ΕΝΕΡΓΕΣ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

Θ. ΧΩΜΑΤΙΔΟΥ, Α. ΤΣΩΗΣ

6.1 ΕΙΣΑΓΩΓΗ

Τα Ενεργά Συστήματα Βάσεων Δεδομένων συνδυάζουν την τεχνολογία των Συστημάτων Βάσεων Δεδομένων με τον Προγραμματισμό Βάσει Κανόνων. Με αυτόν τον τρόπο επεκτείνουν την έννοια της Βάση Δεδομένων, επιτρέποντάς της να περιέχει ενεργά στοιχεία (κανόνες). Χρησιμοποιώντας αυτά τα ενεργά στοιχεία, ο σχεδιαστής της Βάσης Δεδομένων μπορεί πλέον να προκαθορίσει την συμπεριφορά της βάσης απέναντι στα διάφορα πιθανά γεγονότα.

Τα *Ενεργά Συστήματα Βάσεων Δεδομένων (ΕΣΒΔ)* παρουσιάζουν (ή τουλάχιστον πρέπει να παρουσιάζουν) ορισμένα χαρακτηριστικά, ανεξαρτήτως των εφαρμογών που εξυπηρετούν [GDD95]:

- Ένα ΕΣΒΔ *είναι ένα Σύστημα Βάσης Δεδομένων*, επομένως απαιτείται να έχει όλα τα στοιχεία και τις δυνατότητες ενός παραδοσιακού (που στη συνέχεια θα αποκαλείται και *παθητικό*) συστήματος. Δηλαδή, εάν ένας χρήστης αγνοεί όλα τα ενεργά χαρακτηριστικά μπορεί να δουλέψει με το ενεργό σύστημα όπως και με το παθητικό.
- Ένα ΕΣΒΔ *έχει ένα μοντέλο γνώσης*. Το ΕΣΒΔ υποστηρίζει την ενεργή συμπεριφορά και απαιτεί την δυνατότητα καθορισμού κανόνων, οι οποίοι μαζί με τις δυνατότητες καθορισμού δεδομένων αποτελούν το μοντέλο γνώσης.
- Ένα ΕΣΒΔ πρέπει να *μπορεί να αντιδρά σε γεγονότα όλων των ειδών,* ορισμένα από τον χρήστη, την εφαρμογή ή το σύστημα. Φυσικά, ο ορισμός, η αναγνώριση και η σηματοδότηση των γεγονότων ακολουθεί τους σημασιολογικούς κανόνες έτσι όπως αυτοί έχουν οριστεί.
- Ένα ΕΣΒΔ *έχει ένα μοντέλο εκτέλεσης*. Το ΕΣΒΔ αποτιμά συνθήκες, εκτελεί πράξεις και έχει ορισμένη σημασιολογία εκτέλεσης και επεξεργασίας κανόνων, καθώς και ορισμένη σειρά εκτέλεσης κανόνων σε περιπτώσεις σύγκρουσης. Το μοντέλο εκτέλεσης πρέπει να υπακούει στους σημασιολογικούς κανόνες και να βρίσκεται σε συμφωνία με το μοντέλο δοσοληψιών.

- Ένα ΕΣΒΔ *μπορεί να παρουσιάζει πληροφορίες σχετικά με τους κανόνες*, όπως και με τα υπόλοιπα δεδομένα της βάσης.

Τέλος, ένα ΕΣΒΔ πρέπει να είναι εύχρηστο και αποτελεσματικό. Εύχρηστο είναι όταν υπάρχει διαθέσιμη μία γλώσσα ορισμού κανόνων και ένα προγραμματιστικό περιβάλλον που να καθιστά την εργασία αποδοτική (μεταγλωττιστές, ανιχνευτές λαθών, εργαλεία σχεδιασμού, κ.λ.π.). Αποτελεσματικό είναι όταν η λύση που προσφέρει έχει καλύτερη απόδοση από ισοδύναμες λύσεις ¨πάνω¨ από ένα παθητικό σύστημα.

6.2 ΜΟΝΤΕΛΑ ΚΑΙ ΓΛΩΣΣΕΣ ΚΑΝΟΝΩΝ

Βασική έννοια στις Ενεργές Βάσεις Δεδομένων είναι αυτή των *κανόνων (rules)*. Οι κανόνες ορίζονται από τους χρήστες, από τις εφαρμογές, από τους διαχειριστές της βάσης και καθορίζουν την επιθυμητή ¨ενεργή¨ συμπεριφορά. Στην πιο γενική μορφή τους, οι κανόνες αποτελούνται από τα εξής μέρη:

- *Γεγονός (event),* το οποίο προκαλεί την ενεργοποίηση του κανόνα.
- *Συνθήκη (condition),* η οποία ελέγχεται όταν ενεργοποιείται ο κανόνας.
- *Πράξη (action),* η οποία εκτελείται όταν ενεργοποιείται ο κανόνας και ικανοποιείται η συνθήκη.

Κατ' αρχάς ορίζεται ένα σύνολο κανόνων και το ενεργό σύστημα βάσης δεδομένων παρατηρεί τα γεγονότα που λαμβάνουν χώρα. Για κάθε κανόνα, όταν συμβεί το γεγονός που τον ενεργοποιεί, ελέγχεται η συνθήκη που τον συνοδεύει και εφόσον αυτή είναι αληθής, εκτελείται η πράξη.

Χρησιμοποιούνται στην βιβλιογραφία πολλές ονομασίες που περιγράφουν τους κανόνες των ενεργών συστημάτων βάσεων δεδομένων: *κανόνες παραγωγής ή εμπρόθετης αλυσίδας (production or forward-chaining rules), κανόνες ECA (Event-Condition-Action rules), κανόνες κατάστασης-πράξης (situation-action rules), ενεργοποιητές (triggers), προειδοποιητές (alerters).*

Οι κανόνες παραγωγής έχουν τις ρίζες τους στους κανόνες που συναντάμε στην Τεχνητή Νοημοσύνη, οι οποίοι έχουν την μορφή:

πρότυπο →πράξη

και ονομάζονται *κανόνες βασισμένοι σε πρότυπα,* ή *κανόνες CA (Condition -Action)*. Ο κανόνας ενεργοποιείται όταν το πρότυπο ταιριάζει απόλυτα με τα δεδομένα που υπάρχουν στην μνήμη, και η πράξη μεταβάλλει την μνήμη. Στις γλώσσες που υποστηρίζουν τέτοιου είδους κανόνες, θεωρούμε ότι ένας κανόνας ενεργοποιείται μόνο όταν υπάρχουν νέα δεδομένα στην μνήμη που ταιριάζουν με το πρότυπο. Επομένως, αυτοί οι κανόνες ενεργοποιούνται μόνο από γεγονότα, όπως η εισαγωγή, η διαγραφή και η αλλαγή δεδομένων.

Οι κανόνες που έχουν την γενική μορφή που περιγράψαμε παραπάνω (γεγονός - συνθήκη - πράξη *ECA*), καλούνται *κανόνες βασισμένοι σε γεγονότα*:

```
όταν συμβαίνει το γεγονός
εάν ισχύει η συνθήκη
τότε κάνε....
```

και ο κανόνας ενεργοποιείται όταν λαμβάνει χώρα το αντίστοιχο γεγονός.

Μερικά ενεργά συστήματα βάσεων δεδομένων υποστηρίζουν το ένα ή το άλλο είδος κανόνων ή και τα δύο. Υπάρχουν σαφείς διαφορές μεταξύ των κανόνων ECA και CA (Condition - Action). Όταν συμπεριλαμβάνεται το γεγονός που ενεργοποιεί τον κανόνα στον ορισμό του, υπάρχει δυνατότητα να καθοριστούν διαφορετικές πράξεις ανάλογα με το γεγονός που ενεργοποίησε τον κανόνα ακόμα και όταν ισχύει η ίδια συνθήκη. Για παράδειγμα, κανείς

μπορεί να αντιδράσει διαφορετικά στην παραβίαση του ίδιου περιορισμού, εάν αυτή η παραβίαση προήλθε από εισαγωγή κάποιας νέας πλειάδας ή από αλλαγή κάποιας ήδη υπάρχουσας. Αυτή η εξειδικευμένη συμπεριφορά δεν είναι δυνατή με τους κανόνες CA. Από την άλλη μεριά, οι κανόνες CA θεωρούνται γενικά πιο ¨δηλωτικοί¨ και μπορούν να προγραμματιστούν πιο εύκολα στην περίπτωση που αποφασίζει το σύστημα, και όχι ο δημιουργός του κανόνα, πότε πρέπει να αποτιμάται και να ελέγχεται η συνθήκη του κανόνα.

Υπάρχουν απλά και σύνθετα γεγονότα [G94]. Θα εξετάσουμε στην συνέχεια αναλυτικά την έννοια των παραμέτρων γεγονότων, που επιτρέπουν την μεταφορά πληροφορίας από το γεγονός στην συνθήκη και την πράξη. Η ενότητα θα κλείσει με την ανάλυση της σημασιολογίας των όρων "Γεγονός", "Συνθήκη", "Πράξη".

6.2.1 Αιτίες που καθορίζουν ένα γεγονός

Σε ένα ΕΣΒΔ, ο όρος ¨γεγονός¨ καθορίζει την αιτία που προκαλεί την ενεργοποίηση του κανόνα. Αυτή η αιτία μπορεί να είναι:

- *Η μεταβολή των δεδομένων*. Σε μια σχεσιακή βάση δεδομένων, η μεταβολή δεδομένων μπορεί να οφείλεται σε μια λειτουργία εισαγωγής, διαγραφής ή ανανέωσης, σε ένα συγκεκριμένο πίνακα. Σε ένα αντικειμενοστρεφές σύστημα, η μεταβολή δεδομένων μπορεί να οφείλεται σε μια εισαγωγή, διαγραφή ή ανανέωση ενός συγκεκριμένου αντικειμένου, ή ακόμη και σε μια κλήση μεθόδου που μεταβάλλει αντικείμενα.
- *Η επεξεργασία δεδομένων*. Σε μια σχεσιακή βάση δεδομένων, η επεξεργασία δεδομένων οφείλεται στην λειτουργία της επιλογής σε έναν συγκεκριμένο πίνακα δεδομένων, ενώ σε ένα αντικειμενοστρεφές σύστημα βάσης δεδομένων, αυτή μπορεί να προσδιοριστεί ως η επιλογή ενός αντικειμένου ή η κλήση μιας μεθόδου που επεξεργάζεται αντικείμενα.
- *Ο χρόνος ή μια εξωτερική αλλαγή*. Ένα χρονικό γεγονός μπορεί να προσδιορίζει την ενεργοποίηση ενός κανόνα μια απόλυτη χρονική στιγμή (π.χ. την 1η Ιανουαρίου 1995 στις 3μμ), μια επαναλαμβανόμενη χρονική στιγμή (π.χ. κάθε μέρα στις 3μμ) ή σε περιοδικά διαστήματα (π.χ. κάθε 30 λεπτά). Ακόμα, ένα εξωτερικό γεγονός (π.χ. η ¨πτώση ¨ του συστήματος) μπορεί να σηματοδοτεί την ενεργοποίηση ενός κανόνα.
- *Η απαίτηση μιας εφαρμογής ή ενός χρήστη*. Όταν ένα γεγονός καθορίζεται από μια εφαρμογή ή ένα χρήστη, κάθε φορά που η εφαρμογή ειδοποιεί το σύστημα βάσης δεδομένων για την εμφάνιση του γεγονότος, κάθε κανόνας που έχει το γεγονός αυτό ως ενεργοποιητή του, ενεργοποιείται. Με αυτήν την προσέγγιση, η εφαρμογή μπορεί να ελέγξει και να υπολογίσει (ακόμη και χωρίς πρόσβαση στην βάση δεδομένων) πότε το συγκεκριμένο γεγονός θα συμβεί και πότε οι αντίστοιχοι κανόνες θα ενεργοποιηθούν.

6.2.2 Απλά γεγονότα

Η δύναμη και η εκφραστικότητα της γλώσσας καθορίζεται από την πολυπλοκότητα και την ποικιλία των δομών που αυτή υποστηρίζει, και συγκεκριμένα, όσον αφορά στις γλώσσες των κανόνων, από τις δομές που χρησιμοποιούνται για την περιγραφή των γεγονότων. Δεν αρκεί να περιγράφουμε γεγονότα που συμβαίνουν σε μία βάση δεδομένων, όπως η αρχή ή το τέλος μια διαδικασίας, αλλά πρέπει να μπορούμε να υποστηρίζουμε και γεγονότα που σχετίζονται με την αρχή ή το τέλος μιας δοσοληψίας, με ένα συμβάν στο σύστημα που λαμβάνει χώρα μια προκαθορισμένη χρονική στιγμή, κ.ά. Για να υποστηρίξει μια μεγάλη πληθώρα εφαρμογών από διάφορα πεδία, ένα ενεργό σύστημα βάσεων δεδομένων πρέπει να παρέχει πολλές δομές περιγραφής γεγονότων. Οι σημαντικότερες κατηγορίες απλών γεγονότων είναι τα εξής:

- *Χρονικά γεγονότα*: Τα γεγονότα αυτά σηματοδοτούνται όταν φθάσει η κατάλληλη χρονική στιγμή.
- *Γεγονότα μεταβολής δεδομένων:* Τα σημαντικότερα γεγονότα στις σχεσιακές βάσεις δεδομένων είναι η εκτέλεση των πράξεων της εισαγωγής, διαγραφής και μεταβολής (insert, delete, update) δεδομένων. Τα γεγονότα αυτά μπορούν να οριστούν πάνω σε μία συγκεκριμένη σχέση ή σε ορισμένα πεδία μιας σχέσης. Ακόμη, μπορεί να καθοριστεί αν το γεγονός ενεργοποιείται πριν την λειτουργία, ή μετά την ολοκλήρωσή της, με τις λέξεις κλειδιά *BEFORE* και *AFTER*.

  Στις αντικειμενοστρεφείς βάσεις αυτές οι πράξεις γίνονται μέσα από τις μεθόδους των αντικειμένων και τα αντίστοιχα γεγονότα μπορούν να καταταγούν στην κατηγορία "γεγονότα βασισμένα στις τιμές" που περιγράφεται στην συνέχεια.
- *Γεγονότα βασισμένα σε μεθόδους*: Σε ένα αντικειμενοστρεφές περιβάλλον, οι χρήστες χειρίζονται και προσπελαύνουν τα αντικείμενα στέλνοντας μηνύματα σε αυτά. Κάθε τέτοιο μήνυμα προκαλεί την εκτέλεση μιας μεθόδου, και η εκτέλεση μιας μεθόδου σηματοδοτεί ένα γεγονός. Πρέπει να καθορίζεται εάν το γεγονός είναι το σημείο στον χρόνο ακριβώς προτού το αντικείμενο αρχίσει την εκτέλεση της μεθόδου, ή ακριβώς μετά την ολοκλήρωση αυτής. Ο καθορισμός γίνεται με την χρήση των λέξεων κλειδιών *BEFORE* και *AFTER,* και η σύνταξη που καθορίζει το γεγονός είναι:

  ```
  ( BEFORE | AFTER ) ¨.¨ ( class_name | object_name |
      [class_name] ¨*¨ ) ¨. ¨method_name
  ```

  Ένα γεγονός βασισμένο σε μέθοδο σχετίζεται

  - είτε με μια κλάση, εάν δίνεται το όνομά της, οπότε σηματοδοτείται πριν ή μετά από τη εκτέλεση της κατάλληλης μεθόδου πάνω σε οποιοδήποτε αντικείμενο αυτής της κλάσης,
  - είτε με ένα συγκεκριμένο αντικείμενο, εάν καθορίζεται το όνομα του αντικειμένου, οπότε σηματοδοτείται πριν ή μετά από την εκτέλεση της μεθόδου πάνω στο συγκεκριμένο αντικείμενο,
  - είτε με πολλές κλάσεις, οπότε σηματοδοτείται όταν η μέθοδος εκτελείται πάνω σε οποιοδήποτε αντικείμενο οποιαδήποτε κλάσης, όπου έχει οριστεί μια μέθοδος με αυτό το όνομα. Αυτό μπορεί να συμβεί λόγω της κληρονομικότητας μεταξύ των κλάσεων, ή εάν το μοντέλο επιτρέπει τον ορισμό κλάσεων με το ίδιο όνομα και διαφορετικό κυρίως σώμα.

Σε πολλές περιπτώσεις, το γεγονός αναφέρεται σε *γενικές* (*generic*) λειτουργίες, όπως η διαγραφή ή η δημιουργία αντικειμένων. Σε μερικά αντικειμενοστρεφή συστήματα, αυτές οι λειτουργίες παρουσιάζονται ως μέθοδοι: η μέθοδος δημιουργίας ή διαγραφής μιας κλάσης. Επομένως, ο ορισμός ενός γεγονότος πάνω σε μια γενική λειτουργία ισοδυναμεί με τον ορισμό ενός γεγονότος βασισμένου σε μέθοδο πάνω στον τελεστή δημιουργίας ή διαγραφής.

- *Γεγονότα βασισμένα σε τιμές*: Ένα γεγονός μπορεί να σχετιστεί με την μεταβολή της τιμής ενός αντικειμένου με πολλές διαφορετικές μεθόδους, οπότε για να οριστεί θα έπρεπε να καθοριστεί κάθε μία ξεχωριστά από τις περιπτώσεις. Αντ' αυτού, έχουμε τα γεγονότα βασισμένα στις τιμές. Μία λειτουργία πάνω στην τιμή μπορεί να είναι μία ανανέωση ή μία επεξεργασία της τιμής. Επομένως, καθορίζουμε την σηματοδότηση του γεγονότος κάθε φορά που μια λειτουργία τιμής εκτελείται πάνω σε ένα οποιοδήποτε αντικείμενο μιας δεδομένης κλάσης ή σε ένα συγκεκριμένο αντικείμενο. Ακόμη, μπορεί να καθοριστεί αν το

γεγονός ενεργοποιείται πριν την λειτουργία, ή μετά την ολοκλήρωσή της, με τις λέξεις κλειδιά *BEFORE* και *AFTER*.

```
DEFINE EVENT E1 BEFORE/AFTER ¨.¨ (class-name | object-name)
   ¨.update ( ¨attr-name ¨ )¨
```

Η ακριβής σύνταξη διαφέρει στα διάφορα αντικειμενοστρεφή συστήματα. Ο ορισμός αυτού του είδους των γεγονότων προϋποθέτει ότι ο χρήστης μπορεί να προσπελάσει την τιμή των αντικειμένων. Αυτό δεν είναι δυνατόν αν οι τιμές αυτές έχουν οριστεί ως *ιδιωτικές* (*private*), επομένως τα γεγονότα αυτά είναι ορατά μόνο στον προγραμματιστή της αντίστοιχης κλάσης και καλούνται *κανόνες εσωτερικής κλάσης* (*class-defined rules*).

- *Γεγονότα βασισμένα σε δοσοληψίες*: Καθορίζονται από τη αρχή, το τέλος, την αποβολή ή την επιτυχημένη ολοκλήρωση μιας δοσοληψίας -αντίστοιχα *BOT, EOT, ABORT* ή *COMMIT*. Για παράδειγμα, η σύνταξη μπορεί να είναι:

  ```
  DEFINE EVENT E1 ON BOT | EOT | ABORT | COMMIT.
  ```

  Τα γεγονότα αυτά ενεργοποιούνται για κάθε δοσοληψία. Για ετερογενή συστήματα (π.χ. κατανεμημένα) έχουμε και άλλες λειτουργίες, όπως pre-commit, κ.ά. Όμως, εάν θέλουμε να καθορίσουμε γεγονότα σε συγκεκριμένες δοσοληψίες, θα πρέπει να μας παρέχεται από το μοντέλο δοσοληψιών η δυνατότητα καθορισμού ονομαστικών δοσοληψιών. Σε αυτή την περίπτωση, μπορεί να γίνει διάκριση μεταξύ των τύπων των δοσοληψιών και των στιγμιοτύπων τους. Μια *ονομαστική δοσοληψία* είναι η εξειδίκευση ενός τύπου δοσοληψίας, όπου οι εκτελέσεις της διαδικασίας καλούνται *στιγμιότυπα*. Εάν γίνεται από το σύστημα αυτή η διάκριση, μπορούν να οριστούν γεγονότα βασισμένα σε τύπους δοσοληψιών και όχι απλά σε εκτελέσεις αυτών. Τα γεγονότα αυτά ονομάζονται *γεγονότα βασισμένα σε ονομαστικές δοσοληψίες* και ορίζονται σύμφωνα με το ακόλουθο συντακτικό:

  ```
  DEFINE EVENT E1
   ON ( BOT | EOT | ABORT | COMMIT ) [transaction_name]
  ```

- *Γεγονότα ορισμένα από τον χρήστη:* Οι χρήστες ή οι εφαρμογές μπορούν να ορίσουν γεγονότα ανάλογα με τις σημασιολογικές δυνατότητες που παρέχει το σύστημα. Τα γεγονότα αυτά ορίζονται και προσπελαύνονται από τους χρήστες σύμφωνα με το ακόλουθο συντακτικό:

  ```
  DEFINE EVENT event_name...
  ```

  και χρησιμοποιούνται στους κανόνες όπως όλα τα υπόλοιπα είδη γεγονότων. Σε αυτούς μπορούν να υπάρξουν και παράμετροι (που θα μεταφέρουν πληροφορία στην συνθήκη και την πράξη του κανόνα) -με αυτό το θέμα θα ασχοληθούμε στην υποενότητα ¨Παράμετροι γεγονότων¨. Για παράδειγμα, *DEFINE EVENT program-test(program-name, date)...* Τα γεγονότα αυτά μπορεί να ανιχνεύονται από το σύστημα, μπορεί όμως και όχι. Στην τελευταία περίπτωση πρέπει οι χρήστες να ειδοποιούν το σύστημα με μια ειδική λειτουργία:

  ```
  RAISE EVENT event_name...
  ```

  όπου η χρονική στιγμή της λειτουργίας *raise event* είναι η χρονική στιγμή του γεγονότος.

6.2.3 Σύνθετα Γεγονότα

Στην αποτίμηση κανόνων με βάση απλά γεγονότα, αρκεί η εμφάνιση ενός και μόνου σχετικού γεγονότος για την εκκίνηση της διαδικασίας αποτίμησης. Είναι δυνατόν, όμως, να οριστούν και *σύνθετα γεγονότα*, είτε με την χρήση τελεστών είτε με τον συνδυασμό απλών και σύνθετων γεγονότων. Στην συνέχεια παρουσιάζουμε έξι τελεστές γεγονότων, οι τρεις πρώτοι

από αυτούς έχουν πληθικό αριθμό δύο (δηλαδή, συνδέουν δύο απλά ή σύνθετα γεγονότα) και οι τρεις επόμενοι έχουν μοναδιαίο πληθικό αριθμό (δηλαδή, εφαρμόζονται σε ένα μόνο γεγονός).

- *Σύζευξη*: Ο τελεστής σύζευξης δύο γεγονότων *E1* και *E2* σημειώνεται ως *(E1, E2)* και σηματοδοτείται όταν και τα δύο γεγονότα έχουν λάβει χώρα, άσχετα με την σειρά εμφάνισής τους. Το σύνθετο γεγονός αντιστοιχεί στην χρονική στιγμή όπου το δεύτερο γεγονός κάνει την εμφάνισή του.
- *Αλληλουχία*: Ο τελεστής αλληλουχίας δύο γεγονότων *E1* και *E2* σημειώνεται ως *(E1 **;** E2),* όπου τα *E1* και *E2* μπορούν να είναι απλά ή σύνθετα γεγονότα, και σηματοδοτείται όταν πρώτα το *E1* και στην συνέχεια το *E2* έχουν λάβει χώρα. Το σύνθετο γεγονός *(E1 **;** E2)* θεωρείται ότι εμφανίζεται στη χρονική στιγμή του γεγονότος *E2*.
- *Διάζευξη*: Ο τελεστής διάζευξης δύο γεγονότων *E1* και *E2* σημειώνεται ως *(E1 | E2)* και σηματοδοτείται όταν είτε το γεγονός *E1* είτε το γεγονός *E2* έχουν εμφανιστεί. Η δυνατότητα διάζευξης δεν συντελεί τόσο στην υποστήριξη πολυπλοκότερων γεγονότων όσο στον σαφέστερο και συντομότερο ορισμό κανόνων, αφού ο ίδιος κανόνας μπορεί να ενεργοποιείται από περισσότερα του ενός γεγονότα. Με τον τελεστή διάζευξης αποφεύγεται η επανάληψη ορισμού κανόνων. Ιδιαίτερη προσοχή πρέπει να δοθεί στην περίπτωση των σύγχρονων γεγονότων, όπου ο τελεστής αυτός πρέπει να ορίζεται με σαφήνεια.
- *Μοναδικότητα*: Ο τελεστής μοναδικότητας χρησιμοποιείται όπου ένα γεγονός πρέπει να σηματοδοτείται μόνο μια φορά μέσα σε ένα προκαθορισμένο χρονικό διάστημα. Ένα σύνθετο γεγονός *E* ορισμένο με τον τελεστή μοναδικότητας σημειώνεται ως **E IN I,* όπου *I* το χρονικό διάστημα που μας ενδιαφέρει, και θα ενεργοποιήσει τον αντίστοιχο κανόνα μόνο με την πρώτη του εμφάνιση στη διάρκεια του διαστήματος *I*. Όταν δεν καθορίζεται το χρονικό διάστημα *I,* εννοείται το διάστημα από την τρέχουσα χρονική στιγμή ορισμού του σύνθετου γεγονότος έως το άπειρο. Σε αυτή την περίπτωση το γεγονός θα σηματοδοτηθεί μόνο μια φορά, επομένως χρήσιμο θα ήταν μέσα στην πράξη του κανόνα να καθορίζεται και η διαγραφή του γεγονότος, ώστε να αποφεύγεται άσκοπη παρατήρηση.
  Μια παραλλαγή του τελεστή μοναδικότητας, είναι και τελεστής *last (E) IN I,* που σηματοδοτεί την τελευταία εμφάνιση ενός γεγονότος *E* μέσα σε μια προκαθορισμένη περίοδο *I*. Ο καθορισμός του χρονικού διαστήματος *I* είναι υποχρεωτικός, γιατί διαφορετικά είναι αδύνατον να περιοριστεί το διάστημα μέσα στο οποίο θα σηματοδοτηθεί το γεγονός. Η διαφορά των δύο τελεστών είναι ότι στη μεν πρώτη περίπτωση η ενεργοποίηση του κανόνα θα γίνει αμέσως μετά την εμφάνιση του γεγονότος, ενώ στη δεύτερη περίπτωση θα γίνει μετά το πέρας του χρονικού διαστήματος *I*.
- *Πολλαπλή εμφάνιση*: Ο τελεστής πολλαπλής εμφάνισης του γεγονότος *E* σημειώνεται ως *TIMES (n, E) IN I,* και σηματοδοτείται κάθε φορά που το γεγονός εμφανίζεται *n* φορές μέσα στο χρονικό διάστημα *I*. Η συχνότητα *n* δείχνει τον αριθμό των επαναλήψεων του γεγονότος προτού ενεργοποιηθεί ο αντίστοιχος κανόνας. Εάν δεν καθορίζεται χρονικό διάστημα *I,* το σύστημα θεωρεί το διάστημα από την τρέχουσα χρονική στιγμή του ορισμού του σύνθετου γεγονότος έως το άπειρο.
  Επιπλέον, διακρίνουμε και δύο άλλες παραλλαγές του τελεστή. Μπορεί ο συντελεστής συχνότητας να είναι ένα διάστημα ανάμεσα από δύο συχνότητες, οπότε έχουμε *TIMES ([n1 - n2], E) IN I,* όπου το γεγονός σηματοδοτείται όταν κάνει την εμφάνισή του περισσότερες ή ίσες φορές με *n1* και λιγότερες ή ίσες φορές με *n2* μέσα στο διάστημα *I*. Ακόμη, μπορεί ο συντελεστής συχνότητας να έχει την μορφή *TIMES ([>n], E) IN I,* όπου

το γεγονός *E* σηματοδοτείται στο τέλος του διαστήματος *I* εάν έχει εμφανιστεί περισσότερες φορές από *n*. Ο ορισμός *TIMES ([>1], E) IN I* επιτρέπει την άθροιση των εμφανίσεων ενός γεγονότος στην διάρκεια του χρονικού διαστήματος *I* και την περαιτέρω επεξεργασία αυτής της πληροφορίας. Φανερό είναι ότι σε αυτές τις περιπτώσεις ο καθορισμός του χρονικού διαστήματος *I* είναι υποχρεωτικός.

- *Άρνηση:* Ο τελεστής άρνησης ενός γεγονότος *E* σημειώνεται ως *NOT E IN I,* όπου το *I* είναι το χρονικό διάστημα μέσα στο οποίο το γεγονός *E* δεν έχει εμφανιστεί. Μετά το πέρας του διαστήματος *I,* και αν ισχύει η μη εμφάνιση του γεγονότος *E,* ενεργοποιείται ο αντίστοιχος κανόνας.

Εκτός από τους τελεστές που παρουσιάστηκαν παραπάνω, σύνθετα γεγονότα προκύπτουν και ως συνδυασμοί απλών και σύνθετων γεγονότων σε ¨φωλιασμένες¨ δομές. Σε αυτή την περίπτωση, πρέπει να καθορίζεται από το σύστημα η σειρά προτεραιότητας των τελεστών. Σε γενικές γραμμές, όλοι οι συνδυασμοί είναι δυνατοί, όμως δεν έχουν και όλοι φυσική σημασία.

6.2.4 Παράμετροι γεγονότων

Ένα άλλο θέμα που σχετίζεται με την γλώσσα κανόνων είναι η σύνδεση της βάσης δεδομένων την στιγμή της εμφάνισης του γεγονότος με την εκτέλεση του κανόνα, δηλαδή το πέρασμα πληροφορίας από το γεγονός στην συνθήκη και την πράξη. Οι πραγματικές παράμετροι αντιστοιχίζονται στις τυπικές κατά την σηματοδότηση του γεγονότος.

- *Παράμετροι απλών γεγονότων***:** Τα *χρονικά γεγονότα* δεν έχουν παραμέτρους, διότι συμβαίνουν μέσα στην βάση δεδομένων και δεν συνδέονται με τα δεδομένα. Όλα τα άλλα είδη γεγονότων έχουν δύο τύπους παραμέτρων, τις περιβαλλοντολογικές και αυτές που έχουν άμεση σχέση με την φύση των γεγονότων. Περιβαλλοντολογικές παράμετροι είναι:
    - *occ_point (E):* η χρονική στιγμή της σηματοδότησης του γεγονότος *E*
    - *occ_tid (E):* το αναγνωριστικό της δοσοληψίας κατά την διάρκεια της οποίας συμβαίνει το γεγονός E
    - *user_id (E):* το αναγνωριστικό του χρήστη που ξεκίνησε την δοσοληψία κατά την διάρκεια της οποίας συμβαίνει το γεγονός E

  Στα *γεγονότα που είναι βασισμένα στις μεθόδους*, οι τυπικές παράμετροι αντιστοιχούν στα τυπικά ορίσματα της μεθόδου. Μια ακόμη παράμετρος είναι το αναγνωριστικό (*oid*) του αντικειμένου που εκτελεί την μέθοδο. Στα *γεγονότα που βασίζονται στις τιμές* έχουμε το αναγνωριστικό του αντικειμένου και την ανανεωμένη τιμή του αντικειμένου. Απαιτείται ένας μηχανισμός, που να δημιουργεί μια έκδοση του αντικειμένου πριν την αλλαγή της τιμής του, έτσι ώστε να μπορούν να γίνουν συγκρίσεις και να προσπελαύνονται και η καινούργια και η παλαιά τιμή. Η παλαιά τιμή αναφέρεται στην τιμή του αντικειμένου πριν την εκτέλεση της λειτουργίας αλλαγής της και η καινούργια τιμή αναφέρεται στην τιμή του αντικειμένου μετά την εκτέλεση της λειτουργίας αυτής.
- *Παράμετροι σύνθετων γεγονότων*: Διακρίνουμε και εδώ δύο είδη παραμέτρων, τις περιβαλλοντολογικές και αυτές που έχουν άμεση σχέση με το είδος του τελεστή που συνθέτει το γεγονός. *Περιβαλλοντολογική παράμετρος* είναι η *occ_point(E),* δηλαδή η παράμετρος που δηλώνει την χρονική στιγμή της σηματοδότησης του γεγονότος *E,* ενώ οι παράμετροι *occ_tid* και *user_id* δεν έχουν σημασία καθώς τα διάφορα γεγονότα λαμβάνουν χώρα σε διαφορετικές δοσοληψίες. Όσον αφορά τις *παραμέτρους που έχουν σχέση με τον τελεστή που συνθέτει τα γεγονότα* παραθέτουμε τον ακόλουθο πίνακα.

Σύνθετο γεγονός	**παράμετροι των σύνθετων γεγονότων**
(E1 | E2)	παράμετροι του E1 ή του E2
(E1, E2)	παράμετροι του E1 και του E2 (ανεξάρτητα από τη σειρά τους)
(E1 ; E2)	παράμετροι (πρώτα) του E1 και (μετά) του E2
* E	παράμετροι της πρώτης εμφάνισης του E
last (E) IN I	παράμετροι της τελευταίας εμφάνισης του E
TIMES (n, E)	ένωση των παραμέτρων των n εμφανίσεων του E
TIMES ([n1-n2], E) IN I	ένωση των παραμέτρων των n εμφανίσεων του E και του αριθμού των εμφανίσεων
TIMES ([>n], E) IN I	ένωση των παραμέτρων όλων των εμφανίσεων του E μέχρι το τέλος του διαστήματος I και του αριθμού των εμφανίσεων
NOT E IN I	καθόλου παράμετροι

Σχήμα 6.1 Σύνοψη παραμέτρων σύνθετων γεγονότων [G94]

6.2.5 Ανάλυση του όρου "Συνθήκη"

Σε ένα κανόνα ενεργών συστημάτων βάσεων δεδομένων, ο όρος ¨συνθήκη¨ ορίζει λεπτομερώς την συνθήκη που πρέπει να ελεγχθεί μετά την ενεργοποίηση του κανόνα και πριν την εκτέλεση της πράξης. Αυτή η συνθήκη μπορεί να είναι:

- *Κατηγόρημα της βάσης δεδομένων*. Η συνθήκη μπορεί να εξειδικεύει μία ορισμένη τιμή για ένα κατηγόρημα, όπου το κατηγόρημα ορίζεται με μια γλώσσα που αντιστοιχεί στην γλώσσα των ερωτήσεων της βάσης δεδομένων (π.χ. σε ένα ενεργό σχεσιακό σύστημα βάσης δεδομένων η συνθήκη του κανόνα μπορεί να έχει την μορφή της πρότασης where μιας SQL ερώτησης). Για παράδειγμα,

```
DEFINE rule R1
      ON.....
      IF emp.name = ¨Bob¨
      THEN......
```

- *Περιορισμός πράξεων πάνω σε κατηγόρημα της βάσης δεδομένων*. Η αποτίμηση των συνθηκών είναι η πλέον χρονοβόρα εργασία στην εκτέλεση ενός κανόνα, επομένως αυτή που κρίνει και την απόδοση του συστήματος. Σε μερικές περιπτώσεις η συνθήκη μπορεί να εξειδικεύει μία ορισμένη τιμή για ένα κατηγόρημα, όπου το κατηγόρημα ορίζεται με ένα περιορισμένο ρεπερτόριο των δυνατοτήτων της γλώσσας που αντιστοιχεί στην γλώσσα των ερωτήσεων της βάση δεδομένων (π.χ. μπορεί να επιτρέπονται οι λειτουργίες σύγκρισης αλλά όχι και οι ενώσεις ή οι αθροιστικές λειτουργίες).
- *Ερώτηση της βάσης δεδομένων*. Η συνθήκη μπορεί να εκφρασθεί ως μία ερώτηση (query), χρησιμοποιώντας την γλώσσα ερωτήσεων της βάσης δεδομένων. Για παράδειγμα, μια τέτοια συνθήκη μπορεί να επεξεργάζεται όλα τα δεδομένα που έχουν τιμή κάτω ενός συγκεκριμένου ορίου. Αυτή η συνθήκη θα είναι αληθής, είτε αν η ερώτηση επιστρέφει μια κενή απάντηση, είτε αν επιστρέφει μια μη κενή απάντηση. Και σε αυτή την περίπτωση ισχύουν όσα αναφέρθηκαν για τους περιορισμούς των πράξεων πάνω σε κατηγορήματα της βάσης δεδομένων σχετικά με την απόδοση του συστήματος.

- *Εφαρμογή του χρήστη*. Μια συνθήκη κανόνα μπορεί να είναι και η κλήση σε μια εφαρμογή που έχει οριστεί από τον χρήστη. Αν αυτή η διαδικασία επιστρέφει τιμή boolean, και είναι θετική τότε η συνθήκη ισχύει, σε αντίθετη περίπτωση δεν ισχύει. Όμως, μπορεί η διαδικασία να επιστρέφει δεδομένα, οπότε η συνθήκη ισχύει στην περίπτωση όπου η διαδικασία επιστρέφει ένα μη κενό σύνολο δεδομένων, αλλιώς δεν ισχύει. Για παράδειγμα, έστω ότι έχουμε ορίσει την διαδικασία *UpdateSalary*, η οποία επιστρέφει θετική τιμή εάν έχουν γίνει μεταβολές στην τιμή του πεδίου *salary* μιας σχέσης:

```
DEFINE rule R1
      ON.....
      IF UpdateSalary = true
      THEN.....
```

Σε όλες τις παραπάνω περιπτώσεις, εάν η γλώσσα κανόνων υποστηρίζει και παραμέτρους γεγονότων, τότε η γλώσσα που περιγράφει τις συνθήκες συμπεριλαμβάνει και έναν μηχανισμό για αναφορά στις τιμές που μεταφέρονται με τις παραμέτρους αυτές.

Σε μερικές ενεργές βάσεις δεδομένων, οι τιμές ή τα δεδομένα που επεξεργάζεται η συνθήκη μεταφέρονται και στην πράξη του κανόνα, είτε μέσω ενός μηχανισμού παραμέτρων είτε μέσω άλλων χαρακτηριστικών της γλώσσας των κανόνων.

Τέλος, τα περισσότερα συστήματα ενεργών βάσεων δεδομένων επιτρέπουν την απουσία της συνθήκης, ως μέρος του κανόνα. Σε αυτά τα συστήματα, εννοείται ότι η συνθήκη ισχύει πάντοτε.

6.2.6 Ανάλυση του όρου "Πράξη"

Σε ένα κανόνα ενεργών συστημάτων βάσεων δεδομένων, ο όρος ¨πράξη¨ εκτελείται μετά την ενεργοποίηση του κανόνα και εφόσον ο έλεγχος της συνθήκης δείξει την ορθότητα αυτής. Η πράξη μπορεί να είναι:

- *Μία ή περισσότερες λειτουργίες μεταβολής των δεδομένων της βάσης*. Ένα ενεργό σχεσιακό σύστημα βάσης δεδομένων επιτρέπει οι πράξεις των κανόνων να είναι λειτουργίες μεταβολής των δεδομένων, όπως εισαγωγή, διαγραφή ή ανανέωση, και αντίστοιχα ένα αντικειμενοστρεφές σύστημα υποστηρίζει ως ορισμούς πράξεων την δημιουργία, διαγραφή αντικειμένου, ή την κλήση μεθόδων που μεταβάλλουν αντικείμενα. Για παράδειγμα,

```
DEFINE rule R1
      ON event...
      IF condition....
      THEN insert | update | delete
            emp (name= ¨Bob¨, age=¨27¨, dept=¨Toy¨)
```

- *Μία ή περισσότερες λειτουργίες επεξεργασίας των δεδομένων της βάσης*. Ένα ενεργό σχεσιακό σύστημα βάσης δεδομένων επιτρέπει τον καθορισμό λειτουργιών επιλογής ως πράξεις κανόνων, και ένα αντικειμενοστρεφές σύστημα υποστηρίζει την επιλογή αντικειμένων ή την κλήση μεθόδων που επεξεργάζονται αντικείμενα.
- *Άλλες εντολές της βάσης δεδομένων*. Εκτός από τις προηγούμενες λειτουργίες, μια πράξη κανόνα μπορεί να είναι οποιαδήποτε λειτουργία πάνω στη βάση δεδομένων, όπως ορισμός δεδομένων, έλεγχος δοσοληψιών, καθορισμός προνομίων και δικαιωμάτων, κ.ά. Για παράδειγμα,

```
DEFINE rule R1
      ON event...
```

```
IF condition....
THEN abort
```

- *Εφαρμογές του χρήστη*. Η πράξη του κανόνα μπορεί να είναι η κλήση μιας διαδικασίας ορισμένης από τον χρήστη, όπου η διαδικασία μπορεί να προσπελαύνει ή όχι την βάση δεδομένων.

Σε όλες τις παραπάνω περιπτώσεις, εάν η γλώσσα κανόνων υποστηρίζει και παραμέτρους γεγονότων, τότε η γλώσσα που περιγράφει τις πράξεις συμπεριλαμβάνει και έναν μηχανισμό για αναφορά στις τιμές που μεταφέρονται με τις παραμέτρους αυτές. Πολλά συστήματα, τέλος, επιτρέπουν τον ορισμό πολλών πράξεων σε ένα κανόνα, οπότε εμπεριέχουν και μηχανισμούς για τον ορισμό προτεραιοτήτων στην εκτέλεσή τους, η οποία συνήθως είναι ακολουθιακή.

6.2.7 Αναφορά σε μεταβατικές τιμές

Όταν η συνθήκη ενός κανόνα αποτιμάται, ή η πράξη του εκτελείται, αυτό συχνά συμβαίνει σε σχέση με μια αλλαγή της κατάστασης της βάσης δεδομένων (μια μετάβαση), η οποία προκαλεί την ενεργοποίηση του κανόνα. Η μετάβαση αυτή μπορεί να είναι της τάξης μιας απειροελάχιστης αλλαγής των δεδομένων μιας πλειάδας ή μιας ολόκληρης δοσοληψίας.

Υπάρχουν δύο ειδών μηχανισμοί στην γλώσσα του ορισμού των κανόνων για την αναφορά σε μεταβατικές τιμές. Ο πρώτος μηχανισμός αφορά στις παραμέτρους των γεγονότων που μεταφέρουν πληροφορία στις συνθήκες και τις πράξεις των κανόνων και έχει αναπτυχθεί στην υποενότητα 6.2.4 (¨Παράμετροι γεγονότων¨).

Ο δεύτερος μηχανισμός αναφέρεται στην χρήση καθορισμένων ή μη, δεσμευμένων λέξεων. Οι προκαθορισμένες δεσμευμένες λέξεις: *inserted, deleted* και *updated,* αναφέρονται στα δεδομένα που μόλις εισήχθησαν, διεγράφησαν ή ανανεώθηκαν στην βάση δεδομένων. Στην περίπτωση της ενεργοποίησης ενός κανόνα λόγω της ανανέωσης δεδομένων, οι λέξεις νέο και παλαιό (*new* | *current* και *old* | *previous*) μπορούν να χρησιμοποιηθούν για να αναφερθούμε στην καινούργια ή την παλαιά τιμή του δεδομένου που μεταβλήθηκε. Για παράδειγμα,

```
DEFINE rule R1
     ON event...
     IF salary < 1.1 * previous salary
     THEN action....
```

Στην περίπτωση που δεν χρησιμοποιούμε προκαθορισμένες δεσμευμένες λέξεις για την αναφορά στις μεταβατικές τιμές, ο γενικός κανόνας υπαγορεύει ότι, όταν ένας κανόνας ενεργοποιείται από κάποιες αλλαγές των δεδομένων, τότε αναφορές σε αυτά τα δεδομένα στην συνθήκη ή την πράξη του κανόνα αυτού, εμμέσως εννοούν την μεταβατική τιμή που σχετίζεται με τα δεδομένα αυτά.

6.3 ΣΗΜΑΣΙΟΛΟΓΙΑ ΕΠΕΞΕΡΓΑΣΙΑΣ ΚΑΙ ΕΚΤΕΛΕΣΗΣ ΚΑΝΟΝΩΝ

Σε αυτή την ενότητα εξετάζουμε τα διάφορα θέματα που αφορούν στην εκτέλεση και την επεξεργασία των κανόνων, ξεκινώντας από την παρουσίαση ενός απλού αλγορίθμου επεξεργασίας κανόνων, συμπληρώνοντας και επεκτείνοντας το μοντέλο αυτό. Στην συνέχεια, παρουσιάζουμε τους τρόπους σύζευξης των κανόνων με τις δοσοληψίες που τους ενεργοποίησαν, τους μηχανισμούς επίλυσης συγκρούσεων σε σχέση με την σειρά ενεργοποίησης των κανόνων, καθώς και τις διαφορές στην εκτέλεση των κανόνων μετά από κάθε εγγραφή που ικανοποιεί την συνθήκη σε σχέση με την εκτέλεση μετά το σύνολο των

εγγραφών που την ικανοποιούν. Επίσης θα αναφερθούμε και στις προϋποθέσεις για ικανότητα τερματισμού των κανόνων.

6.3.1 Μέγεθος της μονάδας επεξεργασίας

Ένας απλός αλγόριθμος επεξεργασίας κανόνων είναι ο ακόλουθος, παρόμοιος με τον γνωστό *recognize-act cycle* της τεχνητής νοημοσύνης:

```
Όσο υπάρχουν ενεργοποιημένοι κανόνες κάνε:
1.Βρες ένα ενεργοποιημένο κανόνα
2.Αποτίμησε την συνθήκη του
3.Εάν η συνθήκη του ισχύει τότε εκτέλεσε την πράξη που τον
  συνοδεύει
```

Το μέγεθος της μονάδας επεξεργασίας του κανόνα ορίζει την συχνότητα των χρονικών σημείων στα οποία οι κανόνες επεξεργάζονται, δηλαδή την συχνότητα της δραστηριοποίησης του αλγορίθμου. Το μέγεθος αυτό μπορεί να είναι πολύ μικρό, στην περίπτωση που ο κανόνας είναι έτοιμος προς έλεγχο για ενεργοποίηση και εκτέλεση συνεχώς, μέχρι και αρκετά μεγάλο, στην περίπτωση όπου ο κανόνας ενεργοποιείται στα όρια των δοσοληψιών της βάσης δεδομένων.

Αναλυτικότερα, οι κανόνες μπορούν να ενεργοποιούνται *σε κάθε σημείο* κατά την διάρκεια της εκτέλεσης των ενεργειών της βάσης δεδομένων. Αυτό έχει σημασία κυρίως για χρονικά γεγονότα ή για κανόνες που ενεργοποιούνται από συγκεκριμένες καταστάσεις της βάσης δεδομένων.

Σε επόμενο βαθμό, οι κανόνες μπορούν να εκτελούνται *μετά την παραμικρή λειτουργία* της βάσης δεδομένων που ενεργοποίησε ένα κανόνα. Σε ένα σχεσιακό ενεργό σύστημα, αυτό μπορεί να προκληθεί από την αλλαγή (εισαγωγή, διαγραφή, μεταβολή) μιας πλειάδας δεδομένων ενώ σε ένα αντικειμενοστρεφές, από την κλήση μιας μεθόδου ή την μεταβολή μιας τιμής.

Συχνά, το μέγεθος της μονάδας επεξεργασίας μπορεί να είναι *μεγαλύτερο από μια λειτουργία*, δηλαδή ο κανόνας μπορεί να ενεργοποιείται μετά το πέρας μιας πρότασης SQL όπου μεταβλήθηκαν και επεξεργάστηκαν πολλά δεδομένα και πραγματοποιήθηκαν διάφορες λειτουργίες. Ακόμη, οι κανόνες μπορούν να ενεργοποιούνται και να εκτελούνται *στο πέρας κάθε δοσοληψίας*, όπου τα όρια μιας δοσοληψίας συνήθως περιορίζουν έναν αριθμό διαδικασιών της βάσης δεδομένων. Τέλος, το μέγεθος μπορεί να *ορίζεται από τον χρήστη ή την εφαρμογή*, αλλά σε αυτές τις περιπτώσεις υπάρχουν σίγουρα και καθορισμένες τιμές από το σύστημα.

6.3.2 Τρόποι σύζευξης των κανόνων και των δοσοληψιών

Η σχέση του κανόνα και της δοσοληψίας που τον ενεργοποίησε καθορίζει την εκτέλεση του κανόνα σε σχέση

- με την δοσοληψία,
- με το εάν ο κανόνας εκτελείται ως μια μονάδα,
- με το εάν η συνθήκη και/ή η πράξη του εκτελούνται ως ξεχωριστές μονάδες.

Οι σχέσεις αυτές καθορίζονται στο μοντέλο εκτέλεσης και καλούνται *τρόποι σύζευξης* των κανόνων και των δοσοληψιών. Πολλά ερωτήματα γεννιόνται σχετικά με την εκτέλεση των κανόνων:

Πότε πρέπει η εκτέλεση του κανόνα να ξεκινάει ;

- Αμέσως μετά την σηματοδότηση του γεγονότος.

- Στο τέλος της δοσοληψίας του χρήστη αλλά προτού αυτή ολοκληρωθεί επιτυχώς.
- Μετά την επιτυχή ολοκλήρωση της δοσοληψίας που τον ενεργοποίησε.
- Παράλληλα με την δοσοληψία που τον ενεργοποίησε.

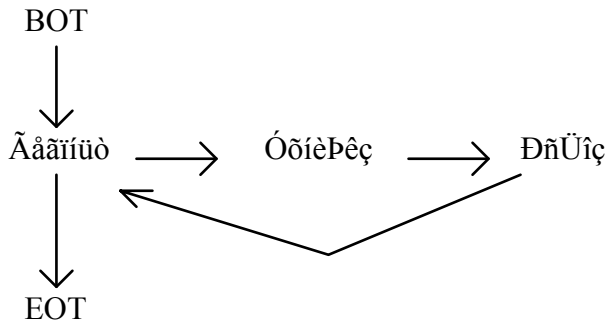
Πότε πρέπει η εκτέλεση του κανόνα να τελειώνει ;

- Προτού συνεχιστεί η εκτέλεση της δοσοληψίας που τον ενεργοποίησε.
- Προτού ολοκληρωθεί επιτυχώς η εκτέλεση της δοσοληψίας που τον ενεργοποίησε.
- Μετά την επιτυχή ολοκλήρωση της δοσοληψίας που τον ενεργοποίησε.
- Μετά την αποβολή της δοσοληψίας που τον ενεργοποίησε.

Πρέπει ο κανόνας να εκτελείται ως μέρος της δοσοληψίας που τον ενεργοποίησε ή ως ξεχωριστή δοσοληψία ;

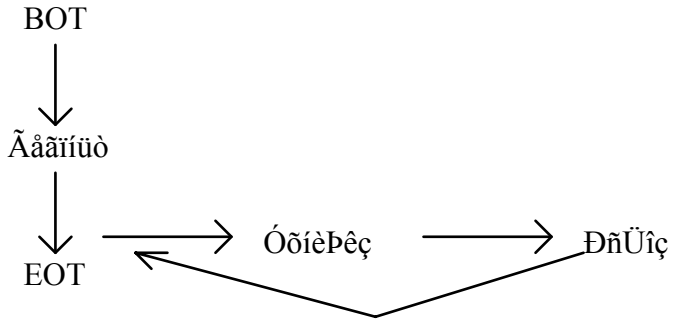
Ως αποτέλεσμα όλων αυτών των ερωτημάτων και των επιλογών, έχουμε τους ακόλουθους τρόπους σύζευξης:

- *Άμεση*: Ο κανόνας εκτελείται αμέσως μόλις σηματοδοτείται το γεγονός που τον ενεργοποιεί. Η εκτέλεση γίνεται ως μια υποδοσοληψία και η συνθήκη και η πράξη του κανόνα μπορούν να έχουν διαφορετικούς τρόπους σύζευξης.



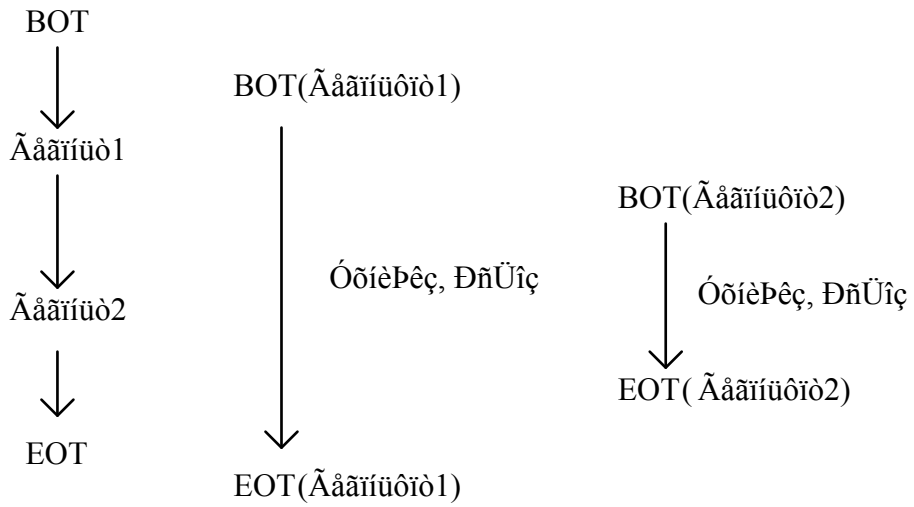
Σχήμα 6.2 Άμεση σύζευξη κανόνα και δοσοληψίας

- *Αναβαλλόμενη*: Η εκτέλεση του κανόνα γίνεται μετά την δοσοληψία που τον ενεργοποίησε αλλά πριν την επιτυχή ολοκλήρωσή της. Η συνθήκη και η πράξη του κανόνα και σε αυτή την περίπτωση μπορούν να έχουν διαφορετικούς τρόπους σύζευξης.



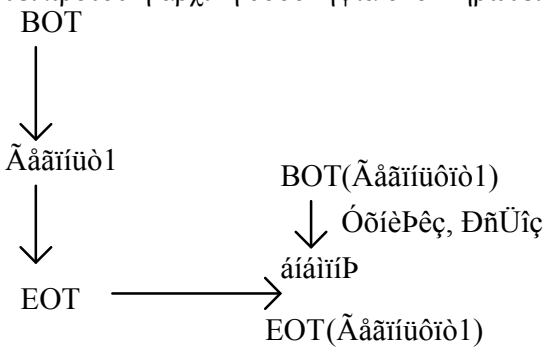
Σχήμα 6.3 Αναβαλλόμενη σύζευξη κανόνα και δοσοληψίας

- *Σε ξεχωριστή δοσοληψία*: Ο κανόνας εκτελείται σε διαφορετική δοσοληψία και δεν εξαρτάται από την δοσοληψία που τον ενεργοποίησε - αυτό σημαίνει ότι αν αυτή αποβληθεί δεν θα γίνει το ίδιο απαραίτητα και για την δοσοληψία του κανόνα.



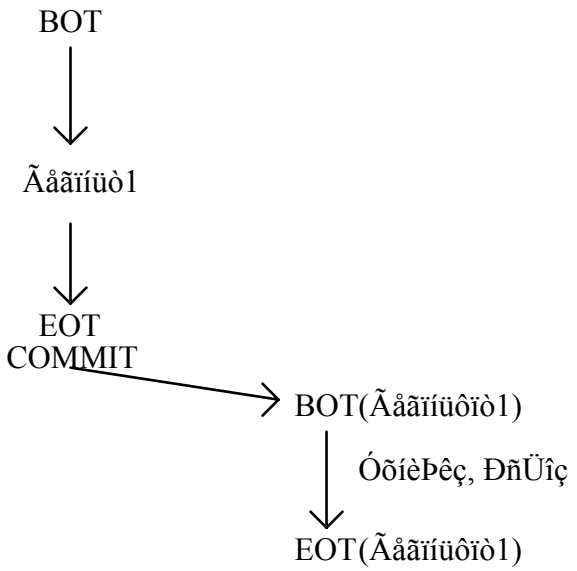
Σχήμα 6.4 Εκτέλεση κανόνα σε ξεχωριστή δοσοληψία

- Σε ξεχωριστή δοσοληψία, εξαρτώμενη όμως από την δοσοληψία που τον ενεργοποίησε:
  1. *παράλληλη*: Ο κανόνας εκτελείται σε ξεχωριστή δοσοληψία, η οποία μπορεί να εξελίσσεται παράλληλα με την δοσοληψία που ενεργοποίησε τον κανόνα, αλλά δεν θα ολοκληρωθεί προτού η αρχική δοσοληψία ολοκληρωθεί επιτυχώς.



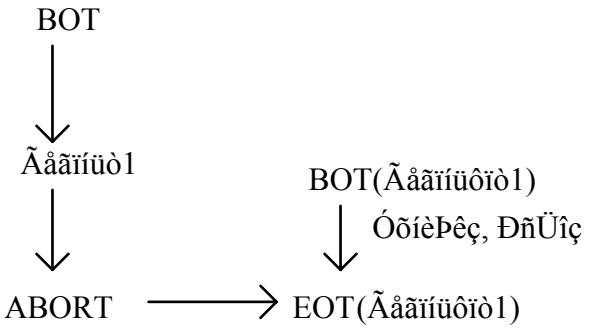
Σχήμα 6.5 Παράλληλη σύζευξη κανόνα και δοσοληψίας

  2. *ακολουθιακή*: Ο κανόνας εκτελείται σε ξεχωριστή δοσοληψία, η οποία δεν μπορεί να ξεκινήσει προτού ολοκληρωθεί επιτυχώς η δοσοληψία που ενεργοποίησε τον κανόνα. Απαραίτητος τρόπος σύζευξης σε περιβάλλοντα όπου οι κανόνες εκτελούν πράξεις που δεν διορθώνονται.



Σχήμα 6.6 Ακολουθιακή σύζευξη κανόνα και δοσοληψίας

3. *αποκλειστικά διαζευκτική:* Ο κανόνας εκτελείται σε ξεχωριστή δοσοληψία, η οποία όμως δεν μπορεί να ολοκληρωθεί αν δεν αποβληθεί η δοσοληψία που ενεργοποίησε τον κανόνα. Αυτός ο τρόπος σύζευξης είναι χρήσιμος σε περιπτώσεις επεξεργασίας με χρονικό περιορισμό και για την περιγραφή κανόνων που εκφράζουν εναλλακτικά σχέδια δράσης.



Σχήμα 6.7 Αποκλειστικά διαζευκτική σύζευξη κανόνα και δοσοληψίας

6.3.3 Μηχανισμοί επίλυσης συγκρούσεων

Στον αλγόριθμο που παρουσιάσαμε στην αρχή της ενότητας, το πρώτο του βήμα απαιτεί την επιλογή ενός ενεργοποιημένου κανόνα. Σε πολλές όμως περιπτώσεις περισσότεροι από ένας κανόνες είναι ενεργοποιημένοι -είτε γιατί το ίδιο γεγονός ενεργοποιεί πολλούς κανόνες, είτε γιατί λόγω του μεγέθους της μονάδας επεξεργασίας των κανόνων, περισσότεροι κανόνες ενεργοποιούνται προτού οι προηγούμενοι εκτελεστούν, είτε γιατί οι κανόνες που δεν επιλέχθηκαν στο πρώτο βήμα εξακολουθούν να είναι ενεργοποιημένοι.

Γενικά πρέπει να υπάρχει κάποιος μηχανισμός που να καθορίζει την σειρά επιλογής των ενεργοποιημένων κανόνων προς εκτέλεση -ο καθορισμός αυτός καλείται *επίλυση συγκρούσεων*

(όρος δανεισμένος από την τεχνητή νοημοσύνη). Παραδείγματα τέτοιων μηχανισμών είναι [WC96]:

- Μπορεί να γίνει αυθαίρετη επιλογή.
- Μπορούν να καθοριστούν *προτεραιότητες* - μια γλώσσα καθορισμού ενεργών κανόνων πρέπει να υποστηρίζει τον ορισμό προτεραιοτήτων. Αυτό μπορεί να γίνει καθορίζοντας την σειρά σε μια ομάδα κανόνων, δηλώνοντας σχετικές προτεραιότητες ανάμεσα σε ζευγάρια κανόνων, είτε δηλώνοντας μια αριθμητική τιμή σε κάθε κανόνα. Οι σχετικές προτεραιότητες είναι η πιο ευέλικτη μέθοδος, καθώς ο καθορισμός προτεραιοτήτων για όλους τους κανόνες συνήθως είναι ανώφελος και άχρηστος και η δήλωση αριθμητικών τιμών στους κανόνες προϋποθέτει την συνεχή ανανέωσή της με την εξέλιξη και τον εμπλουτισμό των συνόλων των κανόνων.
- Η επιλογή μπορεί να στηρίζεται σε στατικές ιδιότητες των κανόνων, όπως η χρονική στιγμή δημιουργίας τους ή των δεδομένων στα οποία αφορούν.
- Η επιλογή μπορεί να στηρίζεται σε δυναμικές ιδιότητες των κανόνων, όπως η σειρά ενεργοποίησής τους (κάτι ανάλογο του FIFO).

6.3.4 Εκτέλεση κανόνα μετά από μοναδική εγγραφή - σύνολο εγγραφών

Ένας κανόνας μπορεί να εκτελείται είτε για κάθε στιγμιότυπο της βάσης δεδομένων που ενεργοποιεί τον κανόνα ή ικανοποιεί την συνθήκη του, είτε για το σύνολο των δεδομένων που τον ενεργοποιούν ή ικανοποιούν την συνθήκη του. Στην πρώτη περίπτωση η εκτέλεση των κανόνων του ενεργού συστήματος βάσης δεδομένων καλείται *προσανατολισμένη προς το στιγμιότυπο (instance-oriented),* ενώ στην δεύτερη *προσανατολισμένη προς το σύνολο (set-oriented)*. Για παράδειγμα, σε ένα σχεσιακό σύστημα, στην προσανατολισμένη προς το στιγμιότυπο εκτέλεση κανόνων, η εκτέλεση ενός κανόνα που ενεργοποιείται από κάποιες διαγραφές σε έναν πίνακα, θα γίνει ως εξής: μετά από κάθε διαγραφή, ενεργοποιείται ο κανόνας, αποτιμάται η συνθήκη του και εφόσον αυτή ισχύει, εκτελείται η πράξη. Σε ένα αντικειμενοστρεφές σύστημα, αντίστοιχα θα είχαμε ενεργοποίηση και εκτέλεση ενός κανόνα μετά την εισαγωγή του νέου αντικειμένου σε μια κλάση.

Στην προσανατολισμένη προς το σύνολο εκτέλεση κανόνων, σε ένα σχεσιακό ενεργό σύστημα βάσεων δεδομένων, για τον παραπάνω κανόνα η εκτέλεση θα γινόταν ως εξής: αφού γίνει το σύνολο των διαγραφών, ενεργοποιείται ο κανόνας, αποτιμάται η συνθήκη του και εκτελείται. Στην περίπτωση του αντικειμενοστρεφούς συστήματος, μετά την ολοκλήρωση όλων των εισαγωγών των αντικειμένων στην συγκεκριμένη κλάση, εκτελείται ο αντίστοιχος κανόνας.

Η διαφορά στο αποτέλεσμα των δύο τρόπων εκτελέσεων του κανόνα, μπορεί να είναι λεπτή, δεν παύει όμως να είναι ουσιαστική. Σε ένα συγκεκριμένο παράδειγμα, όπου ζητείται με την εισαγωγή ενός υπαλλήλου στη βάση να δηλώνεται και ο μισθός του σε σχέση με τον μέσο όρο του μισθού των υπολοίπων υπαλλήλων, το εάν η εκτέλεση του κανόνα θα γίνει μετά από κάθε εισαγωγή νέου υπαλλήλου, ή μετά την εισαγωγή όλων, θα σηματοδοτήσει και διαφορετική τιμή στον μισθό καθενός. Έστω οι σχέσεις:

```
emp(name, emp-no, dept, salary, mgr)
dept(dept, dept-no, floor, mgr)
```

και ο κανόνας:

```
define rule newEmp
```

```
on append emp
  then append to emp(emp.name, emp.emp-no, emp.dept,
avg(emp.salary), emp.mgr)
```

Το θέμα της προσανατολισμένης προς το στιγμιότυπο ή προς το σύνολο εκτέλεσης των κανόνων συνδέεται άμεσα και με το θέμα του μεγέθους της μονάδας επεξεργασίας των κανόνων. Στην περίπτωση όπου το μέγεθος της μονάδας επεξεργασίας είναι το μικρότερο δυνατό - ισοδύναμο με την παραμικρή μεταβολή της βάσης δεδομένων -η εκτέλεση των κανόνων, είτε είναι προσανατολισμένη προς το στιγμιότυπο, είτε προς το σύνολο είναι η ίδια (ο κανόνας εκτελείται μετά από κάθε μεταβολή των δεδομένων).

Στον αλγόριθμο που παρουσιάσαμε στην αρχή του κεφαλαίου, η ροή της εκτέλεσης των κανόνων στην περίπτωση του προσανατολισμού προς το σύνολο είναι ξεκάθαρη. Μετά την ενεργοποίηση του κανόνα, για ολόκληρο το σύνολο των δεδομένων που ικανοποιούν την συνθήκη γίνεται η αποτίμησή της και η εκτέλεση του κανόνα. Στην περίπτωση όμως του προσανατολισμού προς το στιγμιότυπο, παρουσιάζεται το ερώτημα αν ο κανόνας θα εκτελεστεί για το πρώτο στιγμιότυπο μόνο, ή και για τα υπόλοιπα στην συνέχεια.

6.3.5 Ικανότητα τερματισμού κανόνων

Σε όλες τις κατηγορίες αλγορίθμων επεξεργασίας κανόνων, επαναληπτικούς, αναδρομικούς, ακολουθιακούς ή παράλληλους, εμφανίζεται η περίπτωση μη τερματισμού. Δηλαδή, όταν ένας κανόνας ενεργοποιεί ένα άλλον, ή τον εαυτό του αναδρομικά, υπάρχει η πιθανότητα να δημιουργηθεί ένας ατέρμονος κύκλος επεξεργασίας κανόνων. Αυτό μπορεί επίσης να συμβεί σε ένα σύστημα παραγωγής αλλά δεν παρουσιάζει σημαντικό ενδιαφέρον καθώς τέτοια συστήματα συνήθως χρησιμοποιούνται για την ανάπτυξη εφαρμογών ενός μόνο χρήστη, επομένως η πιθανή καταστροφή που μπορεί να προκαλέσει ένας ατέρμονος κύκλος είναι περιορισμένη. Αντίθετα, σε μια βάση δεδομένων, ένας ατέρμονος κύκλος μπορεί να προκαλέσει την υπερχείλιση δεδομένων στην δευτερεύουσα μνήμη και στην καλύτερη περίπτωση, να δεσμεύσει ένα μέρος των πηγών της κεντρικής μονάδας επεξεργασίας ή να κρατήσει επ' αόριστον κλειδωμένα ορισμένα δεδομένα που απαιτούνται από άλλες δοσοληψίες.

Υπάρχουν διάφοροι τρόποι για την αντιμετώπιση αυτής της κατάστασης:

- Αφήνεται στον σχεδιαστή των κανόνων ο έλεγχος ώστε να προβλεφθεί η κατάσταση μη τερματισμού και να αποφευχθεί.
- Ορίζεται ένα άνω όριο αριθμού κανόνων που επεξεργάζεται το σύστημα, ώστε όταν το όριο αυτό συμπληρωθεί, η εκτέλεση των κανόνων να λήγει απότομα. Αυτή η εκδοχή λειτουργεί καλά για τυπικές περιπτώσεις εφαρμογών, όμως σε περιπτώσεις όπου οι κανόνες ορίζονται για διάσχιση δέντρων ή γράφων είναι αδύνατο να τεθεί ένα όριο στο βάθος του κανόνα που θα αποτρέπει τους ατέρμονους κύκλους αλλά και δεν θα αποβάλει την δοσοληψία αδίκως. Σε αυτές τις περιπτώσεις είναι αναγκαία η χρησιμοποίηση ενός εργαλείου που θα αναλύει εκ των προτέρων τις ομάδες των κανόνων ώστε να υποδεικνύει πιθανούς κύκλους.
- Τίθενται συντακτικοί περιορισμοί στον ορισμό των κανόνων, ώστε να μην εμφανιστεί το φαινόμενο μη τερματισμού. Εξασφαλίζεται ότι ένας κανόνας δεν μπορεί να ενεργοποιήσει έναν άλλον, ή ότι οι κανόνες δεν δημιουργούν κύκλους ενεργοποίησης, ή ότι εάν δημιουργούνται τέτοιοι κύκλοι, εξασφαλίζεται ότι κάποιος από τους κανόνες που συμμετέχουν σε αυτούς δεν θα μπορεί να εκτελεστεί διότι δεν θα ικανοποιείται η συνθήκη του.

6.4 ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΩΝ

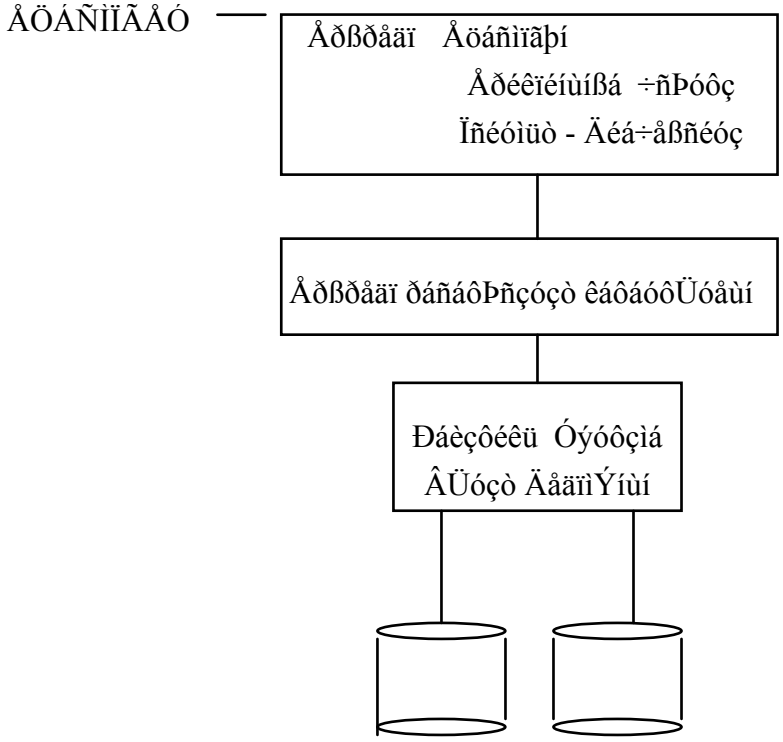
Υπάρχουν πολυάριθμες πιθανές λύσεις αρχιτεκτονικής για τα ενεργά συστήματα βάσεων δεδομένων [Ch92],[WC96], αλλά ξεχωρίζουμε τρεις κατηγορίες:

- Αρχιτεκτονική σε επίπεδα
- Ολοκληρωμένη αρχιτεκτονική
- Αρχιτεκτονική εφαρμογών

Και στις τρεις κατηγορίες απαραίτητο είναι, εκτός των στοιχείων που υπάρχουν και σε ένα παθητικό σύστημα βάσης δεδομένων, να υλοποιηθούν νέα στοιχεία όπως *ο διαχειριστής των κανόνων,* υπεύθυνος για την διαχείριση των κανόνων στο σύστημα, *οι ανιχνευτές γεγονότων,* υπεύθυνοι για την παρακολούθηση των γεγονότων και *ένα τμήμα εκτέλεσης κανόνων,* υπεύθυνο για την εκτέλεση των κανόνων.

6.4.1 Αρχιτεκτονική σε επίπεδα

Σε αυτή την προσέγγιση, τα ενεργά στοιχεία της βάσης δεδομένων τίθενται ¨πάνω¨ από την παραδοσιακή παθητική ΒΔ. Η αρχιτεκτονική αυτή αναφέρεται και ως *χαλαρά συζευγμένη* μορφή, καθώς η διαχείριση και η επεξεργασία των κανόνων είναι τελείως ξεχωριστά τμήματα από το υπόλοιπο σύστημα.



Σχήμα 6.8 Αρχιτεκτονική σε επίπεδα

Ένα επίπεδο ¨χτίζεται¨ πάνω από το παραδοσιακό ΣΔΒΔ μέσω του οποίου όλοι οι ECA κανόνες δρομολογούνται. Το επίπεδο αυτό είναι υπεύθυνο για την παρατήρηση των καταστάσεων και την εκτέλεση των απαραίτητων κανόνων. Αυτό σημαίνει ότι όλες οι δοσοληψίες δρομολογούνται μέσω αυτού του επιπέδου (αν και τελικά επεξεργάζονται από τον

διαχειριστή δοσοληψιών του συστήματος). Το ενεργό τμήμα αντιλαμβάνεται τις εντολές όπως αυτές υποβάλλονται στο σύστημα και τα δεδομένα όπως αυτά επιστρέφονται από το σύστημα προς τον χρήστη και την εφαρμογή.

Υπάρχουν δύο ειδών τρόποι επικοινωνίας των ενεργών τμημάτων του συστήματος με τα παθητικά:

- *Η επικοινωνία προς τα κάτω*. Τα ενεργά στοιχεία χρησιμοποιούν τις υπηρεσίες που τους προσφέρει το παθητικό σύστημα. Για παράδειγμα, ο διαχειριστής κανόνων χρησιμοποιεί για την διαχείριση των κανόνων δομές που υπάρχουν στο μοντέλο δεδομένων του παθητικού συστήματος. Γενικά, το ενεργό επίπεδο χρησιμοποιεί, για την αναπαράσταση των γεγονότων και των κανόνων, τον ορισμό δεδομένων και την γλώσσα χειρισμού των δεδομένων, όπως κάθε άλλη εφαρμογή.
- *Η επικοινωνία προς τα πάνω*. Η επικοινωνία προς τα πάνω απαιτείται σε κάθε περίπτωση όπου τα ενεργά στοιχεία πρέπει να ειδοποιηθούν για την εκτέλεση συγκεκριμένων ενεργειών των παθητικών τμημάτων. Ακόμη είναι απαραίτητη όταν, εξαιτίας της ύπαρξης του ενεργού τμήματος, αλλάζει ο τρόπος λειτουργίας του παθητικού. Αυτό ισχύει για την ανίχνευση των γεγονότων που λαμβάνουν χώρα στην βάση δεδομένων και για την εκτέλεση των κανόνων.

Υπάρχουν βέβαια και κάποιοι περιορισμοί ως προς την κλάση των κανόνων που η αρχιτεκτονική αυτή μπορεί να υποστηρίξει. Για παράδειγμα, η άμεση σύζευξη μεταξύ του κανόνα και της δοσοληψίας που τον ενεργοποίησε δεν μπορεί να υποστηριχτεί, καθώς το ¨ενεργό¨ επίπεδο δεν είναι σε θέση να αποβάλει μια δοσοληψία που εξελίσσεται από το κατώτερο τμήμα του συστήματος. Επίσης, χρονικά και γεγονότα ορισμένα από εφαρμογές δεν υποστηρίζονται παρά μόνο χρησιμοποιώντας την μέθοδο της ερώτησης σε τακτά διαστήματα προς το σύστημα - όσον αφορά στην αποτίμηση των συνθηκών - με όλα τα συναφή μειονεκτήματα που αυτή η μέθοδος παρουσιάζει. Μετά την ενεργοποίηση ενός κανόνα, ο επεξεργαστής των κανόνων αποτιμά την συνθήκη και εκτελεί τις απαιτούμενες πράξεις καλώντας διαδικασίες ή υποβάλλοντας εντολές προς το σύστημα.

Τα πλεονεκτήματα της αρχιτεκτονικής σε επίπεδα είναι:

- Ένα παραδοσιακό σύστημα βάσεων δεδομένων μπορεί να μεταβληθεί σε ενεργό χωρίς μετατροπές των παθητικών στοιχείων του.
- Πολλά διαφορετικής τεχνολογίας παραδοσιακά συστήματα βάσεων δεδομένων μπορούν να μετατραπούν σε ενεργά με την ίδια πλατφόρμα επικοινωνίας ενεργών-παθητικών στοιχείων. Έτσι το ίδιο ενεργό στοιχείο μπορεί να χρησιμοποιηθεί σε ετερογενή παθητικά συστήματα βάσεων δεδομένων.

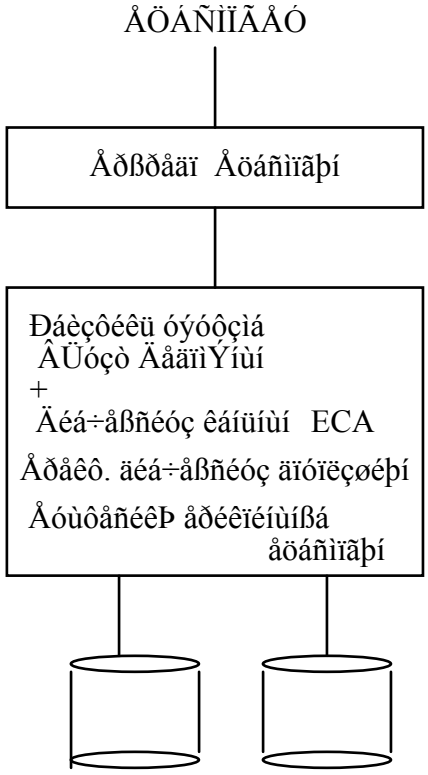
Τα μειονεκτήματα της αρχιτεκτονικής αυτής προσέγγισης είναι:

- Συνήθως παρατηρείται χαμηλή απόδοση, καθώς υπεισέρχεται καθυστέρηση λόγω της επικοινωνίας μεταξύ των παθητικών και των ενεργών στοιχείων.
- Ορισμένα χαρακτηριστικά, όπως οι τρόποι σύζευξης μεταξύ των κανόνων και των δοσοληψιών που τους ενεργοποιούν, ο έλεγχος συγχρονισμού, η άδεια πρόσβασης και επεξεργασίας των κανόνων, δεν υποστηρίζονται από αυτήν την αρχιτεκτονική, καθώς η υλοποίηση του ενεργού τμήματος δεν αλληλεπιδρά με τα υπόλοιπα τμήματα του συστήματος -π.χ. τον διαχειριστή δοσοληψιών, ή τον διαχειριστή κλειδωμάτων (το υπεύθυνο τμήμα για την άδεια πρόσβασης στα δεδομένα).

Μια πρόταση ενεργού συστήματος υλοποιημένου σύμφωνα με αυτήν την αρχιτεκτονική είναι και το Alert [SPAM91].

6.4.2 Ολοκληρωμένη αρχιτεκτονική

Στην ολοκληρωμένη αρχιτεκτονική, τα ενεργά στοιχεία αποτελούν ενιαίο μέρος του παραδοσιακού συστήματος βάσης δεδομένων. Αυτή η προσέγγιση αναφέρεται και ως *σθεναρά συζευγμένη* αρχιτεκτονική, καθώς η διαχείριση και η επεξεργασία των κανόνων αποτελούν αναπόσπαστο μέρος του συστήματος.



Σχήμα 6.9 Ολοκληρωμένη αρχιτεκτονική

Αυτό επιτυγχάνεται είτε μετατρέποντας ένα ήδη υπάρχον παραδοσιακό σύστημα βάσεων δεδομένων, είτε με την χρησιμοποίηση εργαλείων σχεδιασμού για συμβατικά χαρακτηριστικά και σύγχρονη πρόσθεση ενεργής λειτουργικότητας σε αυτά, είτε υλοποιώντας ένα σύστημα εξ' αρχής.

Η βασική ιδέα είναι ότι ο διαχειριστής δεδομένων σε χαμηλό επίπεδο παρατηρεί τις λειτουργίες που γίνονται πάνω στα δεδομένα και ειδοποιεί τα ενεργά στοιχεία όταν συμβεί ένα γεγονός υπεύθυνο για την ενεργοποίηση κανόνα.

Τα πλεονεκτήματα και τα μειονεκτήματα της ολοκληρωμένης αρχιτεκτονικής είναι τα ακριβώς αντίθετα από αυτά της αρχιτεκτονικής σε επίπεδα. Τα πλεονεκτήματα είναι:

- Έχουμε αποτελεσματική απόδοση κατά την διάρκεια της παρακολούθησης των γεγονότων, της αποτίμησης των συνθηκών και της εκτέλεσης των πράξεων, καθώς οι λειτουργίες αυτές συμβαίνουν ως μέρος του συστήματος.

- Η πρόσβαση σε όλα τα υποσυστήματα της βάσης δεδομένων επιτρέπει την υλοποίηση εξειδικευμένων χαρακτηριστικών για τους κανόνες και τις δοσοληψίες, για τον έλεγχο της σύγχρονης ενεργοποίησης κανόνων, κλπ.

Τα μειονεκτήματα είναι:

- Απαιτείται μεγάλη προσπάθεια για την υλοποίηση και την αλλαγή του ήδη υπάρχοντος κώδικα.
- Παραδοσιακά συστήματα βάσεων δεδομένων διαφορετικής τεχνολογίας θα μετατραπούν σε ενεργά συστήματα βάσεων δεδομένων διαφορετικής επίσης τεχνολογίας, καθώς θα απαιτήσουν άλλες προσεγγίσεις κατά την μετατροπή τους.

Σήμερα παραδείγματα συστημάτων που υλοποιούν αυτή την αρχιτεκτονική είναι ανάμεσα σε άλλα και τα ερευνητικά πρωτότυπα Ode, Sentinel, SAMOS.

6.4.3 Αρχιτεκτονική εφαρμογών

Στην αρχιτεκτονική ¨των εφαρμογών¨ - ή *την αρχιτεκτονική που βασίζεται στην φάση της μεταγλώττισης*, όπως αλλιώς συναντάται στη βιβλιογραφία - εναπόκειται στον χρήστη και στις εφαρμογές που αυτός κωδικοποιεί να εμπεριέχουν στοιχεία ενεργής συμπεριφοράς. Δηλαδή, κατά την διάρκεια της μεταγλώττισης των εφαρμογών και των λειτουργιών της βάσης δεδομένων, αυτές μετατρέπονται έτσι ώστε να συμπεριλαμβάνουν τα αποτελέσματα των κανόνων της ενεργής βάσης δεδομένων. Καμία δραστηριότητα δεν απαιτείται κατά την διάρκεια της εκτέλεσης.

Το κύριο πλεονέκτημα αυτής της προσέγγισης είναι ότι δεν χρειάζεται ο έλεγχος και η παρατήρηση των γεγονότων και η αποτίμηση των συνθηκών, οπότε η απόδοση του συστήματος είναι βέλτιστη και η πολυπλοκότητα της υλοποίησης ελάχιστη.

Όμως, παρουσιάζονται και τα ακόλουθα μειονεκτήματα, απόρροια της περιορισμένης εφαρμογής αυτής της αρχιτεκτονικής προσέγγισης στις γλώσσες των εφαρμογών και των κανόνων:

- Η γλώσσα προγραμματισμού που χρησιμοποιείται για την επικοινωνία του χρήστη ή της εφαρμογής με το σύστημα βάσης δεδομένων πρέπει να έχει την δυνατότητα να υποστηρίζει την αποτίμηση συνθηκών και την εκτέλεση πράξεων που απαιτούνται από τους κανόνες.
- Εφόσον τα γεγονότα που αποτελούν τους ενεργοποιητές των κανόνων αναγνωρίζονται από τον μεταγλωττιστή, μειώνεται η ποικιλία αυτών, καθώς χρονικά, σύνθετα και ορισμένα από τις εφαρμογές γεγονότα δεν μπορούν πλέον να υποστηριχτούν.
- Στην περίπτωση που υπάρχει κύκλος ενεργοποίησης κανόνων (δηλαδή, ένα γεγονός αποτελεί την αφετηρία για την ενεργοποίηση ενός κανόνα και αυτός με την σειρά του ενεργοποιεί έναν επόμενο κ.ο.κ.) θα εφαρμοστεί αναδρομική μεταγλώττιση που μπορεί να συνεχιστεί επ' αόριστον, οπότε - αν και σε κάποια άλλη αρχιτεκτονική προσέγγιση μπορεί να μην ήταν απαραίτητο - οι κανόνες που δημιουργούν κύκλους ενεργοποίησης απενεργοποιούνται.

Παράδειγμα συστήματος που υλοποιεί αυτή την αρχιτεκτονική είναι το ερευνητικό πρωτότυπο A-RDL.

6.5 ΣΗΜΑΤΟΔΟΤΗΣΗ ΚΑΙ ΕΚΤΕΛΕΣΗ ΚΑΝΟΝΩΝ

6.5.1 Μέθοδοι σηματοδότησης κανόνων

Κάθε φορά που ένα απλό γεγονός ανιχνεύεται (από τον ανιχνευτή απλών γεγονότων) ελέγχεται εάν η ανίχνευση αυτού συνεπάγεται την σηματοδότηση ενός σύνθετου γεγονότος. Αυτό αποτελεί το καθήκον του ανιχνευτή σύνθετων γεγονότων. Ένας απλός τρόπος για την ανίχνευση σύνθετων γεγονότων είναι ο εξής: καθορίζονται τα σύνθετα γεγονότα, στα οποία συνεισφέρει το απλό γεγονός που μόλις ανιχνεύτηκε, και για κάθε ένα από αυτά ελέγχεται εάν έχουν ανιχνευτεί και τα υπόλοιπα απλά γεγονότα που το απαρτίζουν (και στην κατάλληλη χρονική σειρά, αν αυτό είναι απαραίτητο). Η μέθοδος αυτή προϋποθέτει την αποθήκευση όλων των απλών γεγονότων που ανιχνεύτηκαν στο παρελθόν και την συνεχή εξέτασή τους. Η διαδικασία αυτή πρέπει να επαναλαμβάνεται συνεχώς, κάθε φορά που ανιχνεύεται ένα απλό γεγονός, και για όλα τα σύνθετα γεγονότα.

Ένας πιο αποτελεσματικός τρόπος στηρίζεται στην αυξητική ή ¨βήμα-προς-βήμα¨ ανίχνευση. Σύμφωνα με αυτή, ο ανιχνευτής σύνθετων γεγονότων γνωρίζει την ακολουθία των απλών γεγονότων που αποτελούν ένα σύνθετο γεγονός, για όλα τα σύνθετα γεγονότα. Όταν ανιχνεύεται λοιπόν ένα απλό γεγονός, ελέγχει αν αυτό το γεγονός συνεισφέρει στην ακολουθία των γεγονότων που απαρτίζουν τα σύνθετα γεγονότα, και αν αυτό συμβαίνει, σημειώνει την θέση που τα συστατικά γεγονότα του σύνθετου κατέχουν. Με αυτόν τον τρόπο, μπορούμε συνεχώς να γνωρίζουμε - ή τουλάχιστον γνωρίζει ο ανιχνευτής γεγονότων - σε ποια κατάσταση βρίσκεται η εξέλιξη της ανίχνευσης των σύνθετων γεγονότων, και μόλις ανιχνευτεί το απλό γεγονός που αποτελεί το τελευταίο απαιτούμενο γεγονός για την ολοκλήρωση της ανίχνευσης ενός σύνθετου, ο ανιχνευτής σηματοδοτεί το σύνθετο αυτό γεγονός.

Για την υλοποίηση αυτής της μεθόδου είναι αναγκαία η ύπαρξη ενός μηχανισμού που θα επιτρέπει την παρουσίαση των ακολουθιών των απλών γεγονότων και της κατάστασης της ανίχνευσης στο σύστημα. Ο μηχανισμός αυτός πρέπει να υποστηρίζει τις δομές του ορισμού των γεγονότων της γλώσσας κανόνων (τους τελεστές των σύνθετων γεγονότων, τα διαστήματα παρατήρησης) και τους υπολογισμούς των παραμέτρων (την μεταφορά των παραμέτρων των απλών γεγονότων στις παραμέτρους του σύνθετου που απαρτίζουν).

Προτάσεις για τέτοιους μηχανισμούς στηρίζονται στα:

- *Αυτόματα (automata)*. Χρησιμοποιώντας για την μοντελοποίηση και την ανίχνευση των σύνθετων γεγονότων την δομή των αυτομάτων, η ταξινομημένη χρονολογικά ακολουθία των σύνθετων γεγονότων παρουσιάζεται ως μια ακολουθία τελικών καταστάσεων. Κάθε κατάσταση αναπαριστά ένα απλό γεγονός και το αυτόματο εισέρχεται στην αντίστοιχη κατάσταση μετά την ανίχνευση του αντίστοιχου γεγονότος. Δηλαδή, κάθε φορά που ένα απλό γεγονός ανιχνεύεται, το αυτόματο προχωράει κατά μια κατάσταση. Τελικά, με την ανίχνευση του κατάλληλου σύνθετου γεγονότος το αυτόματο εισέρχεται στην τελική του κατάσταση.
- *Δέντρα (trees)*. Με την χρήση αυτής της δομής, τα σύνθετα γεγονότα αναπαριστώνται σε δέντρα, όπου ο πιο σημαντικός τελεστής της έκφρασης, που συνθέτει το γεγονός, είναι η ρίζα. Όλα τα δέντρα ενώνονται για να σχηματίσουν έναν γράφο, ο οποίος περιέχει κόμβους, φύλλα και άκρα. Οι κόμβοι αναπαριστούν τους τελεστές και μπορούν σε αυτούς να καταλήγουν ή να ξεκινούν πολλά άκρα. Τα φύλλα παρουσιάζουν τα απλά γεγονότα από τα οποία σχηματίζονται τα σύνθετα γεγονότα. Όταν ενεργοποιηθεί ένα απλό γεγονός, ενεργοποιείται το φύλλο που το αναπαριστά και αυτό στην συνέχεια ενεργοποιεί όλους τους κόμβους που συνδέονται με αυτό, κ.ο.κ. Η διαδικασία αυτή συνεχίζεται έως ότου ενεργοποιηθεί ένας κόμβος, ο οποίος αντιστοιχεί στο σύνθετο γεγονός.

- *Δίκτυα Petri (Petri nets)*. Χρησιμοποιώντας την δομή των δικτύων Petri για την αναπαράσταση και την ανίχνευση των σύνθετων γεγονότων, η ταξινομημένη ακολουθία των απλών γεγονότων που απαρτίζουν ένα σύνθετο γεγονός παρουσιάζεται ως ένας γράφος θέσεων και μεταβάσεων. Μία θέση παρουσιάζει το πρότυπο του γεγονότος και σημειώνεται με ένα σύμβολο μόλις συμβεί το γεγονός που αντιστοιχεί σε αυτήν. Όταν όλες οι θέσεις σημειωθούν, τότε ενεργοποιείται η αντίστοιχη μετάβαση και αυτό έχει ως αποτέλεσμα να σημειωθούν περαιτέρω θέσεις. Αυτό σημαίνει ότι προχωρήσαμε ένα βήμα. Εάν σημειωθεί μία θέση που αναπαριστά ένα σύνθετο γεγονός, τότε το γεγονός αυτό σηματοδοτείται.

6.5.2 Μέθοδοι αποτίμησης συνθηκών

Σε πολλά ενεργά συστήματα βάσεων δεδομένων, η διαδικασία της αποτίμησης των συνθηκών των κανόνων εισάγει την μεγαλύτερη καθυστέρηση. Όσον αφορά στη διαδικασία της ανίχνευσης των γεγονότων, είδαμε στην ενότητα ¨Μέθοδοι σηματοδότησης κανόνων¨, ότι συνήθως απαιτείται κάποια επεξεργασία, αλλά, στην επεξεργασία των πλέον πολύπλοκων γεγονότων, η χρήση των δομών που προτείνονται μειώνουν την καθυστέρηση. Όσον αφορά την διαδικασία εκτέλεσης των πράξεων των κανόνων, συχνά πρόκειται για λειτουργίες πάνω σε μεγάλα τμήματα της βάσης ή για κλήση πολύπλοκων εφαρμογών - όμως, χωρίς αυτές, οι πράξεις δεν θα μπορούσαν να εκτελέσουν την απαιτούμενη συμπεριφορά τους. Το ίδιο, συχνά, ισχύει και για τις συνθήκες, μόνο που σε αυτήν την περίπτωση, πολλές λειτουργίες μπορούν να συντομευτούν ή να αποφευχθούν. Το θέμα αυτό, δηλαδή οι μέθοδοι που μειώνουν την καθυστέρηση που εισάγει η αποτίμηση των συνθηκών, αναλύεται στην παρούσα ενότητα.

Η πιο απλή προσέγγιση στην διαδικασία του ελέγχου των συνθηκών είναι να εκτελείται μια ερώτηση για κάθε συνθήκη κανόνα μετά από κάθε αλλαγή της βάσης, και να συγκρίνονται τα καινούρια με τα προηγούμενα αποτελέσματα που επιστρέφει η ερώτηση. Φυσικά αυτή η αντιμετώπιση εισάγει καταστροφικά μεγάλη καθυστέρηση.

Διακρίνουμε μεθόδους που μειώνουν την καθυστέρηση αυτή: τα *δίκτυα διάκρισης* (*discrimination networks*) και την *αυξητική αποτίμηση* (*incremental evaluation*). Τα δίκτυα διάκρισης έχουν τις ρίζες τους στις γλώσσες της τεχνητής νοημοσύνης, και αποτελούν μία δομή δεδομένων που προέρχεται από μία ή περισσότερες συνθήκες κανόνων. Το δίκτυο έχει ως είσοδο τις μεταβολές της βάσης δεδομένων και παράγει στην έξοδό του τις συνθήκες εκείνες που ικανοποιούνται από τις μεταβολές της βάσης. Δημοφιλείς αλγόριθμοι δικτύων διάκρισης είναι οι Rete και TREAT και ο αναγνώστης μπορεί να βρει πολλές πληροφορίες πάνω σε αυτούς στα [F82], [SDLT86], [M87] και σχετικά με τους αλγορίθμους DBCond και DBQuery στο [SLR93]. Εμείς εδώ, θα παρουσιάσουμε ενδεικτικά τους αλγόριθμους Rete και TREAT.

6.5.3 Οι αλγόριθμοι Rete και TREAT

Η βασική ιδέα στον αλγόριθμο Rete [F82] είναι ότι αποθηκεύοντας την υπολογισμένη συνθήκη του κανόνα μεταξύ των φάσεων εκτέλεσης, μπορούμε να υπολογίζουμε το αποτέλεσμα των αλλαγών στην βάση δεδομένων με τον ελάχιστο επιπλέον κόπο. Ο υπολογισμός της συνθήκης αποθηκεύεται σε μια δομή που καλείται δίκτυο διάκρισης (*discrimination network*).

Το δίκτυο *Rete* έχει τους ακόλουθους κόμβους:

- τον κόμβο - ρίζα (*root node*), που χρησιμεύει ως σημείο εισόδου στο δίκτυο,
- τους κόμβους σταθερών (*t-const nodes*), που ελέγχουν τις συνθήκες επιλογής,

- τους κόμβους α-μνήμης (*α-memory nodes*), που αποθηκεύουν τα αποτελέσματα των κόμβων σταθερών,
- τους κόμβους σύζευξης *(and-nodes*), που συνδέουν τα αντικείμενα από δύο κόμβους α-μνήμης ή/και β-μνήμης,
- τους κόμβους β-μνήμης (*β-memory nodes*), οι οποίοι αποθηκεύουν τα αποτελέσματα των κόμβων σύζευξης,
- τους κόμβους (*p - nodes*), οι οποίοι αποθηκεύουν τα αντικείμενα που ταιριάζουν σε μια ολόκληρη συνθήκη κανόνα.

Η φάση ταιριάσματος στον αλγόριθμο Rete εξελίσσεται με το πέρασμα πληροφοριών από την ρίζα δια μέσου του δικτύου. Κάτω από την ρίζα υπάρχει ένας αριθμός κόμβων που αναπαριστούν τους πίνακες της βάσης. Κάτω από αυτούς μπορεί να μην υπάρχει κανένας ή να υπάρχουν πολλοί κόμβοι που φιλτράρουν μια απλή συνθήκη της σχέσης. Μετά την εφαρμογή ενός φίλτρου, οι πλειάδες που ικανοποιούν την συνθήκη αποθηκεύονται σε πίνακες α-memory. Ένας κόμβος με δυο προγόνους εκτελεί μια λειτουργία ένωσης μεταξύ των σχέσεων των προγόνων του και το αποτέλεσμα αποθηκεύεται σε ένα κόμβο β-memory. Στο τέλος του δέντρου, υπάρχει ο κόμβος με το αποτέλεσμα, το οποίο περιέχει το σύνολο προς επίλυση της σύγκρουσης (σύνολο ανταγωνισμού).

Αν και ο αλγόριθμος Rete έχει χρησιμοποιηθεί σε εμπορικά συστήματα παραγωγής, παρουσιάζονται πολλά προβλήματα ιδιαίτερα όταν χρησιμοποιούνται μεγάλες βάσεις δεδομένων: ο πλεονασμός του απαιτούμενου χώρου είναι μεγάλος, καθώς τα ενδιάμεσα αποτελέσματα των κόμβων α- και β-memory αποθηκεύονται και η διατήρηση αυτών των αποτελεσμάτων, ιδίως στην περίπτωση της διαγραφής δεδομένων, εισάγει σημαντική καθυστέρηση.

Ο αλγόριθμος TREAT [M87] είναι μια παραλλαγή του αλγορίθμου Rete, ο οποίος δεν απαιτεί την αποθήκευση των αποτελεσμάτων των λειτουργιών ένωσης στους κόμβους β-memory. Με αυτόν τον τρόπο, το δίκτυο TREAT είναι μια απλούστευση του δικτύου Rete, που περιέχει μόνο κόμβους α-memory. Κατά την διάρκεια της φάσης ταιριάσματος στον αλγόριθμο TREAT, η πληροφορία περνάει, μέσω του δικτύου, από την ρίζα προς τους κόμβους α-memory, με τον ίδιο τρόπο όπως και στον αλγόριθμο Rete. Η καινούρια πλειάδα μπορεί να χρησιμοποιηθεί σε μια σχεσιακή ερώτηση με τους άλλους κόμβους α-memory των άλλων σχέσεων σε ένα κανόνα.

Η ερώτηση του αλγορίθμου TREAT σε σχέση με την αντίστοιχη του αλγορίθμου Rete, είναι πιο οικονομική γιατί γίνεται λιγότερη χρήση της μνήμης (καθώς δεν υπάρχουν κόμβοι β-memory), και γρηγορότερη επεξεργασία της διαγραφής πλειάδων (καθώς δεν χρειάζεται να ανανεώνονται οι κόμβοι β-memory).

Παρουσιάζουμε ακολούθως ένα παράδειγμα [P94]. Έστω σε ένα σχεσιακό μοντέλο, οι σχέσεις:

```
module(code, subject, level)
course(title, duration)
contents(title, code)
```

Ένας κανόνας μπορεί να είναι:

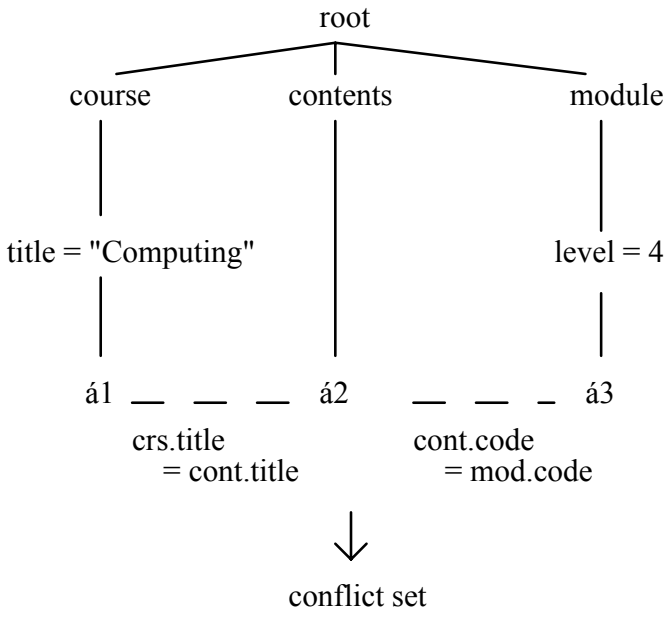
```
if crs.title = ¨Computing¨ and crs.title = cont.title
and mod.level = 4 and mod.code = cont.code
```

```
from crs in course, cont in content, mod in module
then <action>
```

Έστω ότι εισάγουμε την εγγραφή:

```
(code = 301, subject = maths, level = 4)
```

Ο αλγόριθμος θα δημιουργήσει ένα + αντικείμενο (αντικείμενο εισαγωγής) t που θα περιέχει την τιμή της πλειάδας. Το αντικείμενο θα διοχετευθεί από τον κόμβο-ρίζα σε όλους τους κόμβους που συνδέονται με αυτόν, ώστε να ελεγχθεί εάν κάποιες συνθήκες κανόνων ικανοποιούνται μετά την εισαγωγή αυτής της εγγραφής. Θα ταιριάξει μόνο με την συνθήκη πάνω από τον κόμβο α3, *level = 4*. Θα εισαχθεί στον κόμβο α3. Στην συνέχεια, εάν οι κόμβοι α2 και α1 δεν είναι άδειοι, θα γίνει σύνδεση είτε με τον α2, είτε με τον α1. Η επιλογή γίνεται αυθαίρετα. Έπειτα, εάν το σύνολο επίλυσης δεν είναι άδειο, προστίθεται σε αυτό, και δημιουργείται η νέα, τελική τιμή του συνόλου επίλυσης. Τα περιεχόμενα αυτού περνούν στον κόμβο-P και ανανεώνεται το σύνολο κανόνων προς εκτέλεση (rule agenda). Στον αλγόριθμο Rete, ισχύουν τα παραπάνω, μόνο που τα αποτελέσματα της ένωσης των α3 και α2 (ή α1) αποθηκεύονται σε έναν κόμβο β-μνήμης, και στη συνέχεια ενώνονται και με τον άλλο κόμβο α1 (ή α2). Τα δίκτυα για τους αλγορίθμους θα έχουν την μορφή των σχημάτων 6.10 και 6.11.

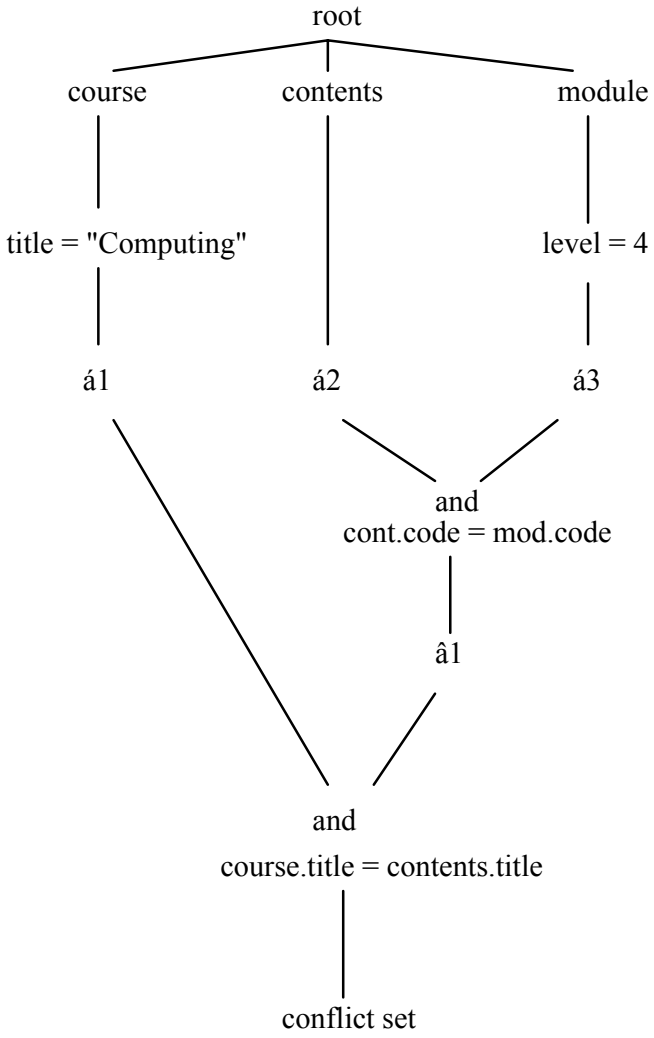


Σχήμα 6.10 Το δίκτυο TREAT για το παράδειγμα

Τα ειδικά οφέλη που παράγονται από τους κόμβους αποθήκευσης α- και β-μνήμης είναι:

- *κόμβοι α-μνήμης*: αποθηκεύουν τα αποτελέσματα της λειτουργίας της επιλογής πάνω στις σχέσεις, οι οποίες στην συνέχεια πρέπει να ενωθούν με νέες πλειάδες, ώστε να εισαχθούν στην βάση δεδομένων. Αυτή η διαδικασία μειώνει το μέγεθος των σχέσεων, οι οποίες ενώνονται κατά την διάρκεια του ταιριάσματος και επιτρέπει να εκτελεστούν επιλογές πινάκων μόνο μία φορά, όταν η πλειάδα εισαχθεί παρά πολλές φορές κατά την διάρκεια της επεξεργασίας.

- *κόμβοι β-μνήμης:* αποθηκεύουν τα αποτελέσματα της λειτουργίας της ένωσης μεταξύ των κόμβων α-μνήμης, τα οποία στην συνέχεια πρέπει να ενωθούν με άλλους κόμβους α-μνήμης. Αυτή η διαδικασία μειώνει επίσης το μέγεθος των σχέσεων που πρέπει να ενωθούν κατά την διάρκεια της φάσης ταιριάσματος.



Σχήμα 6.11 Το δίκτυο Rete για το παράδειγμα

Τα ειδικά κόστη που σχετίζονται με τους κόμβους αποθήκευσης α- και β-μνήμης είναι:

- *κόμβοι α-μνήμης:* κάθε απλή αναφορά σε μια σχέση σε έναν κανόνα πιθανά οδηγεί στη δημιουργία ενός κόμβου α-μνήμης. Στην περίπτωση όπου υπάρχει ένας σημαντικός αριθμός κανόνων, αυτή η πρόσθετη απαίτηση αποθήκευσης μπορεί να είναι μεγαλύτερη από την αρχική βάση δεδομένων. Επιπλέον, οι διαγραμμένες πλειάδες πρέπει να αφαιρούνται από τους κόμβους α-μνήμης.
- *κόμβοι β-μνήμης*: κάθε απλή ένωση στην συνθήκη ενός κανόνα πρέπει να υλοποιηθεί, το οποίο μπορεί να οδηγήσει σε καθυστερήσεις και αυξημένες απαιτήσεις αποθήκευσης πολύ μεγαλύτερες από το μέγεθος της βάσης δεδομένων. Επίσης, απαιτείται ανανέωση στην εισαγωγή και την διαγραφή των πλειάδων στους πίνακες της βάσης δεδομένων.

Συμπερασματικά, η ουσία της προσέγγισης των αλγορίθμων Rete/TREAT βασίζεται στο εξής: η επιπρόσθετη αποθήκευση που καταλαμβάνουν οι κόμβοι α-μνήμης δικαιολογείται από το κέρδος στην ταχύτητα, που είναι το αποτέλεσμα της ένωσης μικρότερων σχέσεων και της εκτέλεσης λιγότερων επιλογών στις πλειάδες που επεξεργαζόμαστε.

Εμπειρικές μελέτες και προσομοιώσεις απέδειξαν ότι για μια πλειάδα εφαρμογών ο αλγόριθμος TREAT ξεπερνά σε αποτελεσματικότητα και αποδόσεις τον αλγόριθμο Rete [WH92]. Επίσης, βελτιώσεις του αλγορίθμου TREAT σε προγράμματα και πειράματα έδειξαν ότι σε μερικές περιπτώσεις είναι απαραίτητο να παράγονται παρά να αποθηκεύονται τα αποτελέσματα των κόμβων α-memory.

6.5.4 Έλεγχος συντονισμού

Τα παθητικά συστήματα βάσεων δεδομένων επιτρέπουν σε πολλούς χρήστες και εφαρμογές να προσπελαύνουν το σύστημα ταυτόχρονα, γιαυτό και έχουν μηχανισμούς ελέγχου συντονισμού, ώστε να μην επιτρέπουν λειτουργίες που εκτελούνται από διαφορετικούς χρήστες ή εφαρμογές να αλληλεπιδρούν μεταξύ τους. Τα ενεργά συστήματα βάσεων δεδομένων επιπρόσθετα πρέπει να υποστηρίζουν μηχανισμούς ελέγχου συντονισμού, εκτός από τα δεδομένα, και για τους κανόνες.

Όπως και οι άλλες λειτουργίες της βάσης δεδομένων, οι συνθήκες και οι πράξεις των κανόνων διαβάζουν και γράφουν δεδομένα στην βάση. Στις περισσότερες ενεργές βάσεις υπάρχουν μηχανισμοί ελέγχου συντονισμού για την ανίχνευση των γεγονότων, την αποτίμηση των συνθηκών και την εκτέλεση των πράξεων. Συνήθως χρησιμοποιούνται οι παραδοσιακοί μηχανισμοί, ιδίως αν η αποτίμηση των συνθηκών και η εκτέλεση των πράξεων γίνεται μέσω του επεξεργαστή ερωτήσεων ή των εντολών προς την βάση. Επιπλέον, απαιτούνται ειδικοί μηχανισμοί για τον έλεγχο συντονισμού των λειτουργιών στους κανόνες και τις ομάδες αυτών (rule sets) - εκτός και αν οι κανόνες αποτελούν μέρος του σχήματος ή αντιμετωπίζονται ως οντότητες πρώτης κλάσης, οπότε οι παραδοσιακοί μηχανισμοί είναι επαρκείς.

Ο έλεγχος συντονισμού εξασφαλίζει την συνέπεια των αντικειμένων κατά την διάρκεια μιας δοσοληψίας, και το ίδιο ισχύει και για τους κανόνες. Για παράδειγμα, δεν είναι επιθυμητό ένας κανόνας να ενεργοποιείται από κάποιο γεγονός στην διάρκεια μιας δοσοληψίας και στην συνέχεια να μην ενεργοποιείται από ένα γεγονός ίδιου τύπου, κατά την διάρκεια της ίδιας δοσοληψίας, διότι μια άλλη σύγχρονα εξελισσόμενη δοσοληψία διέγραψε τον κανόνα. Αντίστοιχα, δεν θέλουμε ένας κανόνας να ενεργοποιείται από ένα γεγονός, εάν αυτός ο κανόνας δεν είχε ενεργοποιηθεί πρωτύτερα από γεγονός ίδιου τύπου, διότι μια σύγχρονα εξελισσόμενη δοσοληψία τον δημιούργησε στο ενδιάμεσο διάστημα. Αυτές οι απαιτήσεις καλούνται ¨*συνέπεια μέσα στις δοσοληψίες*¨ - διότι απαιτείται από μια ομάδα κανόνων να είναι συνεπής κατά την διάρκεια μιας δοσοληψίας, και ¨*συνέπεια ανάμεσα στις δοσοληψίες*¨ - διότι απαιτείται να υπάρχει συνέπεια ανάμεσα σε διαφορετικές δοσοληψίες (η ενεργοποίηση ενός κανόνα δεν μπορεί να γίνει από μια δοσοληψία που προηγήθηκε αυτής που τον δημιούργησε ή που έπεται αυτής που τον διέγραψε).

Ένας τετριμμένος μηχανισμός ελέγχου συντονισμού είναι ο περιορισμός της δημιουργίας και διαγραφής κανόνων κατά την διάρκεια δοσοληψιών, οι οποίες δεν εκτελούνται ταυτόχρονα με δοσοληψίες των χρηστών ή των εφαρμογών. Αν όμως επιζητείται η ελεύθερη δημιουργία και καταστροφή κανόνων, οι μηχανισμοί μπορεί να στηρίζονται στο σύστημα κλειδώματος της βάσης δεδομένων -δηλαδή, κατά τη διάρκεια λειτουργίας εγγραφής σε ένα κανόνα, αυτός τίθεται σε κατάσταση ¨αποκλειστικού κλειδώματος¨, ώστε να είναι απρόσιτος για τις υπόλοιπες δοσοληψίες.

6.6 ΑΝΑΛΥΣΗ ΚΑΝΟΝΩΝ

Ένα σημαντικό πρόβλημα στην σχεδίαση των εφαρμογών είναι η δυνατότητα πρόβλεψης της συμπεριφοράς των κανόνων σε όλα τα πιθανά σενάρια. Το πρόβλημα επιδεινώνεται όταν προστίθενται νέοι κανόνες σε μία εφαρμογή, οι οποίοι αλληλεπιδρούν με τους ήδη υπάρχοντες κανόνες με απρόσμενα επακόλουθα. Για μερικές γλώσσες κανόνων είναι δυνατόν να εκτελεστεί στατική αυτόματη ανάλυση σε ομάδες κανόνων ώστε να προβλεφθούν κάποιες όψεις της συμπεριφοράς των κανόνων- βλ. [AWH92], [BW94].

Οι τεχνικές ανάλυσης μπορούν να καθορίσουν εάν ένα σύνολο κανόνων θα τερματίσει (*rule termination*), εάν οι κανόνες αυτοί διέπονται από την ιδιότητα της συρροής (*confluence*), ή εάν χαρακτηρίζονται από παρατηρήσιμη ντετερμινιστική συμπεριφορά (*observable determinism*).

Πιο συγκεκριμένα, *η ιδιότητα του τερματισμού των κανόνων* σημαίνει ότι η επεξεργασία χρήσης των κανόνων θα τερματίσει μετά από οποιοδήποτε σύνολο αλλαγών στην βάση δεδομένων και σε οποιαδήποτε κατάσταση και αν βρίσκεται αρχικά η βάση.

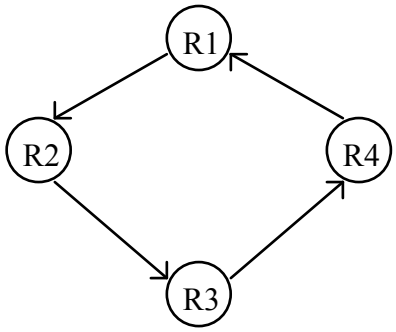
Η *ιδιότητα της συρροής* δηλώνει ότι η σειρά της εκτέλεσης κανόνων που δεν έχουν καθορισμένη προτεραιότητα δεν επιφέρει διαφορά στην τελική κατάσταση της βάσης δεδομένων, δηλαδή ότι εάν πολλοί κανόνες ενεργοποιούνται συγχρόνως κατά την διάρκεια επεξεργασίας, η τελική κατάσταση της βάσης, μετά το πέρας της επεξεργασίας των κανόνων, δεν εξαρτάται από την σειρά εκτέλεσης των κανόνων - σε αυτήν την περίπτωση οι κανόνες διέπονται από την ιδιότητα της συρροής.

Η *ιδιότητα της παρατηρήσιμης ντετερμινιστικής συμπεριφοράς* έγκειται στο ότι η σειρά εκτέλεσης κανόνων που δεν έχουν καθορισμένες προτεραιότητες δεν επιφέρει διαφορές στην σειρά ή την εμφάνιση των παρατηρούμενων πράξεων των κανόνων. *Αισθητή* ή *παρατήρησιμη* (*observable*) καλείται η πράξη που είναι ορατή και στο περιβάλλον (για παράδειγμα, η επεξεργασία δεδομένων ή η εντολή επιστροφής της βάσης σε προηγούμενη κατάσταση (*roll-back statement*)).

Στη γενική περίπτωση, είναι πολύ δύσκολο να αποφανθούμε για αυτές τις ιδιότητες. Όμως, οι συντηρητικοί αλγόριθμοι της στατικής ανάλυσης που αναπτύχθηκαν εγγυώνται ότι:

- ένα σύνολο κανόνων θα τερματίσει ή μπορεί να μην τερματίσει.
- ένα σύνολο κανόνων διέπεται από την ιδιότητα της συρροής ή μπορεί να μην διέπεται από αυτήν.
- ένα σύνολο κανόνων χαρακτηρίζεται από ντετερμινιστική συμπεριφορά ή μπορεί να μην χαρακτηρίζεται από αυτήν.

Επιπλέον, όταν η απάντηση είναι ¨αρνητική¨ για τις παραπάνω ιδιότητες, οι αλγόριθμοι απομονώνουν τους κανόνες που είναι υπεύθυνοι για το πρόβλημα και καθορίζουν κριτήρια, που όταν ικανοποιούνται, εγγυώνται την ιδιότητα. Με αυτόν τον τρόπο, η ανάλυση αποτελεί την βάση ενός ενεργού περιβάλλοντος, όπου ο προγραμματιστής κανόνων χρησιμοποιεί τον αναλυτή για την απαραίτητη πληροφορία σχετικά με την συμπεριφορά των κανόνων. Όταν οι παραπάνω ιδιότητες δεν εγγυώνται, ο χρήστης μπορεί να επαληθεύσει εάν τα κριτήρια ικανοποιούνται ή και να τροποποιήσει τα σύνολα κανόνων και να δοκιμάσει εκ νέου.



Σχήμα 6.11 Κυκλικός γράφος ενεργοποίησης

Το μοντέλο εκτέλεσης βασίζεται στον προσανατολισμένο γράφο εκτέλεσης (*directed execution graph*), ο οποίος έχει μια διακεκριμένη αρχική κατάσταση που αναπαριστά την αρχή της επεξεργασίας των κανόνων και καμία ή περισσότερες τελικές καταστάσεις που αναπαριστούν το πέρας της επεξεργασίας. Τα μονοπάτια στον γράφο παριστούν όλες τις πιθανές αλληλουχίες εκτέλεσης κατά την διάρκεια της επεξεργασίας των κανόνων. Οι κλάδοι στον γράφο είναι το αποτέλεσμα της επιλογής διαφορετικών κανόνων όταν περισσότεροι του ενός κανόνων είναι ενεργοποιημένοι και κατάλληλοι για εκτέλεση. Φανερό είναι ότι ο γράφος ενός πλήρως ταξινομημένου σε προτεραιότητες συνόλου κανόνων δεν έχει κλάδους.

6.6.1 Τερματισμός κανόνων

Το βασικό θεώρημα για τον *τερματισμό* (*termination*) των κανόνων είναι το εξής: *εάν δεν υπάρχουν κύκλοι στον γράφο που περιγράφει ένα σύνολο κανόνων, τότε αυτοί θα τερματίσουν*. Βέβαια, υπάρχουν περιπτώσεις όπου υπάρχει ένας κύκλος στον γράφο ενεργοποίησης αλλά η παρουσία άλλων ιδιοτήτων εγγυάται τον τερματισμό. Για παράδειγμα, όταν η πράξη του κανόνα r, που ανήκει στον κύκλο, αφαιρεί από τον πίνακα t στοιχεία και καμία άλλη πράξη άλλου κανόνα δεν εισάγει στον πίνακα αυτόν στοιχεία, τελικά, σε κάποια στιγμή, η πράξη του κανόνα δεν θα έχει κανένα αποτέλεσμα. Άλλο παράδειγμα είναι, όταν η πράξη ενός κανόνα r που μετέχει στον κύκλο εκτελεί κάποια μονοτονική ανανέωση (π.χ. αυξάνει την τιμή κάποιας μεταβλητής), κάποια στιγμή η συνθήκη ενός κανόνα r' του κύκλου δεν θα ισχύει (γιατί η τιμή της μεταβλητής θα έχει ξεπεράσει ένα όριο).

Αν και μερικές από αυτές τις περιπτώσεις ανιχνεύονται αυτόματα, σύμφωνα με την τεχνική ανάλυσης, ο αναλυτής καταστρώνει τον γράφο και ειδοποιεί τον χρήστη για τους κύκλους που εμφανίζονται. Στην συνέχεια, αυτός ελέγχει εάν για κάθε ένα κύκλο, κάποια συνθήκη δεν θα ισχύει στο μέλλον ή εάν κάποια πράξη δεν θα έχει μετά από έναν ορισμένο αριθμό επαναλήψεων αποτέλεσμα, οπότε και οι κανόνες θα τερματίσουν κανονικά.

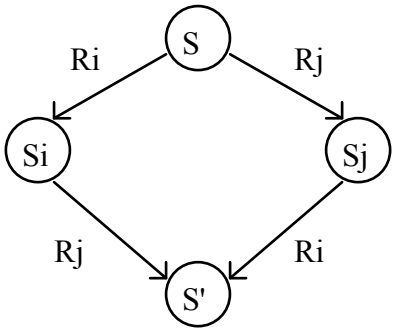
6.6.2 Ιδιότητα συρροής (Confluence)

Εάν όλοι οι κανόνες σε ένα σύνολο κανόνων έχουν καθορισμένες προτεραιότητες ή εάν ποτέ δύο ή περισσότεροι κανόνες δεν ενεργοποιούνται συγχρόνως, η επεξεργασία των κανόνων είναι πλήρως ντετερμινιστική και ισχύει η ιδιότητα της συρροής. Όμως, όταν δεν ισχύουν τα παραπάνω, τίποτα δεν εγγυάται την ιδιότητα αυτή και η ανάλυση κανόνων μπορεί να είναι μια αρκετά περίπλοκη διαδικασία.

Αρχικά πρέπει να καθοριστεί εάν δύο κανόνες είναι *ανταλλάξιμοι (commutative)*. Δύο κανόνες R1 και R2 καλούνται ανταλλάξιμοι εάν σε μια δεδομένη κατάσταση S της βάσης δεδομένων στην οποία είναι και οι δύο κανόνες ενεργοποιημένοι, η επεξεργασία του κανόνα R1

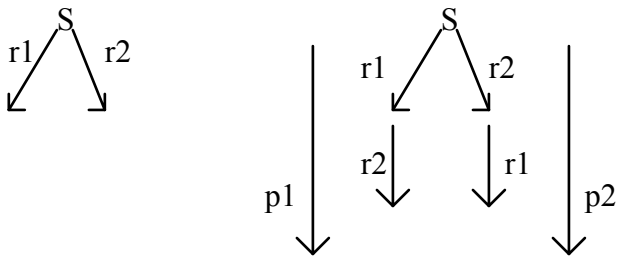
και στην συνέχεια του κανόνα R2 παράγει την ίδια κατάσταση S' όπως και η επεξεργασία του κανόνα R2 και στην συνέχεια του κανόνα R1.

Μία μέθοδος του ελέγχου της ιδιότητας της συρροής είναι ο έλεγχος εάν όλα τα ζευγάρια των κανόνων είναι ανταλλάξιμα μεταξύ τους. Όμως σε πολλές περιπτώσεις αυτή η μέθοδος είναι πολύ περιοριστική. Για παράδειγμα, εάν κάποιοι από τους κανόνες έχουν καθορισμένες προτεραιότητες ή εάν δεν γίνεται να ενεργοποιούνται συγχρόνως, τότε δεν είναι απαραίτητο να είναι και ανταλλάξιμοι μεταξύ τους για να ισχύει η ιδιότητα της συρροής.



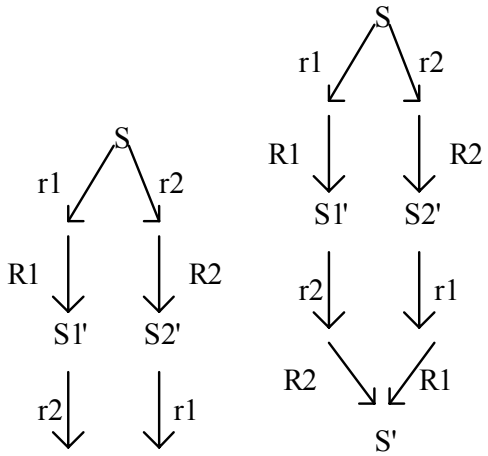
Σχήμα 6.12 Ανταλλάξιμοι κανόνες

Σύμφωνα με τις προηγούμενες παρατηρήσεις, στο [AWH92] βρίσκουμε τον ακόλουθο αλγόριθμο: Καθορίζουμε το σύνολο ανταγωνισμού που ενεργοποιείται στην αρχική κατάσταση S. Έστω οι κανόνες r1 και r2, οι οποίοι δεν έχουν καθορισμένες προτεραιότητες. Θεωρούμε ένα μονοπάτι p1 από τον r1 στον r2 και αντίστοιχα ένα μονοπάτι p2 από τον r2 στον r1.



Σχήμα 6.13 Πρώτο βήμα του αλγόριθμου ελέγχου συρροής

Εισάγουμε μεταξύ των r1 και r2 (αντίστοιχα των r2 και r1) όλους τους κανόνες που ενεργοποιούνται από τον κανόνα r1 (αντίστοιχα από τον κανόνα r2) και έχουν υψηλότερη προτεραιότητα από τον r2 (αντίστοιχα από τον r1). Καταλήγουμε σε μια τελική κατάσταση S' προσθέτοντας στο μονοπάτι p1 την αλληλουχία κανόνων R2 και στο μονοπάτι p2 την αλληλουχία κανόνων R1.



Σχήμα 6.13 Δεύτερο και τρίτο βήμα του αλγόριθμου ελέγχου συρροής

Δεδομένου ότι οι αλληλουχίες κανόνων R1 και R2 είναι ανταλλάξιμοι μεταξύ τους, κάθε γράφος εκτέλεσης που κατασκευάζεται σύμφωνα με την προηγούμενη διαδικασία έχει μία μοναδική τελική κατάσταση, δηλαδή οι κανόνες του συνόλου R διέπονται από τη ιδιότητα της συρροής.

6.6.3 Παρατηρήσιμη ντετερμινιστική συμπεριφορά (Observable determinism)

Σε μερικές γλώσσες παραγωγής κανόνων, μπορεί η τελική κατάσταση της βάσης δεδομένων να μην είναι το μοναδικό αποτέλεσμα της επεξεργασίας κανόνων -μερικές πράξεις των κανόνων μπορεί να είναι ορατές στο περιβάλλον κατά την διάρκεια της επεξεργασίας. Σε αυτή την περίπτωση, ο χρήστης επιθυμεί να καθορίσει εάν ένα σύνολο κανόνων είναι αισθητά ντετερμινιστικό - δηλαδή, εάν η σειρά και η εμφάνιση των αξιοπαρατήρητων πράξεων των κανόνων είναι ίδια, ανεξαρτήτως του κανόνα που επιλέγεται όταν πολλαπλοί κανόνες είναι ενεργοποιημένοι.

Η ανάλυση της ιδιότητας αυτής γίνεται κάνοντας χρήση των αποτελεσμάτων της μερικής συρροής. Προσθέτουμε ένα πίνακα T στην βάση δεδομένων, στον οποίο καταγράφουμε τους κανόνες εκείνους, των οποίων οι πράξεις είναι αξιοπαρατήρητες. Κατόπιν ελέγχουμε το σύνολο κανόνων που μας ενδιαφέρει, εάν διέπεται από την ιδιότητα της συρροής σε σχέση με τον πίνακα T που μόλις ορίσαμε. Εάν αυτό ισχύει και δεν υπάρχουν αόριστα μονοπάτια στον γράφο του συνόλου τότε οι κανόνες του συνόλου είναι παρατηρήσιμα ντετερμινιστικοί. Περισσότερες λεπτομέρειες καθώς και οι ορισμοί και τα θεωρήματα που περιγράφουν πλήρως τα παραπάνω βρίσκουμε στο [AWH92].

Δυστυχώς οι τεχνικές της ανάλυσης κανόνων παρουσιάζουν και κάποια μειονεκτήματα:

- Στην γενική περίπτωση, οι τεχνικές είναι *συντηρητικές (conservative)*. Δηλαδή, η ανάλυση μπορεί να καθορίσει πότε ένα σύνολο κανόνων θα τερματίσει ή εάν κάποιοι κανόνες μπορεί να μην τερματίσουν. Δεν είναι απόλυτα τα αποτελέσματα της ανάλυσης για το εάν κάποιοι κανόνες δεν θα τερματίσουν σίγουρα.
- Οι τεχνικές ανάλυσης είναι στατικές μέθοδοι και όχι δυναμικές, δηλαδή δεν λαμβάνουν υπ' όψη τους την πραγματική τωρινή κατάσταση της βάσης δεδομένων. Επομένως, μπορεί για παράδειγμα, να αποφανθούν ότι κάποιοι κανόνες δεν θα τερματίσουν αλλά η κατάσταση κατά την ώρα της εκτέλεσης της βάσης δεδομένων να εξασφαλίζει τον τερματισμό των κανόνων αυτών.

- Οι τεχνικές της ανάλυσης κανόνων εξαρτώνται από τις δομές της γλώσσας κανόνων. Κατά την ανάλυση των κανόνων αναλύονται και επεξεργάζονται ανεξάρτητα γεγονότα, συνθήκες και πράξεις κανόνων. Στην περίπτωση όπου τα γεγονότα είναι απλά και αντιστοιχούν σε συγκεκριμένες λειτουργίες, οι συνθήκες και οι πράξεις καθορίζονται με την χρήση μιας αλγεβρικής γλώσσας ερωτήσεων (όπως η SQL), η ανάλυση μπορεί να είναι μια απλή διαδικασία. Στην αντίθετη περίπτωση, όπου τα γεγονότα είναι σύνθετα και ορισμένα από εξωτερικές διαδικασίες, οι συνθήκες και οι πράξεις των κανόνων καθορίζονται από μια γλώσσα προγραμματισμού εφαρμογών, η ανάλυση μπορεί να γίνει δύσκολη και περίπλοκη διαδικασία.
- Οι τεχνικές ανάλυσης εξαρτώνται άμεσα από την σημασιολογία της εκτέλεσης των κανόνων, η οποία διαφέρει σημαντικά στα διάφορα ενεργά συστήματα.

Παρά, όμως, αυτά τα μειονεκτήματα, οι τεχνικές ανάλυσης είναι χρήσιμες και υποστηρίζουν την εργασία του σχεδιαστή των κανόνων. Όταν αρχικά οριστεί ένα σύνολο κανόνων, οι τεχνικές βοηθούν στην πρόβλεψη της συμπεριφοράς των κανόνων, και όταν αργότερα, προστεθούν και άλλοι κανόνες, οι τεχνικές εξηγούν πώς αυτοί οι νέοι κανόνες θα επηρεάσουν την ολική συμπεριφορά του συστήματος.

Αν και οι μέθοδοι που περιγράφονται αναλυτικά στο [AWH92] αναπτύχθηκαν και παρουσιάστηκαν στα πλαίσια του συστήματος κανόνων Starburst, μπορούν εύκολα να τροποποιηθούν και να προσαρμοστούν και σε άλλες γλώσσες κανόνων. Σημαντικό επίσης είναι ότι *τα βασικά στοιχεία των τεχνικών ανάλυσης μπορούν να παραμείνουν αμετάβλητα:* ο γράφος ενεργοποίησης για την ανάλυση τερματισμού, τα λήμματα που ορίζουν την συρροή, η έννοια της μερικής συρροής και η χρήση της στην ανάλυση της αισθητής ντετερμινιστικής συμπεριφοράς.

6.7 ΕΜΠΟΡΙΚΑ ΣΥΣΤΗΜΑΤΑ

Στην ενότητα αυτή θα παρουσιάσουμε τον τρόπο με τον οποίο δυο κλασικά σχεσιακά συστήματα, η Oracle και η DB2, αποκτούν ενεργή συμπεριφορά. Επιπλέον, θα παρουσιάσουμε και τον τρόπο με τον οποίο οι ενεργοποιητές αναμένεται ότι θα τυποποιούνται στο SQL-3 στάνταρτ.

6.7.1 ORACLE

Η Oracle παρακολουθεί στενά την εξέλιξη του SQL στάνταρτ και εδώ και αρκετά χρόνια δίνει την δυνατότητα στους χρήστες της να ορίσουν και να χρησιμοποιήσουν *ενεργοποιητές* (triggers). Οι αρχικές μορφές των ενεργοποιητών ήταν αρκετά πρωτόγονες αλλά από την έκδοση 7.2 και μετά οι ενεργοποιητές της Oracle είναι πολύ κοντά στην πρόταση της SQL-3.

Οι ενεργοποιητές στην Oracle ουσιαστικά υλοποιούν κανόνες της μορφής Γεγονός-Πράξη (event-action) ενώ υπάρχει τρόπος επιβολής κάποιας συνθήκης μέσα στο σώμα της πράξης (μέσω μιας `WHERE` συνθήκης). Τα γεγονότα που μπορούν να χρησιμοποιηθούν είναι αυστηρά προκαθορισμένα και περιορίζονται σε συνδυασμό απλών γεγονότων μεταβολής δεδομένων. Δηλαδή, μόνο οι πράξεις της εισαγωγής (`INSERT`), διαγραφής (`DELETE`) και μεταβολής (`UPDATE`) δεδομένων (σε κάποιο πίνακα) μπορούν να προκαλέσουν ενεργοποίηση.

Η "πράξη" ενός ενεργοποιητή μπορεί να εκτελεστεί ακριβώς πριν ή ακριβώς μετά την εκτέλεση του γεγονότος που την προκάλεσε. Το πότε ακριβώς θα γίνει η εκτέλεση καθορίζεται στον ορισμό του ενεργοποιητή. Σε κάθε περίπτωση, η εκτέλεση της "πράξης" ενός ενεργοποιητή γίνεται ως τμήμα της δοσοληψίας που προκάλεσε την ενεργοποίηση (έχουμε δηλαδή άμεση σύζευξη). Για την ακρίβεια δημιουργείται μια υποδοσοληψία η οποία

περιλαμβάνει: 1) το γεγονός (την εντολή μεταβολής των δεδομένων) που προκάλεσε την ενεργοποίηση και 2) το τμήμα "πράξης" του ενεργοποιητή. Αυτή η υποδοσοληψία μπορεί να αναιρεθεί (roll back) χωρίς να χρειαστεί να αναιρεθεί η κυρίως δοσοληψία του χρήστη.

Το τμήμα της "πράξης" ενός ενεργοποιητή είναι ένα γενικό PL/SQL block. Η εκτέλεσή του μπορεί να γίνεται για κάθε πλειάδα ξεχωριστά ή συνολικά για όλες τις πλειάδες που μεταβλήθηκαν από το συγκεκριμένο γεγονός (όλες οι πλειάδες που έγιναν `INSERT, DELETE` ή `UPDATE`). Η Oracle, δηλαδή, υποστηρίζει και ενεργοποιητές προσανατολισμένους προς το στιγμιότυπο αλλά και ενεργοποιητές προσανατολισμένους προς το σύνολο.

Κατά την εκτέλεση της "πράξης" ενός ενεργοποιητή προσανατολισμένου προς το στιγμιότυπο ο χρήστης μπορεί να αναφερθεί στην παλιά και την νέα τιμή της πλειάδας μέσα από ειδικές ονομασίας που καθορίζονται κατά τον ορισμό του κανόνα (`OLD AS..., NEW AS...`). Στην περίπτωση, όμως, που έχουμε ενεργοποιητές προσανατολισμένους προς το σύνολο δεν μπορούμε να αναφερθούμε στις νέες και παλιές τιμές.

Η σημασιολογία εκτέλεσης των κανόνων δεν είναι πολύ αυστηρά ορισμένη. Συγκεκριμένα δεν υπάρχει ξεκάθαρος ορισμός της σειράς με την οποία θα εκτελεστούν οι διάφοροι ενεργοποιητές που τυχαίνει να παρακολουθούν το ίδιο γεγονός.

Η σύνταξη για τον ορισμό των ενεργοποιητών στην Oracle είναι η παρακάτω:

```
<Oracle-trigger>::=CREATE TRIGGER <trigger-name>
                   { BEFORE | AFTER } <trigger-events>
                   ON <table-name>
                   [ [ REFERENCING <references> ]
                   FOR EACH ROW
                   [ WHERE ( <conditions> ) ] ] <PL/SQL>

<trigger event>::= INSERT | DELETE | UPDATE
                   [ OF <column-names> ]

<reference>::=     OLD AS <old-value-tuple-name>      |
                   NEW AS <new-value-tuple-name>
```

6.7.2 DB2

Οι ενεργοποιητές στην DB2 ορίστηκαν σχετικά πρόσφατα από την IBM η οποία, μετά την εμπειρία που είχε με το ερευνητικό πρότυπο Starbust, προσπάθησε να ορίσει αυστηρά την σημασιολογία των ενεργοποιητών. Ιδιαίτερο πρόβλημα υπάρχει σε σχέση με την επιβολή των αναφορικών περιορισμών ορθότητας. Οι πράξεις των ενεργοποιητών μπορεί να προκαλούν παραβίαση των περιορισμών αυτών ενώ ταυτόχρονα η επιβολή των περιορισμών αυτών μπορεί να προκαλέσει την εκτέλεση εντολών που γενούν γεγονότα. Έτσι, υπάρχει πρόβλημα στον ορισμό της σειράς με την οποία θα εκτελούνται αυτές οι ενέργειες από την βάση. Η IBM για να λύσει αυτό το πρόβλημα πρότεινε μια δική της μεθοδολογία η οποία στην συνέχεια υιοθετήθηκε και από το SQL-3.

Οι ενεργοποιητές στην DB2 είναι ουσιαστικά κανόνες της μορφής ECA. Το τμήμα "γεγονός" μπορεί να είναι μόνο κάποιο απλό γεγονός μεταβολής δεδομένων (εισαγωγή, διαγραφή ή μεταβολή). Σε αντίθεση με την ORACLE, δεν υποστηρίζει συνδυασμό γεγονότων. Το τμήμα της "συνθήκης" είναι μια γενική SQL `WHERE` συνθήκη η οποία πρέπει να αληθεύει για κάποια πλειάδα προκειμένου να γίνει η εκτέλεση του κανόνα. Σε αυτό το θέμα η DB2 είναι

πιο πλήρης σε σχέση με την ORACLE, αφού επιτρέπει τον ορισμό τμήματος συνθήκης για όλους τους τύπους των ενεργοποιητών.

Ο ενεργοποιητής μπορεί να οριστεί να εκτελεστεί ακριβώς πριν ή ακριβώς μετά την εκτέλεση του γεγονότος που το ενεργοποίησε όπως ακριβώς και στην Oracle. Η εκτέλεση της "πράξης" του ενεργοποιητή μπορεί να γίνει για κάθε πλειάδα ξεχωριστά ή μια φορά συνολικά για το γεγονός (την εντολή) που προκάλεσε την ενεργοποίηση.

Κατά την εκτέλεση της "πράξης" του ενεργοποιητή, στην DB2 μπορούμε πάντα να αναφερθούμε στις παλιές και νέες τιμές μέσω δεσμευμένων όρων. Όταν ο ενεργοποιητής είναι προσανατολισμένος προς το στιγμιότυπο (ενεργοποίηση για κάθε πλειάδα) τότε το τμήμα "πράξης" του ενεργοποιητή έχει στην διάθεσή του τόσο τις παλιές όσο και τις νέες τιμές της συγκεκριμένης πλειάδας. Διαφορετικά, στην περίπτωση ενεργοποιητή προσανατολισμένου προς το σύνολο (ενεργοποίηση συνολικά για το γεγονός), μπορούμε να αναφερθούμε στο σύνολο (την σχέση) των νέων πλειάδων και στο σύνολο (την σχέση) των παλιών πλειάδων (αυτές οι προσωρινές σχέσεις λέγονται και *σχέσεις δέλτα*). Προφανώς, όταν το γεγονός του ενεργοποιητή είναι η πράξη `INSERT` δεν υπάρχουν παλιές τιμές ενώ όταν η πράξη είναι `DELETE` δεν υπάρχουν νέες τιμές.

Η "πράξη" ενός ενεργοποιητή μπορεί να είναι γενικά κάποια SQL εντολή αλλά αυτή η εντολή δεν μπορεί να περιέχει εντολές ελέγχου της δοσοληψίας (`ABORT, COMMIT,...`). Η "πράξη" θα εκτελεστεί στην ίδια δοσοληψία με το γεγονός που την προκάλεσε. Όπως και στην Oracle, έχουμε υποδοσοληψίες οι οποίες μπορούν να αναιρεθούν με κατάλληλες `ABORT` εντολές χωρίς να χρειαστεί να αναιρεθεί η συνολική δοσοληψία του χρήστη.

Η DB2 κάνει ειδική μεταχείριση στους ενεργοποιητές που εκτελούνται πριν από το γεγονός που τους ενεργοποίησε. Η "πράξη" αυτών των ενεργοποιητών δεν επιτρέπεται να μεταβάλει την κατάσταση της βάσης αλλά μόνο να μεταβάλει τις νέες τιμές των πλειάδων που προκαλούν την ενεργοποίηση. Έτσι, αυτού του είδους οι ενεργοποιητές δεν μπορούν να προκαλέσουν την αλυσιδωτή ενεργοποίηση άλλων ενεργοποιητών.

Η σημασιολογία της εκτέλεσης των ενεργοποιητών είναι καλά ορισμένη στην DB2. Στην περίπτωση ταυτόχρονης ενεργοποίησης από το ίδιο γεγονός υπάρχει μια προκαθορισμένη σειρά με την οποία θα γίνει η εκτέλεση. Αυτή η σειρά ορίζεται με βάση την χρονική στιγμή ορισμού των ενεργοποιητών.

Το συντακτικό ορισμού των ενεργοποιητών στην DB2 είναι το παρακάτω:

```
<DB2-trigger>::= CREATE TRIGGER <trigger-name>
                 { BEFORE | AFTER } <trigger-event>
                 ON <table-name>
                 [ REFERENCING <references> ]
                 FOR EACH { ROW | STATEMENT }
                 WHERE ( <SQL-condition> )
                 <SQL-procedure-statements>

<trigger event>::= INSERT | DELETE | UPDATE
                 [ OF <column-names> ]

<reference>::=   OLD AS <old-value-tuple-name>      |
                 NEW AS <new-value-tuple-name>      |
                 OLD_TABLE AS <old-value-table-name> |
```

```
NEW_TABLE AS <new-value-table-name>
```

6.7.3 SQL-3

Το προσεχές πρότυπο SQL-3 [ISO94] θα περιέχει πολλά στοιχεία σχετικά με τους ενεργοποιητές - αν και πιο στενά συνδεδεμένα με τους αναφορικούς περιορισμούς ορθότητας, όπως αυτοί περιγράφονται στο πρότυπο SQL92 [MS93]. Το σύστημα DB2 είναι πολύ κοντά στο προτεινόμενο SQL-3 και σχεδόν όλα όσα αναφέραμε στην προηγούμενη ενότητα ισχύουν για το SQL-3. Στην συνέχεια θα προσπαθήσουμε να επαναλάβουμε τα χαρακτηριστικά των ενεργοποιητών όπως προβλέπονται στο SQL-3.

Η γραμματική ορισμού των ενεργοποιητών στην SQL-3 είναι η εξής:

```
<trigger definition>::=
        CREATE TRIGGER <trigger-name>
        <trigger action time> <trigger event>
        ON <table name>
        [ REFERENCING <old or new values alias list> ]

<trigger action time>::=
        BEFORE |
        AFTER

<trigger event>::=
        INSERT |
        DELETE |
        UPDATE [ OF <column-names> ]

<trihgger column list>::= <column name list>

<triggered action>::=
        [ FOR EACH { ROW | STATEMENT } ]
         [ WHEN ( <search condition> ) ]
        <triggered SQL statement>

<triggered SQL statement>::=
        <SQL procedure statement> |
        BEGIN ATOMIC
         { <SQL procedure statement> <semicolon> }...
        END

<old or new values alias list>::=
        <old or new values alias>...

<old or new values alias>::=
        OLD [ ROW ] [ AS ] <old values correlation name> |
        NEW [ ROW ] [ AS ] <new values correlation name> |
```

```
OLD TABLE [ AS ] <old values table alias> |
NEW TABLE [ AS ] <new values table alias>
```

Στην SQL-3 οι ενεργοποιητές ορίζονται ως μέρος του σχήματος της βάσης. Έχουν κάποιο μοναδικό όνομα και αναφέρονται αναγκαστικά σε ένα και μόνο πίνακα της βάσης (base table).

Υπάρχουν δυο βασικές κατηγορίες ενεργοποιητών: *αυτοί που εκτελούνται πριν το γεγονός που τα προκάλεσε* (BEFORE triggers) και *αυτοί που εκτελούνται μετά* (AFTER triggers).

Κάθε ενεργοποιητής ορίζει κάποιο "γεγονός" που προκαλεί την ενεργοποίηση και έχει ένα τμήμα "πράξης" που καθορίζει τι θα εκτελεστεί όταν γίνει η ενεργοποίηση. Επιπλέον μπορεί να υπάρχει και ένα τμήμα "συνθήκης".

Για να ενεργοποιηθεί ένας ενεργοποιητής πρέπει να συμβεί το "γεγονός" που ορίζει ο ενεργοποιητής. Τα "γεγονότα" για τους ενεργοποιητές της SQL-3 είναι μόνο οι απλές πράξεις της εισαγωγής, διαγραφής και μεταβολής πλειάδων (δηλαδή οι εντολές SQL: `INSERT`, `DELETE` και `UPDATE`). Αν για παράδειγμα, οριστεί ένας ενεργοποιητής πάνω στον πίνακα Α και για το γεγονός `INSERT` τότε οποτεδήποτε συμβαίνει εισαγωγή μιας νέας εγγραφής στον πίνακα Α έχουμε ενεργοποίηση του συγκεκριμένου ενεργοποιητή.

Σε κάθε ενεργοποιητή μπορεί να υπάρχει προαιρετικά ένα τμήμα "συνθήκης". Αυτή η συνθήκη αποτιμάται ακριβώς πριν την εκτέλεση του τμήματος "πράξη" και αφού έχει γίνει η ενεργοποίηση. Το τμήμα "πράξης" εκτελείται μόνο αν η συνθήκη ισχύει. Η συνθήκη έχει την γενική μορφή μιας δήλωσης `WHERE` της SQL.

Στον ορισμό του ενεργοποιητή μπορεί να καθοριστεί αν το τμήμα "πράξη" θα εκτελείται μια φορά για κάθε πλειάδα του γεγονότος (`FOR EACH ROW`) ή μόνο μία φορά για όλες τις πλειάδες μαζί (`FOR EACH STATEMENT`). Αν για παράδειγμα, έχουμε την εισαγωγή 5 πλειάδων στον πίνακα Α με μια εντολή `INSERT` και ο παραπάνω ενεργοποιητής είχε οριστεί "`FOR EACH ROW`" τότε θα είχαμε 5 εκτελέσεις της "πράξης" του ενεργοποιητή. Διαφορετικά θα είχαμε μόνο μία εκτέλεση.

Το τμήμα της "πράξης" είναι μια SQL δήλωση (μία ή περισσότερες "SQL procedure statement").

Τόσο μέσα στο τμήμα της "πράξης" όσο και μέσα στο τμήμα "συνθήκης" μπορούν να υπάρχουν αναφορές στις παλιές και νέες τιμές των πλειάδων. Στην περίπτωση που ο ενεργοποιητής είναι δηλωμένος "`FOR EACH STATEMENT`" τότε οι παλιές τιμές των πλειάδων θεωρούμε ότι αποτελούν ένα προσωρινό πίνακα στον οποίο μπορούν να υπάρχουν αναφορές μέσα στο σώμα της δήλωσης του ενεργοποιητή. Το ίδιο ισχύει και για τις νέες τιμές. Τα ονόματα με τα οποία γίνονται οι αναφορές στις παλιές και νέες τιμές καθορίζονται από της δηλώσεις "`OLD ROW AS …`", "`NEW ROW AS …`", "`OLD TABLE AS …`" και "`NEW TABLE AS …`".

6.8 ΑΝΑΦΟΡΕΣ

[AWH92] A.Aiken, J.Widom, J.M.Hellerstein. Behavior of database production rules: Termination, confluence, and observable determinism. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 59-68, San Diego, California, June 1992.

[BW94] E.Baralis, J.Widom. An algebraic approach to rule analysis in expert database systems. In *Proceedings of the 20th International Conference on Very Large Data Bases*, Santiago, Chile, September 1994.

[Ch92] S. Chakravarthy. Architectures and Monitoring Techniques for Active Database: An Evaluation. Tech. Report UF-CIS-TR-92-041, University of Florida, November 1992.

[F82] C.L.Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. In *Artificial Intelligence*, 19:17 - 37, 1982.

[G94] S. Gatziu. *Events in an Active Object-Oriented Database System*. PhD thesis, University of Zurich, 1994.

[GGD95] S. Gatziu, A.Geppert and K.R.Dittrich. A Designer's Benchmark for Active Database Management Systems: 007 meets the BEAST. In T.Sellis, editor, *Rules in Database Systems*, Proc. of the 2nd Int' l Workshop RIDS '95, pages 309-323, Springer, Athens, September 1995.

[ISO94] ISO-ANSI working draft: Database language SQL3, 1994. X3H2/94/080 and SOU/003.

[M87] D.P.Miranker. TREAT: A better match algorithm for AI production systems. In *Proc. of the AAAI Conf. on Artificial Intelligence*, pages 42-47, August 1987.

[MS93] J.Melton and A.R.Simon. *Understanding the New SQL: a Complete Guide*. Morgan Kaufmann, San Fransisco, California, 1993.

[P94] N.W.Paton. Implementing production rules for Object-Oriented Databases. Technical report, Dept. of Computing and electrical Engineering, Heriot-Watt University, Edinburgh, Scotland, 1994.

[SDLT86] M.I.Schor, T.P.Daly, H.S.Lee, and B.R.Tibbitts. Advances in RETE pattern matching. In *Proc. of the 5th National Conf. on Artificial Intelligence*, pages 226 - 232, Philadelphia, Pennsylvania, August 1986.

[SLR93] T.Sellis, C.-C.Lin, L.Raschid. Coupling Production Systems and Database Systems: A Homogeneous Approach. *IEEE Transactions on Knowledge and Data Engineering*, 5(2), April 1993.

[SPAM91] U.Schreier, H.Pirahesh, R.Agrawal and C.Mohan. Alert: An achitecture for transforming a passive dbms into an active dbms. In *Proceedings 17th International Conference on Very Large Data Bases*, pages 469-478, Barcelona (Catalonia, Spain), Sept.1991.

[WC96] J. Widom and S.Ceri. *Active Database Systems - Triggers and Rules for Advanced Database Processing*. Morgan Kaufman Publishers Inc., 1996.

[WH92] Y.-W.Wang and E.N.Hanson. A performance comparison of the Rete and TREAT algorithms for testing database rule conditions. In *Proc. of the 8th Int'l. Conf. on Data Engineering*, pages 88-97, Tempe, Arizona, Feb 1992.