organization"*, CH2107-1/84/0000/0274$01.00, IEEE, 1984.

[Γιαλούρης, 1993] Γιαλούρης Κ., *"Εμπειρα Συστήματα: συμβολή στη δομή φλοιών ανάπτυξης - Εφαρμογές στη γεωργία"*, Διδακτορική Διατριβή, Γεωργικό Πανεπιστήμιο, Αθήνα, 1993.

[Clarke, 1991] Clarke R., *"A Contingency Approach to the Application Software Generations"*, Data Base, pp. 23-34, Summer 1991.

[Clocksin, Mellish, 1987] Clocksin W.F, Mellish C.S., *"Programming in Prolog"*, 3th Edition, 1987.

[Conery, 1987] Conery J., *"Parallel Execution of Logic Programs"*, Klumer Academic Publishers, 1987.

[Cordingleys, 1989] Cordingleys E., *"Knowledge Elicitation Techniques for Knowledge based Systems"*, in Knowledge Elicitation Technicques", ed. by Diaper Dan, 1989.

[Deen, Thomas, 1990] Deen S.M., Thomas G.P., eds. *"Data and Knowledge Base Integration"*, Proc. of the working Conf., Keele,1989, Pitman, 1990.

[Deliyanni, Kowalsky, 1979] Deliyanni N., Kowalsky R.A.,*"Logic and Semantic Networks"*, Commun. ACM Vol.22, No.3, pp.184-192, 1979.

[Diaper, 1989] Diaper D., *"Knowledge Elicitation, prenciples, techniques and applications"*, Ellis Harwood Edition/Halsten Press Edition, 1989.

[Farris, Singleton, 1989] Farris C. D. , Singleton P. *"Combining Prolog with an RDBMS for Applications in Software Configuration Management"*, Proc. of the working Conf., Keele, 1989, ed. Deen Thomas 1990.

[Gammack, 1987] Gammack J., *"Different Techniques and Different Aspects on Declarative Knowledge"*, in Knowledge Acquisition for Expert Systems ed. Alison Kidd, 1987.

[Garey, Johnshon,1979] Garey M.R., Johnson D.S, *"A Guid to the Theory of NP-Completeness"*, Freeman, 1979.

[Garner, 1987] Garner B. J., *"Expert Systems: from Database to Knowledge Base"*, Information and Software Technology, Vol. 29, No. 2, pp. 60-65, March/April 1987.

[Gomez, Chandrasekaran, 1981] Gomez F., Chandrasekaran B., *"Knowledge Organization and Distribution for Medical diagnosis"*, IEEE Transactions on Systems, Man and Cybernetics, Vol. SMC-11, No.1, Jan. 1981.

[Guenther, Lehman, Schonfeld, 1986] Guenther F., Lehman H., Schonfeld W., *"A theory for the representation of Knowledge"*, IBM J.Res. Develop, Vol. 30, No. 1, January 1986.

[Hardzband, Maryanski, 1985] Hardzband D., Maryanski F., *"Enhansing Knowledge Representation in Engineering Data bases"*, Computer, Sept, 1985.

[Hodil, Butler, Richardson, 1986] Hodil E.D, Butler C.W., Richardson G. L. , *"Knowledge-based systems in the commercial environment"*, IBM Systems Journal, Vol. 25, No. 2, 1986.

[Ip, Holden, 1989] Ip S., Holden T., *"A Knowledge Assistant for he Design of Information Systems"*, Proc. of the working Conf., Keele,1989 , ed. Deen, Thomas, 1990.

[Irani, Shing, 1984] Keki I., Shing Y., *"Implementation of Very Large PROLOG-based Knowledge Bases on Data flow archtectures"*, 1st Conf. on AI Applications, Sheraton, Denver Tech Centre Dec. 5-7, 1984, CH2107-1/84/0000/0454$01.00,IEEE, 1984.

[Kastner, Hong, 1984] Kastner J. K., Hong S. J. *"A review of expert systems"*, European Journal of Operation Research, North Holland, Vol. 18, pp.285-292, 1984.

[Kellogg, 1984] Kellogg C., *"The Transition from Data Management to Knowledge Management"*, International Conference on Data Engineering. Los Angeles, CA, USA, pp. 467-472, April 24-27 1984.

[Kidd, 1987] Kidd A., *"Knowledge Acquisition for Expert Systems"*, A Practical Handbook, Plenum Press, 1987.

[Kim, Novick, 1993], Kim S., Novic M., *"Using clustering techniques to support case reasoning"*, International Journal of Computer Applications in Technology, Vol. 6, No. 2, pt. 3, pp. 57-73, 1993.

[Κόλλιας, 1986] Κόλλιας Ι., "Δομές Δεδομένων", τόμοι Ι,ΙΙ, Αθήνα, 1986.

[Laffey, Perkins, Nguyen, 1984] Laffey T.J., Perkins W.A., Nguyen T.A., *"Reasoning about fault diagnosis with LES"*, CH2107-1/84/0000/0267$01.00, IEEE, 1984.

[Laffey, Cox, Schmidt, Kao, Read, 1988] Laffey T.J., Cox P., Schmidt J., Kao S., Read J., *"Real-Time Knowledge-Based Systems"*, AI Magazine, pp.27-45, Spring 1988.

[Lee R, 1983] Lee R., *"Application Software and Organizational Change: Issues in the Representation of Knowledge"*, Information Systems Vol. 8, No. 3, pp. 187-194, 1983.

[Leitheiser, 1986] Leitheiser R., *"Computer support for Knowledge workers: A review of laboratory experiments"*, DATA BASE, pp.17-35, Spring 1986.

[Leung, Felix Wong, Lam, 1989] Leung K.S, Felix Wong W. S., Lam W, *"Applications of a Novel Fuzzy Expert System Shell"*, Expert Systems, Vol.6, No.1, February 1989.

[Li, 1984] Li D., *"A PROLOG Database System"*, Research studies Press, 1984.

[Lin, 1989] Lin T.Y., " *"Neighborhood Systems and Approximation in Reletional Databases and Knowledge Bases"*, proc. fourth int. symposium on methodologies for intelligent systems, 1989

[Lizhu, Deyuan, Zhengping, Liping, 1988] Lizhu Z., Deyuan Y., Zhengping F., Liping Z., *"QKBMS/75 -- A Knowledge Base Manegement System Growing From Relational DBMS and Logic Programming Language"*, Conference on The Role of AI in Databases and Information Systems, Guangzhou, PR China, 4-8 July 1988, AI in Databases and Information Systems (DS-3), eds. Meersman R., Shi Z., North-Holland, 1990.

[MacKellar, Maryanski, 1984] MacKellar B., Maryanski F., *"Reasoning by Analogy in Knowledge Base Systems"*, Proc. of the 4th Int. Conference on Data Engineering, LA, CA, USA , pp. 467-472, Feb 1-5, 1984.

[Meersman, Shi, Kung, 1990] Meersman R.A., Shi Z., Kung C., Ed. *"Artificial Ntelligence In Databases and Information Systems"* (DS-

3), Proceedings of the IFIP TC2/TC8/WG 2.6/WG 8.1 Working Conference on The Role of AI in Databases and Information Systems, Guangzhou, PR China, 4-8 July, 1988, North- Holland, 1990.

[Minsky, 1975] Minsky M., *"A Framework For Representing Knowledge"*, in The Psychology of Computer Vision, ed. P.H. Winston, MacGraw-Hill, New York, pp.211-277, 1975.

[Mohania, Sarda, 1994], Mohania M., Sarda N., *"A heuristic for rule alocation in distributed deductive database systems"*, Rules in Database Systems, Proceedings of the 1st International Wrkshop on Rules in Database Systems, p.385-400, 1994.

[Moore, Hawkinson, Knickerbocker, Churchman, 1984] Moore R., Hawkinson L., Knickerbocker K., Churchman L., *"A Real-Time Expert System for process control"*, 1st Conf. on AI Applications, Sheraton, Denver Tech Centre Dec. 5-7, 1984, CH2107-1/84/0000/0454$01.00, IEEE, 1984.

[Nait Abdallah, 1991] Nait Abdallah M.A. , *"Kernel Knowledge versus belt Knowledge in default reasoning: a logical approach"*, International Coonference on Computing and Information, Ottawa, Canada, May 27-29, 1991, Proceedings, Lecture Notes in Computer Science, V.497, Spinger-Verlag, pp.675-686, 1991.

[Obretenov, Agelov, Mihaylov, Dishlieva and Kirova, 1988] Obretenov D., Agelov Z., Mihaylov J., Dishlieva P., Kirova N., *"A Knowledge-based approach to relational database design"*, Data &

Knowledge Engineering Vol. 3, pp. 173-180 , North- Holland, 1988.

[Παναγιωτόπουλος, 1988] Παναγιωτόπουλος Ι.-Χ.,*"Νέες Μορφές Τεχνολογίας - Εμπειρα Συστήματα - Turbo Prolog"*, εκδ. Σταμούλης, Πειραιάς, 1988.

[Παναγιωτόπουλος, Παπαθανασίου, 1991] Παναγιωτόπουλος Ι.-Χ., Παπαθανασίου Ε., *"Μετασχηματισμός Βάσεων Δεδομένων σε Βάσεις Γνώσης: περίπτωση Μικροϋπολογιστών και Δικτύων αυτών"*, πρακτικά 3ου Πανελλήνιου συνεδρίου Πληροφορικής, τόμος ΙΙ, σελ. 74-85, Αθήνα, Μάϊος 1991.

[Παναγιωτόπουλος, Τσιρόπουλος, 1988] Παναγιωτόπουλος Ι.-Χ., Τσιρόπουλος Ι., *"Ελαχιστοποίηση απαιτήσεων μνήμησε μεγάλες Τράπεζες Γνώσης για μικροϋπολογιστές και τοπικά δίκτυα αυτών"*, 2ο Πανελλήνιο Συνέδριο Πληροφορικής, Θεσσαλονίκη, 4-6 Νοεμ. 1988.

[Panayiotopoulos, Papathanassiou, 1994] Panayiotopoulos J.-C., Papathanassiou E., *"The RAM-Real-Time Problem in the case of Large Knowledge Bases"*, Journal of Information & Optimization Sciences, Vol. 15 , No.1, pp.41-48, 1994

[Piatetsky-Shapiro, Frawley,1991] Piatetsky-Shapiro G., Frawley W., *"Knowledge discovery in Databases"*, AAAI Press / MIT Press, 1991.

[Quinlan, 1986] Quinlan J. R., "Induction of Decision Trees", Machine Learning, Vol. 1, No.1, pp.81-106, 1986.

[Quinlan, 1990] Quinlan J. R.,*"Learning Logical Definitions from Relations"*, Machine Learning, Vol.5, No.3, pp.239-266, 1990.

[Randall, Douglas, 1982] Randall D., Douglas L., *"Knowledge-Based Systems in Artificial Intelligence"*, McGraw-Hill International Book Comp. 1982.

[Rasmussen, 1992] Rasmussen E., *"Clustering Algorithms"*, Information Retrieval, Data Structures & Algorithms, ed. Frakes W., Baeza-Yates R., Prentice All, 1992.

[Raynal, 1988] Raynal M., *"Distributed Algorithms and Protocols"*, Wiley, 1990.

[Raz and Botten, 1992], Raz T., Botten N., *"The Knowledge base partitioning problem: Mathematical Formulation and Heuristic clustering"*, Data & Knowledge Engineering, North-Holland, Vol. 8, pp. 329-337, 1982.

[Reiter, 1990] Reiter R., *"Integrity Constraints for Knowledge Bases"*, Working Conference on The Role of Artificial Intelligence in Databases and Information Systems, Guangzhou, PR China, 4-8 July 1988, AI in Databases and Information Systems (DS-3), ed. Meersman R., Shi Z., North-Holland, 1990.

[Ringland, Duce, 1988] Ringland G. A. and Duce D. A., *"Approaches to Knowledge Representation"*, Research Studies Press LTD, England, 1988.

[Rundensteiner, 1988] Rundensteiner A., *"The role of AI in Databeses versus the role of Database Theory in AI: an opinion."*, Conference

on The Role of AI in Databases and Information Systems, Guangzhou, PR China, 4-8 July 1988, AI in Databases and Information Systems (DS-3), ed. Meersman R., Shi Z., North-Holland, 1990.

[Schalkoff, 1990], Schalkoff R., *"Artificial Intelligence: An Engineering Approach"*, McGraw-Hill, 1990.

[Schwelger, 1994], Schwelger J., *"Facilitating teamwork of autonomous systems with a distributed real-time knowledge base"*, Proceedings IEEE International Conference on Robotics and Automation pt. 4 1994. Publ by IEEE, IEEE Service Center, Piscataway, NJ, USA, p. 2883-2888, 1994.

[Sowa, 1990] Sowa J., *"Knowledge Representation in Databases, Expert Systems, and Natural Language"*, Conf. on The Role of AI in Databases and Information Systems, Guangzhou, PR China, 4-8 July 1988, AI in Databases and Information Systems (DS-3), ed. Meersman R., Shi Z., North-Holland, 1990.

[Sridhar, Murty, 1994], Sridhar V., Murty M.N., *"Knowledge-based clustering approach for data abstraction"*, Knowledge-Based Systems, Vol. 7, No. 2, pp. 103-13, June 1994.

[Su,Shi,Wang,Hu,Shi and Wang, 1988] Su B., Shi C., Wang K., Hu P., Shi H., Wang J., *"The Architecture of a Distributed Knowledge Base System"*, Conference on The Role of AI in Databases and Information Systems, Guangzhou, PR China, 4-8 July 1988, AI in

Databases and Information Systems (DS-3), ed. Meersman R., Shi Z., North-Holland, 1990.

[Takizawa, Katsumata, 1989] Takizawa M., Katsumata M., *"Integration of Database Systems at the Navigational Level by Using Prolog"*, Proc. of the working Conf., Keele,1989, eds. Deen, Thomas, 1990.

[Thompson, Thompson, 1986] Thompson B., Thompson W, *"Finding rules in data"*, Nov. 1986, Byte.

[Townsend, 1986] Townsend C., *"Introduction to Turbo Prolog"*, Singapore, 1986.

[Vedder, 1989] Vedder R., *"PC-based Expert System Shells: some desirable and less desirable characteristics"*, Expert Systems, Vol.6, No.1, Feb. 1989 .

[Wiederhold, 1985] Wiederhold G, *"Knowledge versus data"*, in "On Knowledge based Management systems" ed. dy Brodie, Mylopoulos, 1985.

[Williams, Bainbridge, 1988] Williams T., Bainbridge B., *"Rule Based Systems"*, in Approaches to Knowledge Representation, ed. Ringland G. A., Duce D.A., Research Studies Press LTD, England, 1988.

[1988, Turbo Prolog] *Turbo Prolog* v.2.0, manual, Borland, 1988.