ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

*Διπλωματική Εργασία*

# Grid Resource Brokering

## Σημασιολογικός μετασχηματισμός αιτημάτων για πόρους σε συστήματα Grid

***Κοντοκώστας Δημήτριος***

*A.M.: 1999030040*

*Εξεταστική επιτροπή:*

καθ. Σαμολαδάς Βασίλης *(επιβλέπων)*

καθ. Κοντογιάννης Κώστας

καθ. Κουμπαράκης Μανώλης

*Χανιά, Ιούλιος 2004*

# Περιεχόμενα

# Εισαγωγή

Η υποδομή του Grid θα έχει τη δυνατότητα να συνδέει δυναμικά πόρους ως σύνολο για να υποστηρίζει την εκτέλεση μιας μεγάλης κλίμακας, διεξοδική σε πόρους και κατανεμημένης εφαρμογής. Η αναζήτηση πόρων σε ένα περιβάλλον Grid μπορεί να είναι μια πολύ σύνθετη και δύσκολη εργασία. Ένα ζήτημα είναι η σωστός προσδιορισμός των αιτημάτων των πόρων όταν εκφράζεται στο χαμηλού επιπέδου σχήμα μιας βάσης δεδομένων για την περιγραφή των πόρων. Με μια ευθύ λύση, τα αιτήματα των πόρων μπορεί να είναι πολύ μεγάλα, ιδιαίτερα όταν υπάρχουν πολλαπλάσιες εναλλακτικές προδιαγραφές που θα ικανοποιούσαν μια ανάθεση (π.χ. ζητάμε όλα τα λειτουργικά συστήματα που ανήκουν στην οικογένεια UNIX). Τέτοια αιτήματα είναι δύσκολο να συντεθούν με το χέρι.

Αναπτύσσουμε έναν σημασιολογικό μετασχηματισμό των αιτημάτων για πόρους, από μια υψηλού επιπέδου γλώσσα, όπως η DataLog, που επιτρέπει στα αιτήματα να διευκρινιστούν χρησιμοποιώντας ισχυρά κατηγορήματα (predicates), σε αιτήματα πάνω σε ένα χαμηλού επιπέδου σχήμα περιγραφής πόρων. Η μηχανή μετασχηματισμού μας είναι εκτατή, δεδομένου ότι χρησιμοποιεί μια τεχνολογική βάση γνώσεων για να καθοδηγήσει το μετασχηματισμό.

Η δομή της αναφοράς έχει ως εξής: στο 1$^{ο}$ κεφάλαιο εξετάζουμε γενικά τα συστήματα Grid, εφαρμογές πάνω σε αυτά καθώς και το τι πρόκειται να υλοποιηθεί στο μέλλον. Στο 2$^{ο}$ κεφάλαιο μελετάμε πιο αναλυτικά ένα σύστημα Grid από πλευράς υλοποίησης με αναφορά στο Globus Toolkit [7] και αναλύουμε τα λειτουργικά κομμάτια από τα οποία αποτελείται. Στο επόμενο κεφάλαιο αναφέρουμε τη σχετικά πρόσφατη βιβλιογραφία πάνω στο πεδίο της αναζήτησης πόρων σε συστήματα Grid. Στη συνέχεια το 4$^{ο}$ κεφάλαιο περιέχει μια ανάλυση για το GLUE Schema [17], μια προσπάθεια για τη δημιουργία ενός στάνταρ για τη περιγραφή πόρων σε Grid συστήματα, τέλος ακολουθεί μια ανάλυση της υλοποίησής μας.

# Κεφάλαιο 1, Grid Systems

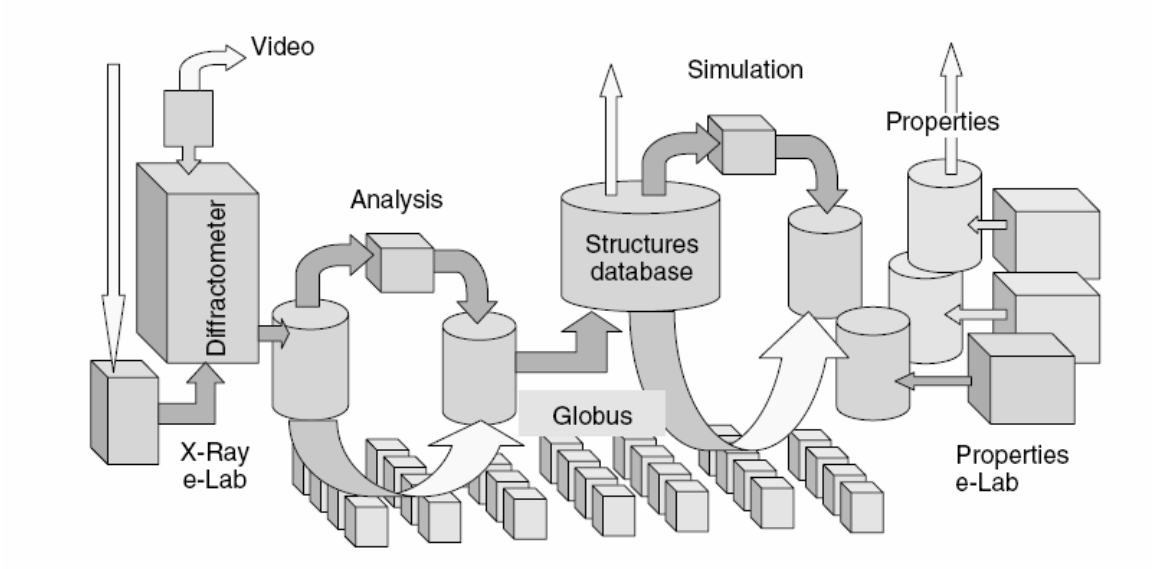
Το Grid είναι μια υποδομή για τον υπολογισμό και τη διαχείριση δεδομένων που θα παράσχει την ηλεκτρονική ενίσχυση για την υποστήριξη μιας 'παγκόσμιας κοινωνίας' στην επιχείρηση, την κυβέρνηση, την έρευνα, την επιστήμη και την ψυχαγωγία. Ενσωματώνει τη δικτύωση, την επικοινωνία, τον υπολογισμό και την πληροφορία για να παρέχει μια εικονική πλατφόρμα για τον υπολογισμό και τη διαχείριση δεδομένων με τον ίδιο τρόπο που το Διαδίκτυο ενσωματώνει τους πόρους (resources) για να διαμορφώσει μια εικονική πλατφόρμα για την πληροφορία.

Τα μεγάλης κλίμακας δικτυώματα είναι πραγματικά κατανεμημένα, ετερογενή και δυναμικά συστήματα που υπόσχονται αποτελεσματικά άπειρους υπολογιστικούς κύκλους και αποθήκευση, καθώς επίσης και πρόσβαση σε όργανα, συσκευές απεικόνισης, κ.ο.κ. αδιαφορώντας για τη γεωγραφική τους θέση.

Η πραγματικότητα είναι ότι για να επιτύχουν αυτήν την υπόσχεση, πρέπει να αναπτυχθούν σύνθετα συστήματα λογισμικού και υπηρεσίες, τα οποία επιτρέπουν την πρόσβαση με έναν φιλικό προς το χρήστη τρόπο, επιτρέπουν αποτελεσματική συνένωση των πόρων, και επιβάλλουν τις πολιτικές που επιτρέπουν στις κοινότητες των χρηστών να συντονίζουν πόρους με ένα σταθερό και αποδοτικό τρόπο. Είτε οι χρήστες χρησιμοποιούν ένα πόρο (έναν ενιαίο υπολογιστή, ένα αρχείο δεδομένων, κ.λπ.), είτε χρησιμοποιούν διάφορους πόρους στο σύνολο ως ένα συντονισμένο 'εικονικό υπολογιστή', το Grid δίνει ένα ενιαίο και ομοιόμορφο τρόπο στους χρήστες για να αλληλεπιδράσουν με τους πόρους και παρέχει μια περιεκτική και ισχυρή πλατφόρμα για σφαιρικό υπολογισμό (global computing) και διαχείριση δεδομένων.

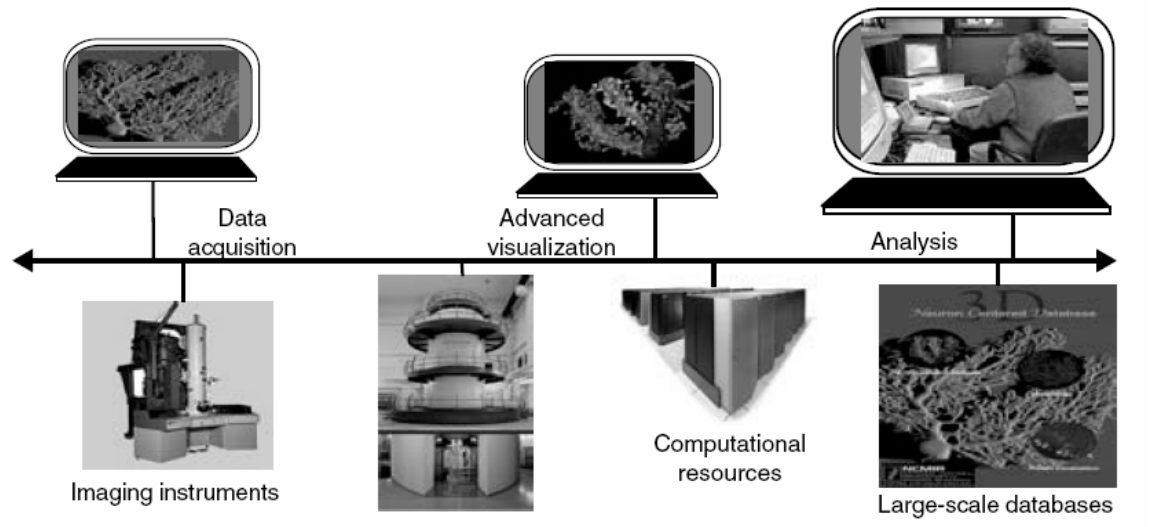
## 1.1 Εφαρμογές πάνω σε συστήματα Δικτυωμάτων

Σε αυτή την ενότητα θα δούμε μερικά παραδείγματα για την ευρεία χρήση των Grid συστημάτων σε αρκετά χρήσιμες εφαρμογές.



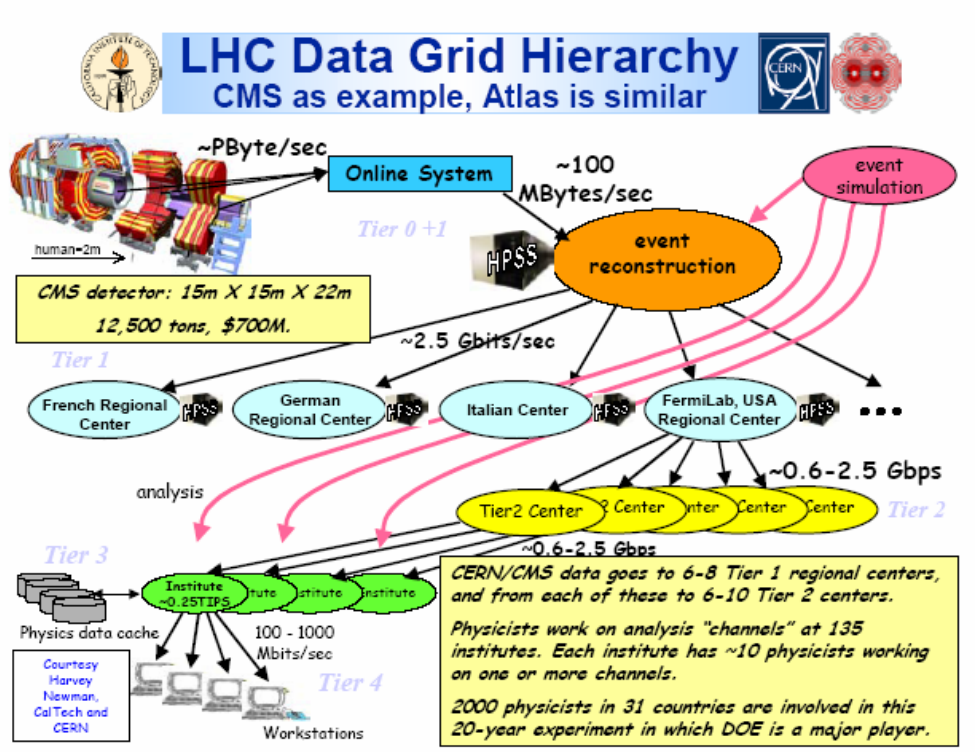

**Εικόνα 1-1**, Σύστημα Grid συνδυαστικής χημείας

Στην εικόνα 1-1 έχουμε ένα παράδειγμα εφαρμογής Grid σε συνδυαστική χημεία. Τα πειράματα τρέχουν παράλληλα σε ένα 'έξυπνο εργαστήριο' που χρησιμοποιεί ένα μικρής έκτασης Grid για να υπολογίζει αποτελέσματα σε πραγματικό χρόνο.




**Εικόνα 1-2,** Telescience application

Στην εικόνα 1-2 βλέπουμε τον πως οι διάφοροι πόροι συνδυάζονται για μια νευρο-χειρουργική Τήλε-ιατρική εφαρμογή (http://www.npaci.edu/Alpha/telescience.html).




**Εικόνα 1-3,** High Energy Physics Data Analysis

Στην εικόνα 1-3 βλέπουμε τη χρήση του Grid στην επιστήμη, που τίθεται αναγκαία για να κάνει δυνατή την ανακάλυψη και την έρευνα νέων κατευθύνσεων.

Για την κατασκευή ενός συστήματος Grid από την πλευρά του υλικού δεν υπάρχει κάποιος χρυσός κανόνας που πρέπει να ακολουθήσει κάποιος - υπόκειται καθαρά στον τύπο της εφαρμογής. Μεγάλο ρόλο πάντως παίζει η τοπολογία του δικτύου (bandwidth, latency) και ανάλογα με την εφαρμογή (υπολογιστική, δεδομένων) οι επιμέρους κόμβοι μπορεί να είναι υπερ-υπολογιστές, clusters ή και συμβατικοί υπολογιστές (desktop, laptop, PDA's στο μέλλον).

## 1.3 Το μέλλον των Grid Συστημάτων

Κατά πολλούς τρόπους, η έρευνα και η ανάπτυξη μεγάλης κλίμακας Grid συστημάτων μόλις αρχίζει. Αναμένονται μεγάλες αλλαγές στις εφαρμογές και στην τεχνολογία του Grid στην επόμενη δεκαετία, κατά την οποία θα εξελιχθούν οι υπάρχουσες τεχνολογίες και θα ενσωματωθούν καινούργιες. Ακόμη και συσκευές όπως PDA's και αισθητήρες θα μπορούν να συνδεθούν στο Grid, δίνοντας τη δυνατότητα Petabytes δεδομένων και petaflops υπολογιστικών πόρων να αποτελούν ένα Grid σύστημα με - άνευ προηγουμένου - ετερογένεια και διακύμανση απόδοσης συσκευών. Την επόμενη δεκαετία το λογισμικό από το οποίο αποτελούνται τα Grid θα γίνει πιο εκλεπτυσμένο, υποστηρίζοντας μεγάλη ανομοιογένεια, ευελιξία και προσαρμοστηκότητα. Οι εφαρμογές θα χρησιμοποιούν τα Grid με περίπλοκους τρόπους, προσαρμοσμένους στις δυναμικές διαμορφώσεις των πόρων και τις παραλλαγές απόδοσης για να επιτύχουν τον στόχο του αυτόνομου υπολογισμού.

Για να που ολοκληρωθούν αυτά τα τεχνικά και πειθαρχικά επιτεύγματα απαιτείται μια απέραντη προσπάθεια έρευνας, ανάπτυξης και επέκτασης από την κοινότητα. Σήμερα, πολλές ομάδες κοιτάζουν πέρα από τις προκλήσεις της ανάπτυξης των σημερινών πλεγμάτων, στις προκλήσεις έρευνας και ανάπτυξης του μέλλοντος. Μια τέτοια προσπάθεια είναι η ανάπτυξη του GLUE Schema [17] , που προβλεπει στην προτυποποίηση του τρόπου περιγραφής πόρων, θα την εξετάσουμε αναλυτικά στο κεφάλαιο 4. προσπάθεια επίσης καταβάλλεται και από το Global Grid Forum (GGF) [4] μια ακόμη ομάδα η οποία αναπτύσσει 2 πρότυπα για την ανάπτυξη Grid συστημάτων, το Open Grid Services Architecture (OGSA) [5] και το Open Grid Services Interface (OGSI) [6].

Το OGSA είναι ένα εξελισσόμενο πρότυπο για το οποίο υπάρχει μεγάλη βιομηχανική υποστήριξη. Ουσιαστικά αλλάζει το πρότυπο προγραμματισμού έτσι ώστε να υποστηρίζει την έννοια των εφαρμογών που διατίθενται ως Web Services. Αυτό παράγει πολλαπλά οφέλη, που περιλαμβάνουν:

- Ένα κοινό και ανοικτό πρότυπο με το οποίο μπορούν να προσεγγιστούν οι διάφορες υπηρεσίες Grid που χρησιμοποιούν επιμέρους πρότυπα όπως το SOAP, XML, κτλ.
- Η δυνατότητα να προστεθούν και να ενσωματωθούν πρόσθετες υπηρεσίες.
- Ένα κοινό τρόπο για να βρεθούν, να προσδιοριστούν, και να χρησιμοποιηθούν οι νέες υπηρεσίες Grid (UDDI) .

Το OGSI είναι ένα πρότυπο που συμπληρώνει το OGSA και καθορίζει τις διεπαφές και τα πρωτόκολλα που θα χρησιμοποιηθούν μεταξύ των ποικίλων υπηρεσιών σε ένα περιβάλλον Grid. Το OGSI είναι το πρότυπο που θα παράσχει τη διαλειτουργικότητα μεταξύ των Grid που σχεδιάζονται χρησιμοποιώντας την αρχιτεκτονική OGSA.

Είναι αναμφισβήτητο ότι ο καθορισμός προτύπων τα οποία είναι κοινά αποδεκτά είναι το κλειδί για την ανάπτυξη και την εξέλιξη των Grid συστημάτων. Τα OGSA – OGSI και το GLUE Schema είναι το πρώτο βήμα για την προτυποποίηση των λειτουργιών του Grid, σίγουρα αναμένονται εξελίξεις των τελευταίων και καθορισμός καινούργιων καθώς οι απαιτήσεις του μέλλοντος το επιβάλλουν.

# Κεφάλαιο 2, GLOBUS toolkit

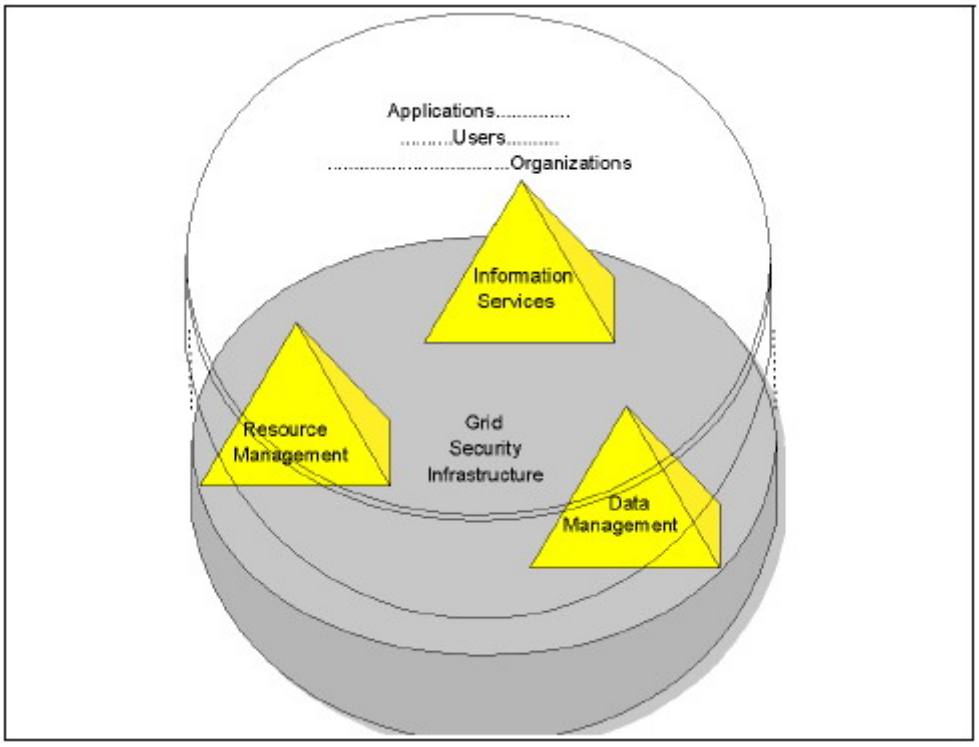
Το Globus Project [7] είναι μια συνδυασμένη προσπάθεια από ερευνητές και προγραμματιστές από όλο τον κόσμο εστιασμένη στα υπολογιστικά Grid και είναι οργανωμένο σε τέσσερις δραστηριότητες: στην έρευνα, στην ανάπτυξη λογισμικού, σε πραγματικές δοκιμές και σε εφαρμογές.

Με στενή συνεργασία με πραγματικά προγράμματα Grid στην επιστημονική έρευνα και τη βιομηχανία, αναπτύσσει και προωθεί τα τυποποιημένα πρωτόκολλα Grid (GGF [4]). Για να επιτρέψει τη δια-λειτουργικότητα και την κοινή υποδομή, αναπτύσσει και προωθεί τυποποιημένο λογισμικό (APIs) και 'Grid SDKs' για να επιτρέψει τη διανομή και τη φορητότητα του κώδικα. Επιπλέον προσφέρεται ένα αρθρωτό "σύνολο τεχνολογιών" και επιτρέπει *την επαυξητική* ανάπτυξη των εργαλείων και των εφαρμογών πάνω σε Grid. Με όλα τα παραπάνω αποτελεί μια ανοικτή πηγή - βάση λογισμικού ως αναφορά για την οικοδόμηση της υποδομής και των εφαρμογών του Grid.

Στα πλαίσια αυτής της διπλωματικής θα αναφερθούμε στο περιβάλλον του Globus toolkit και στα βασικά κομμάτια από τα οποία αυτό αποτελείται. Ουσιαστικά αποτελείται από την υποδομή της ασφάλειας και 3 βασικές υπηρεσίες:

- **Υποδομή Ασφάλειας** (Grid Security Infrastructure), παρέχει συναρτήσεις ασφάλειας όπως ατομική ή αμοιβαία πιστοποίηση, εμπιστευτική επικοινωνία, και εξουσιοδότηση
- **Υπηρεσίες Διαχείρισης Δεδομένων** (Data Management Services), προσφέρουν υποστήριξη για μεταφορά αρχείων σε μηχανές ανάμεσα στο Grid και τη διαχείριση των μεταφορών αυτών.
- **Υπηρεσίες Διαχείρισης Πόρων** (Resource Management Services), υποστηρίζουν κατανομή πόρων, παράδοση διεργασίας (σε απομακρυσμένο μηχάνημα και λήψη αποτελεσμάτων) και διαχείριση της διεργασίας αυτής.
- **Υπηρεσίες Πληροφορίας** (Information Services), προσφέρουν υπηρεσίες για τη συλλογή πληροφοριών πάνω στο Grid και ερωτήσεις πάνω στις υπηρεσίες αυτές.

Η υποδομή της ασφάλειας, δίνει τη βάση πάνω στην οποία ‘χτίζονται’ οι υπόλοιπες υπηρεσίες, βλέπε εικόνα 2-1. Στις παρακάτω υποενότητες οι παραπάνω υπηρεσίες αναλύονται ξεχωριστά., με περισσότερη έμφαση στις Υπηρεσίες Πληροφορίας.




**Εικόνα 2–1**, υπηρεσίες του Globus

## 2.1 Υπηρεσίες Ασφάλειας

Το Globus toolkit χρησιμοποιεί την υποδομή ασφάλειας Grid (GSI) για να κάνει δυνατή την ασφαλή πιστοποίηση ταυτότητας και της επικοινωνίας πάνω από ένα ανοικτό δίκτυο. Παρέχει επίσης διάφορες χρήσιμες υπηρεσίες, συμπεριλαμβανομένης της αμοιβαίας πιστοποίησης ταυτότητας και ενιαίο (single) sign-on. Το GSI βασίζεται στη κρυπτογράφηση με δημόσιο κλειδί, X.509 πιστοποιητικά, και το ασφαλές πρωτόκολλο επικοινωνίας SSL. Επίσης επεκτάσεις σε αυτά τα πρότυπα έχουν προστεθεί για την υποστήριξη επιπλέον λειτουργικότητας.

Τα αρχικά κίνητρα πίσω από το GSI είναι:

- Η ανάγκη για την ασφαλή επικοινωνία μεταξύ των στοιχείων ενός υπολογιστικού Grid.
- Η ανάγκη να υποστηριχθεί η ασφάλεια μέχρι τα όρια της εφαρμογής, απαγορεύοντας ένα κεντρικά-ρυθμισμένο σύστημα ασφάλειας.
- Η ανάγκη να υποστηριχθεί "ενιαίο sign-on" για τους χρήστες του Grid, συμπεριλαμβανομένου delegation of credentials για τους υπολογισμούς που περιλαμβάνουν πολλούς πόρους.

Το βασικό κομμάτι του GSI είναι το CAS, Το οποίο επιτρέπει στους προμηθευτές των πόρων να προσδιορίσουν τις πολιτικές ελέγχου πρόσβασης σε αυτούς γενικά (π.χ. σε ομάδες χρηστών), αλλά και λεπτομερή εξουσιοδότηση - πολιτική διαχείρισης ελέγχου πρόσβασης σε συγκεκριμένους χρήστες. Τέλος, αν και οι προμηθευτές των πόρων διατηρούν την απόλυτη κυριότητα πάνω σε αυτούς, τίθένται οι ίδιοι κάτω από τις καθημερινές πολιτικές διαχείρισης του Grid (π.χ. προσθήκη και διαγραφή των χρηστών, τροποποίηση των προνομίων τους κ.τ.λ.).

## 2.2 Υπηρεσίες Διαχείρισης Δεδομένων

Κατά οικοδόμηση ενός Grid, σημαντικότερο στοιχείο μέσα σε αυτό είναι τα δεδομένα, για τα οποία είναι αναγκαίο να καθορισθούν οι απαιτήσεις τους, το πώς θα ‘κινούνται’ αυτά γύρω από την υποδομή του Grid και τέλος να ορισθεί η πρόσβαση στα απαραίτητα δεδομένα κατά τρόπο ασφαλή και αποδοτικό. Αυτές τις προδιαγραφές έρχεται να καλύψει η υπηρεσία διαχείρισης δεδομένων η οποία αποτελείται από τα παρακάτω κομμάτια :

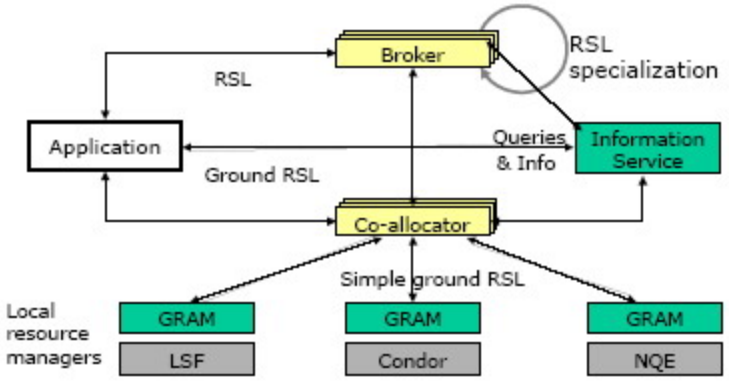
- Το GridFTP, ένα υψηλής απόδοσης, ασφαλές και αξιόπιστο πρωτόκολλο μεταφοράς δεδομένων, βελτιστοποιημένο για high-bandwidth wide-area networks το οποίο είναι βασισμένο στο γνωστό FTP. Έχει επιλεχθεί ένα σύνολο χαρακτηριστικών πρωτοκόλλου και επεκτάσεις οι οποίες καθορίστηκαν σε IETF

RFCs καθώς επίσης προστέθηκαν χαρακτηριστικά για να καλύψουν τις απαιτήσεις από τα τρέχοντα προγράμματα Grid.

- Σφαιρική πρόσβαση στη δευτεροβάθμια αποθήκευση (GASS), που χρησιμοποιείται για να μεταφέρει τα αρχεία μεταξύ του πελάτη GRAM και του κεντρικού εξυπηρετητή GRAM. Το GASS παρέχει επίσης τις βιβλιοθήκες και ευκολίες για το άνοιγμα, το κλείσιμο, και προσκόμιση δεδομένων από τα σύνολα δεδομένων στο περιβάλλον Globus.
- Αξιόπιστη υπηρεσία μεταφοράς αρχείων (RFT) είναι μια υπηρεσία βασισμένη σε OGSA που παρέχει την διεπαφή για την επίβλεψη και τον έλεγχο των μεταφορών αρχείων τρίτων χρησιμοποιώντας τους κεντρικούς εξυπηρετητές GridFTP.
- Υπηρεσία θέσης αντιγράφου (RLS) η οποία διατηρεί και παρέχει την πρόσβαση στις πληροφορίες χαρτογράφησης - αντιστοίχησης από τα λογικά ονόματα για τα αντικείμενα δεδομένων σε target ονόματα. Τα τελευταία μπορούν να αντιπροσωπεύσουν τις φυσικές θέσεις των αντικειμένων, ή μια είσοδο στο RLS που χαρτογραφεί ένα άλλο επίπεδο λογικής ονομασίας.

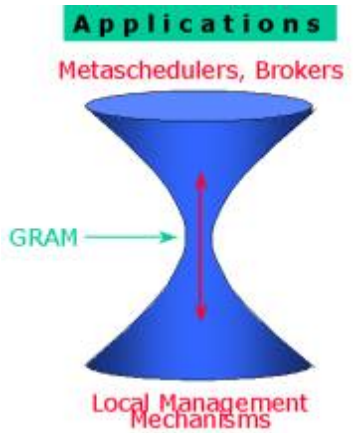
## 2.3 Υπηρεσίες Διαχείρισης Πόρων

Η υπηρεσία αυτή, γνωστή και ως GRAM (*Globus Resource Allocation Manager*) παρέχει μια ασφαλή και ελεγχόμενη απομακρυσμένη πρόσβαση σε ετερογενής πόρους, καθώς και την διαχείριση απομακρυσμένων υπολογισμών. Το GRAM απλοποιεί τη χρήση των απομακρυσμένων συστημάτων, με την παροχή μιας ενιαίας, ομοιόμορφης και ευέλικτης διεπαφής για την αίτηση και τη χρησιμοποίηση απομακρυσμένων πόρων για την εκτέλεση εργασιών. Η πιο κοινή χρήση (και η καλύτερα υποστηριγμένη) του GRAM είναι απομακρυσμένη υποβολή και έλεγχος εργασιών και χρησιμοποιείται για να υποστηρίξει τις διανεμημένες εφαρμογές υπολογισμού.



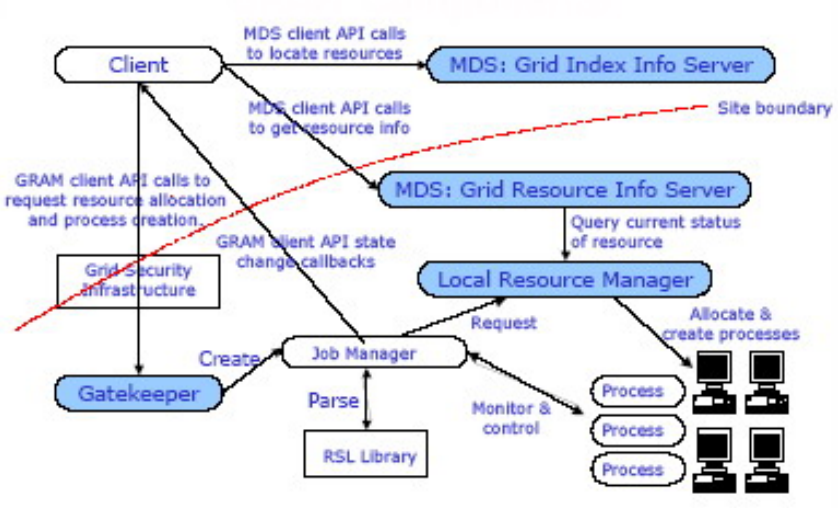

**Εικόνα 2-2**, Resource Management Architecture

Το GRAM χρησιμοποιεί την υποδομή ασφάλειας Grid (GSI) ώστε να παρέχει την αμοιβαία πιστοποίηση ταυτότητας των χρηστών και των απομακρυσμένων πόρων. Ο στόχος του είναι να μειώσει τον αριθμό μηχανισμών που απαιτούνται για τη χρησιμοποίηση των απομακρυσμένων πόρων (όπως χρονοπρογραμματισμός, συστήματα αναμονής, συστήματα reservation, και διεπαφές ελέγχου). Αυτή η ικανότητα παρομοιάζεται (όπως και πολλά άλλα κομμάτια του Globus) με το λαιμό της κλεψύδρας, με τις εφαρμογές και τις υψηλότερου επιπέδου υπηρεσίες (όπως brokers, metaschedulers) επάνω και τους τοπικούς μηχανισμούς ελέγχου και πρόσβασης από κάτω, όπως φαίνεται στην εικόνα 2-3. Και οι δύο πλευρές εργάζονται μόνο με το GRAM, έτσι ο αριθμός των αλληλεπιδράσεων, τα APIs και τα πρωτόκολλα που πρέπει να χρησιμοποιηθούν μειώνονται πολύ.




**Εικόνα 2-3**,

Τέλος η RSL (Resource Specification Language) μια γλώσσα που παρέχει το GRAM, δίνει τη σύνταξη για την ανταλλαγή πληροφορίας μεταξύ των επιμέρους κομματιών του. Η πληροφορία που ανταλλάσσεται μπορεί να είναι είτε για περιγραφή προδιαγραφών πόρων, είτε για τη διευθέτηση (configuration) μιας εργασίας (εκτελέσιμο, ορίσματα, κατάλογος, μεταβλητές περιβάλλοντος, ...). Επίσης το Globus παρέχει και συναρτήσεις για τη διαχείριση της RSL.




**Εικόνα 2-4**, GRAM Components

Στην εικόνα 2-4 φαίνεται η σχηματική αναπαράσταση και οι αλληλεπιδράσεις των επιμέρους υπηρεσιών του Globus με το GRAM.

## 2.4 Υπηρεσίες Πληροφορίας

Οι υπηρεσίες πληροφορίας είναι ένα ζωτικής σημασίας κομμάτι της υποδομής του Grid. Διατηρεί τη γνώση για τη διαθεσιμότητα, την ικανότητα καθώς και την τρέχουσα χρησιμοποίηση των πόρων. Μέσα σε οποιοδήποτε Grid, τόσο οι πόροι δεδομένων όσο και οι πόροι CPU θα κυμανθούν, ανάλογα με τη διαθεσιμότητά τους για να επεξεργαστούν και να μοιραστούν δεδομένα, καθώς αυτοί οι πόροι δεσμεύονται και απελευθερώνονται, ανανεώνουν τη διαθεσιμότητά τους στις υπηρεσίες πληροφορίας έτσι ώστε ο πελάτης, ο 'μεσίτης' (broker) ή ο διαχειριστής των πόρων του Grid να

χρησιμοποιήσει αυτή την πληροφορία για να λάβει ενημερωμένες αποφάσεις σχετικά με τις αναθέσεις των πόρων. Η υπηρεσία πληροφορίας έχει ως σκοπό να παρέχει:

- Αποδοτική παράδοση της πληροφορίας κατάστασης από μια ατομική πηγή
- Κοινούς μηχανισμούς έρευνας και ανακάλυψης πόρων σε όλες τις οντότητες Grid

Οι προμηθευτές υπηρεσιών πληροφορίας είναι προγράμματα που παρέχουν τις πληροφορίες στον κατάλογο για την κατάσταση των πόρων. Παραδείγματα της πληροφορίας που συγκεντρώνονται :

- Στατική πληροφορία του Host (όπως για το λειτουργικό σύστημα όνομα και έκδοση, για τον επεξεργαστή κατασκευαστή, μοντέλο, ταχύτητα, cache , αριθμός επεξεργαστών, για τη μνήμη τη συνολική φυσική και εικονική, για συσκευές, για υπηρεσίες όπως τύπο, πρωτόκολλο, port κ.ο.κ.
- Δυναμική πληροφορία για το φόρτο εργασίας του κόμβου, τις καταχωρήσεις στη σειρά αναμονής, κ.ο.κ.
- Πληροφορία για σύστημα αποθήκευσης, όπως συνολικός αποθηκευτικός χώρος, ελεύθερος χώρος, κ.ο.κ.
- Πληροφορία δικτύου, όπως bandwidth , λανθάνουσα κατάσταση (latency), μετρημένη και προβλεφθείσα κ.ο.κ.
- Ιδιαίτερα Δυναμική πληροφορία για την διαθέσιμη ελεύθερη φυσική και εικονική μνήμη, ελεύθερος αριθμός επεξεργαστών, κ.ο.κ.

Η υπηρεσία πληροφοριών (GIS, *Grid Information Service*), επίσης γνωστή και ως υπηρεσία ελέγχου και εύρεσης (MDS, *Monitoring and Discovery Service*), παρέχει τις υπηρεσίες πληροφορίας στο Globus. Το MDS χρησιμοποιεί το ελαφρύ πρωτόκολλο πρόσβασης καταλόγου (LDAP) ως διεπαφή για την πληροφορία των πόρων (από την πρόσφατη έκδοση 3.0, λόγω της αναφοράς στο OGSA - OGSI η πληροφορία είναι αποθηκευμένη σε XML και οι ερωτήσεις γίνονται σε XPath).

Το MDS παρέχει την πρόσβαση στις στατικές και δυναμικές πληροφορίες των πόρων. Βασικά, περιέχει τα ακόλουθα συστατικά:

- Υπηρεσία πληροφοριών των πόρων Grid (GRIS, *Grid Resource Information Service*)
- Υπηρεσία πληροφοριών δεικτών Grid (GIIS, *Grid Index Information Service*)
- Προμηθευτές πληροφοριών
- Πελάτη MDS

Η εικόνα 2-5 αντιπροσωπεύει μια εννοιολογική άποψη των τμημάτων του MDS. Όπως εικονογραφείται, οι πληροφορίες των πόρων λαμβάνονται από τον προμηθευτή πληροφοριών και περνούν στο GRIS. Το GRIS καταχωρεί τις τοπικές πληροφορίες του στο GIIS, το οποίο μπορεί προαιρετικά επίσης να καταχωρηθεί σε ένα άλλο GIIS, κ.ο.κ. . Οι πελάτες MDS μπορούν να ρωτήσουν τις πληροφορίες των πόρων άμεσα στο GRIS (για τους τοπικούς πόρους) ή / και σε ένα GIIS (για τους 'ευρείς' πόρους του Grid).

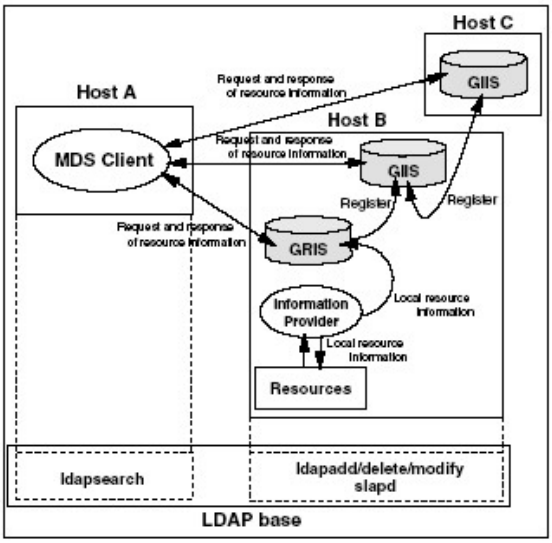
### 2.4.1 Υπηρεσία πληροφοριών των πόρων του Grid (GRIS)

Το GRIS είναι η αποθήκη των τοπικών πληροφοριών των πόρων που προέρχονται από τους προμηθευτές πληροφορίας. Το GRIS είναι σε θέση να καταχωρήσει τις πληροφορίες σε ένα GIIS, αλλά το ίδιο το GRIS δεν μπορεί να καταχωρήσει κάποιο GIIS ή άλλο GRIS. Οι τοπικές πληροφορίες που διατηρούνται από το GRIS ενημερώνονται όταν ζητείται, και εναποθηκεύονται για μια χρονική περίοδο γνωστή ως time-to-live (TTL). Εάν κανένα αίτημα για αυτές δεν παραληφθεί από το GRIS, οι πληροφορίες που θα ξεπεράσουν το TTL θα διαγραφούν. Τέλος εάν ένα πιο πρόσφατο αίτημα παραληφθεί, ο GRIS θα καλέσει τον σχετικό προμηθευτή πληροφορίας για να ανακτήσει την πιο πρόσφατη.

### 2.4.2 Υπηρεσία πληροφοριών δεικτών Grid (GIIS)

Το GIIS είναι η αποθήκη που περιέχει τους δείκτες για την πληροφορία των πόρων που καταχωρούνται από το GRIS ή και άλλα GIISs. Θα μπορούσαμε να τον δούμε ως κεντρικό εξυπηρετητή πληροφορίας. Το GIIS έχει έναν ιεραρχικό μηχανισμό, αντίστοιχο με αυτόν του DNS, και κάθε GIIS έχει δικό του όνομα. Αυτό σημαίνει οι χρήστες ότι

πελατών μπορούν να διευκρινίσουν το όνομα ενός κόμβου GIIS για την αναζήτηση πληροφορίας.




**Εικόνα 2-5**, σχηματική απεικόνιση του MDS

### 2.4.3 Προμηθευτές πληροφοριών

Οι προμηθευτές πληροφοριών μεταφράζουν τις ιδιότητες και την κατάσταση των τοπικών πόρων στο σχήμα που καθορίζεται από το GRIS. Προκειμένου να προσθέσει κάποιος δικούς του πόρους που χρησιμοποιούνται από το MDS, πρέπει να δημιουργήσει συγκεκριμένους προμηθευτές πληροφορίας για να μεταφέρει τις ιδιότητες και τη κατάσταση τους στο GRIS.

### 2.4.4 Πελάτης MDS

Ο πελάτης MDS είναι βασισμένος στο μοντέλο LDAP και στην εντολή *ldapsearch* , ή ένα ισοδύναμο API. Μια αναζήτηση πληροφορίας για τους πόρους στο περιβάλλον Grid εκτελείται αρχικά από τον πελάτη MDS.

# Κεφάλαιο 3, Έρευνα που έχει γίνει στην περιοχή της αναζήτησης πόρων σε Grid συστήματα

Το πρόβλημα της εύρεσης και διαχείρισης πόρων είναι ένα από τα πιο βασικά προβλήματα των Grid συστημάτων λόγω της ετερογένειας και δυναμικότητας που τα χαρακτηρίζει, το οποίο πολλοί ερευνητές προσπάθησαν να λύσουν. Σε αυτό το κεφάλαιο θα εξετάσουμε μερικές από αυτές τις προτάσεις από τη σχετικά πρόσφατη βιβλιογραφία.

## 3.1 Σχεδιασμός και αξιολόγηση δομής επιλογής πόρων σε εφαρμογές Grid

Το 2002 στο [12] πρότειναν μια απλή γλώσσα βασισμένη σε μια τεχνική για matching η οποία ονομάζεται Set Matching και επεκτείνει την τεχνική που χρησιμοποιείται στο Condor [11]. Η τεχνική αυτή υποστηρίζει την επιλογή τόσο ενός πόρου όσο και πολλαπλών, καθώς επίσης και την δυνατότητα να χρησιμοποιηθούν υπομονάδες αντιστοίχησης (mapping modules) ξεχωριστά για κάθε εφαρμογή.

Η γλώσσα που χρησιμοποιείται ονομάζεται Set-Extended ClassAd και επεκτείνει την απλή ClassAd γλώσσα ενσωματώνοντας aggregate συναρτήσεις (Max, Min, Sum), συναρτήσεις όπως η Suffix, SetSize και μπορεί να χρησιμοποιήσει τις συναρτήσεις αυτές μέσα στις εκφράσεις που δείχνουν τους περιορισμούς,

π.χ. Sum(other.Memory)>5GB, Rank=1/Max(execution-time).

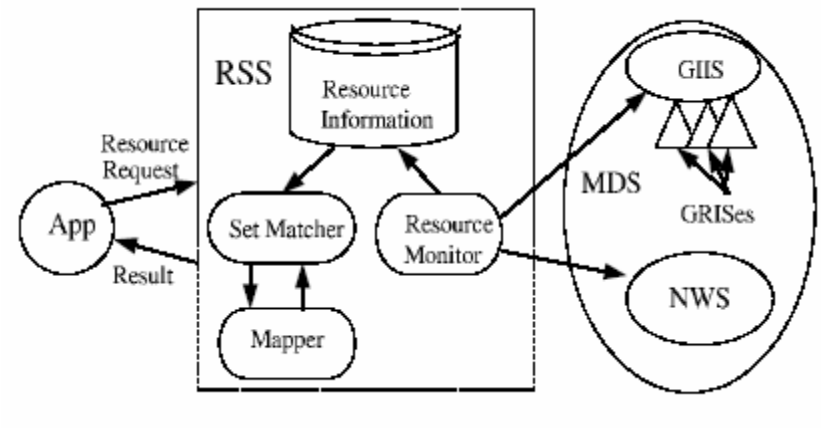
Η δομή της ερώτησης μπορεί να έχει τα παρακάτω στοιχεία:

- Είδος ClassAd, αν είναι Set-Extended ή απλό.
- Είδος υπηρεσίας, σύγχρονη ή ασύγχρονη. Με τη σύγχρονη δηλώνουμε ότι απαιτούμε πόρους άμεσα, ενώ με την ασύγχρονη ότι θέλουμε πόρους που θα γίνουν διαθέσιμοι έως μια χρονική περίοδο.
- Περιγραφή εργασίας, χαρακτηριστικά της εργασίας / μοντέλο απόδοσης

- Το πρόγραμμα Mapper που θα χρησιμοποιήσουμε, αναφέρεται παρακάτω.
- Περιορισμούς, του χρήστη για τους πόρους.
- Κατάταξη (Rank), θέτουμε κριτήρια για την κατάταξη των πόρων.

Στην εικόνα 3-1 βλέπουμε την αρχιτεκτονική του επιλογέα πόρων Πιο αναλυτικά, η υπηρεσία επιλογής πόρων (Resource Selection Service, RSS) όπως φαίνεται, χρησιμοποιεί το MDS [ενότητα 2.4] για να ανανεώνει την πληροφορία των πόρων και την υπηρεσία κατάστασης δικτύου (Network Weather Service, NWS) - ένα κατανεμημένο σύστημα που περιοδικά ελέγχει και δυναμικά προβλέπει την απόδοση δικτυακών και υπολογιστικών πόρων. Αποτελείται από 3 βασικές μονάδες

- Ελεγκτής Πόρων (Resource Monitor), ο οποίος λειτουργεί σαν ένα GRIS [2.4.1] και είναι υπεύθυνος να ρωτά το MDS για να ανανεώνει και να κρατά σε προσωρινή μνήμη (cache) την πληροφορία.
- Set Matcher, χρησιμοποιεί τον αλγόριθμο set-matching για να ταιριάξει αιτήσεις εργασιών με τους καλύτερους δυνατούς πόρους (χρησιμοποιεί Greedy αλγόριθμο γιατί το πρόβλημα είναι NP-Complete).
- Αντιστοιχιστής (Mapper), επειδή πολλές φορές η τοπολογία των πόρων και οι υπολογιστικές απαιτήσεις συνδέονται άμεσα με τον χρόνο εκτέλεσης της εργασίας, η δουλεία του mapper είναι να αποφασίσει την τοπολογία και να δεσμεύσει τον φόρτο εργασίας (workload) των πόρων.




**Εικόνα 3-1**, Αρχιτεκτονική του επιλογέα πόρων

Επειδή η δουλειά του Mapper είναι στενά συνδεδεμένη με την εφαρμογή, είναι δύσκολο να γίνει γενικευμένη, έτσι αφήνεται στον χρήστη να ορίσει συναρτήσεις κόστους ως βιβλιοθήκες οι οποίες συνδέονται δυναμικά με τον Mapper.

Μια ενδεικτική ερώτηση θα μπορούσε να είναι η εξής:

```
[
ServiceType="Synchronous";
Type="Set";
iter=100;alpha=0;x=100;y=100;z=100;
A=370; B=254; startup=30; C=0.0000138;
comtime=(other.RLatency+y*x*B/other.RBandwidth
        +other.LLatency+y*x*B/other.Lbandwidth);
exectime=(computime_comtime)*iter+startup;
Mapper=[type="dll";libname="cactus"; func="mapper"];
requirements=Suffix(other.machine,domains)
        && Sum(other.MemorySize)>=(1.757+C*z*x*y);
domains={cs.utk.edu,ucsd.edu};
rank=Min(1/exectime)
]
```

## 3.2 Διαχείριση και κατανόηση κατανεμημένων πολιτικών με χρήση με Classified Advertisements

Το 2003 οι [14] προτείνουν αλγόριθμους οι οποίοι επεκτείνουν το ClassAds matchmaking στο κατανεμημένο σύστημα Condor [11] για εύρεση πόρων. Οι αλγόριθμοι αυτοί μπορούν να κατανοήσουν για ποιο λόγο κάποιες ερωτήσεις δεν επιστρέφουν αποτελέσματα ή είναι από μόνες τους αντιφατικές, έτσι προτείνουν τι μπορεί να αλλαχθεί στην ερώτηση αυτή, ώστε να επιστραφούν αποτελέσματα.

Πρέπει να αναφέρθεί πως στο σύστημα Condor [11] εκτός από τους περιορισμούς που θέτει ο χρήστης για μια εργασία (CPU, Memory, Os, …) και οι πόροι μπορεί να έχουν από μόνοι τους περιορισμούς (π.χ. ένας κόμβος δέχεται εκτελέσιμα μικρού μεγέθους από

τις 9π.μ. και 5 μ.μ., δεν δέχεται εργασίες του χρήστη ‘Χ’, εργασίες γίνονται δεκτές μόνο αν έχει μικρό workload, … ) τους οποίους πρέπει να λάβουν υπόψιν οι παραπάνω αλγόριθμοι. Έτσι μπορεί να συμβαίνουν 2 περιπτώσεις όταν δεν επιστρέφονται αποτελέσματα (matches):

- Κανένας κόμβος δεν ταιριάζει με τις απαιτήσεις του χρήστη.
- Οι περιορισμοί όλων των κόμβων απορρίπτουν της εργασία.

Στην $1^{η}$ περίπτωση πρέπει αρχικά να προσδιοριστεί το κατηγόρημα ή ο συνδυασμός των κατηγορημάτων που δημιουργούν το πρόβλημα. Στη συνέχεια προτείνονται αλλαγές στις απαιτήσεις του χρήστη, μπορεί επιπλέον να εντοπιστούν συγκρούσεις, όταν 2 ή περισσότερα κατηγορήματα δεν μπορούν να συνυπάρξουν στην ίδια ερώτηση. Ακόμη και στην περίπτωση που από την λογική των απαιτήσεων, δεν μπορούν να τροποποιηθούν οι τελευταίες, χρήσιμα συμπεράσματα μπορούν να προκύψουν από την ανάλυση.

Στη $2^{η}$ περίπτωση προσπαθούν να βρουν το πιο κοντινό πεδίο τιμών σε αυτό της εργασίας που ζητείται, μπορεί να προσδιορισθούν επίσης και πόσα μηχανήματα θα επιστρεφόταν για μια εργασία με τιμές στο παραπάνω πεδίο. Προτείνουν επίσης εναλλακτικά πεδία τιμών και την απόσταση που έχουν από το τρέχων (με κάποιες μετρικές) και τα μηχανήματα που θα επέστρεφαν.

Και στις δυο περιπτώσεις ο υπολογισμός γίνεται γεωμετρικά, με την αναπαράσταση των ερωτήσεων σε Ν-διάστατο χώρο όπου κάθε διάσταση αναπαριστά μια μεταβλητή, και οι προτάσεις αναπαριστώνται στο χώρο αυτό από υπερ-τετράγωνα (hyper rectangles). Με διάφορες μετρικές βρίσκονται οι κοντινότερες τιμές στο υπερ-τετράγωνο, προσπαθούν να το επεκτείνουν για να τις περιλάβει και διάφορες άλλες μεθόδους που δεν ανήκουν στο αντικείμενο της εργασίας αυτής.

Παραθέτουμε 2 παραδείγματα για το πως δουλεύουν οι αλγόριθμοι αυτοί :

**Ερώτηση:**

```
(other.Foo == "bar") &&
(other.Memory > 2096) &&
(other.Arch == "INTEL") &&
(other.OpSys == "LINUX") &&
(other.Disk >= 14)
```

**Απάντηση:**

The results of `condor q -analyze` are:

```
Run analysis summary. Of 839
machines,
839 were rejected by the job's
requirements
No successful match recorded.
Predicate Matches Suggestion
--------- ------- ----------
1 Foo == "bar"     0 REMOVE
2 Memory > 2096    0 MODIFY TO 1911
3 OpSys == "LINUX" 616
4 Arch == "INTEL"  740
5 Disk >= 14       835
```

**Ερώτηση:**

```
(other.OpSys == "SOLARIS") &&
(other.Arch == "ALPHA") &&
(other.Disk >= DiskUsage)
```

**Απάντηση:**

The results of `condor q -analyze` are:

```
Run analysis summary. Of 839
machines,
839 were rejected by the job's
requirements
No successful match recorded.
Predicate Matches    Suggestion
--------- -------    ----------
1 Arch == "ALPHA"     4 MODIFY TO
                       "INTEL"
2 OpSys == "SOLARIS"  120
3 Disk >= 14          838
Conflicts:
predicates: 1, 2
```

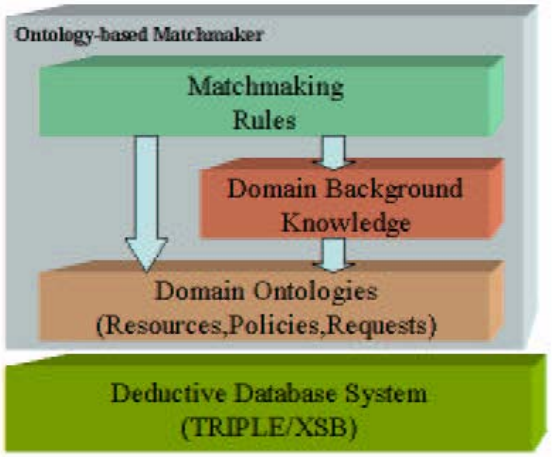
## 3.3 Ταίριασμα πόρων (Resource matching) με χρήση οντολογιών

Το 2004 στο [15] προτείνουν μια ευέλικτη και επεκτάσιμη προσέγγιση για τη λύση του resource matching σε συστήματα Grid χρησιμοποιώντας τεχνολογίες του Semantic Web. Σχεδίασαν ένα πρωτότυπο επιλογέα πόρων (resource selector) σε TRIPLE βασισμένο σε οντολογίες (RDF, OWL) το οποίο αξιοποιεί τις οντολογίες αυτές, τη γνώση που του έχουν περάσει στο υπόβαθρο καθώς και κανόνες που έχουν εισάγει για να λύσει το resource matching στο Grid. Προσπαθούν ουσιαστικά να υλοποιήσουν αυτό που οι [13] ονομάζουν 'knowledge-oriented Grid' .

Ο βασισμένος σε οντολογίες matchmaker αποτελείται από τρία βασικά συστατικά:

1. Ο*ντολογίες*,
2. *Γνώση Υποβάθρου*
3. Κ*ανόνες για matchmaking*

Η εικόνα 3-2 παρουσιάζει τη σχέση μεταξύ αυτών των συστατικών. Τα βέλη δείχνουν την εξάρτηση μεταξύ των διαφορετικών συστατικών, π.χ. η γνώση υποβάθρου χρησιμοποιεί το λεξιλόγιο από τις οντολογίες για να συλλάβει τις βασικές πληροφορίες, οι κανόνες αντιστοιχίας χρησιμοποιούν τις οντολογίες και τη γνώση υποβάθρου για να ταιριάξουν τους πόρους με ένα αίτημα. Ο matchmaker έχει χτιστεί πάνω σε TRIPLE/XSB συμπερασματική βάση δεδομένων (deductive database). Παρακάτω θα εξετάσουμε τη λειτουργία κάθε συστατικού.




**Εικόνα 3-χ**, Ontology-based Matchmaker Architecture

**Οντολογίες**. Αναπτύχθηκαν τρεις χρησιμοποιώντας το RDF-Schema και κάθε μια καθορίζει τα αντικείμενα, τις ιδιότητες των αντικειμένων, και τις σχέσεις μεταξύ των αντικειμένων. Είναι οι εξής:

- *Οντολογία των πόρων*. Η οντολογία των πόρων παρέχει ένα αφηρημένο μοντέλο για την περιγραφή των πόρων.
- *Οντολογία αιτημάτων των πόρων*. Αυτή η οντολογία συλλαμβάνει ένα αίτημα, τις ιδιότητες του (*π.χ.,* ιδιοκτήτης αιτήματος) και τα χαρακτηριστικά του (*π.χ.,* JobType="MPI").
- *οντολογία πολιτικών*. Συλλαμβάνει τις πολιτικές έγκρισης και χρήσης των πόρων.

Δεδομένου ότι οι οντολογίες αναπαριστώνται από πρότυπα Semantic Web, μπορούν εύκολα να ανταλλαχθούν και να μοιραστούν από διαφορετικά εργαλεία όπως άλλα συστήματα βασισμένα σε κανόνες ή γνώση

**Η Γνώση Υποβάθρου (Domain Background Knowledge)** περιέχει γνώση η οποία δεν περιέχεται στην οντολογία και χρησιμοποιείται κατά την διαδικασία του matchmaking. Η γνώση αναπαριστάται με μορφή κανόνων οι οποίοι χρησιμοποιούν το λεξιλόγιο των οντολογιών για προσθέσουν επιπρόσθετα αξιώματα που δεν μπορούν να εκφραστούν μόνο από τις τελευταίες. Οι κανόνες αυτοί καθορίζουν ποια λειτουργικά είναι συμβατά μεταξύ τους και το εκφράζουν μεταβατικά και συμμετρικά. Καθορίζουν επίσης υποκατάστατα λειτουργικά, για να αποφασίσουν ποια λειτουργικά μπορούν να αντικατασταθούν από άλλα σε μια αίτηση.

**Οι Κανόνες για Matchmaking** ορίζουν τους περιορισμούς ανάμεσα στους πόρους και τις αιτήσεις. Εκτός από τους απλούς περιορισμούς (αλφαριθμητικούς, ισότητας, ... ) η TRIPLE μπορεί να κρίνει τους περιορισμούς σε επίπεδο αντικειμένων και τις σχέσεις τους, όπως αυτές εκφράζονται σε RDF και γνώση υποβάθρου.

**Η TRIPLE/XSB** σποτελεί μια συμπερασματική βάση δεδομένων (deductive database) η οποία υποστηρίζει την αποθήκευση RDFSchema και την TRIPLE rule language και έχει χτιστεί πάνω από την συμπερασματική βάση XSB. Η TRIPLE/XSB υπολογίζει τους κανόνες για matchmaking και σε συνδυασμό με τη γνώση που έχει στο υπόβαθρο και τις οντολογίες βρίσκει το καλύτερο match για τις απαιτήσει του χρήστη.

Αναφέρουμε τα επιθυμητά αποτελέσματα που προσφέρει μια αρχιτεκτονική σαν και αυτή, βασισμένη στο Semantic Web:

- *Διανομή και Συντηρησιμότητα* Οι οντολογίες είναι εύκολο να διαμοιραστούν, να διατηρηθούν και να γίνουν κατανοητοί σε σχέση με τα επίπεδα σχήματα σχεσιακών βάσεων.
- *Διμερείς Περιορισμοί.* Τόσο η αίτηση, όσο και οι πόροι μπορούν να έχουν περιορισμούς οι οποίοι λαμβάνονται υπόψιν στο Matchmaking.

- *Πολύπλευρο ταίριασμα* Ένας χρήστης μπορεί να υποβάλει ένα αίτημα που απαιτεί πολλαπλούς ταυτόχρονους πόρους, όπου κάθε ένας ταιριάζει με την απαίτησή του.
- *Έλεγχος ακεραιότητας.* Ο matchmaker μπορεί να χρησιμοποιήσει γνώση για να προσδιορίσει τις ασυνέπειες στις περιγραφές των πόρων πριν ακόμη τους δεχθεί ως διαθέσιμους. Έλεγχος ακεραιότητας μπορεί επίσης να γίνει για το αίτημα να σιγουρευτεί ότι δεν υπάρχει καμία σύγκρουση στις απαιτήσεις των πόρων.
- *Εκφραστικότητα.* Λόγω της ασυμμετρικής περιγραφής, το αίτημα μπορεί να διαμορφωθεί συγκεκριμένα για τις εξαρτώμενες από το πεδίο εφαρμογές. Τα υψηλού επιπέδου χαρακτηριστικά εφαρμογής μπορούν να παρασχεθούν από το χρήστη.
- *Ευελιξία και Εκτατότητα,* Νέες έννοιες και περιορισμοί μπορούν να προστεθούν εύκολα στην οντολογία.

Το μόνο μειονέκτημα αυτής υλοποίησης είναι η απόδοση της, η οποία εξαρτάται άμεσα από τον αριθμό των κανόνων και το μέγεθος της βάσης, και καθώς αυτά μεγαλώνουν πέφτει αρκετά. Για την χρησιμοποίησή του Semantic Web σε συστήματα Grid αλλά και γενικότερα, τίθεται αναγκαία η ανάπτυξη νέων και πιο αποδοτικών εργαλείων που θα μπορούν να σταθούν ανάμεσα στα υπάρχοντα σχεσιακά συστήματα. Η τρέχουσα απόδοσή τους βέβαια δεν αποτελεί έκπληξη, καθώς είναι μια πολύ πρόσφατη τεχνολογία που δεν έχει ωριμάσει αρκετά.

# Κεφάλαιο 4, Το σχήμα GLUE ( GLUE schema )

Το GLUE (*Grid Laboratory Uniform Environment*) Schema [17] αναπτύχθηκε σαν μέρος του DataTAG Project [16], που είχε σαν στόχο την δημιουργία μια μεγάλης κλίμακας διηπειρωτική δοκιμή Grid περιλαμβάνοντας το ευρωπαϊκό πρόγραμμα DataGrid [19], το Globus [7] και σχετικά προγράμματα Grid στις ΗΠΑ. Είναι ένα πρόγραμμα που ερευνά τις προηγμένες τεχνολογίες δικτύωσης και ζητήματα δια-λειτουργικότητας μεταξύ των Grid διαφορετικών περιοχών. Το GLUE Schema [17] στοχεύει στο να καθορίσει ένα κοινό εννοιολογικό πρότυπο δεδομένων και να ορίσει κοινές / δια-λειτουργικές προδιαγραφές πληροφορίας που θα χρησιμοποιούνται για τον έλεγχο και την ανακάλυψη πόρων του Grid.

Πιο συγκεκριμένα, επικεντρώνεται στο να περιγράψει όλους τους πόρους που συμμετέχουν σε ένα Grid οι οποίοι πρέπει να ελέγχονται και να αναζητούνται. Σαν τελικό στόχο έχει να μοντελοποιηθεί σωστά ώστε να παράγει ένα σχήμα το οποίο θα μπορεί να χρησιμοποιηθεί σαν Grid Information Service (GIIS), και η ίδια πληροφορία θα μπορεί να ανακτηθεί από GIIS's διαφορετικής τεχνολογίας (π.χ. R-GMA [9], MDS 2 / 3 [8]). Ο πίνακας 3-1 δείχνει τις τεχνολογίες που χρησιμοποιούν διάφοροι GIIS's

Η τρέχων έκδοση του GLUE Schema [17] είναι η 1.1 και αποτελείται από τέσσερα επιμέρους σχήματα :

- **Computing Element (CE)**, περιγραφή υπολογιστικών μονάδων (με το οποίο και θα ασχοληθούμε στη συνέχεια).
- **Storage Element (SE)**, περιγραφή δίσκων και μονάδων αποθήκευσης πληροφορίας.
- **CE – SE Binding**, δείχνει τη σχέση μεταξύ μονάδων αποθήκευσης και υπολογιστικών μονάδων.
- **Network** (δεν έχει ολοκληρωθεί ακόμη).

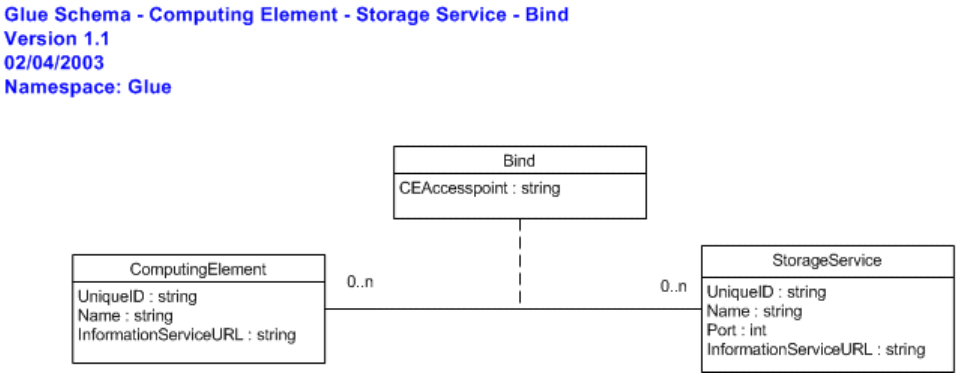
| | MDS 2.x | R-GMA | MDS 3.x |
|---|---|---|---|
| DATA Model | LDAP | Relational | XML |
| Communication | LDAP | HTTP | Independent |
| Storage | in-memory | In-memory or DBMS | in-memory or xIndice (XML DB) |
| Query language | LDAP | SQL | XPath (later XQuery, XSLT) |
| Resource Information Service | GRIS | Producer | Service |
| Aggregation Service | GIIS | Archiver | Index Service |
| Directory Service | GIIS | Registry | Index Service |

**Πίνακας 4–1**, Grid Information Services

Ο πίνακας 4-1 δείχνει τις διάφορες τεχνολογίες που χρησιμοποιούνται από τους GIS και την ετερογένεια που έρχεται να καλύψει το GLUE Schema.

## 4.1 Σχέση μεταξύ μονάδων αποθήκευσης και υπολογισμού

Η δρομολόγηση μιας διεργασίας συχνά απαιτεί τόσο μια υπηρεσία υπολογισμού για να τρέξει όσο και μια υπηρεσία αποθήκευσης για να παρέχει την αποθήκευση της εξόδου ή ενός αρχείου εισόδου. Η επιλογή ενός τέτοιου κατάλληλου ζευγαριού γίνεται στη βάση των στατικών σχέσεων μεταξύ αυτών των υπηρεσιών που καθορίζονται από το διαχειριστή του συστήματος.

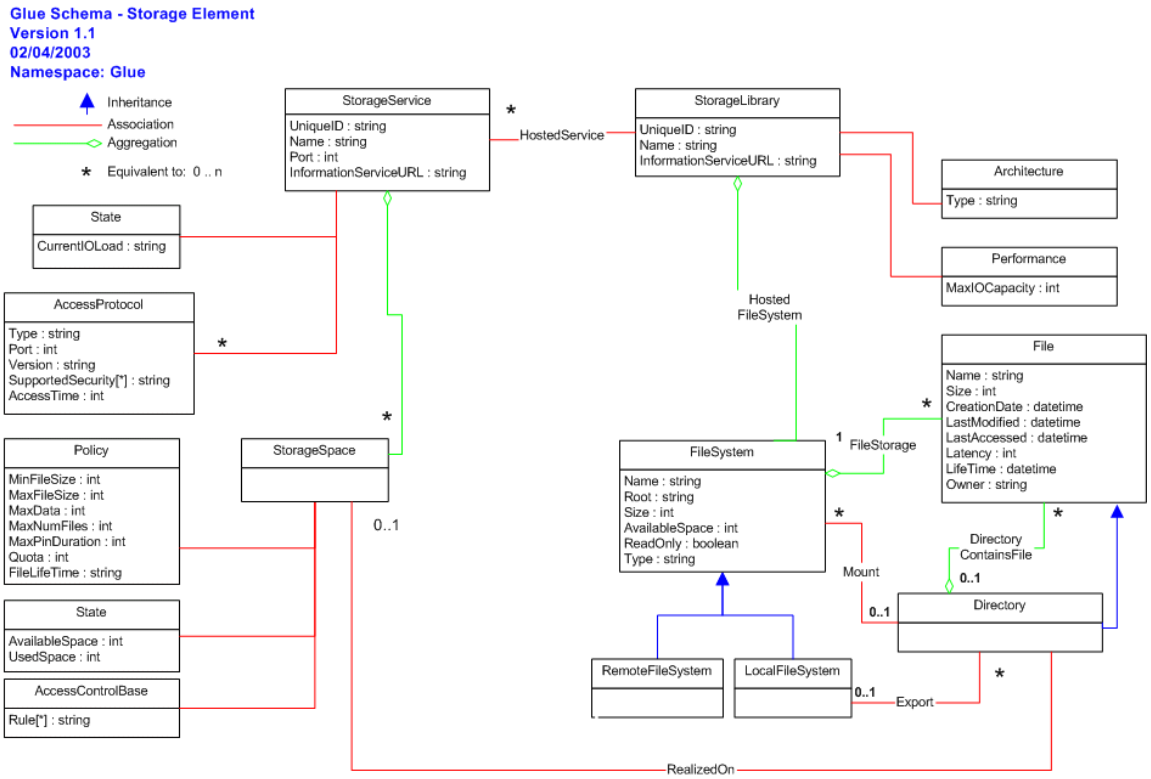



**Εικόνα 4–1**, Computing Element UML class Diagram

## 4.2 Περιγραφή μονάδων αποθήκευσης (SE)

Για κάθε μονάδα αποθήκευσης καθορίζονται πλήρως οι παρακάτω έννοιες :

- **Αποθηκευτικός Χώρος** (Storage space), δίνει ένα σύνολο πολιτικών (MaxFileSize, MinFileSize, MaxData, MaxNumFiles, MaxPinDuration), και προνομίων πάνω σε καταλόγους (directories).
- **Μόνιμος τύπος αρχείου (**Permanent file type), ένα αρχείο που αποθηκεύεται σε έναν αποθηκευτικό χώρο και μπορεί να αφαιρεθεί μόνο από τον ιδιοκτήτη ή από το διαχειριστή του συστήματος.
- **'Ευμετάβλητος' τύπος αρχείου** (Volatile file type), ένα αρχείο που αποθηκεύεται σε έναν αποθηκευτικό χώρο και μπορεί να αφαιρεθεί από την υπηρεσία αποθήκευσης όταν απαιτείται ελεύθερος χώρος. Είναι συνήθως αρχεία που έχουν συγκεκριμένη περίοδο ζωής, η οποία καθορίζεται από το σύστημα ή το χρήστη.
- **Ανθεκτικός τύπος αρχείου (**Durable file type), ένα αρχείο που αποθηκεύεται σε έναν αποθηκευτικό χώρο που προορίζεται να αφαιρεθεί το συντομότερο δυνατόν, αλλά δεν πρέπει να διαγραφεί από την υπηρεσία αποθήκευσης. Συνδέεται με συγκεκριμένη περίοδο ζωής (ίσως μεγαλύτερη από αυτή ενός ευμετάβλητου αρχείου), αλλά όταν λήγει η διάρκεια ζωής του δεν διαγράφεται, αλλά ο διαχειριστής του συστήματος ειδοποιείται. Κατά συνέπεια, η έννοια ενός "ανθεκτικού" αρχείου έχει τα χαρακτηριστικά γνωρίσματα και των Ευμετάβλητων και μόνιμων αρχείων.
- **Υπηρεσία Αποθήκευσης** (Storage Service), προσδιορίζεται από ένα μοναδικό URI και διαχειρίζεται τους πόρους δίσκων και ταινιών σε επίπεδο Αποθηκευτικού Χώρου. Κάθε Αποθηκευτικός Χώρος συνδέεται με μια εικονική οργάνωση (VO) και ένα σύνολο από συγκεκριμένες πολιτικές πάνω σε αυτή, έχοντας όλες τις λεπτομέρειες του υλικού κρυμμένες. Επίσης εκτελεί τη μεταφορά αρχείων προς ή από τον Αποθηκευτικό Χώρο, χρησιμοποιώντας ένα σύνολο από υπηρεσίες μετακίνησης δεδομένων (π.χ. GridFTP). Τέλος ρυθμίζει την πολιτική για τη διάρκεια ζωή των αρχείων (lifetime policy).




**Εικόνα 4–2**, Storage Element UML class Diagram

## 4.3 Περιγραφή Υπολογιστικών Μονάδων (CE)

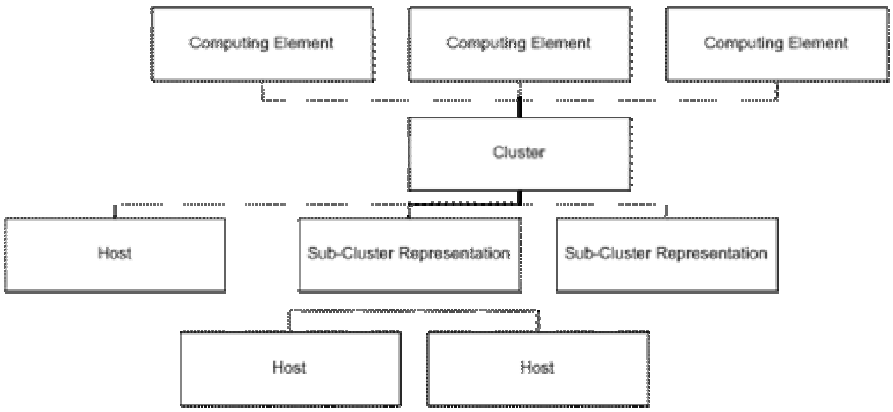
Σε αυτό το κομμάτι περιγράφονται πλήρως όλα τα επιμέρους τμήματα που αποτελούν ένα υπολογιστικό σύστημα, είτε αυτό αποτελείται από έναν μόνο υπολογιστή (node), είτε από σύνθεση πολλών ομάδων κόμβων (clusters / subclusters).

Τα βασικά στοιχεία που περιγράφουν μια υπολογιστική μονάδα είναι τα εξής :

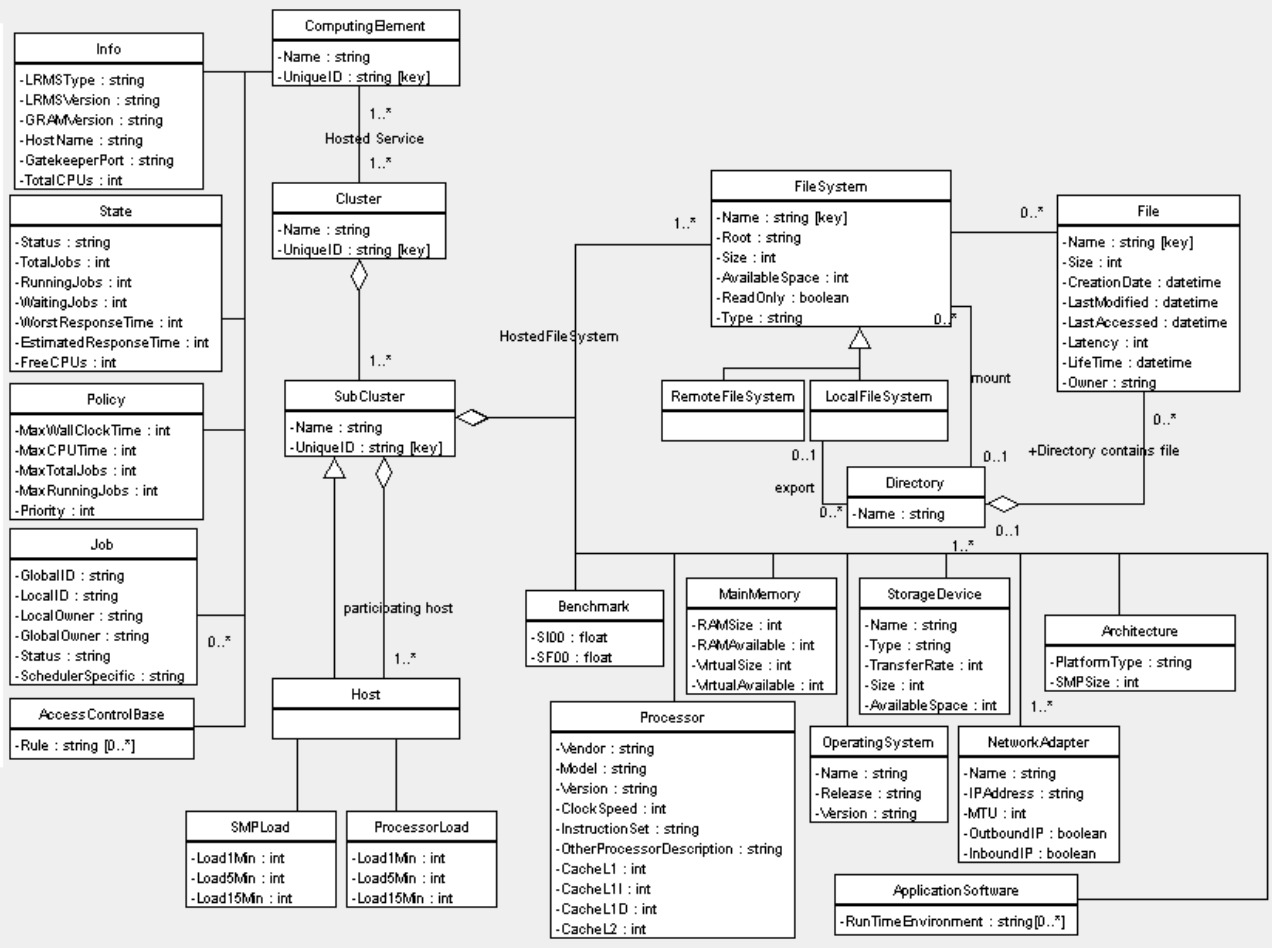
- **Computing Element**, δηλώνει ένα σημείο εισόδου σε μια ουρά (queuing system) και υπάρχει αποκλειστικά μία ουρά για κάθε Computing Element, για την αναπαράσταση παραπάνω της μιας ουράς σε ένα σύστημα μπορούμε να δημιουργήσουμε περισσότερα Computing Element. Η πληροφορία που περιέχει

είναι αποκλειστικά για την ουρά, ενώ οποιαδήποτε άλλη πληροφορία για φυσικούς πόρους περιέχεται στο Cluster.

- **Cluster**, Το Cluster είναι μια οντότητα που ομαδοποιεί Subclusters και κόμβους και είναι δυνατόν να περιέχεται σε περισσότερους του ενός Computing Elements.
- **Subcluster**, Το Subcluster είναι μια ομοιογενής συλλογή κόμβων, όπου η ομοιογένεια καθορίζεται στους κόμβους που έχουν στα αντίστοιχα κατηγορήματα (attributes) τους, ίδιες τιμές. Το subcluster κληρονομεί από τους κόμβους το σύνολο των ιδιοτήτων για τις οποίες οι ομοιογενείς τιμές βεβαιώνονται, καθώς επίσης και τον αριθμό τους.
- **Host**, χαρακτηρίζει κάθε υπολογιστικό κόμβο ανάλογα με τη σύνθεσή του (π.χ. επεξεργαστής, κύρια μνήμη, προγράμματα, κ.τ.λ.)




**Εικόνα 4–3**, παράδειγμα αναπαράστασης υπολογιστικής μονάδας




**Εικόνα 4–4**, Computing Element UML class Diagram

Στο παράρτημα Α.1 παρατίθεται το σχεσιακό σχήμα για το Computing Element, πάνω στο οποίο θα βασιστούμε στη συνέχεια.

# Κεφάλαιο 5, Η δουλειά μας

Είδαμε από τη βιβλιογραφία πως η τρέχουσα ερευνητική τάση είναι η δημιουργία πιο εκφραστικών γλωσσών, πιο έξυπνων αλγορίθμων που θα αναλάβουν την διαδικασία της αναζήτησης – μεσιτείας (brokering) πόρων για τα συστήματα Grid, με αποκορύφωμα την υλοποίηση του knowledge-oriented Grid που χρησιμοποιεί την τεχνολογία του semantic web. Αν και το τελευταίο αποτελεί μια πιο ολοκληρωμένη λύση, και έχουν αναπτυχθεί συστήματα που υποστηρίζουν την τεχνολογία αυτή, δεν είναι τόσο αποδοτικά ώστε να χρησιμοποιηθούν σε πραγματικά συστήματα Grid. Η υλοποίησή μας υποστηρίζει ένα σημαντικό ποσοστό της επιπλέον λειτουργικότητας που υπόσχεται το παραπάνω σύστημα με απλή σύνταξη ερωτήσεων και χρήση της ήδη υπάρχουσας τεχνολογίας.

Η υλοποίησή μας δίνει την δυνατότητα στο χρήστη του Grid να κάνει ερωτήσεις για πόρους σε μια υψηλού επιπέδου γλώσσα με σύνταξη σαν της DataLog. Η υλοποίηση βασίζεται σε ένα parser, τον οποίο από εδώ και στο εξής θα ονομάζουμε *εικονική βάση DataLog*. Ο parser υποστηρίζει το GLUE CE schema και μετασχηματίζει την ερώτησή μας με χρήση ειδικών κατηγορημάτων. τα κατηγορήματα αυτά, τα οποία από εδώ και στο εξής θα ονομάζουμε *εικονικές συναρτήσεις*, υπολογίζονται από μια βάση γνώσης, για να πάρουμε σαν έξοδο μια ερώτηση σε χαμηλού επιπέδου γλώσσα που υποστηρίζεται από GIS.

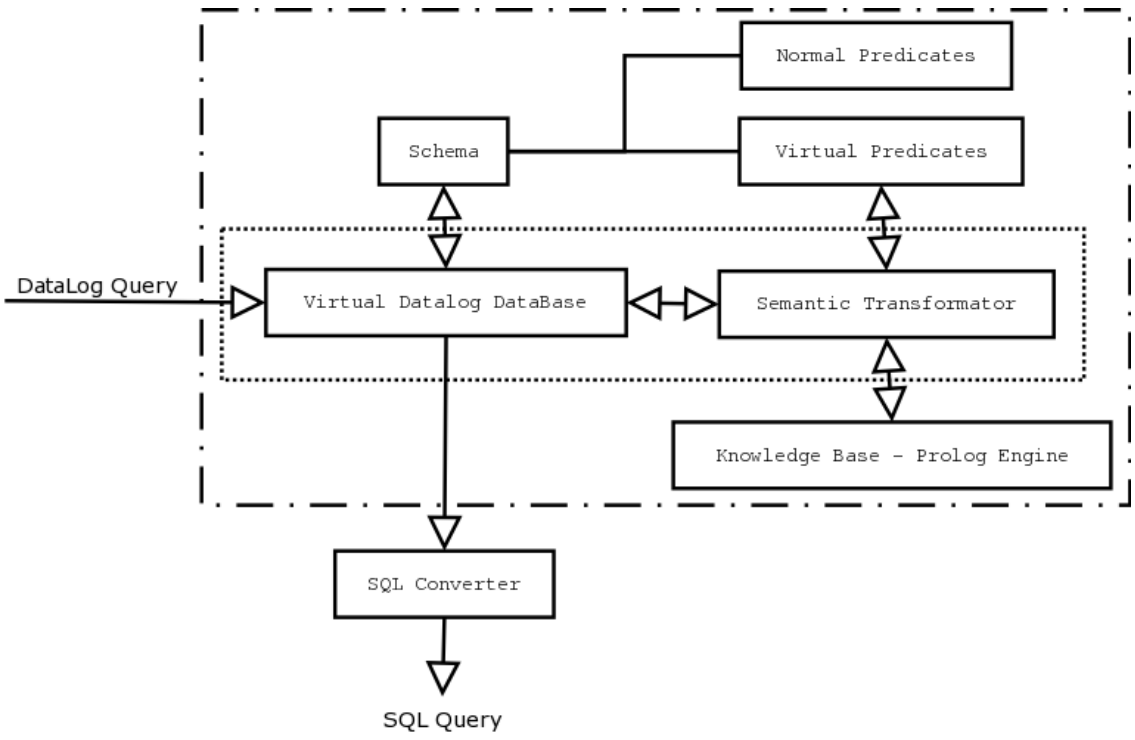
Πιο συγκεκριμένα, η εικονική βάση DataLog περιέχει όλους τους πίνακες (ή αλλιώς συναρτήσεις) για το GLUE CE schema, αλλά χωρίς τα δεδομένα, και χρησιμοποιεί συναρτήσεις που υπολογίζονται από μια prolog engine και ορίζονται από το χρήστη για να αρχικοποιήσει μεταβλητές που χρησιμοποιούνται στην ερώτηση. Στο τέλος μετασχηματίζει την ερώτηση σε μια ‘βελτιωμένη’ DataLog ερώτηση ή σε μια γλώσσα που υποστηρίζεται από GIS, στην υλοποίηση μας υποστηρίζεται η SQL που υπακούει στο GLUE CE Relational Schema (παράρτημα Α.1).

Χρησιμοποιήσαμε τη prolog γιατί είναι μια πολύ ισχυρή γλώσσα υπολογισμού με δυνατότητες αναδρομικού υπολογισμού, τη Datalog σαν γλώσσα ερωτήσεων γιατί είναι επίσης πολύ ισχυρή γλώσσα και είναι πολύ κοντά στο σχεσιακό μοντέλο και την SQL σαν μια ενδεικτική target γλώσσα γιατί είναι από τις πιο διαδεδομένες γλώσσες ερωτήσεων.

## 5.1 Σημασιολογικά χαρακτηριστικά (Semantic features)

Η σχηματική δομή της υλοποίησής μας φαίνεται αναλυτικά στην εικόνα 5-1. Όπως παρατηρούμε αποτελείται από 5 βασικά κομμάτια:

- Την Εικονική βάση DataLog.
- Τον Σημασιολογικό Μετασχηματιστή
- Το Schema στο οποίο υπακούει η DataLog
- Τη Βάση Γνώσης – Prolog Engine
- Και τον Μετατροπέα σε SQL




**Εικόνα 5-1**, σχηματική αναπαράσταση της αρχιτεκτονικής μας

Το Schema αποτελείται από 2 ειδών συναρτήσεις, τις απλές, όπου αναφέρονται στους πίνακες με τα δεδομένα του GLUE Schema και τις εικονικές, όπου τις ορίζουμε για να υπολογιστούν με τη χρήση της Prolog και να ενσωματώσουμε τα αποτελέσματά τους στην τελική ερώτηση. Η εικονική βάση DataLog είναι το βασικό κομμάτι της εφαρμογής μας από το οποίο εξαρτώνται όλα τα υπόλοιπα. Για μια ερώτηση που του δίνεται, συμβουλεύεται το Schema ώστε να δημιουργήσει μια σύνθετη δομή που θα αναλύσουμε στην ενότητα 5.4 Και αφού τελειώσει με την ανάλυση της ερώτησης, εξετάζει αν χρησιμοποιήθηκαν εικονικές συναρτήσεις ώστε να τις στείλει στον Σημασιολογικό Μετασχηματιστή. Ο τελευταίος, αφού συμβουλευτεί από το σχήμα τον τύπο των ορισμάτων των εικονικών συναρτήσεων, τις μετατρέπει σε Prolog ερωτήσεις και τις στέλνει στη Prolog Engine, από όπου και παίρνει το αποτέλεσμα και το ενσωματώνει στην αρχική δομή, εξετάζουμε με λεπτομέρεια τον τρόπο στην ενότητα 5.5.

Π.χ. αν βρεθεί η συνάρτηση

```
virtual_OS_family("unix",Os)
```

θα μας επιστραφεί από τον Σημασιολογικό Μετασχηματιστή η έκφραση

```
Os = "linux"
OR Os = "solaris"
OR Os = "aix"
OR Os = "bsd"
OR Os = "sunos"
```

Την οποία και θα ενσωματώσουμε στη δομή που κρατάμε για την ερώτηση, αντικαθιστώντας την με την εικονική συνάρτηση.

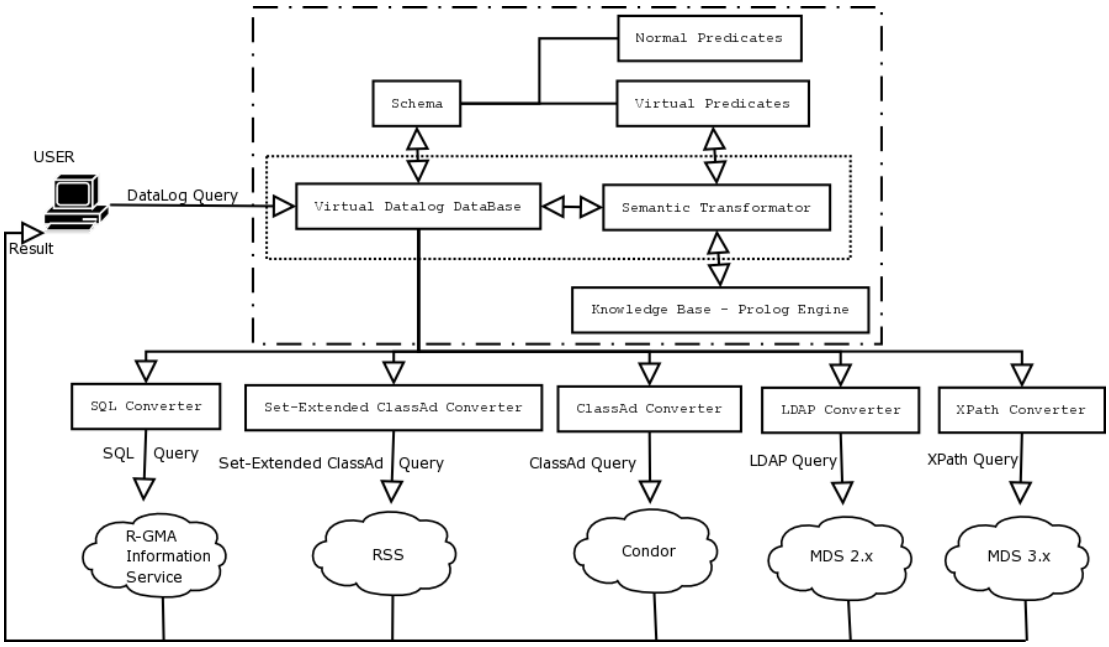
Τέλος αφού μετασχηματιστούν όλες οι εικονικές συναρτήσεις, η δομή μας περνάει από τον Μετατροπέα SQL που την μετατρέπει σε μια SQL ερώτηση.

Αξίζει να σημειωθεί πως εκτός από την εικονική βάση DataLog και τον Σημασιολογικό Μετασχηματιστή τα οποία μαζί παρέχουν τη λειτουργικότητα που θέλουμε, τα υπόλοιπα κομμάτια μπορούν πολύ εύκολα να επεκταθούν, τροποποιηθούν ή και να αντικατασταθούν.

Π.χ. ο Μετατροπέας SQL να αντικατασταθεί σε Μετατροπέα LDAP ή XPath, να μεγαλώσουμε τη Βάση Γνώσης (προσθήκη νέων συναρτήσεων) ή να επεκτείνουμε το Schema (να ενσωματώσουμε το GLUE SE / CE-SE / Network).

### 5.1.1 Πλεονεκτήματα που προσφέρει η υλοποίησή μας

Το βασικό πλεονέκτημα της υλοποίησής μας είναι ότι μπορεί να λειτουργήσει από την πλευρά του χρήστη και για αυτό το λόγο μπορεί να χρησιμοποιηθεί παράλληλα με οποιοδήποτε σύστημα (εφόσον βέβαια δημιουργηθεί κατάλληλος μετατροπέας). Η εικόνα 5-2 δείχνει ένα παράδειγμα.




**Εικόνα 5-2**, παράδειγμα χρήσης

Έτσι ένας απλός χρήστη του MDS ή ακόμη και ένας χρήστης των υλοποιήσεων των ενοτήτων 3.1, 3.2 μπορεί να προσθέσει σε μια εφαρμογή Grid, τη λειτουργικότητα που του παρέχουμε. Πρέπει να διευκρινίσουμε πως η εικόνα δεν είναι εντελώς ακριβής, είναι δύσκολο να υλοποιηθεί ένα σύστημα που να παρέχει όλους τους απεικονιζόμενους

μετατροπείς, κυρίως λόγω των διαφορών στην απεικόνισης της πληροφορίας, ακόμη και στο GLUE Schema, υπάρχουν αρκετές δομικές διαφορές ανάμεσα στο Relational και το XML Schema. Θα ήταν πιο σωστό να αντιστοιχούσε ξεχωριστό σύστημα για κάθε μετατροπέα, αλλά χωρίς βλάβη της γενικότητας και για λόγους απλοποίησης το αναπαραστήσαμε με αυτόν τον τρόπο.

Όσον αφορά την υλοποίηση της ενότητας 3.3, αν και εμείς παρέχουμε ένα σημαντικό κομμάτι της ευελιξίας που δίνει ένα τέτοιο σύστημα , σε καμία περίπτωση δεν μπορούμε να την παρέχουμε ολόκληρη, αλλά και δεν είναι αυτός ο σκοπός μας. Η βασική μας διαφορά είναι ότι το πρώτο αποτελεί μια ολοκληρωμένη λύση βασισμένη στο Semantic Web, αλλά θέτει αναγκαία την χρησιμοποίηση νέων τεχνολογιών και συστημάτων τα οποία δεν έχουν ωριμάσει αρκετά (σε σχέση με τα παραδοσιακά σχεσιακά μοντέλα), ενώ εμείς προσφέρουμε ένα χρήσιμο υποσύνολο της λειτουργικότητας του που μπορεί όμως να χρησιμοποιηθεί από υπάρχοντα συστήματα χωρίς την τροποποίηση της αρχιτεκτονικής τους.

## 5.2 Η γλώσσα DataLog

Σε αυτή την ενότητα αναφέρουμε την σημαντική της γλώσσας DataLog, η οποία αποτελεί μια θεωρητική συμπερασματική βάση δεδομένων (deductive database). Έχει χτιστεί γύρω από ένα λογικό μοντέλο και προσφέρει μια γλώσσα διατύπωσης που επιτρέπει την έκφραση λογικών ερωτήσεων.

Η γλώσσα διατύπωσης ερωτήσεών της Datalog συσχετίζεται με τη γλώσσα Prolog και κυρίως με ένα υποσύνολο της, έχοντας τις παρακάτω διαφορές :

1. Απαγορεύεται η χρήση συναρτησιακών όρων (functional terms).
2. Απαγορεύεται η χρήση σύνθετων όρων (π.χ. f(f(…)) )
3. Απαγορεύεται η χρήση της άρνησης σε αλφαριθμητικά (negative literal).
4. Δεν υποστηρίζονται μηχανισμοί της Prolog όπως backtracking

Η διαφορά της με σχεσιακές γλώσσες όπως η SQL είναι ότι οι τελευταίες προσφέρουν μια περιορισμένη μορφή λογικής έκφρασης, σε αντίθεση με τις συμπερασματικές βάσεις που είναι μια εξελιγμένη μορφή σχεσιακών συστημάτων που συμπεριλαμβάνουν κανόνες οι οποίοι μπορούν να εξάγουν συμπεράσματα εκτός από τα γεγονότα που είναι περασμένα στη βάση.

Το λογικό μοντέλο των συμπερασματικών βάσεων σχετίζεται πολύ με το σχεσιακό μοντέλο και ειδικότερα, με το συγγενικό υπολογισμό μέσω σχεσιακής άλγεβρας. Η τελευταία όμως είναι ανεπαρκής για την έκφραση των ερωτήσεων σε βάσεις δεδομένων, μια βασική ατέλεια είναι η έλλειψη αναδρομής (recursion), η οποία δεν επιτρέπει την έκφραση των αναδρομικών ορισμών στο transitive closure ενός γράφου.

## 5.3 Ερωτήσεις DataLog

Είναι σημαντικό να αναφερθεί αρχικά, ότι οι Datalog ερωτήσεις δεν μπορούν να υποστηρίζουν άρνηση (not(…)) και αναδρομικές συναρτήσεις (recursion) γιατί αυτό δεν είναι δυνατόν να γίνει από τη στιγμή που η βάση μας δεν έχει τα πραγματικά δεδομένα. Επιπλέον η εικονική μας βάση δεν υποστηρίζει κανόνες αλλά είναι πλήρως συμβατή με μια Datalog βάση με κανόνες επειδή υποστηρίζουμε στην ερώτηση και διάζευξη, OR (και δεν έχουμε αναδρομή).

Οι συναρτήσεις για το GLUE CE Schema έγιναν με βάση το Relational Schema (παράρτημα Α.1) χρησιμοποιώντας το όνομα του πίνακα για το όνομα της συνάρτησης και για κάθε κατηγόρημα (attribute) δημιουργούσαμε ένα όρισμα (argument) με τη σειρά που αυτά έχουν οριστεί.

Π.χ. για τον πίνακα GlueCluster με κατηγορήματα :

UniqueID

Name

InformationServiceURL

Ορίσαμε την συνάρτηση glueCluster(_,_,_) με ένα προς ένα αντιστοίχηση στα ορίσματα με τα κατηγορήματα, με τη σειρά που εμφανίζονται τα τελευταία.

Εκτός από το GLUE CE Schema (το οποίο μπορεί εύκολα να επεκταθεί ή και να αλλαχθεί) έχουμε ορίσει κάποιες ενδεικτικές συναρτήσεις που υπολογίζονται με τη prolog engine τις οποίες θα δούμε αναλυτικά αργότερα. Επιπλέον συναρτήσεις μπορούν εύκολα να ενσωματωθούν και αφήνεται στην ευχέρεια και φαντασία του προγραμματιστή να τις ορίσει - εμείς απλά δίνουμε την πλατφόρμα πάνω στην οποία μπορεί να το κάνει αυτό.

Η γραμματική της ερώτησης που μπορούμε να υποβάλλουμε βρίσκεται στο παράρτημα Α.2 . Η υλοποίηση έγινε σε C++ με τα εργαλεία Flex και Bison.

## 5.4 Αλγόριθμος και Δομές Δεδομένων

Σε αυτή την ενότητα θα δούμε τον βασικό αλγόριθμο που χρησιμοποιήσαμε, καθώς και τις δομές δεδομένων που δημιουργήσαμε για να υλοποιήσουμε την εικονική βάση DataLog.

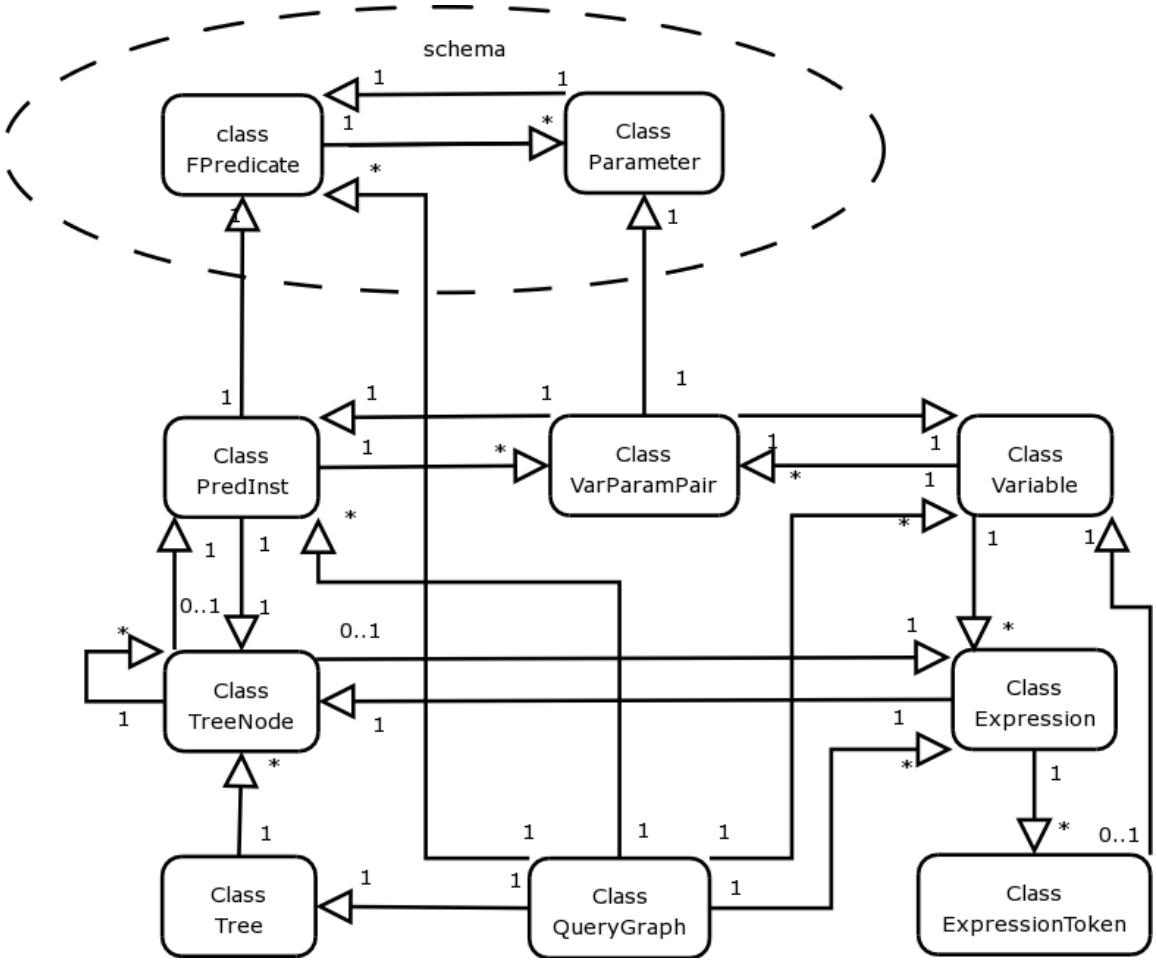
Πριν αρχίσει η συντακτική ανάλυση της ερώτησης, αρχικοποιείται το schema στο οποίο θα πρέπει να είναι συμβατή η ερώτηση που θα κάνουμε. Αυτό γίνεται από μια συνάρτηση που (‘με το χέρι’) δημιουργούμε μια λίστα με κλάσεις που ορίζουν το όνομα κάθε συνάρτησης (predicate), τι ορίσματα παίρνει, πόσα και τι τύπου είναι το καθένα. Επίσης στο αρχικό schema κρατάμε πληροφορία για την σύνθεση μιας SQL ερώτησης, θα μπορούσαμε σχετικά εύκολα να αλλάξουμε αυτή την πληροφορία και λίγο τον κώδικα για να συνθέσουμε ερώτηση σε μια άλλη γλώσσα (LDAP, XPath, …).

Καθώς γίνεται η συντακτική ανάλυση δημιουργούμαι τις παρακάτω δομές :

- Λίστα από μεταβλητές, οι μεταβλητές που χρησιμοποιούνται στην ερώτηση.
- Λίστα από εκφράσεις, μια έκφραση μπορεί να είναι η εξής : X+3<=5*Y-3/X

- Λίστα από χρησιμοποιημένες συναρτήσεις, που κρατάει την πληροφορία ποια συνάρτηση του schema είναι και ποια ορίσματα έχουν αρχικοποιηθεί και με ποιες μεταβλητές.
- Λίστα από χρησιμοποιημένες εικονικές συναρτήσεις, που κρατάει την πληροφορία ποιες συναρτήσεις πρέπει να υπολογιστούν μέσω της prolog engine και όλα όσα κρατάει η προηγούμενη λίστα.
- Ένα δέντρο, που αποθηκεύει την δομή των λογικών προτάσεων της ερώτησης, οι κόμβοι του είναι AND / OR και τα φύλλα εκφράσεις ή συναρτήσεις,

Οι σχέσεις μεταξύ των δομών αυτών φαίνεται από την εικόνα 6-1.

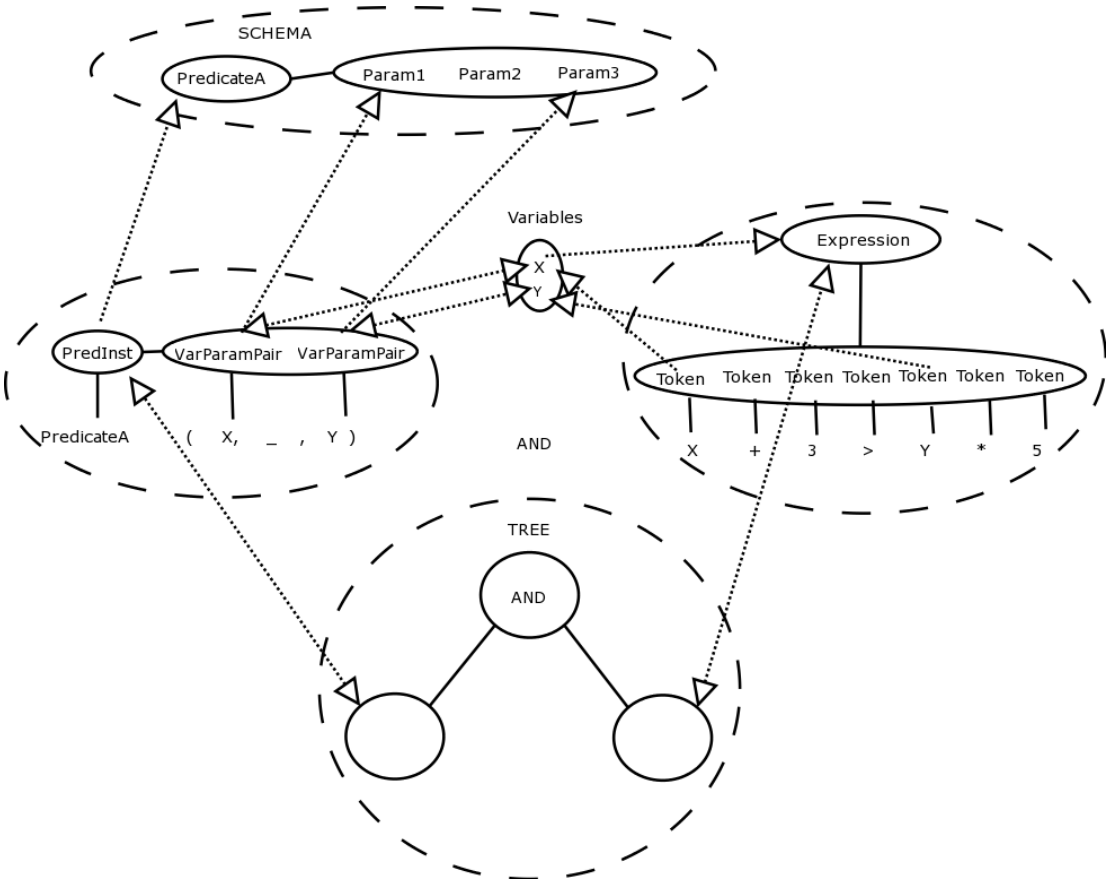



**Εικόνα 5-1**, σχέσεις μεταξύ των κλάσεων

Στη συνέχεια βλέπουμε ποιες συναρτήσεις είναι εικονικές, τις δίνουμε στο Σημασιολογικό Μετασχηματιστή και τις μετασχηματίζουμε με τον τρόπο που θα δούμε στην επόμενη ενότητα ώστε να μπουν στην δομή του δέντρου που έχουμε ήδη. Τέλος συνθέτουμε από το δέντρο την ερώτηση σε SQL.

Για την χρήση όσο το δυνατόν λιγότερης μνήμης μόνο το αρχικό schema, οι μεταβλητές και οι τιμές (int, strings) καταλαμβάνουν μνήμη, όλα τα υπόλοιπα είναι αναφορές σε αυτά.

Π.χ. η κλάση PredInst περιέχει ένα δείκτη στο αρχικό σχήμα και κρατάει μια λίστα μόνο με τα ορίσματα που έχουν αρχικοποιηθεί, ενώ τα τελευταία έχουν 2 δείκτες, ένα στο αντίστοιχο όρισμα στο schema και ένα στη μεταβλητή που το αρχικοποιεί.




**Εικόνα 5-2**, παράδειγμα δομών που κρατάμε

Η χρονική πολυπλοκότητα του συντακτικού αναλυτή για τη δημιουργία της δομής που κρατάμε, για την δημιουργία των prolog ερωτήσεων και για τον μετασχηματισμό της απάντησης των τελευταίων είναι γραμμική ενώ για τη δημιουργία της τελικής ερώτησης είναι πολυωνυμική ($O(N^2)$). Μπορεί να εκφραστεί ως συνάρτηση του πλήθους των μεταβλητών, των συναρτήσεων και των τιμών (int, string, …) που περιέχονται στην ερώτηση. Η συνολική πολυπλοκότητα της υλοποίησης μας εξαρτάται από την πολυπλοκότητα του υπολογισμού της prolog.

Η χωρική πολυπλοκότητα, εκτός του υπολογισμού της prolog είναι γραμμική. Μπορεί να εκφραστεί ως συνάρτηση του αριθμού των μεταβλητών και το πλήθος των τιμών (int, string, …) που περιέχονται στην ερώτηση συν το χώρο που κρατάμε για το αρχικό σχήμα.

## 5.5 Σημασιολογικός Μετασχηματισμός

Σε αυτή την ενότητα θα δούμε αναλυτικά πως μπορούμε να χρησιμοποιήσουμε την prolog για να υπολογίσουμε και να αρχικοποιήσουμε μεταβλητές στην τελική γλώσσα.

Χρησιμοποιήσαμε την SWI-Prolog γιατί είναι ελεύθερο λογισμικό, είναι μια αρκετά καλή υλοποίηση συμβατή με την ISO Prolog, δίνει C++ Interface και παρέχει αρκετές ενσωματωμένες συναρτήσεις.

### 5.5.1 Μετασχηματισμός εικονικής συνάρτησης για ερώτηση στη Prolog

Κάθε εικονική συνάρτηση που χρησιμοποιείται στην ερώτηση στέλνεται στον σημασιολογικό μετασχηματιστή. Αφού την πάρει ο τελευταίος, ελέγχει από το σχήμα το όνομα, τον αριθμό των ορισμάτων και τον τύπο τους, αν είναι για είσοδο / έξοδο και τι τύπου είναι (int, float, …). Στη συνέχεια δημιουργεί μια συνάρτηση με το ίδιο όνομα και

με ορίσματα, την τιμή που αντιστοιχεί στην μεταβλητή, αν αυτή είναι για είσοδο και την ίδια την μεταβλητή αν αυτή είναι για έξοδο

Π.χ. για την εικονική συνάρτηση

virtual_comp_OS(C,Os)

όπου το 1$^{ο}$ όρισμα είναι είσοδος και το 2$^{ο}$ έξοδος και στην ερώτηση υπάρχει το :

C="msdos"

θα δημιουργήσει την συνάρτηση

virtual_comp_OS("msdos",Os)

Τέλος στέλνει στην Prolog την εξής ερώτηση :

Transform( virtual_comp_OS("msdos",Os), R).

Όπου τη συνάρτηση που δημιουργήσαμε τη δίνουμε σαν είσοδο, και παίρνουμε το R σαν έξοδο. Εξετάζουμε την μορφή της εξόδου στην επόμενη ενότητα.

### 5.5.2 Μετασχηματισμός δέντρου για την έξοδο της Prolog

Για να γενικεύσουμε την χρήση των συναρτήσεων και να μπορέσουμε να υποστηρίξουμε σύνθετα αποτελέσματα, η απάντηση της prolog για κάθε ερώτηση που της υποβάλουμε πρέπει να έχει μια συγκεκριμένη μορφή. Είναι ένα δέντρο αντίστοιχο με αυτό που παράγει ένας συντακτικός αναλυτής καθώς εξετάζει μια έκφραση.

Ένα δέντρο στη prolog μπορεί να αναπαρασταθεί με την εξής μορφή :

Tree(x1,tree(x1,tree(x2,tree(x3,tree(void))),tree(…), tree(…))

Εμείς εμπνευστήκαμε από την παραπάνω μορφή και επιλέξαμε την παρακάτω σύνταξη :

node (type, node(type,…),node(type,...),…).

ή node(void) αν δεν επιστρέψει κανένα αποτέλεσμα.

Όπου το type υποδηλώνει τι τύπου είναι ο τρέχων κόμβος και μπορεί να είναι ένα από τα εξής:

- **and / or**, και ο κόμβος μπορεί να έχει απεριόριστο αριθμό παιδιών.
- **eq(=) / neq(<>) / gt(>) / lt(<) / get(>=) / let(<=)**, και ο κόμβος πρέπει να έχει ακριβώς 2 παιδιά.

- **mul / div / plus / minus,** και ο κόμβος μπορεί να έχει απεριόριστο αριθμό παιδιών.
- **func**, δηλώνει συναρτήσεις, όπου το πρώτο όρισμα είναι το όνομα και τα υπόλοιπα τα ορίσματα τις συνάρτησής μας. Είναι συναρτήσεις που προορίζονται για την target γλώσσα.
- **int / float / char / str / id**, και ο κόμβος μπορεί να έχει μια τιμή (φύλλο).
- **par_o / par_c**, δηλώνει το άνοιγμα και το κλείσιμο συναρτήσεων και δεν έχει κανένα παιδί. Χρησιμοποιείται μόνο για τις αριθμητικές εκφράσεις, για τις λογικές δεν χρειάζεται από τη στιγμή που υπάρχει το δέντρο.

Π.χ. αν θέλαμε να επιστρέψουμε την έκφραση :

```
A>8.35+3 and ( X="sss" or 5=Y*3-34/X) and Y>X
```

Θα έπρεπε από τη prolog να δημιουργήσουμε το εξής δέντρο :

```
node(and ,
      node( gt ,
            node( id , A ) , node( plus ,
                  node( float , 8.35 ) ,
                  node( int , 3 ) ,
                  node( int , 3 ) ) ) ,
            node( or ,
                  node( eq ,
                        node( id , X ) ,
                        node( str , sss ) ) ,
                  node( eq ,
                        node( int , 5 ) ,
                        node( minus ,
                              node( mul ,
                                    node( id , Y ) ,
                                    node( int , 3 ) ) ,
                              node( div ,
                                    node( int , 34 ) ,
                                    node( id , X ) ) ) ) ) )
            node( gt ,
                  node( id , Y ) ,
                  node( id , X ) )  )
```

Παίρνοντας αυτό το δέντρο σαν απάντηση από τη prolog, για κάθε and / or δημιουργούμε κόμβους στο δέντρο που κρατάει ο συντακτικός αναλυτής στη θέση που ήταν η συνάρτηση που καλέσαμε, ενώ με τα υπόλοιπα δημιουργούμε Expressions τα οποία προσθέτουμε κάτω από αυτούς τους κόμβους και στην αντίστοιχη λίστα με Expressions που κρατάμε.

### 5.5.3 Προσθήκη νέων συναρτήσεων

Για να προσθέσει κάποιος μια νέα συνάρτηση, το μόνο που χρειάζεται να κάνει είναι να την ορίσει στο schema (όνομα, αριθμός / τύπος ορισμάτων) και φυσικά να γράψει κώδικα σε prolog. Ο κώδικας αυτός όμως έχει κάποιους περιορισμούς.

Επειδή από τη C++ καλούμε για κάθε εικονική συνάρτηση την εξής ερώτηση σε prolog :

transform(myFunction(arg1, arg2, …), Result).

πρέπει να οριστεί ακριβώς η παραπάνω συνάρτηση, και όχι απλά η

myFunction(arg1, arg2, …)

Τέλος το Result πρέπει να υπακούσει στη μορφή που αναφέραμε στην προηγούμενη υπό-ενότητα.

### 5.5.4 Συναρτήσεις που υλοποιήσαμε

Υλοποιήσαμε 5 ενδεικτικές συναρτήσεις για να δούμε πως μπορεί να χρησιμοποιηθεί η prolog για την βελτιστοποίηση την τελικής ερώτησης.

Οι 3 πρώτες αφορούν τα λειτουργικά συστήματα

- συνάρτηση που επιστρέφει λειτουργικά συμβατά με αυτό που της δίνουμε (αυτά περιορίζονται στην οικογένεια των windows)
- συνάρτηση που επιστρέφει τα λειτουργικά που ακολουθούν ένα standard (POSIX, MS-DOS, WIN32,…)
- συνάρτηση που επιστρέφει όλα τα λειτουργικά που ανήκουν σε μια οικογένεια (UNIX, Windows)

Επιπλέον υλοποιήσαμε 2 συναρτήσεις πάνω στους επεξεργαστές. Τους κατηγοροποιήσαμε με βάση την απόδοση τους (μέσω προσωπικής εμπειρίας), έτσι μπορούμε να ζητάμε :

- επεξεργαστές τουλάχιστον αντίστοιχους με αυτόν που δίνουμε (αν δεν υπάρχει αυτός που δίνουμε, ψάχνει τον κοντινότερο με ταχύτητα ρολογιού +-100).
- επεξεργαστές με ανώτερο όριο απόδοσης τον επεξεργαστή αυτόν που δίνουμε (αν δεν υπάρχει αυτός που δίνουμε, ψάχνει τον κοντινότερο με ταχύτητα ρολογιού +-100).

Π.χ. γνωρίζουμε ότι ένας επεξεργαστής AMD Athlon στα 700 είναι αντίστοιχος με έναν Intel Pentium III 800.

## 5.6 Παραδείγματα χρήσης

Σε αυτό το κεφάλαιο θα αναλύσουμε διάφορα παραδείγματα ερωτήσεων. Θα δούμε την SQL έξοδο πάνω στο GLUE RDBMS Schema [παράρτημα Α.1] από τις Datalog ερωτήσεις που υποβάλουμε. Αξίζει να σημειωθεί ότι η υλοποίησή μας είναι αρκετά γρήγορη, κανένα από τα παραδείγματα που δοκιμάσαμε δεν ξεπέρασε χρόνο εκτέλεσης μεγαλύτερο από 0.2 δευτερόλεπτα, ακόμη και αυτά που χρησιμοποιούσαν τη prolog engine, με αρκετά χαμηλά επίπεδα μνήμης.

### 5.6.1 Παραδείγματα πάνω στο Glue Schema

Εδώ θα δούμε κάποιες συχνές ερωτήσεις που συναντάμε στη βιβλιογραφία, οι πρώτες είναι σχετικά απλές, ενώ στο τέλος θα δούμε κάποιες πιο σύνθετες. Σε αυτά τα παραδείγματα δεν χρησιμοποιούμε καθόλου τον σημασιολογικό μετασχηματισμό.

**Παράδειγμα 1, με ένα πίνακα**

Εδώ ζητάμε να μας επιστρέψει τη διεύθυνση των sub clusters που έχουν μνήμη μεγαλύτερη από 512 MB, αρχιτεκτονική Alpha και τρέχουν λειτουργικό σύστημα Solaris και η ταχύτητα του επεξεργαστή τους είναι μεγαλύτερη από 600 κύκλους ανά δευτερόλεπτο.

**Ερώτηση:**

```
? glueSubCluster(_,_,_,Mem,_,_,_, "alpha",_,"solaris",_,_,_,_,C,_,_,_,_,_,_,_,_,_,_,_,URL)
and Mem >= 512
and C >= 600.
```

**Μετασχηματισμένη ερώτηση:**

```
SELECT t0.InformationServiceURL , t0.OSName , t0.PlatformType , t0.RAMSize
FROM glueSubCluster t0
WHERE
 ( t0.RAMSize >= 512
   AND t0.PlatformType = "alpha"
   AND t0.OSName = "solaris"
   AND t0.ClockSpeed > = 600  )
```

### Παράδειγμα 2, με join δυο πινάκων

Ζητάμε να μας επιστρέψει τα sub clusters που έχουν σκληρό δίσκο μεγαλύτερο από 10 GB, αρχιτεκτονική Alpha και τρέχουν λειτουργικό σύστημα Solaris.

**Ερώτηση:**

```
? glueSubCluster(F,_,_,Mem,_,_,_, "alpha",_,"solaris",_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_)
and glueHostRemoteFileSystem(F,_,_,Size,_,_,_)
and Size >=10000.
```

**Μετασχηματισμένη ερώτηση:**

```
SELECT t0.Size , t1.OSName , t1.PlatformType , t1.RAMSize , t1.UniqueID
FROM glueHostRemoteFileSystem t0 ,
    glueSubCluster t1
```

```
WHERE t0.GlueSubClusterUniqueID = t1.UniqueID
AND ( t0.Size >= 10000
  AND t1.PlatformType = "alpha"
  AND t1.OSName = "solaris"  )
```

### 5.6.2 Παραδείγματα με χρήση σημασιολογικού μετασχηματισμού

Εδώ θα δούμε κάποιες σύνθετες ερωτήσεις οι οποίες κάνουν χρήση των εικονικών συναρτήσεων prolog. Θα δούμε τις ερωτήσεις σε datalog και πως αυτές μετασχηματίζονται ώστε να συμπεριλάβουν και τις απαντήσεις της prolog engine.

**Παράδειγμα 1, με ένα πίνακα και δύο εικονικές συναρτήσεις**

Η ερώτηση μας επιστρέφει τη μνήμη (συνολική και διαθέσιμη), το λειτουργικό τον επεξεργαστή (κατασκευαστή, μοντέλο και ταχύτητα) καθώς και τη διεύθυνση του sub cluster, ζητώντας το λειτουργικό να είναι συμβατό με msdos και ο επεξεργαστής να είναι τουλάχιστον αντίστοιχος με Intel Pentium III με ταχύτητα ρολογιού στα 450, τέλος για τη μνήμη ζητάμε να έχει συνολική μεταξύ 256 MB και 512 MB ή διαθέσιμη μεγαλύτερη από 200 MB

**Ερώτηση:**

```
? glueSubCluster(_,_,_,Mem,MemAv,_,_,_,_,Os,_,_,V,M,_,C,_,_,_,_,_,_,_,_,_,_,URL)
  and virtual_comp_OS("msdos",Os)
  and virtual_atleast_like_processor("intel", "pentium3", "450" , V, M, C)
  and ((Mem > 256 and Mem<512)
     OR (MemAv <200)).
```

**Μετασχηματισμένη ερώτηση:**

```
SELECT t0.InformationServiceURL , t0.Version , t0.Model , t0.Vendor , t0.OSName ,
t0.RAMAvailable , t0.RAMSize
FROM glueSubCluster t0
WHERE
```

```
( ( ( t0.OSName = "winxp"
      OR t0.OSName = "win98"
      OR t0.OSName = "win95"
      OR t0.OSName = "msdos" )
    AND ( ( t0.Vendor = "intel"
        AND t0.Model = "pentium4"
        AND t0.ClockSpeed  >= 1800 )
      OR ( t0.Vendor = "amd"
        AND t0.Model = "athlon"
        AND t0.ClockSpeed >= 450 )
      OR ( t0.Vendor = "intel"
        AND t0.Model = "pentium3"
        AND t0.ClockSpeed >= 450 )
      OR ( t0.Vendor = "intel"
        AND t0.Model = "pentium2"
        AND t0.ClockSpeed  >= 550 )
      OR ( t0.Vendor = "intel"
        AND t0.Model = "celeron"
        AND t0.ClockSpeed >= 600 )))
  AND ( ( t0.RAMSize > 256
      AND t0.RAMSize < 512 )
    OR  t0.RAMAvailable < 200 )
```

**Παράδειγμα 2, με join τριών πίνακων και χρήση μιας εικονικής συνάρτησης**

Στην ερώτηση αυτή ζητάμε το cluster να έχει τουλάχιστον 2 ελευθέρους επεξεργαστές, σκληρό δίσκο μεγαλύτερο από 30GB, το λειτουργικό σύστημα να ανήκει στην οικογένεια του UNIX και να έχει αρχιτεκτονική Intel. Μας επιστρέφει εκτός από τα παραπάνω to Information Service URL cluster και του computing element, το Host name - port και το όνομα του μηχανήματος.

**Ερώτηση:**

```
? glueCE(ID,Name,GClusterID,_,_,_,_,Host,Port,_,_,_,_,_,_,FreeCPUs,_,_,_,_,_,
GRISURL) and glueSubCluster(F,_,GClusterID,_,_,_,_,
"intel",_,Os,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,URL)
and glueHostRemoteFileSystem(F,_,_,Size,_,_,_)
and virtual_OS_family("unix",Os)
and Size >30000
and FreeCPUs>2.
```

**Μετασχηματισμένη ερώτηση:**

```
SELECT t0.Size , t1.InformationServiceURL , t1.OSName , t1.PlatformType ,
t1.UniqueID , t2.InformationServiceURL , t2.FreeCpus , t2.GatekeeperPort ,
t2.HostName , t2.GlueClusterUniqueID , t2.Name , t2.UniqueID
FROM glueHostRemoteFileSystem t0 ,
    glueSubCluster t1 ,
    glueCE t2
WHERE t0.GlueSubClusterUniqueID = t1.UniqueID
AND t1.GlueClusterUniqueID = t2.GlueClusterUniqueID
AND ( ( t1.OSName = "linux"
     OR t1.OSName = "solaris"
     OR t1.OSName = "aix"
     OR t1.OSName = "bsd"
     OR t1.OSName = "sunos" )
  AND t0.Size = 30000
  AND t1.PlatformType = "intel"
  AND t2.FreeCpus > 2  )
```

# Κεφάλαιο 6, Συνεισφορά της δουλειάς μας και μελλοντικές βελτιώσεις

Η βασική λειτουργικότητα που παρέχουμε είναι ότι προσφέρουμε μια βάση γνώσης (knowledge base) σε ένα σύστημα ερωτήσεων και δίνουμε στον χρήστη τη δυνατότητα να εκφράσει πολύπλοκες ερωτήσεις με απλή και σύντομη σύνταξη, χωρίς να χρειαστεί να αλλάξουμε την τεχνολογία που χρησιμοποιούμε. Ένα κύριο πλεονεκτήματά της υλοποίησης μας είναι η ευελιξία της, μπορεί πολύ εύκολα να τροποποιηθεί και να ενσωματώσει καινούργιες συναρτήσεις, να τροποποιήσει ή και να αλλάξει εντελώς το schema στο οποίο υπακούει ανάλογα με τις ανάγκες κάθε εφαρμογής. Μπορεί να χρησιμοποιηθεί από έναν απλό χρήστη του Grid ο οποίος ορίζει τις δικές του συναρτήσεις ανάλογα με τις ανάγκες του, για να υποβάλει ερώτηση σε ένα GIS, ακόμη από το ίδιο το GIS το οποίο όμως κοινοποιεί τις υλοποιημένες συναρτήσεις του στους χρήστες και δίνει DataLog interface.

Αν και η βασική λειτουργικότητα που θέλαμε να πετύχουμε παρέχεται ήδη, βελτιώσεις μπορούν να γίνουν σε ορισμένους τομείς ώστε να η υλοποίηση μας να γίνει ακόμη πιο ευέλικτη και πιο επεκτάσιμη - γενικευμένη.

Αρχικά θα μπορούσε να βελτιωθεί η διεπαφή με το χρήστη (interface) η οποία σε αυτή την φάση είναι αρκετά απλή, θα μπορούσε ίσως να δοθεί σαν web service συμβατό με τα πρότυπα OGSA – OGSI για να ενσωματωθεί σε πραγματικά συστήματα Grid. Επιπλέον το schema το οποίο θα συμβουλευόμαστε θα μπορούσε να αρχικοποιείται από εξωτερικό αρχείο για να υποστηρίζει και άλλα σχήματα, να επεκτείνει το τρέχων σχήμα, ή να ορίσει νέες εικονικές συναρτήσεις πιο εύκολα. Στην τρέχουσα κατάσταση αυτό γίνεται από μια συνάρτηση, και ενώ μπορεί εύκολα να αλλάξει, θα πρέπει να ξανακάνουμε compile τον κώδικα. Τέλος θα μπορούσαμε να υποστηρίξουμε κανόνες DataLog. Αυτό δεν προσθέτει κάτι επιπλέον στη λειτουργικότητα αλλά το μειονέκτημα της τρέχουσας έκδοσης είναι ότι δεν δίνει ευελιξία στο ποιες μεταβλητές πρέπει να επιστραφούν από τη σημαντική της DataLog (DataLog semantics).

# Παράρτημα Α

## A.1 GLUE CE Relational Schema

Παρατίθεται το σχεσιακό σχήμα για το Computing Element, όπως δίνεται από το [18]

### Glue-CE R-GMA schema

### GlueCE

CE Glue Schema Table

| | | |
|---|---|---|
| **UniqueID** | VARCHAR(128) | A CE Unique ID |
| Name | VARCHAR(255) | name of this CE could be the name of the local queue associated with it |
| GlueClusterUniqueID | VARCHAR(100) | Relates to GlueCluster |
| TotalCPUs | INT | Number of CPUs available to the queue |
| LRMSType | VARCHAR(255) | Name of local resource management system |
| LRMSVersion | VARCHAR(255) | Version of local resource management system |
| GRAMVersion | VARCHAR(255) | The GRAM version |
| HostName | VARCHAR(128) | Fully qualified hostname for host where gatekeeper runs |
| GatekeeperPort | VARCHAR(128) | Port number for the gatekeeper |
| RunningJobs | INT | Number of jobs in a running state |
| WaitingJobs | INT | Number of jobs that are in a state different than running |
| TotalJobs | INT | Number of jobs in the CE |
| Status | VARCHAR(255) | States a queue can be in |
| WorstResponseTime | INT | Worst time between job submission till when job |

| | | |
|---|---|---|
| | | starts its execution |
| EstimatedResponseTime | INT | Estimated time between job submission till when job starts its execution |
| FreeCpus | INT | Number of free CPUs available to a scheduler |
| Priority | INT | Info about the Queue Priority |
| MaxRunningJobs | INT | The maximum number of jobs allowed to be running |
| MaxTotalJobs | INT | The maximum allowed number of jobs in the queue |
| MaxCPUTime | INT | The maximum CPU time allowed for jobs submitted to the CE in mins |
| MaxWallClockTime | INT | The maximum wall clock time allowed for jobs submitted to the CE in mins |
| InformationServiceURL | VARCHAR(128) | URL of the GRIS |

## GlueCEAccessControlBaseRule

Info of a VO which users are allowed to access the CE

| | | |
|---|---|---|
| **GlueCEUniqueID** | VARCHAR(128) | Relates users to CE |
| **Value** | VARCHAR(128) | The rule that grant/deny access of this service |

## GlueCluster

A cluster (a cluster serves many CEs)

| | | |
|---|---|---|
| **UniqueID** | VARCHAR(100) | The unique Identifier for the cluster |
| Name | VARCHAR(255) | The name of the Cluster |
| InformationServiceURL | VARCHAR(128) | URL of the GRIS |

## GlueSubCluster

A subcluster (part of a cluster)

| | | |
|---|---|---|
| **UniqueID** | VARCHAR(100) | A unique ID for the SubCluster |
| Name | VARCHAR(255) | The name of the SubCluster |
| GlueClusterUniqueID | VARCHAR(255) | Relates to GlueCluster (instead of GlueChunkKey) |
| RAMSize | INT | The amount of RAM |
| RAMAvailable | INT | The amount of free RAM |
| VirtualSize | INT | The amount of virtual memory |
| VirtualAvailable | INT | The amount of available virtual memory |
| PlatformType | VARCHAR(128) | Informally describes the platform type of the computing element |
| SMPSize | INT | Informally describes the platform type of the computing element |
| OSName | VARCHAR(255) | The name of the operating system |
| OSRelease | VARCHAR(255) | The release of the operating system |
| OSVersion | VARCHAR(255) | The version of the operating system |
| Vendor | VARCHAR(255) | Name of the CPU vendor |
| Model | VARCHAR(255) | Model of the CPU |
| Version | VARCHAR(255) | Version of the CPU |
| ClockSpeed | INT | MHz associated with the CPUs |
| InstructionSet | VARCHAR(255) | Processor instruction set |
| OtherProcessorDescription | VARCHAR(255) | Other processor description |
| CacheL1 | INT | first-level unified cache size (in kb) of a cpu |
| CacheL1I | INT | first-level instruction cache size (in kb) of a cpu |
| CacheL1D | INT | first-level data cache size (in kb) of a cpu |
| CacheL2 | INT | second-level unified cache size (in kb) of a cpu |
| BenchmarkSF00 | INT | SpecFloat2000 of the nodes associated to the subcluster |
| BenchmarkSI00 | INT | SpecInt2000 of the nodes associated to the |

| | | |
|---|---|---|
| | | subcluster |
| InboundIP | VARCHAR(1) | "y" if inbound IP connections are allowed - otherwise "n" |
| OutboundIP | VARCHAR(1) | "y" if outbound IP connections are allowed - otherwise "n" |
| InformationServiceURL | VARCHAR(128) | URL of the GRIS |

## GlueSubClusterSoftwareRunTimeEnvironment

Relates software to SubCluster

| | | |
|---|---|---|
| **Value** | VARCHAR(255) | Application software |
| **GlueSubClusterUniqueID** | VARCHAR(100) | Relates software to SubCluster |

## GlueHostRemoteFileSystem

A remote file system accessed via a network-related service

| | | |
|---|---|---|
| **GlueSubClusterUniqueID** | VARCHAR(245) | Relates to GlueSubCluster (instead of GlueChunkKey) |
| **Name** | VARCHAR(245) | The name for the file system |
| Root | VARCHAR(255) | Path name or other information defining the root of the file system |
| Size | INT | Size of the file system in bytes |
| AvailableSpace | INT | Amount of free space in bytes for the file system |
| ReadOnly | VARCHAR(5) | Is this read only |
| Type | VARCHAR(128) | File system type |

## A.2 Γραμματική συντακτικού αναλυτή

letter [a-zA-Z]

digit [0-9]

integerValue {digit}+

floatValue {digit}*"."{digit}+(("E"|"e")("+"|"-")?{digit}+)?

boolValue ([Tt][Rr][Uu][Ee])|([Ff][Aa][Ll][Ss][Ee])

and_op ([Aa][Nn][Dd])

or_op ([Oo][Rr])

mod ([Mm][oO][Dd])

simpleCharValue [A-Za-z0-9`~!@#$%^&*()\-_+=|\[\]{};:,./? ]

charEscapeSeq "\\"[ar0tn\'\"\\]

charValue "\'"({simpleCharValue}|{charEscapeSeq})"\'"

string "\""(({simpleCharValue}|{charEscapeSeq})*)"\""

value {string}|{charValue}|{boolValue}|{floatValue}|{integerValue}

pred [a-z]({letter}|{digit}|[_])*

var ([_])|((([_]{letter})|([A-Z]))({letter}|{digit}|[_])*)

“%” : comments until new line

VALUE : boolValue | integerValue | floatValue | string | charValue

VARIABLE : var

PREDICATE :pred

PLUS_MINUS : “+” | “-“

MUL_DIV : “*” | “/”

MOD

GL : “>” | “<” | “<=” | ”>=”

EQ : “=” | “!=”

AND_OP : and_op

OR_OP : or_op

GIVEN : “:-“

END_CLAUSE :”.”

primary_expr : VARIABLE | PREDICATE '(' argument_expr_list ')' | VALUE
| '(' expr ')'
argument_expr_list : VARIABLE | VALUE | argument_expr_list ',' VARIABLE
| argument_expr_list ',' VALUE
unary_expr : primary_expr | PLUS_MINUS primary_expr
math_expr : unary_expr | math_expr MUL_DIV unary_expr
| math_expr MOD unary_expr | math_expr PLUS_MINUS unary_expr
eq_expr: math_expr | eq_expr GL math_expr | eq_expr EQ math_expr
logical_and_expr :eq_expr | logical_and_expr AND_OP eq_expr
expr: logical_and_expr | expr OR_OP logical_and_expr
Query : '?' expr END_CLAUSE

*Οι παρακάτω 2 κανόνες υπάρχουν στην γραμματική αλλά για μελλοντική υποστήριξη :*
Defpred : PREDICATE '(' argument_expr_list ')' GIVEN expr END_CLAUSE
defpred_expr_list : defpred | defpred_expr_list defpred

Program : query | defpred_expr_list query

# Βιβλιογραφία – Αναφορές


1. Mark Baker, Rajkumar Buyya and Domenico Laforenza: The Grid: International Efforts in Global Computing.
2. Fran Berman, Geoffrey Fox, and Tony Hey: The Grid: past, present, future.
3. Ian Foster, Carl Kesselman, Steven Tuecke: The Anatomy of the Grid, Enabling Scalable Virtual Organizations.
4. GGF, **http://www.gridforum.org**.
5. OGSA, http://www.globus.org/ogsa.
6. OGSI, http://www.ggf.org/ogsi-wg.
7. The Globus Project Web Site, **http://www.globus.org**.
8. Globus MDS, **http://www.globus.org/mds**.
9. R-GMA - "DataGrid Information and Monitoring Services Architecture: Design, Requirements and Evaluation Criteria", Technical Report, DataGrid, 2002.
10. Legion Worldwide Virtual Computer Home Page, **http://legion.virginia.edu**.
11. The Condor Project, **http://www.cs.wisc.edu/condor/**.
12. Chuang Liu, Lingyun Yang, Ian Foster, Dave Angulo, 2002, Design and Evaluation of a resource selection framework for Grid applications.
13. Carole A. Goble, David De Roure, Nigel R. Shadbolt, and Alvaro A. A. Fernandes, Enhancing Services and Applications with Knowledge and Semantics
14. Nicholas Coleman, Rajesh Raman, Miron Livny and Marvin Solomon, 2003, Distributed Policy Management and Comprehension with Classified Advertisements.
15. Hongsuda Tangmunarunkit, Stefan Decker, Carl Kesselman, 2004, Ontology-based Resource Matching in the Grid - The Grid meets the Semantic Web.
16. DataTAG Project, **http://www.cnaf.infn.it/~sergio/datatag/**.
17. GLUE Schema Official Documents, **http://www.cnaf.infn.it/~sergio/datatag/glue/**.
18. GLUE CE RDBMS Schema, **http://hepunx.rl.ac.uk/edg/wp3/documentation/doc/schemas/Glue-CE.htm**.
19. DataGrid, **http://www.eu-datagrid.org/**.
20. Jeffrey D. Ullman, Principles of DataBase and Knowledge – Base Systems