ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ
ΜΕΣΟΛΟΓΓΙΟΥ

ΤΜΗΜΑ ΕΦΑΡΜΟΓΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΣΤΗΝ
ΔΙΟΚΗΣΗ ΚΑΙ ΤΗΝ ΟΙΚΟΝΟΜΙΑ

ΜΟΝΤΕΛΑ ΔΕΔΟΜΕΝΩΝ ΓΙΑ ΧΡΟΝΙΚΕΣ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

ΕΙΣΗΓΗΤΗΣ: ΜΑΚΡΗΣ ΧΡΗΣΤΟΣ
ΕΠΙΚΟΥΡΟΣ ΚΑΘΗΓΗΤΗΣ

ΣΠΟΥΔΑΣΤΕΣ	ΑΜ
ΡΙΖΟΥ ΓΙΑΝΝΟΥΛΑ	7885
ΤΣΙΧΛΗΣ ΝΙΚΟΛΑΟΣ	7965

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1: ΣΧΕΣΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΔΙΑΧΕΙΡΙΣΗΣ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

Το εισαγωγικό αυτό κεφάλαιο προορίζεται να χρησιμεύσει σαν ένα γρήγορο μάθημα εμπέδωσης του σχεσιακού μοντέλου, συνοψίζοντας κάποιες έννοιες τις οποίες εν *συντομία* θα αναλύσουμε, η γνώση των οποίων κρίνεται αναγκαία για την κατανόηση των επόμενων κεφαλαίων:

- Types (τύποι)
- Relation values (τιμές σχέσης)
- Relation variables (μεταβλητές σχέσης)
- Integrity constraints (περιορισμοί ακεραιότητας)
- Relational operators (σχεσιακοί τελεστές)
- The relational model (*το* σχεσιακό μοντέλο)

Το παρακάτω παράδειγμα (figure 1.1) δείχνει ένα σύνολο από τιμές για μια βάση δεδομένων.

παράδειγμα 1.1

Β.Δ. Προμηθευτών -Αποστολών

S

S#	SNAME	STATUS	CITY
S1	SMITH	20	LONDON
S2	JONES	10	PARIS
S3	BLAKE	30	PARIS
S4	CLARK	20	LONDON
S5	ADAMS	30	ATHENS

SP

S#	P#
S1	P1
S1	P2
S1	P3
S1	P4
S1	P5
S1	P6
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4
S4	P5

Συνολικά, η βάση δεδομένων προορίζεται να ερμηνευθεί ως εξής:
Η σχέση S δείχνει τους προμηθευτές που είναι αυτήν την περίοδο στο πλαίσιο της σύμβασης. Κάθε προμηθευτής έχει έναν αριθμό προμηθευτή (S#), μοναδικός για εκείνο τον προμηθευτή, ένα όνομα προμηθευτή (SNAME), όχι απαραιτήτως μοναδικό (*αν και* οι τιμές δειγμάτων SNAME που παρουσιάζονται στο παράδειγμα 1.1 συμβαίνει να είναι μοναδικές) *μια* θέση (STATUS) και μια τοποθεσία (CITY).
Η σχέση SP δείχνει τις πιθανές αποστολές των μερών από τους προμηθευτές. Παραδείγματος χάριν, οι τιμές δειγμάτων στο παράδειγμα 1.1 δείχνουν μεταξύ άλλων ότι ο S1 προμηθευτής είναι αυτήν την περίοδο ικανός να τροφοδοτήσει το μέρος P1. Σαφώς, ο συνδυασμός S# τιμής και P# τιμής για μια δεδομένη "πιθανή αποστολή" είναι μοναδικός για εκείνη την αποστολή.

1.1 Types (τύποι)

Οι σχέσεις στο σχεσιακό μοντέλο καθορίζονται από τύπους. Τι είναι ένας όμως τύπος; Είναι ένα πεπερασμένο (μετρήσιμο) σύνολο τιμών. Πεπερασμένο επειδή ασχολούμαστε με συγκροτήματα ηλεκτρονικών υπολογιστών, τα οποία είναι πεπερασμένα εξ ορισμού. Περιλαμβάνουν το σύνολο όλων των ακέραιων αριθμών (type integer) και το σύνολο όλων των σειρών χαρακτήρα (type char). Κάθε τιμή έχει, ή είναι κάποιος τύπος. *Στην* πραγματικότητα, κάθε τιμή είναι ακριβώς ένας τύπος.
Επιπλέον ισχύουν:

α)κάθε μεταβλητή πρέπει να είναι κάποιου τύπου, που σημαίνει ότι κάθε πιθανή τιμή της ζητούμενης μεταβλητής, είναι μια τιμή του εν λόγω τύπου

β)κάθε ιδιότητα κάθε σχέσης πρέπει να είναι κάποιου τύπου, που σημαίνει ότι κάθε πιθανή τιμή της ζητούμενης ιδιοτήτας, είναι μια τιμή του εν λόγω τύπου

γ)κάθε τελεστής που επιστρέφει ένα αποτέλεσμα θα πρέπει να είναι κάποιου τύπου, δηλαδή κάθε πιθανό αποτέλεσμα που επιστρέφεται από επίκληση του ζητούμενου τελεστή, θα πρέπει να είναι του ίδιου τύπου

δ)κάθε παράμετρος κάθε τελεστή θα πρέπει να είναι κάποιου τύπου, που σημαίνει ότι κάθε πιθανό όρισμα που μπορεί να αντικαταστήσει την εν λόγω παράμετρο είναι μια τιμή του εν λόγω τύπου

ε)γενικότερα, κάθε έκφραση που δηλώνεται θα πρέπει να είναι κάποιου τύπου

Οποιοσδήποτε τύπος που δίνεται, είτε ορίζεται από το σύστημα είτε ορίζεται απ' το χρήστη. Απ' τους τύπους που προαναφέραμε (integer, char), και οι δυο ορίζονται απ΄ το σύστημα. Οποιοσδήποτε τύπος, ανεξάρτητα από το εάν καθορίζεται από το σύστημα ή όχι, μπορεί να χρησιμοποιηθεί ως βάση για τις μεταβλητές, τις ιδιότητες, τους τελεστές, και τις παραμέτρους.
Οποιοσδήποτε τύπος που δίνεται είναι είτε κλιμακωτός είτε μη κλιμακωτός. Μη κλιμακωτός είναι ο τύπος που έχει ένα σύνολο ορατών απ' το χρήστη ιδιοτήτων. Ειδικότερα οι σχεσιακοί τύποι είναι μη κλιμακωτοί, επειδή κάθε σχεσιακός τύπος έχει ένα σύνολο ορατών ιδιοτήτων. Κλιμακωτός τύπος σημαίνει ότι δεν έχει ένα σύνολο ορατών χαρακτηριστικών, όπως οι οριζόμενοι απ'το σύστημα τύποι integer.

1.2 Relation values (τιμές σχέσης)

Δίνοντας μια συλλογή του τύπου Ti(i = 1, 2..., n), *το* r είναι μια σχέση (ή τιμή σχέσης) σε εκείνους τους τύπους εάν αποτελείται από δύο μέρη, έναν τίτλο και ένα σώμα, όπου:

- τίτλος είναι ένα σύνολο n ιδιοτήτων, μια για κάθε Ti. Κάθε ιδιότητα αποτελείται από ένα όνομα ιδιοτήτων Ai και το αντίστοιχο όνομα τύπου Ti.
- *το* σώμα είναι ένα σύνολο από m πλειάδες, όπου κάθε πλειάδα είναι ένα σύνολο n τμημάτων, ένα για κάθε ιδιότητα στον τίτλο. Έστω ότι t είναι μια τέτοια πλειάδα. Τότε, κάθε συστατικό του t αποτελείται από το όνομα ιδιοτήτων Ai *και μια* αντίστοιχη τιμή vi του τύπου Ti.

Οι τιμές m *και* n καλούνται αριθμός στοιχείων συνόλου και βαθμός, της σχέσης r.

Σημαντικά σημεία που προκύπτουν απ' αυτόν τον ορισμό είναι τα παρακάτω:

1. Από μια συνοπτική εικόνα μιας σχέσης, βλέπουμε οτι ο τίτλος είναι η γραμμή των ονομάτων των στηλών και το σώμα είναι το σύνολο των δεδομένων της σειράς
2. Υπάρχει μια λογική διαφορά μεταξύ του ονόματος μιας ιδιότητας *και μιας* ιδιότητας. Συχνά χρησιμοποιούμε εκφράσεις όπως "attribute A"(ιδιότητα A), αλλά αυτές οι εκφράσεις θα πρέπει να σημαίνουν μια ιδιότητα της οποίας το όνομα είναι A
3. Μια σχέση δεν περιέχει πλειάδες. Περιέχει ένα σώμα, και εκείνο το σώμα περιέχει τις πλειάδες
4. Μια σχέση πρώτου βαθμού ονομάζεται μοναδιαία, μια σχέση δευτέρου βαθμού ονομάζεται δυαδική, μια σχέση τρίτου βαθμού ονομάζεται τριαδική και μια σχέση n βαθμού λέγεται n-αδική. Μια σχέση με βαθμό 0 λέγεται μηδενική (nullary)
5. Καμιά σχέση δεν περιέχει ποτέ διπλότυπες πλειάδες
   Οι πλειάδες t1 *και* t2 είναι διπλότυπες εάν και μόνο εάν περιέχουν ακριβώς τις ίδιες ιδιότητες A1,A2,….,An και για κάθε i(i=1,2,3,…,n), *η* τιμή του Ai *στο* t1 είναι ίση με τη τιμή του Ai *στο* t2
6. Δεν υπάρχει καμιά πάνω προς τα κάτω(top-to-bottom) σειρά για τις πλειάδες μιας σχέσης
7. Δεν υπάρχει καμιά δεξιά προς αριστερά(left-to-right) σειρά για τις ιδιότητες μιας σχέσης
8. Κάθε πλειάδα κάθε σχέσης περιέχει μια τιμή για κάθε ιδιότητα γι' αυτή τη σχέση. Οι σχέσεις αυτές είναι σε πρώτη κανονική μορφή

Οι σχέσεις και η έννοιά τους

Λαμβάνοντας υπόψη μας μια σχέση r, ο τίτλος του r μπορεί να θεωρηθεί ένα ορισμένο κατηγόρημα (δηλ., μια λειτουργία με ένα σύνολο παραμέτρων, το οποίο επιστρέφει μια τιμή αληθινή ή ψευδή όταν επικαλείται). Το κατηγόρημα που αντιστοιχεί στη σχέση r είναι το κατηγόρημα σχέσης γι' αυτή την σχέση.

Ακόμα, κάθε πλειάδα στο σώμα του r μπορεί να θεωρηθεί μια ορισμένη πρόταση (δηλ., μια δήλωση που είναι είτε αληθινή είτε ψευδή). Οι προτάσεις αντιστοιχούν στις επικλήσεις του κατηγορήματος σχέσης. Λαμβάνονται από αυτό το κατηγόρημα με την αντικατάσταση των επιχειρημάτων του κατάλληλου τύπου για τις παραμέτρους του κατηγορήματος.

Γενικότερα, μπορούμε να πούμε ότι το κατηγόρημα που αντιστοιχεί σε μια δεδομένη σχέση είναι η προοριζόμενη ερμηνεία ή η έννοια γι' αυτή την σχέση, και οι προτάσεις που αντιστοιχούν στις πλειάδες αυτής της σχέσης γίνονται κατανοητές από τη σύμβαση για να είναι αυτές που αξιολογούνται σε αληθινές.

1.3 Relation variables (μεταβλητές σχέσης)

Εξετάστε τη βάση δεδομένων προμηθευτής και αποστολές όπως φαίνεται στο σχήμα 1.1. Όπως σημειώνεται σε προηγούμενο τμήμα, αυτός ο αριθμός παρουσιάζει δύο τιμές σχέσης. Συγκεκριμένα, οι τιμές σχέσης που μπορούν να εμφανιστούν στη βάση δεδομένων σε κάποιο ιδιαίτερο χρόνο. Αλλά φυσικά, εάν επρόκειτο να εξετάσουμε τη βάση δεδομένων σε κάποιο διαφορετικό χρόνο, πιθανώς θα δούμε δύο διαφορετικές τιμές σχέσης. Με άλλα λόγια, η S *και* SP σε αυτή την βάση δεδομένων είναι πραγματικά μεταβλητές: μεταβλητές σχέσης, ή με άλλα λόγια μεταβλητές των οποίων οι επιτρεπόμενες τιμές είναι τιμές σχέσης (διαφορετικές τιμές σχέσης στους διαφορετικούς χρόνους). Παραδείγματος χάριν, υποθέστε η μεταβλητή σχέσης S αυτήν την περίοδο έχει την τιμή στο σχήμα 1.1, και υποθέστε ότι διαγράφουμε τη πλειάδα για τους προμηθευτές στο Παρίσι:

DELETE S WHERE CITY = 'Paris';

Τότε το αποτέλεσμα φαίνεται στο παρακάτω πίνακα:

S#	SNAME	STATUS	CITY
S1	SMITH	20	LONDON
S4	CLARK	20	LONDON
S5	ADAMS	30	ATHENS

Αυτό που έχει συμβεί εδώ είναι ότι η παλαιά τιμή σχέσης *του* S έχει αντικατασταθεί *στο* σύνολο από μια εξ ολοκλήρου νέα τιμή σχέσης. Φυσικά, η παλαιά τιμή (με πέντε πλειάδες) και η νέα (με τρεις) είναι κάπως παρόμοιες, αλλά βεβαίως είναι διαφορετικές τιμές. Πράγματι, η ΔΙΑΓΡΑΦΗ που παρουσιάστηκε είναι στενογραφία για την ακόλουθη σχεσιακή ανάθεση:

S: = S WHERE NOT (CITY = 'Paris');

Όπως σε όλες τις αναθέσεις, αυτό που συμβαίνει εδώ, είναι ότι:

1. η έκφραση στη δεξιά πλευρά αξιολογείται, και έπειτα
2. το αποτέλεσμα αυτής της αξιολόγησης ορίζεται στη μεταβλητή στην αριστερή πλευρά.

Μια παρατήρηση στην ενημέρωση

Επιστρέφουμε για μια στιγμή στο ζήτημα της ενημέρωσης των πλειάδων, επειδή υπάρχει ένα άλλο θέμα που θίγεται σχετικά με αυτό το ζήτημα: Η σχεσιακή ανάθεση και οι εντολές INSERT, DELETE *και* UPDATE είναι όλες συνολικού επιπέδου τελεστές. Η UPDATE παραδείγματος χάριν, ενημερώνει ένα σύνολο πλειάδων, μιλώντας αόριστα (ακριβέστερα, αντικαθιστά ένα σύνολο πλειάδων στη μεταβλητή σχέση, από ένα άλλο τέτοιο σύνολο πλειάδων). Ανεπίσημα, μιλάμε συχνά για, παραδείγματος χάριν, την ενημέρωση κάποιας μεμονωμένης πλειάδας, αλλά πρέπει σαφώς να γίνει κατανοητό ότι:

(1) τέτοια συζήτηση πραγματικά σημαίνει ότι το σύνολο των πλειάδων που ενημερώνουμε πρέπει να έχει ένα αριθμό στοιχείων συνόλου, και
(2) μερικές φορές ενημερώνοντας ένα σύνολο πλειάδων, ένα είναι συνήθως απίθανο να ενημερωθεί!

Υποθέστε, παραδείγματος χάριν, ότι οι προμηθευτές της μεταβλητής σχέσης S υπόκεινται στον περιορισμό ακεραιότητας ότι οι προμηθευτές S1 *και* S4 βρίσκονται πάντα στην ίδια πόλη. Κατόπιν οποιαδήποτε «μονή- πλειάδα» που ενημερώνεται στο S, εάν προσπαθεί να αλλάξει την πόλη για ακριβώς έναν από αυτούς τους δύο προμηθευτές πρέπει απαραιτήτως να αποτύχει. Άντ' αυτού, και οι δύο πρέπει να ενημερωθούν ταυτόχρονα, όπως εδώ:

```
UPDATE S WHERE S# = S# ('S1') OR S# = S# ('S4')
        {CITY := some value} ;
```

Κατά συνέπεια, αυτό που σημαίνει πραγματικά όταν λέμε οτι "η πλειάδα t ενημερώνεται *σε* t', είναι ότι αντικαθιστούμε τη πλειάδα t με μια άλλη πλειάδα t'.

Κλειδιά

Έστω ότι K είναι ένα υποσύνολο του συνόλου ιδιοτήτων της μεταβλητής σχέσης R. Κατόπιν το K λέγεται ότι είναι πρωτεύων κλειδί (ή απλά κλειδί) *για το* R εάν και μόνο εάν κατέχει και τις δύο από τις ακόλουθες ιδιότητες:

- **μοναδικότητα:** Καμία πιθανή τιμή του R δεν περιέχει δύο ευδιάκριτες πλειάδες με την ίδια τιμή για το K.
- ***μη αναγωγιμότητα:*** Κανένα κατάλληλο υποσύνολο του K δεν έχει την μοναδικότητα.

Στην περίπτωση των προμηθευτών και των αποστολών, τα μόνα τέτοια κλειδιά είναι {S # } για τη σχέση S *και* {S #, P #} για τη σχέση SP. Σημειώστε τις εσωκλειώμενες αγκύλες: Είναι σημαντικό να γίνει κατανοητό ότι τα κλειδιά είναι πάντα σύνολα ιδιοτήτων, κι όχι μόνο ιδιότητες (ακόμα και όταν περιλαμβάνει το σύνολο μόνο μια ιδιότητα, όπως κάνει στην περίπτωση της σχέσης S). Γι' αυτόν τον λόγο, εσωκλείουμε πάντα το σχετικό όνομα ιδιοτήτων (S) στις αγκύλες.

Έστω R1 *και* R2 είναι μεταβλητές σχέσης. Έστω ότι το K είναι πρωτεύων κλειδί για *την* R1. Έστω ακόμα ότι το K είναι ένα υποσύνολο των ιδιοτήτων της R2 το οποίο περιέχει ακριβώς τις ίδιες ιδιότητες με το K. Κατόπιν *το* FK λέγεται ότι είναι ένα ξένο κλειδί εάν και μόνο εάν για όλο το χρόνο, κάθε πλειάδα στην τρέχουσα τιμή *της* R2 έχει μια τιμή *για το* FK που είναι ίση με την τιμή του K σε κάποιες πλειάδες στην τρέχουσα τιμή της R1. Στην περίπτωση των προμηθευτών και των αποστολών, το μόνο ξένο κλειδί είναι το S # στη σχέση SP, η οποία ταιριάζει με *το* πρωτεύων κλειδί της πλειάδας S.

Λειτουργικές εξαρτήσεις

Ας υποθέσουμε ότι τα A *και* B είναι υποσύνολα του συνόλου των ιδιοτήτων της σχέσης R. Τότε η λειτουργική εξάρτηση A->B ισχύει για *το* R εάν και μόνο εάν *σε* κάθε πιθανή τιμή *του* R, όποτε δύο πλειάδες έχουν την ίδια τιμή για το A, έχουν επίσης την ίδια τιμή για το B. Παραδείγματος χάριν, υποθέστε ότι υπήρχε ένας κανόνας για τη βάση δεδομένων προμηθευτής και αποστολές με σκοπό, εάν δύο προμηθευτές βρίσκονται στην ίδια πόλη συγχρόνως, τότε πρέπει να έχουν την ίδια θέση εκείνη την περίοδο. Άρα η λειτουργική εξάρτηση

{CITY} -> {STATUS}

θα ίσχυε για τους προμηθευτές της σχέσης S. Σημεία που εμφανίζονται:

α)εάν το K είναι πρωτεύων κλειδί *της* R *και* το A είναι κάποιο σύνολο ιδιοτήτων *της* R, τότε η λειτουργική εξάρτηση K - > A ισχύει απαραιτήτως για *το* R

β)οι λειτουργικές εξαρτήσεις είναι η βάση της θεωρίας κανονικοποίησης, μέχρι και συμπεριλαμβανομένης της κανονικής μορφής Boyce/Codd normal form

1.4 Integrity constraints (περιορισμοί ακεραιότητας)

Ένας περιορισμός ακεραιότητας, ή περιορισμός για συντομία, μπορεί να θεωρηθεί, ως *μια* Boolean έκφραση (επίσης γνωστή ως λογική, υποθετική, ή truth-valued έκφραση) που απαιτείται να αξιολογηθεί σε αληθή. Ταξινομούμε αυτούς *τους* περιορισμούς στη βάση δεδομένων, σε περιορισμούς σχέσεων, ιδιοτήτων, και περιορισμούς τύπων. *Στην* ουσία:

- ένας περιορισμός βάσεων δεδομένων είναι ένας περιορισμός στις τιμές που μια δεδομένη βάση δεδομένων επιτρέπεται να έχει

  π.χ.

  ```
  CONSTRAINT DBC1 IS_EMPTY ((S JOIN SP)
                                WHERE STATUS <20
                                AND P# = P# ('P6'));
  ```

  Σημαίνει ότι κανένας προμηθευτής με θέση μικρότερη από 20 δεν μπορεί να υποστηρίζει το μέρος P6
- ένας περιορισμός σχέσης είναι ένας περιορισμός στις τιμές που μια δεδομένη σχέση επιτρέπεται να έχει

  π.χ.

  ```
  CONSTRAINT RBC1 IS_EMPTY (S WHERE STATUS<1
                                OR STATUS>100);
  ```

  Σημαίνει οτι οι τιμές θέσης των προμηθευτών πρέπει να είναι αποκλειστικά στη σειρά 1 έως 100
- ένας περιορισμός ιδιοτήτων είναι ένας περιορισμός στις τιμές που μια δεδομένη ιδιότητα επιτρέπεται να έχει

  π.χ.

  ```
  STATUS INTEGER
  ```

- ένας περιορισμός τύπων είναι, ακριβώς, ένας καθορισμός του συνόλου τιμών που αποτελούν έναν δεδομένο τύπο
  π.χ.

```
TYPE S POSSREP {C CHAR
                CONSTRAINT SUBSTR (C, 1, 1) ='S'
                AND LENGTH (C) >=2
                AND LENGTH (C) <=5  };
```

  Αυτός ο καθορισμός τύπων περιορίζει τους αριθμούς προμηθευτών να είναι τέτοιος που να μπορούν να αντιπροσωπευθούν από μια σειρά χαρακτήρων C που αποτελείται από δύο έως πέντε χαρακτήρες, των οποίων ο πρώτος πρέπει να είναι το "S".

1.5 Relational operators (σχεσιακοί τελεστές)

Το σχεσιακό μοντέλο περιλαμβάνει ένα απέραντο σύνολο γενικών τελεστών γνωστό συλλογικά ως σχεσιακή άλγεβρα (οι τελεστές είναι γενικοί επειδή ισχύουν για όλες τις πιθανές σχέσεις). Σε αυτό το τμήμα, καθορίζουμε αυτούς τους τελεστές στους οποίους θα στηριζόμαστε σε μεγάλο ποσοστό στα επόμενα κεφάλαια. Δίνουμε επίσης μερικά παραδείγματα, αλλά μόνο εκεί όπου ότι οι τελεστές είναι εντελώς άγνωστοι.

Καθένας από τους τελεστές που συζητάμε παίρνει είτε μια είτε δύο σχέσεις ως τελεστέους και επιστρέφει μια άλλη σχέση σαν αποτέλεσμα.

Πολύ σημαντικό το γεγονός ότι το αποτέλεσμα είναι πάντα μια άλλη σχέση που αναφέρεται ως (σχεσιακή) ιδιότητα περάτωσης. Είναι αυτή η ιδιότητα η οποία μας επιτρέπει να γράψουμε τις τοποθετημένες σχεσιακές εκφράσεις.

Rename (Μετονομασία)

Έστω ότι A είναι μια σχέση με μια ιδιότητα X και χωρίς την ιδιότητα Υ. Κατόπιν τη μετονομασία

A RENAME X AS Y

παράγει μια σχέση που διαφέρει από την A στο ότι το όνομα των ιδιοτήτων είναι Υ αντί του X.

Union (ένωση), Intersect (τομή), και Minus (διαφορά)

Έστω A και B είναι σχέσεις του ίδιου τύπου σχέσης. Τότε:

- η ένωση αυτών των σχέσεων, A UNION B, είναι μια σχέση του ίδιου τύπου, με το σώμα που αποτελείται από όλες τις πλειάδες t έτσι ώστε το t να εμφανίζεται στο A ή το B ή και στα δύο.
- η τομή αυτών των σχέσεων, A INTERSECT B, είναι μια σχέση του ίδιου τύπου, με το σώμα να αποτελείται από όλες τις πλειάδες t, έτσι ώστε το t να εμφανίζεται και στο A και στο B.
- Η διαφορά μεταξύ αυτών των σχέσεων, A MINUS B (σ' αυτή τη σειρά), είναι μια σχέση του ίδιου τύπου, με το σώμα να αποτελείται από όλες τις πλειάδες t έτσι ώστε το t να εμφανίζεται στο A και όχι στο B.

Να σημειώσουμε ότι η ένωση και η τομή είναι συσχετιζόμενα, πράγμα που επιτρέπει στις περιττές παρενθέσεις να παραλειφθούν από μια συνεχή ακολουθία ενώσεων ή τομών. Παραδείγματος χάριν οι εκφράσεις

A UNION (B UNION C)

και

(A UNION B) UNION C

μπορούν και οι δύο αναμφίβολα να συντομευτούν ως εξής:

A UNION B UNION C

Έστω ότι S είναι ένα σύνολο σχέσεων όλες του ίδιου τύπου σχέσης, RT. Τότε:
(1)Εάν το S περιέχει μόνο μια σχέση r, τότε η ένωση και η τομή όλων των σχέσεων *στο* S ορίζονται και οι δύο να είναι r
(2)Εάν το S δεν περιέχει καμία σχέση, τότε:

- ✓ η ένωση όλων των σχέσεων στο S ορίζεται να είναι η κενή σχέση του τύπου RT
- ✓ *η* τομή όλων των σχέσεων στο S καθορίζεται για να είναι η "καθολική" σχέση του τύπου RT - δηλαδή εκείνη η μοναδική σχέση του τύπου RT που περιέχει όλα τα πιθανές πλειάδες του τύπου TUPLE{H}, όπου {H} είναι η επικεφαλίδα του τύπου σχέσης RT

Restrict (Περιορισμός)
Δώστε στη σχέση *A* τις ιδιότητες X και Y (ενδεχομένως κι άλλες), κι έστω ότι θ είναι ένας τελεστής ("=", ">", "<"). Έτσι, η έκφραση X θ Y είναι καλοσχηματισμένη και, λαμβάνοντας υπόψη τις ιδιαίτερες τιμές για το X και το Y, αξιολογείται σε μια αληθή τιμή (true or false). Κατόπιν ο περιορισμός θ της σχέσης *A* στις ιδιότητες X και Y (σε εκείνη την σειρά)

A WHERE X *θ* Y

είναι μια σχέση με τον ίδιο τίτλο με το A και με το σώμα που αποτελείται από όλες τις πλειάδες του A, έτσι ώστε η έκφραση X θ Y να είναι αληθινή για την εν λόγω πλειάδα.

Project (σχεδιασμός)
Έστω ότι η σχέση A έχει τις ιδιότητες X,Y,....,Z. Τότε ο σχεδιασμός της σχέσης A στο X,Y,.....Z

A{ X,Y,.....Z}

είναι μια σχέση με:

i. ένα τίτλο που προήλθε από τον τίτλο του A αφαιρώντας όλες τις ιδιότητες που δεν αναφέρθηκαν στο σύνολο { X, Y ...., Z} , και

ii. ένα σώμα που αποτελείται απ' όλες τις πλειάδες { X χ, Υ y,..., Z z} έτσι ώστε μια πλειάδα να εμφανίζεται στο A με X την τιμή χ, Υ την τιμή y..., *και* Z την τιμή z

Join (συνένωση)
Δώστε στις σχέσεις A και B τις ιδιότητες
X1, X2..., Xm, Y1, Y2..., Yn

και

Y1, Y2..., Yn, Z1, Z2..., Zp

αντίστοιχα. Δηλαδή οι Y ιδιότητες Y1, Y2..., Yn είναι μόνο κοινές για τις δύο σχέσεις, οι X ιδιότητες X1, X2,..., Xm είναι οι άλλες ιδιότητες του A, και οι Z ιδιότητες Z1, Z2..., Zp είναι οι άλλες ιδιότητες του B. Παρατηρήστε ότι:

- ✓ Μπορούμε να υποθέσουμε (χάρη στην ιδιότητα του τελεστή RENAME) ότι καμία ιδιότητα Xi (i = 1,2,..., m) *δεν* έχει το ίδιο όνομα με οποιεσδήποτε ιδιότητες Zj (j = 1,2,... ,p)
- ✓ κάθε ιδιότητα Yk (K = 1,2,..., n) έχει τον ίδιο τύπο και στο A και *στο* B).

Τώρα εξετάστε { X1, X2..., Xm} , {Y1, Y2..., Yn} , *και* {Z1, Z2..., Zp} *ως τρεις* σύνθετες ιδιότητες X, Y, και Z αντίστοιχα. Κατόπιν η συνένωση του A και *του* B

A JOIN B

είναι μια σχέση με τον τίτλο {X, Y, Z} και το σώμα να αποτελείται από όλες τις πλειάδες {X x, Y y, Z z}, έτσι ώστε μια πλειάδα να εμφανίζεται στο A με X την τιμή x , *στο* Y την τιμή y και μια πλειάδα εμφανίζεται στο B *με* Y την τιμή y *και* Z την τιμή z.

Όπως στην ένωση και τη τομή, η συνένωση είναι συσχετιζόμενη, και επιτρέπει στις περιττές παρενθέσεις να παραλειφθούν από μια συνεχή ακολουθία συνενώσεων. Παραδείγματος χάριν, οι εκφράσεις

A JOIN (B JOIN C) *και* (A JOIN B) JOIN C

μπορούν και οι δύο να συντομευτούν σε

A JOIN B JOIN C

Έστω ότι S είναι ένα σύνολο σχέσεων. Κατόπιν:

- εάν *το* S περιέχει μόνο μια σχέση r, τότε η συνένωση όλων των σχέσεων στο S ορίζεται να είναι απλά r.
- εάν *το* S δεν περιέχει καμία σχέση, τότε η συνένωση όλων των σχέσεων στο S ορίζεται να είναι TABLE_DEE

Extend (επέκταση)
Έστω A μια σχέση. Κατόπιν η επέκταση

EXTEND A ADD exp AS Z

είναι μια σχέση με:

- *με* τίτλο που αποτελείται από τον τίτλο του A επεκτεινόμενος με τις ιδιότητες του Z, και
- ένα σώμα που αποτελείται από όλες τις πλειάδες t, έτσι ώστε *το* t *να* είναι πλειάδα του A επεκτεινόμενο με μια τιμή για την ιδιότητα Z, που υπολογίζεται με την αξιολόγηση exp *σ'* αυτή τη πλειάδα του A

Η σχέση A δεν πρέπει να έχει την ιδιότητα Z και το exp δεν πρέπει να αναφερθεί στο Z.
Εδώ είναι ένα απλό παράδειγμα EXTEND:

EXTEND S ADD (STATUS * 3) AS TRIPLE

Το αποτέλεσμα φαίνεται παρακάτω(εφαρμόζεται στο αρχικό παράδειγμα 1.1):

S#	SNAME	STATUS	CITY	TRIPLE
S1	Smith	20	London	60
S2	Jones	10	Paris	30
S3	Blake	30	Paris	90
S4	Clark	20	London	60
S5	Adams	30	Athens	90

Summarize (σύνοψη)
Έστω ότι A *και* B είναι δύο σχέσεις. Κατόπιν η παρουσίαση της πληροφορίας
SUMMARIZE A PER B ADD summary AS Z

είναι μια σχέση που καθορίζεται ως εξής:

- πρώτα, το B πρέπει να είναι του ίδιου τύπου με κάποια προβολή του A, δηλαδή κάθε ιδιότητα του B πρέπει να είναι ιδιότητα του A. Έστω οι ιδιότητες αυτής *της* προβολής ότι είναι A1, A2....,An
- *ο* τίτλος του αποτελέσματος αποτελείται από τον τίτλο *του* B που επεκτείνεται με τις ιδιότητες της Z

το σώμα αυτού του αποτελέσματος αποτελείται απ' όλες τις πλειάδες t, έτσι ώστε *το* t *να* είναι πλειάδα *του* B που επεκτείνεται με μία τιμή για την ιδιότητα Z. Αυτή τιμή Z υπολογίζεται με την αξιολόγηση του αθροίσματος όλων των πλειάδων του A που έχουν την ίδια τιμή για τις ιδιότητες {A1, A2,..., An).

Εδώ είναι ένα απλό παράδειγμα:

SUMMARIZE SP PER S {S#} ADD COUNT AS P_COUNT

Το αποτέλεσμα παρουσιάζεται στο παρακάτω σχήμα:

S#	P_COUNT
S1	6
S2	2
S3	1
S4	3
S5	0

Αν προσπαθούσαμε να το κάνουμε με τη γνωστή μας SQL, το αποτέλεσμα αυτό θα έβγαινε απ' τον παρακάτω κώδικα:

```
SELECT S#, COUNT (*) AS P_COUNT
FROM SP
GROUP BY S#
```

Group and ungroup (ομαδοποίησε και χώρισε)

Εξετάστε τους ακόλουθους δύο τύπους σχέσης:

RELATION (S# S#, P# P#)
RELATION {S# S#, P#_REL RELATION {P# P#}}

Θα αναφερόμαστε σε αυτούς τους δύο τύπους ως RT1 *και* RT2, αντίστοιχα.
Τώρα έστω ότι SP *και* SP' είναι σχέσεις των τύπων RT1 *και* RT2, αντίστοιχα (οι σχέσεις στα σχήματα 1.1 και 1.3, μπορούν αντίστοιχα να ληφθούν ως παραδείγματα των πιθανών τιμών για αυτές τις δύο σχέσεις). Κατόπιν η έκφραση

SP GROUP {P#} AS P#_REL

παράγει μια σχέση που καθορίζεται ως εξής:

1. πρώτα, ο τίτλος μοιάζει με αυτό:

   {S# S#, P#_REL RELATION {P# P#}}

   με άλλα λόγια, ο τίτλος περιέχει μια σχέση P#_REL
2. δεύτερον, το σώμα περιέχει ακριβώς μια πλειάδα για κάθε ευδιάκριτη τιμή S# την τιμή στο SP

Πηγαίνοντας στην UNGROUP τώρα, η έκφραση

SP' UNGROUP P#_REL

παράγει μια σχέση που ορίζεται ακολούθως:

1) ο τίτλος μοιάζει με αυτό:

   {S# S#, P#,P#}

   Με άλλα λόγια ο τίτλος περιλαμβάνει μια ιδιότητα P#

2) το σώμα περιλαμβάνει ακριβώς μια πλειάδα για κάθε συνδυασμό μιας πλειάδας *στο* SP', *και* μια πλειάδα στη τιμή P#_REL

1.6 The relational model (*το* σχεσιακό μοντέλο)

Το σχεσιακό μοντέλο αποτελείται από τα ακόλουθα πέντε συστατικά:

1.Μια απέραντη συλλογή κλιμακωτών τύπων (που περιλαμβάνουν ειδικότερα τον τύπο Boolean ή τιμή αληθείας)
2.Μια γεννήτρια τύπων σχέσης και μια προοριζόμενη ερμηνεία για τις σχέσεις των τύπων που παράγονται με αυτόν τον τρόπο
3.Εγκαταστάσεις για τον καθορισμό των μεταβλητών σχέσης τέτοιων παραγμένων τύπων σχέσης
4. Ένας σχεσιακός τελεστής ανάθεσης για την ανάθεση των τιμών σχέσης σε τέτοιες μεταβλητές σχέσης
5.Μια απέραντη συλλογή των γενικών σχεσιακών τελεστών για τον προσδιορισμό των τιμών σχέσης από άλλες τιμές σχέσης

Σ' αυτό το σημείο θα προσθέσουμε κάποια σχόλια γι' αυτά τα πέντε συστατικά.

(1)Οι κλιμακωτοί τύποι μπορούν να καθοριστούν απ' το σύστημα ή να καθοριστούν από το χρήστη. Θα πρέπει να είναι διαθέσιμα τα κατάλληλα μέσα για τους χρήστες για να καθορίσουν τους κλιμακωτούς τύπους τους (αυτή η απαίτηση υπονοείται, από το γεγονός ότι το σύνολο τέτοιων τύπων είναι απέραντο). Πρέπει επομένως να είναι διαθέσιμα τα μέσα για τους χρήστες για να καθορίσουν τους κλιμακωτούς τελεστές τους, δεδομένου ότι οι τύποι χωρίς τελεστές είναι άχρηστοι.
   Ο μόνος κλιμακωτός τύπος που απαιτείται να οριστεί απ' το σύστημα είναι ο τύπος Boolean

(2)Ο δημιουργός τύπων σχέσης επιτρέπει στους χρήστες να διευκρινίσουν οποιοδήποτε απαραίτητο τύπο σχέσης

(3)Οι διαδικασίες για τον καθορισμό των μεταβλητών σχέσης πρέπει να είναι διαθέσιμες. Οι μεταβλητές σχέσης είναι οι μόνες μεταβλητές που επιτρέπονται σε μια σχεσιακή βάση δεδομένων

(4)Οι μεταβλητές είναι αναθεωρήσιμες εξ ορισμού. Κάθε είδος μεταβλητής υπόκειται *στην* ανάθεση, και οι μεταβλητές σχέσης δεν είναι καμία εξαίρεση

(5)Οι γενικοί σχεσιακοί τελεστές είναι οι τελεστές που αποτελούν τη σχεσιακή άλγεβρα, και είναι επομένως ενσωματωμένοι.

1.7 ΤΟ TUTORIAL D

Τα παραδείγματα κωδικοποίησης σε αυτό το *βιβλίο* εκφράζονται *σε μια* γλώσσα αποκαλούμενη TUTORIAL D. Σε αυτό το κεφάλαιο προσφέρουμε μια συνοπτική περιγραφή των σημαντικότερων χαρακτηριστικών γνωρισμάτων *της* γλώσσας. Το σώμα *που* ακολουθεί αποτελείται από μια γραμματική BNF *για την* έκδοση *του* TUTORIAL D που θα χρησιμοποιούμε στα παραδείγματά *μας*.

Έστω ότι < xyz > δηλώνει μια αυθαίρετη συντακτική κατηγορία (δηλ., οτιδήποτε εμφανίζεται *στην* αριστερή πλευρά κάποιου κανόνα παραγωγής BNF):

- η έκφραση < xyz list > δείχνει μια ακολουθία μηδενός ή *περισσότερων* < xyz>s στην οποία κάθε ένα < xyz > είναι χωρισμένο από το επόμενο (εάν υπάρχει επόμενο) από ένα τουλάχιστον διάστημα.

- η έκφραση < xyz commalist > δείχνει μια ακολουθία μηδενός ή περισσότερων < xyz>s στο οποίο κάθε ένα < xyz > είναι χωρισμένο από το επόμενο (εάν υπάρχει επόμενο) από ένα κόμμα, καθώς επίσης και, προαιρετικά, ένα ή περισσότερα διαστήματα είτε πριν είτε μετά από το κόμμα (ή τα δύο).

- η έκφραση < ne xyz list > (όπου το ne αντιπροσωπεύει "nonempty" δηλαδή η λίστα δεν είναι άδεια) δείχνει μία < xyz list > που περιέχει τουλάχιστον ένα < xyz >.

- η έκφραση < ne xyz commalist > ορίζεται ανάλογα

Μερικές πιο προκαταρκτικές παρατηρήσεις:

- **Πρώτον,** όλες οι συντακτικές κατηγορίες της μορφής <... name > καθορίζονται να είναι < identifier>s (αναγνωριστές).

- **Δεύτερον,** μερικοί από τους κανόνες παραγωγής BNF συνοδεύονται από μια πεζή εξήγηση ορισμένων πρόσθετων κανόνων σύνταξης ή της αντίστοιχης σημασιολογίας ή και τα δύο-αλλά μόνο σε περιπτώσεις όπου τέτοια περαιτέρω εξήγηση φαίνεται απαραίτητη.

- **Τρίτον,** τα σύμβολα αγκύλης "{" και "}" στη γραμματική χρησιμοποιούνται για να εσωκλείσουν τις λίστες των στοιχείων, όταν η εν λόγω λίστα προορίζεται να δείξει ένα σύνολο κάποιου είδους (που υπονοεί ειδικότερα ότι (1) η σειρά στην οποία τα στοιχεία εμφανίζονται μέσα σ' αυτή τη λίστα είναι ασήμαντη και περαιτέρω ότι (2) κανένα στοιχείο δεν εμφανίζεται μέσα σ' αυτή τη λίστα περισσότερο από μία φορά).

1.8 scalar type definitions (Ορισμοί κλιμακωτών τύπων)

<scalar type def>
::=TYPE <scalar type name> <ne possrep def list>;

<possrep def>

::= POSSREP [<possrep name>]

{<ne possrep component def commalist>

[<possrep constraint del>]}

Το < Possrep name > μπορεί να παραλειφθεί εάν και μόνο εάν *το* < possrep def > είναι το μοναδικό μέσα στο εφαρμόσιμο < scalar type def >.

<possrep component def>

::=< possrep component name> <type>

<possrep constraint def>

::=constraint <bool exp>

Το < bool exp > είναι οποιαδήποτε έκφραση *που* παράγει *μια* τιμή αλήθειας (truth value). Στο πλαίσιο εντός του < possrep constraint def> *μια* < bool exp > δεν επιτρέπεται να αναφέρει μεταβλητές, αλλά το < possrep component name>s από τον περιορισμό < possrep def > μπορεί να χρησιμοποιηθεί για να δείξει τα αντίστοιχα συστατικά της πιθανής εφαρμόσιμης αντιπροσώπευσης μιας αυθαίρετης τιμής του εν λόγω κλιμακωτού τύπου.
Σημείωση: αφού οι τιμές αλήθειας είναι κλιμακωτές τιμές, τότε μία < bool exp > είναι *μια* ειδική περίπτωση μίας < scalar exp >.

<relation comp>
::= <relation exp> <relation comp op> <relation exp>

Μία <relation comp > ("σύγκριση σχέσης") είναι μια ειδική περίπτωση ενός < bool exp >. Οι σχέσεις που δείχνονται από τα δύο < relation exp>s πρέπει να είναι του ίδιου τύπου.

<relation comp op>
::= =| □ | □ | □ | □ | □

Σημείωση: Τα σύμβολα "□" *και* "□" δείχνουν "το υποσύνολο" και το "κατάλληλο υποσύνολο". Αντίστοιχα, τα σύμβολα "□" *και* "□" δείχνουν "το υπερσύνολο" *και* "το κατάλληλο υπερσύνολο," αντίστοιχα.

1.9 relational definitions (Σχεσιακοί ορισμοί)

<relvar def>
::=VAR <relvar name>
RELATION {<attribute commalist>}
<ne candidate key def list>
<foreign key def list>;

Η σχέση που καθορίζεται από ένα < relvar def > έχει τύπο σχέσης (relation type) RELATION (A1 T1, A2 T2..., An Tn), όπου A1T1, A2 T2..., An Tn είναι ιδιότητες διευκρινισμένες στην <attribute commalist> μέσα στο < relvar def >.

Σημείωση: *το* TUTORIAL D σκόπιμα δεν παρέχει ένα σαφές "define relation type" τελεστή. Αντ' αυτού, οι τύποι σχέσεων μπορούν απλά να χρησιμοποιηθούν "εὐθύγραμμα," ὡς τμήμα ενός < relvar def >.

<attribute>
::= <attribute name> <attribute type>

Το <attribute type>δηλαδή ο τύπος ιδιοτήτων μπορεί να είναι οποιοσδήποτε τύπος που καθορίζεται από το σύστημα ή από το χρήστη.

<candidate key def>
::= KEY { <attribute name commalist>}

Χρησιμοποιούμε το κλειδί KEY για να σημάνουμε ένα κλειδί υποψηφίων, για λόγους απλότητας. Κάθε σχέση απαιτείται να έχει τουλάχιστον ένα κλειδί υποψηφίων.

<foreign key def>
::= FOREIGN KEY { <foreign key component commalist> }
REFERENCES <relvar name>

Τα <foreign key component>s αφότου έχει γίνει οποιαδήποτε απαραίτητη μετονομασία ιδιοτήτων, διευκρινίζει ότι οι ιδιότητες του ξένου κλειδιού καλούνται Al, A2,..., An. Κατόπιν η σχέση που προσδιορίζεται από το < relvar name > πρέπει να περιλάβει ένα σύνολο ιδιοτήτων αποκαλούμενων A1, A2...,An και αυτό το σύνολο των ιδιοτήτων πρέπει να έχει οριστεί ως ένα κλειδί υποψηφίων γι' αυτή τη σχέση.

<foreign key component>
::= <attribute name>
I RENAME (<ne renaming commalist>)

Το μεμονωμένο <renaming>*s* εκτελείται *στη* σειρά όπως γράφεται. Οι παρενθέσεις μπορούν να παραλειφθούν εάν η λίστα περιέχει ακριβώς ένα <renaming>.

<renaming>
::=< attribute name> AS <attribute name>

1.10 Relational expressions (Σχεσιακές εκφράσεις)

<Relation exp>
::= <relvar name>
I <relation selector inv>
I <other relation op inv>
I <with exp>
I <introduced name>
I (<relation exp>)

<relation selector inv>
::= RELATION [{<attribute commalist> }]
{<tuple selector inv commalist>}

Στο TUTORIAL D μια επίκληση επιλογέων σχέσης για τον τύπο σχέσης RT αποτελείται συνήθως από τη λέξη κλειδί RELATION, που ακολουθείται από μια λίστα πλειάδων των επικλήσεων επιλογέων που εσωκλείονται στις αγκύλες. Κάθε μια απ' αυτές τις πλειάδες επικλήσεις επιλογέων πρέπει να επιλέξει μια πλειάδα του τύπου TUPLE {H}, όπου {H} είναι ο τίτλος του τύπου σχέσης RT.

<project>
::= <relation exp>
{ [ALL BUT] <attribute name commalist> }

Το <relation exp> δεν πρέπει να είναι <nonproject>.

<nonproject>
::= <rename>

I <where>
I
I <union>
I <intersect>
I <minus>
I <join>
I <extend>
I <summarize>
I <group>
I <ungroup>

<rename>

::= <relation exp> RENAME (<ne renaming commalist>)

Η< relation exp > *δεν πρέπει να* είναι < nonproject >. Το μεμονωμένο < renaming>*s* εκτελείται *στη* σειρά όπως γράφεται. Οι παρενθέσεις μπορούν να παραλειφθούν εάν η λίστα περιέχει ακριβώς ένα <renaming >.

<where>
::= <relation exp> WHERE <bool exp>

Το<relation exp > *δεν πρέπει να* είναι < nonproject >. Το < bool exp > είναι οποιαδήποτε έκφραση που παράγει μια τιμή αλήθειας, τέτοιες εκφράσεις επιτρέπονται για να περιλάβουν μια αναφορά σε μια ιδιότητα της σχέσης που δείχνεται από το < relation exp > (*π.χ.*, S WHERE CITY = "LONDON")

<union>

::= <relation exp> UNION <relation exp>

Το <relation exp>s δεν πρέπει να είναι <nonproject>s, εκτός από το ότι το ένα ή και τα δύο μπορεί να είναι μια άλλη <union>.

<Intersect>
::= <relation exp> INTERSECT <relation exp>

Το <relation exp>s δεν πρέπει να είναι < nonproject>s, εκτός από το ότι το ένα ή και τα δύο μπορεί να είναι ένα άλλο < Intersect >.

<minus>
::=<relation exp> MINUS <relation exp>

Το <relation exp>s *δεν πρέπει να* είναι <nonproject>s.

<join>
::= <relation exp> JOIN <relation exp>

Το <relation exp>s δεν πρέπει να είναι < nonproject>s, εκτός από το ότι το ένα ή και τα δύο μπορεί να είναι άλλο < join >.

<extend>
::= EXTEND <relation exp>
ADD (<ne extend add commalist>)

Το <relation exp> *δεν πρέπει να* είναι ένα < nonproject >. Το μεμονωμένο < extend add> εκτελείται στη σειρά όπως γράφεται. Οι παρενθέσεις μπορούν να παραλειφθούν εάν το commalist περιέχει ακριβώς ένα < extend add >.

<summarize>

::= SUMMARIZE <relation exp> PER <relation exp

ADD (<ne summarize add commalist>)

Το <relation exp>s *δεν* πρέπει *να* είναι < nonproject>s. Το μεμονωμένο < summarize add> εκτελείται στη σειρά όπως γράφεται. Οι παρενθέσεις μπορούν να παραλειφθούν εάν η λίστα περιέχει ακριβώς ένα <summarize add>.

<group>

::= <relation exp>
GROUP {[ALL BUT] <attribute name commalist>)
AS <attribute name>

Το <relation exp> *δεν* πρέπει *να* είναι <nonproject>.

<ungroup>

::= <relation exp> UNGROUP <attribute name>

Το <relation exp> δεν πρέπει να είναι <nonproject>. Οι ιδιότητες *που* προσδιορίζονται από *το* <attribute name> πρέπει *να* είναι *του* ίδιου τύπου σχέσης.

<with exp>

::= WITH <ne name intro commalist> : <exp>

ένα < with exp > είναι μια σχεσιακή έκφραση, μια έκφραση πλειάδας, ή μια κλιμακωτή έκφραση όπως το < exp >. Μετά από την άνω και κάτω τελεία είναι μια σχεσιακή έκφραση, μια έκφραση πλειάδας, ή μια κλιμακωτή έκφραση διαδοχικά. Σε όλες τις περιπτώσεις, το μεμονωμένο <name intro> εκτελείται στη σειρά όπως γράφεται, και *η* σημασιολογία του < with exp > ορίζεται να είναι η ίδια με αυτές όπως μιας έκδοσης < exp > στην οποία κάθε περιστατικό κάθε εισαχθέντος ονόματος αντικαθίσταται από το κείμενο της αντίστοιχης έκφρασης.

<name intro>

::= <exp> AS <introduced name>

Το<introduced name > μπορεί να χρησιμοποιηθεί μέσα στον περιορισμό <with exp > οπουδήποτε ένα < exp > (εσωκλειόμενο σε παρένθεση εάν είναι απαραίτητο) θα επιτρεπόταν.

1.11 relational assignments (Σχεσιακές αναθέσεις)

<relation assignment>
::= <ne relation assign commalist>;

Η<relation assignment> *στο* tutorial D είναι μια πολλαπλάσια ανάθεση. Το μεμονωμένο < relation assign> εκτελείται στη σειρά όπως γράφεται.

<relation assign>

::= <relvar name> := <relation exp>

I <relation insert>

I <relation delete>

I <relation update>

Σύμφωνα με τα παραπάνω οι σχέσεις δηλώνονται από το < relvar name > *και το* < relation exp> πρέπει να είναι του ίδιου τύπου.

<relation insert>
::= INSERT <relvar name> <relation exp>

Το μεταβλητή σχέσης δηλώνεται από το < relvar name > και η σχέση δηλώνεται από το < relation exp > *που* πρέπει να είναι του ίδιου τύπου.

<relation delete>
::= DELETE <relvar name> [WHERE <bool exp>]

< Το bool exp > επιτρέπεται να περιλάβει μια αναφορά σε μια ιδιότητα της σχέσης που δηλώνεται από το < relvar name >

<relation update>
::= UPDATE <relvar name> [WHERE <bool exp>]
{<ne attribute update Commalist>}

όλες *οι* μεμονωμένες < attribute update>s εκτελούνται *στη* σειρά όπως γράφονται. Οι αγκύλες μπορούν να παραλειφθούν εάν η λίστα περιέχει ακριβώς ένα < attribute update >.

<attribute update>
::= <attribute name> := <exp>

Οι διευκρινισμένες ιδιότητες και το < exp > πρέπει να είναι του ίδιου τύπου , επίσης το< exp > επιτρέπεται να περιλάβει μια αναφορά σε μια ιδιότητα της σχέσης.

1.12 Contraint définitions (Ορισμοί περιορισμού)

<Constraint def>
::= CONSTRAINT <constraint name> <bool exp> ;

< Το bool exp > δεν πρέπει να αναφέρει οποιεσδήποτε μεταβλητές εκτός από τις σχέσεις.
Σημείωση: Μια ειδική μορφή του < bool exp >, IS_EMPTY (< relation exp>), αξιολογείται σε αληθινή εάν το σώμα της σχέσης που δηλώνεται από το < relation exp > είναι κενό (δηλ., δεν περιέχει καμία πλειάδα) και ψευδή στην αντίθετη περίπτωση.

ΚΕΦΑΛΑΙΟ 2: ΧΡΟΝΟΣ ΚΑΙ Η ΒΑΣΗ ΔΕΔΟΜΈΝΩΝ

2.1 Εισαγωγή

Μια χρονική βάση δεδομένων μπορεί να θεωρηθεί, ως βάση δεδομένων που περιέχει τα ιστορικά στοιχεία αντί ή εκτός από τα τρέχοντα στοιχεία. Τέτοιες βάσεις δεδομένων ήταν υπό ενεργό έρευνα από την αρχή της δεκαετίας του '80 (ενδεχομένως και νωρίτερα). Μερικές από αυτές τις έρευνες υποστηρίζουν την θέση ότι τα στοιχεία σε μια τέτοια βάση δεδομένων, μόλις παρεμβληθούν, δεν πρέπει ποτέ να διαγραφούν ή να αλλάξουν από καμιά άποψη, οπότε σ'αυτήν την περίπτωση η βάση δεδομένων μπορεί να θεωρηθεί, ως περιορισμός των ιστορικών στοιχείων.

Οι συμβατικές βάσεις δεδομένων, σε αντίθεση με τις χρονικές, είναι τυπικά στο άλλο άκρο. Μια τέτοια βάση δεδομένων περιέχει μόνο τρέχοντα στοιχεία, τα οποία μπορούν να αλλάζουν ή να διαγράφονται μόλις οι προτάσεις που αντιπροσωπεύονται από αυτά *το* στοιχεία παύουν να είναι αυτές που αξιολογούνται σε αληθινές. Θα ονομάζουμε μια τέτοια βάση δεδομένων μη-χρονική, προκειμένου να υπογραμμίσουμε το γεγονός ότι περιέχει μόνο τρέχοντα στοιχεία (οι μη-χρονικές βάσεις δεδομένων καλούνται μερικές φορές βάσεις δεδομένων στιγμιότυπων).

Τα ιδιαίτερα χαρακτηριστικά γνωρίσματα μιας χρονικής βάσης δεδομένων είναι αρκετά, και προφανώς ο ίδιος ο χρόνος. Η έρευνα των χρονικών βάσεων δεδομένων έχει περιλάβει ένα ορισμένο ποσό έρευνας σχετικά με τη φύση του χρόνου. Εδώ είναι μερικές από τις ερωτήσεις που έχουν ερευνηθεί:

- Ο χρόνος έχει μια αρχή ή ένα τέλος;
- Είναι ο χρόνος μια συνέχεια ή διαιρείται σε ιδιαίτερα τμήματα;
- Ποιος είναι ο καλύτερος τρόπος να χαρακτηριστεί η σημαντική έννοια "τώρα" (μερικές φορές γνωστή ως "κινούμενο σημείο τώρα");

Εάν τα στοιχεία γενικά μπορούν να θεωρηθούν ως κωδικοποιημένη αντιπροσώπευση των προτάσεων, τότε τα χρονικά στοιχεία ειδικότερα μπορούν να θεωρηθούν ως κωδικοποιημένη αντιπροσώπευση σφραγισμένων χρονικά προτάσεων, που σημαίνει τις προτάσεις που περιλαμβάνουν ένα ή περισσότερα επιχειρήματα κάποιου χρονικού τύπου. Σε μια χρονική βάση δεδομένων, κάθε πρόταση είναι χρόνος που σφραγίζεται υπό την προηγούμενη έννοια. Επομένως μπορούμε να καθορίσουμε μια χρονική σχέση στην οποία κάθε πλειάδα περιλαμβάνει τουλάχιστον ένα χρονικά σφραγισμένο τύπο. Περαιτέρω μπορούμε να καθορίσουμε μια χρονική σχέση της οποίας ο τίτλος είναι κάποιας χρονικής σχέσης. Τέλος, να ορίσουμε μια χρονική βάση δεδομένων στην οποία όλες οι σχέσεις είναι χρονικές.

Ακόμη όμως κι αν οι αρχικές σχέσεις στη βάση δεδομένων είναι όλες χρονικές, πολλές σχέσεις που μπορούν να προέλθουν από αυτήν την βάση δεδομένων δεν θα είναι χρονικές. Παραδείγματος χάριν, η απάντηση στην ερώτηση "πάρε τα ονόματα όλων των υπαλλήλων που έχουμε προσλάβει", μπορεί να περιέχεται σε κάποια χρονική βάση δεδομένων, αλλά το αποτέλεσμα δεν είναι το ίδιο σε μια χρονική σχέση. Και θα ήταν ένα παράξενο DBMS (πρόγραμμα διαχείρισης βάσεων δεδομένων) στην πραγματικότητα, το οποίο θα μας επέτρεπε να περιλαμβάνουμε αποτελέσματα τα οποία δεν μπορούν από μόνα τους να κρατηθούν σε μια βάση δεδομένων.

2.2 Χρονικά σφραγισμένες προτάσεις(timestamped propositions)

Είμαστε τώρα σε θέση να αρχίσουμε την έρευνά μας με μερικά από τα ζητήματα που περιβάλλουν τις χρονικές βάσεις δεδομένων. Αρχίζουμε με το να παρουσιάσουμε τον τρόπο με τον οποίο οι άνθρωποι εκφράζουν τις χρονικά σφραγισμένες προτάσεις στη φυσική γλώσσα. Εδώ είναι τρία παραδείγματα (ονομαζόμενα T1, T2, *και* T3):

T1: Ο προμηθευτής S1 διορίστηκε *και* είναι τοποθετημένος κάτω από σύμβαση την 1η Ιουλίου 1999.
T2: Ο προμηθευτής S1 ήταν στο πλαίσιο της σύμβασης από την 1η Ιουλίου 1999.
T3: Ο προμηθευτής S1 ήταν στο πλαίσιο της σύμβασης κατά τη διάρκεια του διαστήματος από την 1η Ιουλίου 1999, μέχρι την παρούσα ημέρα.

Κάθε μια από αυτές τις τρεις προτάσεις αντιπροσωπεύει μια πιθανή ερμηνεία για μια πλειάδα που μοιάζει με:

S#	FROM
S1	July 1st, 1999

Όπως μπορείτε να δείτε, η πλειάδα έχει δύο ιδιότητες, S # *και* FROM, με τιμές τον προμηθευτή με αριθμό S1 και τη χρονικά σφραγισμένη πρόταση 1η Ιουλίου 1999. Επιπλέον, οποιεσδήποτε από τις τρεις ερμηνείες μπορεί να είναι κατάλληλες για αυτή την πλειάδα εάν εμφανίζονται σε κάποια βάση δεδομένων που αντιπροσωπεύει την τρέχουσα παρούσα κατάσταση σε κάποια επιχείρηση. Οι προθέσεις **on, since** *και* **during,** χαρακτηρίζουν τις τρεις ερμηνείες.

Αν και έχουμε αναφέρει τις τρεις πιθανές ερμηνείες, μπορεί να υποστηριχτεί ότι *οι* προτάσεις T1, T2, *και* T3 λένε το ίδιο πράγμα με ελαφρώς διαφορετικό τρόπο. Πράγματι, θεωρούμε ότι *το* T2 *και* T3 είναι ισοδύναμα, αλλά όχι *το* T1 *και* T2 (ή *το* T1 *και* T3).

- ✓ Το T1 υπονοεί ότι ο προμηθευτής S1 δεν ήταν στο πλαίσιο της σύμβασης την αμέσως προηγούμενη ημέρα της καθορισμένης ημερομηνίας διορισμού. Το T2 ούτε δηλώνει αυτό το γεγονός ούτε το υπονοεί.

- ✓ Υποθέστε ότι σήμερα ("the present day") είναι 1η Μαΐου 2000. Τότε *το* T2 δηλώνει ότι ο προμηθευτής S1 ήταν στο πλαίσιο της σύμβασης κάθε ημέρα από

την 1η Ιουλίου 1999, έως την 1η Μαΐου 2000. Το T1 ούτε δηλώνει αυτό *το* γεγονός, ούτε το υπονοεί.

Κατά συνέπεια, κανένα από τα T1 *και* T2 υπονοεί το άλλο, και επομένως δεν είναι ισοδύναμα.

Οι πλειάδες στις συμβατικές βάσεις δεδομένων συχνά περιλαμβάνουν τα πράγματα όπως την ημερομηνία του διορισμού, και τις προτάσεις όπως *το* T2 (ή T3). Εάν τέτοια συμβαίνουν εδώ (δηλ. .. εάν *το* T1 προορίζεται να είναι ισοδύναμο με *το* T2 *και* T3 τελικά), τότε η παρούσα διατύπωση δεν είναι αρκετά επαρκής για τον στόχο μας. Μπορούμε να τη βελτιώσουμε με να διαμορφώσουμε εκ νέου την ερμηνεία:

- T1: Ο προμηθευτής S1 *διορίστηκε* πρόσφατα στις 1η Ιουλίου *το* 1999, *και η* σύμβαση δεν έχει τελειώσει.

Εάν αυτή η έκδοση του T1 είναι πραγματικά αυτό που οι υποτιθέμενες 2 πλειάδες σημαίνουν, τότε το T2 με την παρούσα μορφή του δεν είναι επαρκές. Πρέπει να διαμορφωθεί εκ νέου έτσι:

- T2: Ο προμηθευτής S1 δεν ήταν στο πλαίσιο της σύμβασης στις 30 Ιουνίου 1999, αλλά ήταν από την 1η Ιουλίου του 1999.

Φυσικά, και το T3 χρειάζεται μια ανάλογη διευκρίνιση:

- T3: Ο προμηθευτής S1 δεν ήταν στο πλαίσιο της σύμβασης στις 30 Ιουνίου, το 1999, αλλά ήταν έτσι κατά τη διάρκεια του διαστήματος από την 1η Ιουλίου 1999, μέχρι την παρούσα ημέρα.

Παρατηρήστε τώρα ότι *το* T1 εκφράζει έναν χρόνο στον οποίο ένα ορισμένο γεγονός πραγματοποιήθηκε, ενώ *το* T2 *και* T3 εκφράζουν ένα διάστημα του χρόνου κατά τη διάρκεια του οποίου μία ορισμένη κατάσταση συνεχίζεται. Έχουμε επιλέξει σκόπιμα ένα παράδειγμα στο οποίο μία ορισμένη κατάσταση μπορεί να συνεχίζεται από τις πληροφορίες σχετικά με ένα ορισμένο γεγονός: Από το γεγονός (το πιο πρόσφατο διορισμό του προμηθευτή S1) που εμφανίζεται στις 1η Ιουλίου του 1999 (και η σύμβαση *δεν* έχει διακοπεί στη συνέχεια), ότι ο προμηθευτής ήταν στην κατάσταση του να είναι στο πλαίσιο της σύμβασης από εκείνη την ημερομηνία μέχρι την παρούσα ημέρα. Η συμβατική τεχνολογία βάσεων δεδομένων μπορεί να χειριστεί τις χρονικές στιγμές (χρόνοι στις οποίες *τα* γεγονότα εμφανίζονται) πολύ καλά. Εντούτοις, δεν μπορεί να χειριστεί τα χρονικά διαστήματα (διαστήματα του χρόνου κατά τη διάρκεια των οποίων οι καταστάσεις συνεχίζονται) .

Παρατήρηση: *Παρόλο που το T2 και T3 είναι λογικά ισοδύναμα, υπάρχει σημαντική διαφορά στην μορφή, δηλαδή τα κατηγορήματα τα οποία είναι τρέχοντα, είναι διαφορετικά. Για να είμαστε συγκεκριμένοι, το T3 αναφέρεται σε ένα χρονικό διάστημα (με ένα συγκεκριμένο αρχικό σημείο και ένα τελικό σημείο), ενώ το T2 αναφέρεται μόνο σε μια συγκεκριμένη χρονική στιγμή. Κατά συνέπεια, η μορφή T2 δεν μπορεί να χρησιμοποιηθεί*

για τα ιστορικά αρχεία, ενώ το Τ3 μπορεί να αντικαταστήσει τη φράση "σήμερα" σε εκείνη την πρόταση με κάποια ρητή ημερομηνία, για παράδειγμα 1η Μαΐου του 2000. (φυσικά, το Τ3 θα αντιστοιχούσε κανονικά σε τρεις πλειάδες, κι όχι σε δυο.)Τα αντίστοιχα κατηγορήματα φαίνονται κάπως έτσι:

- *Τ2: Ο προμηθευτής Sx ήταν στο πλαίσιο της σύμβασης από την ημερομηνία d.*
- *Τ3: Ο προμηθευτής Sx ήταν στο πλαίσιο της σύμβασης κατά τη διάρκεια του διαστήματος από την ημερομηνία b μέχρι σήμερα e.*

Οι χρονικές βάσεις δεδομένων μπορούν να περιέχουν πληροφορίες σχετικά με το μέλλον καθώς επίσης και το παρελθόν.

2.3 Θεμελιώδης χρόνος εναντίον του χρόνου συναλλαγής

Σε μια συμβατική (δηλ., μη-χρονική) βάση δεδομένων, οτιδήποτε μπορεί να ενημερωθεί. Αλλά εάν τα ιστορικά στοιχεία μπορούσαν να ενημερωθούν, θα βρίσκαμε τον εαυτό μας αντιμέτωπο με την πιθανότητα η ιστορία να αλλάξει!

Είναι σημαντικό να γίνει κατανοητό ότι η βάση δεδομένων δεν περιέχει "τον πραγματικό κόσμο", περιέχει μόνο τη γνώση ή τις πεποιθήσεις *μας για* τον πραγματικό κόσμο. Και ενώ είναι λογικό να βεβαιωθεί ότι το παρελθόν είναι αμετάβλητο και η ιστορία δεν μπορεί υπό αυτήν τη μορφή ποτέ να αλλάξει, είναι εξίσου λογικό να βεβαιωθεί ότι οι πεποιθήσεις μας για αυτήν μπορούν να αλλάξουν. Σε ένα πλαίσιο βάσεων δεδομένων, όταν μιλάμε για την "ενημερωμένη ιστορία", αυτό που εννοούμε πραγματικά είναι να ενημερώνουμε τις πεποιθήσεις *μας για* την ιστορία, και όχι την ιστορία

Ας εξετάσουμε το ακόλουθο απλό παράδειγμα:

Έστω ότι p είναι η πρόταση: "ο προμηθευτής S1 ήταν στο πλαίσιο της σύμβασης." Υποθέστε ότι αυτή η κατάσταση εξασφαλίζεται από *τον* 1η Ιούλιου του 1999, μέχρι την 1η Μάιου του 2000, και επομένως παρεμβάλλουμε την ακόλουθη πλειάδα στη βάση δεδομένων:

S#	FROM	TO
S1	July 1st 1999	May 1st 2000

Σημειώστε πολύ προσεκτικά ότι αυτή η πλειάδα δεν αντιστοιχεί στην πρόταση p. Μάλλον, αντιστοιχεί σε αυτό που καλείται σφραγισμένος χρόνος επέκτασης *της* πρότασης p που μπορεί να δηλωθεί έτσι: "Ο προμηθευτής S1 ήταν στο πλαίσιο της σύμβασης από την 1η Ιουλίου *του* 1999, έως την 1η Μαΐου του 2000". Ο θεμελιώδης χρόνος για *το* p σε αυτό το παράδειγμα είναι το διάστημα του χρόνου κατά τη διάρκεια του οποίου (σύμφωνα με τις τρέχουσες πεποιθήσεις μας) *το* p ήταν στην πραγματικότητα αληθινό.

Σημείωση: *Υποθέτουμε για απλότητα ότι το διευκρινισμένο διάστημα (από την 1η Ιουλίου του 1999, μέχρι την 1η Μαΐου του 2000) είναι ο μόνος θεμελιώδης χρόνος για το p .*

Γενικότερα, ο θεμελιώδης χρόνος για μια δεδομένη πρόταση είναι ένα σύνολο διαστημάτων, όχι μόνο ένα ενιαίο διάστημα. Παραδείγματος χάριν, εάν ο προμηθευτής S1 ήταν επίσης στο πλαίσιο της σύμβασης, προηγουμένως, από την 1η Ιανουαρίου του 1998, μέχρι την 1η Απριλίου του 1998, τότε ο σχετικός θεμελιώδης χρόνος θα περιελάμβανε σαφώς δύο διαστήματα, όχι μόνο ένα. Αλλά θα υποθέσουμε για την απλότητα σε όλο το υπόλοιπο αυτού του τμήματος ότι οι θεμελιώδεις χρόνοι είναι πράγματι ενιαία διαστήματα.

Υποθέστε ότι μαθαίνουμε ότι ο προμηθευτής S1 "η σύμβαση του οποίου άρχισε στην πραγματικότητα στις 1η Ιουνίου, και όχι στις 1η Ιουλίου", επομένως αντικαθιστάμε την αρχική πλειάδα με αυτή:

S#	FROM	TO
S1	June 1st 1999	May 1st 2000

Κατά συνέπεια, ο θεμελιώδης χρόνος για την πρόταση p είναι τώρα το διάστημα από την 1η Ιουνίου του 1999, έως την 1η Μαΐου του 2000. Με άλλα λόγια, μόλις "ενημερώσαμε την πεποίθησή μας για την ιστορία". Αυτό βεβαίως *που* δεν έχουμε κάνει, είναι ότι *η* αναπροσαρμογή που μόλις αποδώσαμε δεν μπορεί να αλλάξει το ιστορικό γεγονός ότι η βάση δεδομένων παρουσίασε προηγουμένως τον προμηθευτή S1 στο πλαίσιο της σύμβασης από την 1η Ιουλίου (και όχι από την 1η Ιουνίου) του 1999.

Τέλος, υποθέστε ότι ανακαλύπτουμε ότι ένα λάθος έχει γίνει και ο προμηθευτής S1 *δεν* ήταν ποτέ στο πλαίσιο της σύμβασης. Επομένως διαγράφουμε την πλειάδα για τον προμηθευτή S1 εξ ολοκλήρου. Η πρόταση p είναι τώρα ψευδή, κατά συνέπεια, δεν υπάρχει τώρα κανένας θεμελιώδης χρόνος. (μπορούμε να πούμε ότι ο θεμελιώδης χρόνος είναι τώρα ένα κενό σύνολο διαστημάτων).

Τώρα υποθέστε ότι η αρχική πλειάδα παρεμβλήθηκε στο χρονικό διάστημα t1 *και* αντικαταστάθηκε στο t2, και εκείνη η πλειάδα αντικατάστασης διαγράφηκε έπειτα στο χρονικό διάστημα t3. Τότε θα λέγαμε ότι το διάστημα από *το* t1 *στο* t2 ήταν ο χρόνος συναλλαγής, όχι για την πρόταση p υπό αυτήν τη μορφή, αλλά για την σφραγισμένη χρονικά επέκταση του p από 1η Ιουλίου του 1999 έως την 1η Μαΐου του 2000.
Δηλαδή το διάστημα από *το* t1 *στο* t2 είναι ο χρόνος κατά τη διάρκεια του οποίου η βάση δεδομένων βεβαίωσε ότι αυτή η ιδιαίτερη σφραγισμένη χρονικά επέκταση *του* p ήταν αληθινή. Επιπλέον, η λογοτεχνία θα έλεγε ότι το διάστημα από το t2 *στο* t3 ήταν ο χρόνος συναλλαγής για τη σφραγισμένη χρονικά επέκταση *του* p *με θεμελιώδη* σφραγισμένο χρόνο από την 1η Ιουνίου του 1999 έως την 1η Μαΐου του 2000. Δηλαδή το διάστημα από *το* t2 *στο* t3 είναι ο χρόνος κατά τη διάρκεια του οποίου η βάση δεδομένων βεβαίωσε ότι αυτή η ιδιαίτερη σφραγισμένη χρονικά επέκταση *του* p ήταν αληθινή.

Σημείωση: *οι θεμελιώδης χρόνοι μπορούν να ενημερωθούν, αλλά οι χρόνοι συναλλαγής δεν μπορούν. Οι θεμελιώδης χρόνοι απεικονίζουν τις πεποιθήσεις μας για την ιστορία, και αυτές οι πεποιθήσεις μπορούν να αλλάξουν. Οι χρόνοι συναλλαγής, σε αντίθεση, απεικονίζουν την ιστορία υπό αυτήν τη μορφή, και η ιστορία δεν μπορεί να αλλάξει. Πράγματι, οι χρόνοι συναλλαγής ρυθμίζονται από το σύστημα, όχι από κάποιο ανθρώπινο χρήστη.*

2.4 Μερικά θεμελιώδη ζητήματα

Η ιδέα ότι ένα διάστημα αρχίζει με τον χρόνο b *και τελειώνει* με τον χρόνο e μπορεί να θεωρηθεί ως σύνολο όλων των χρόνων t έτσι ώστε b $\leq$ t $\leq$ e (όπου "<" σημαίνει "νωρίτερα από). Αν και προφανές, αυτή η απλή έννοια έχει τις πολυάριθμες εκτεταμένες συνέπειες. Οδηγεί επίσης άμεσα σε μια σειρά αρκετά θεμελιωδών ζητημάτων!

- "όλοι οι χρόνοι *τ* έτσι ώστε *το* b $\leq$t $\leq$e αυξάνει το φάσμα των άπειρων συνόλων και όλων των εννοιολογικών και υπολογιστικών δυσκολιών που τέτοια σύνολα πάσχουν;

ΑΠΑΝΤΗΣΗ: *να* εμφανίζεται, αλλά απομακρύνουμε το φάσμα και παρακάμπτουμε *τις* δυσκολίες με την υιοθέτηση της υπόθεσης ότι η υπόδειξη ως προς το χρόνο αποτελείται από μια πεπερασμένη ακολουθία ιδιαίτερων, αδιαίρετων χρονικών τμημάτων (όπου ένα χρονικό τμήμα είναι η μικρότερη χρονική μονάδα που το σύστημα είναι ικανό αντιπροσώπευσης). Το διάστημα που αρχίζει με το χρόνο b και ο χρόνος e που τελειώνει περιλαμβάνει έτσι έναν πεπερασμένο αριθμό τέτοιων τμημάτων.

- Οι προτάσεις T1, T2, *και* T3 από την παράγραφο φαίνονται να υποθέτουν ότι τα χρονικά τμημάτων είναι ημέρες, αλλά σίγουρα το σύστημα υποστηρίζει τις χρονικές μονάδες στα μικροσκοπικά μέρη ενός δευτερολέπτου. Εάν ο S1 ήταν προμηθευτής στις 1^{η} Ιουλίου του 1999, αλλά όχι στις 30 Ιουνίου του 1999, τι μπορεί να γίνει για το θεωρούμενο διάστημα του χρόνου από την αρχή της ημέρας τις 1^{η} Ιουλίου του 1999, μέχρι την ίδια την στιγμή του διορισμού, κατά τη διάρκεια της οποίας ο S1 δεν ήταν ακόμα επίσημα στο πλαίσιο της σύμβασης;

ΑΠΑΝΤΗΣΗ: Πρέπει να διακρίνουμε προσεκτικά μεταξύ των χρονικών τμημάτων υπό αυτήν τη μορφή, τα οποία είναι οι μικρότερες χρονικές μονάδες που το σύστημα είναι ικανό να αντιπροσωπεύσει, και των χρονικών μονάδων που είναι σχετικές για κάποιο ιδιαίτερο σκοπό, και οι οποίες μπορεί να είναι ημέρες ή μήνες ή χιλιοστά του δευτερολέπτου (κ.λπ., κ.λπ.). Καλούμε αυτά τα τελευταία χρονικά σημεία μονάδων προκειμένου να τονιστεί το γεγονός ότι για το προσιτό σκοπό αυτά είναι αδιαίρετα. Ένα χρονικό σημείο είναι ένα σύνολο χρονικών τμημάτων που εκτείνεται από ένα "τμήμα ορίου" στο επόμενο (π.χ. από τα μεσάνυχτα μια ημέρας στα μεσάνυχτα της επόμενης). Τα χρονικά σημεία έχουν μια διάρκεια (μια ημέρα στο παράδειγμα). Τυπικά, τα χρονικά σημεία είναι πράγματι σημεία που είναι αδιαίρετα, και η έννοια της διάρκειας αυστηρά δεν ισχύει.

- η υπόδειξη ως προς το χρόνο μπορεί να θεωρηθεί (για κάποιο συγκεκριμένο σκοπό) ως πεπερασμένη ακολουθία χρονικών σημείων, και μπορούμε να αναφερθούμε στο χρονικό σημείο που πετυχαίνει ή που προηγείται αμέσως από οποιοδήποτε δεδομένο σημείο. Είναι αυτό σωστό;

ΑΠΑΝΤΗΣΗ: Ναι, μέχρι ενός σημείου η εν λόγω ύπαρξη σημείου, είναι φυσικά, το τέλος του χρόνου! Και μέχρι ενός σημείου, η αρχή του χρόνου. Η αρχή του χρόνου είναι

ένα χρονικό σημείο που δεν έχει κανέναν προκάτοχο και το τέλος του χρόνου είναι ένα χρονικό σημείο που δεν έχει κανέναν διάδοχο.

Κλείνουμε αυτό το τμήμα με μερικές τελικές παρατηρήσεις σχετικά με το τρέχων παράδειγμα. Για να είμαστε συγκεκριμένοι, από αυτό το σημείο υποθέτουμε, αρκετά ρεαλιστικά, ότι:
1. Κανένας προμηθευτής δεν μπορεί να τελειώσει μια σύμβαση μια ημέρα και να αρχίσει άλλη την αμέσως επόμενη ημέρα.
2. Κανένας προμηθευτής δεν μπορεί να είναι στο πλαίσιο δύο ευδιάκριτων συμβάσεων συγχρόνως.
3. Οι συμβάσεις προμηθευτών μπορούν να είναι open-ended, ένας προμηθευτής μπορεί να είναι αυτήν την περίοδο στο πλαίσιο της σύμβασης και η τελευταία ημέρα αυτής της σύμβασης μπορεί να είναι άγνωστη.

2.5 Προσθήκη γνωρισμάτων

Σε αυτό το σημείο χρησιμοποιούμε τη βάση δεδομένων προμηθευτής-αποστολές ως βάση για μερικά από τα προβλήματα που προκύπτουν όταν προσπαθούμε να προσθέσουμε χρονικά χαρακτηριστικά γνωρίσματα σε μια συμβατική (δηλ., μη-χρονική) βάση δεδομένων.

Η πρώτη αλλαγή που εισάγουμε δεν είναι χρονική, είναι ένα θέμα απλοποίησης. Για να είμαστε συγκεκριμένοι, απλοποιούμε τη σχέση S, με τη ρίψη όλων των ιδιοτήτων εκτός από την ιδιότητα S#. Έτσι η σχέση είναι:

Supplier S# is currently under contract

Το παρακάτω σχήμα παρουσιάζει ένα σύνολο τιμών για τη βάση δεδομένων μετά από αυτήν την απλοποίηση. Όπως ήδη έχει λεχθεί, αυτή η αναθεωρημένη βάση δεδομένων είναι ακόμα καθαρά συμβατική και δεν περιλαμβάνει καμία χρονική πτυχή.

Παράδειγμα 2.1
Τιμές για την ΒΔ του παραδείγματος προμηθευτών-αποστολών

S#
S1
S2
S3
S4
S5

S#	P#
S1	P1
S1	P2
S1	P3
S1	P4
S1	P5
S1	P6
S2	P1
S2	P2
S3	P2
S4	P2
S4	P4
S4	P5

Τώρα, μπορεί να σκεφτείτε ότι η απλουστευμένη βάση δεδομένων του παραπάνω σχήματος είναι πάρα πολύ απλή. Εντούτοις, είναι τέλεια επαρκής ως βάση για σχεδόν όλα τα θέματα που θέλουμε να θίξουμε σε αυτό το μέρος του βιβλίου.

Στη συνέχεια θα εξετάσουμε μερικούς απλούς περιορισμούς και ερωτήματα ενάντια σε αυτήν την αναθεωρημένη βάση δεδομένων. Στα επόμενα δύο τμήματα, θα δούμε τι συμβαίνει σε αυτούς τους περιορισμούς και ερωτήματα όταν η βάση επεκτείνεται για να ενσωματώσει τα διάφορα χρονικά χαρακτηριστικά γνωρίσματα.

Περιορισμοί (constraints)
(αρχική βάση δεδομένων):

Οι μόνοι περιορισμοί που θέλουμε να εξετάσουμε εδώ είναι οι διάφοροι περιορισμοί κλειδιού. Σας υπενθυμίζουμε ότι {S#} *και* {S #, P #} είναι τα πρωτεύοντα κλειδιά για τις σχέσεις S *και* SP αντίστοιχα, και το {S#} είναι ένα ξένο κλειδί στη SP *που* αναφέρεται στο πρωτεύον κλειδί S.

ΣΗΜΕΙΩΣΗ: Από την άποψη του σχεδίου ταξινόμησης που περιγράφεται στο κεφάλαιο I, οι περιορισμοί πρωτεύοντος κλειδιού είναι και περιορισμοί σχέσεων και ο περιορισμός ξένου κλειδιού είναι ένας περιορισμός βάσεων δεδομένων.

Ερωτήματα (queries)
(αρχική βάση δεδομένων):

Εξετάζουμε ακριβώς δύο ερωτήματα, και τα δύο πολύ απλά:

- ✓ **QUERY A:** πάρτε τους αριθμούς των προμηθευτών για τους προμηθευτές που είναι αυτήν την περίοδο ικανοί να προμηθεύσουν τουλάχιστον ένα μέρος.

  Εδώ είναι *το* **TUTORIAL D** αυτού του ερωτήματος:
  SP {S#}

- ✓ **QUERY B:** πάρτε τους αριθμούς των προμηθευτών για τους προμηθευτές που είναι αυτήν την περίοδο ανίκανοι να προμηθεύσουν οποιαδήποτε μέρη.

Διατύπωση **TUTORIAL D**:

S {S#} MINUS SP {S#}

Παρατηρήστε ότι το ερώτημα Α περιλαμβάνει μια απλή προβολή και το ερώτημα Β περιλαμβάνει τη διαφορά μεταξύ των δύο αυτών προβολών.

2.6 "ΗΜΙΧΡΟΝΙΚΗ" προσέγγιση της Β.Δ

Για να επαναλάβουμε, θέλουμε να προχωρήσουμε ήπια και να κάνουμε τις χρονικές αναθεωρήσεις μας στη βάση δεδομένων αποστολών-προμηθευτών με αποσπασματικό τρόπο. Η πρώτη αναθεώρηση περιλαμβάνει τις ημιχρονικές σχέσεις S *και* SP προσθέτοντας μια σφραγισμένη χρονικά ιδιότητα SINCE, σε καθεμιά *και* μετονομάζοντας τις δυο σχέσεις αναλόγως.

Παράδειγμα 2.2
Προμηθευτές-αποστολές
Ενδεικτικές ημιχρονικές τιμές

S_SINCE

S#	SINCE
S1	d04
S2	d07
S3	d03
S4	d04
S5	d02

SP_SINCE

S#	P#	SINCE
S1	P1	d04
S1	P2	d05
S1	P3	d09
S1	P4	d05
S1	P5	d04
S1	P6	d06
S2	P1	d08
S2	P2	d09
S3	P2	d08
S4	P2	d06
S4	P4	d04
S4	P5	d05

Για απλότητα, δεν παρουσιάζουμε τις πραγματικά χρονικά σφραγισμένες τιμές στο παραπάνω σχήμα. Άντ' αυτού, χρησιμοποιούμε τα σύμβολα της μορφής d01, d02 *και* ούτω καθ' εξής, όπου το d μπορούμε να πούμε ότι είναι η "μέρα". Υποθέτουμε ότι η $1^{η}$ ημέρα προηγείται αμέσως της $2^{ης}$ ημέρας, η $2^{η}$ ημέρα προηγείται αμέσως της $3^{ης}$ ημέρας, και τα λοιπά...

Το κατηγόρημα για την σχέση S_SINCE είναι:

Από την ημέρα SINCE (και όχι την ημέρα αμέσως πριν από την ημέρα SINCE), ο προμηθευτής S # ήταν στο πλαίσιο της σύμβασης.

Και το κατηγόρημα για την σχέση SP _SINCE είναι:

Από την ημέρα SINCE (και όχι την ημέρα αμέσως πριν από την ημέρα SINCE), ο προμηθευτής S # *ήταν σε θέση να προμηθεύσει το μέρος P #.*

Περιορισμοί (constraints)
(ημιχρονική βάση δεδομένων):

Τα πρωτεύοντα και τα ξένα κλειδιά για την ημιχρονική βάση δεδομένων του σχήματος 2.2 είναι τα ίδια που ήταν για την αρχική βάση δεδομένων του σχήματος 2.1.

Σημείωση: *έχουμε καθορίσει τις δύο ιδιότητες SINCE να είναι τύπου DATE, που αντιπροσωπεύει τις Gregorian (Γρηγοριανές) ημερομηνίες, που σημαίνει τις ημερομηνίες που είναι ακριβείς στην ημέρα και περιορίζονται από τους κανόνες του γρηγοριανού ημερολογίου (π.χ. " 31 Απριλίου ,2005" και "29η Φεβρουαρίου, 2100" δεν είναι έγκυρες ημερομηνίες).*

```
VAR S_SINCE RELATION {S# S#, SINCE DATE}
        KEY {S#};

VAR SP_SINCE RELATION {S# S#, P# P#, SINCE DATE}
        KEY {S#, P#}
        FOREIGN KEY {S#} REFERENCES S_SINCE;
```

Εντούτοις, χρειαζόμαστε έναν πρόσθετο περιορισμό, επιπλέον του ξένου βασικού περιορισμού από SP_SINCE *σε* S_SINCE, για να εκφράσουμε το γεγονός ότι κανένας προμηθευτής δεν μπορεί να προμηθεύσει οποιοδήποτε μέρος προτού τοποθετηθεί αυτός ο προμηθευτής στο πλαίσιο της σύμβασης:

```
CONSTRAINT XST1 /* επιπρόσθετος ημιχρονικός περιορισμός N.1 */
        IS EMPTY (((S SINCE RENAME SINCE AS SS) JOIN
                        (SP SINCE RENAME SINCE AS SPS))
                WHERE SPS < SS);
```

Η διαίσθηση πίσω από αυτήν την διατύπωση είναι ότι εάν η πλειάδα SP *στη* σχέση SP_SINCE αναφέρεται στην πλειάδα S *στο* S_SINCE, τότε η τιμή SINCE *στην* SP *δεν* πρέπει να είναι λιγότερο από αυτή *στο* S.

Παρατηρούμε ότι, δίνοντας μια ημιχρονική βάση δεδομένων όπως αυτήν του σχήματος 2.2, θα πρέπει πιθανώς να δηλώσουμε πολλούς περιορισμούς της ίδιας γενικής και μάλλον δυσκίνητης φύσης με τον περιορισμό XST1.

Ερωτήματα (Queries)
(ημιχρονική βάση δεδομένων):

* **QUERY A:** Πάρτε τους αριθμούς των προμηθευτών για τους προμηθευτές που είναι αυτήν την περίοδο ικανοί να προμηθεύσουν τουλάχιστον ένα μέρος, δείχνοντας σε κάθε περίπτωση την ημερομηνία από τότε που είναι ικανοί να το κάνουν.

Εάν ο προμηθευτής Sx είναι αυτήν την περίοδο ικανός να προμηθεύσει διαφορετικά μέρη, τότε ο Sx ήταν σε θέση να προμηθεύσει τουλάχιστον ένα μέρος από την πιο πρόωρη ημερομηνία SINCE που παρουσιάζεται για τον Sx στην σχέση SP_SINCE (π.χ., εάν Sx είναι S1, τότε η πιο πρόωρη ημερομηνία SINCE είναι d04).

Εδώ είναι το TUTORIAL D του ερωτήματος:

SUMMARIZE SP PER SP {S#} ADD MIN (SINCE) AS SINCE

Το αποτέλεσμα μοιάζει με αυτό:

S#	SINCE
S1	d04
S2	d08
S3	d08
S4	d04

- **QUERY B:** Πάρτε τους αριθμούς προμηθευτών για τους προμηθευτές που είναι αυτήν την περίοδο ανίκανοι να προμηθεύσουν οποιαδήποτε μέρη, δείχνοντας σε κάθε περίπτωση την ημερομηνία από τότε που είναι ανίκανοι να το κάνουν.

Στα στοιχεία δειγμάτων μας υπάρχει ακριβώς ένα προμηθευτής (συγκεκριμένα ο προμηθευτής S5) που είναι αυτήν την περίοδο ανίκανος να προμηθεύσει οποιαδήποτε μέρη. Εντούτοις, δεν μπορούμε να ανακαλύψουμε την ημερομηνία από τότε που ο S5 είναι ανίκανος να προμηθεύσει οποιαδήποτε μέρη, επειδή υπάρχουν ανεπαρκείς πληροφορίες στη βάση δεδομένων, αφού η βάση δεδομένων είναι ακόμα ημιχρονική. Παραδείγματος χάριν, υποθέστε ότι η τρέχουσα ημέρα είναι η $10^{η}$, και τότε ίσως είναι η περίπτωση όπου ο S5 ήταν σε θέση να προμηθεύσει τουλάχιστον ένα μέρος από την $2^{η}$ ημέρα, όταν τοποθετήθηκε ο S5 αρχικά στο πλαίσιο της σύμβασης, μέχρι τόσο αργά όσο *την* $9^{η}$ ημέρα ή, στην άλλη περίπτωση, ίσως είναι η περίπτωση όπου ο S5 δεν ήταν σε θέση να προμηθεύσει τίποτα.

Προκειμένου να υπάρξει οποιαδήποτε ελπίδα απάντησης του ερωτήματος B, πρέπει να ολοκληρώσουμε τη "χρονοτριβή" της βάσης δεδομένων μας, ή τουλάχιστον τη μερίδα SP από αυτή. Για να είμαστε ακριβέστεροι, πρέπει να διατηρήσουμε τα ιστορικά αρχεία *στη* βάση δεδομένων που παρουσιάζουν ποιοι προμηθευτές ήταν σε θέση να προμηθεύσουν ποια μέρη και πότε.

2.7 "ΠΛΗΡΩΣ ΧΡΟΝΙΚΗ" προσέγγιση της Β.Δ

Το σχήμα 2.3 παρουσιάζει μια "πλήρως χρονική" έκδοση της βάσης των προμηθευτών και των αποστολών. Παρατηρήστε ότι οι ιδιότητες SINCE έχουν γίνει ιδιότητες FROM, και κάθε σχέση έχει αποκτήσει μια πρόσθετη χρονικά σφραγισμένη ιδιότητα, αποκαλούμενη TO (και για αυτόν τον λόγο έχουμε αντικαταστήσει *την* _SINCE *σε* _FROM_ TO στη σχέση ονομάτων).Οι ιδιότητες FROM *και* TO μαζί εκφράζουν την έννοια ενός διαστήματος χρόνου κατά τη διάρκεια του οποίου (σύμφωνα με τις τρέχουσες πεποιθήσεις μας) κάποια πρόταση ήταν αληθινή.

Παράδειγμα 2.3
Β.Δ. προμηθευτών-αποστολών
$1^{η}$ πλήρως χρονική έκδοση με την χρήση των ιδιοτήτων FROM *και* TO

S_FROM_TO

S#	FROM	TO
S1	d04	d10
S2	d02	d04
S2	d07	d10
S3	d03	d10
S4	d04	d10
S5	d02	d10

SP_FROM_TO

S#	P#	FROM	TO
S1	P1	d04	d10
S1	P2	d05	d10
S1	P3	d09	d10
S1	P4	d05	d10
S1	P5	d04	d10
S1	P6	d06	d10
S2	P1	d02	d04
S2	P1	d08	d10
S2	P2	d03	d03
S2	P2	d09	d10
S3	P2	d08	d10
S4	P2	d06	d09
S4	P4	d04	d08
S4	P5	d05	d10

Επειδή τώρα διατηρούμε τα ιστορικά αρχεία, υπάρχουν περισσότερες πλειάδες σε αυτήν την βάση δεδομένων από ότι υπήρξαν στις προηγούμενες. Στην πραγματικότητα, η πλήρως χρονική βάση δεδομένων του σχήματος 2.3 περιλαμβάνει όλες τις πληροφορίες σε σχέση με την ημιχρονική του σχήματος 2.2. Καθαρά χάριν του παραδείγματος, έχουμε παρουσιάσει την τιμή TO για δύο από τις αποστολές προμηθευτών S4's ως ημερομηνία πριν από την τρέχουσα ημερομηνία (δηλ., έχουμε μετατρέψει εκείνες τις δύο αποστολές από "τρέχουσες" σε "ιστορικές" πληροφορίες). Αυτή η πλήρως χρονική βάση δεδομένων περιλαμβάνει επίσης τις ιστορικές πληροφορίες σχετικά με ένα προηγούμενο διάστημα του χρόνου, από d02 σε d04, κατά τη διάρκεια του οποίου ο προμηθευτής S2 ήταν προηγουμένως στο πλαίσιο της σύμβασης και ικανός να προμηθεύσει ορισμένα μέρη.

Το κατηγόρημα για την σχέση S_FROM_TO είναι:

Από την ημέρα FROM (και όχι την ημέρα αμέσως πριν την FROM) στην ημέρα TO (και όχι την ημέρα αμέσως μετά την TO) ο προμηθευτής S# ήταν στο πλαίσιο της σύμβασης.

Το κατηγόρημα για την σχέση SP _ FROM_TO είναι:

Από την ημέρα FROM (και όχι την ημέρα αμέσως πριν την FROM) ως την ημέρα TO (και όχι την ημέρα αμέσως μετά την TO) ο προμηθευτής S# ήταν ικανός να προμηθεύσει το μέρος P#.

Περιορισμοί (constraints)
(πλήρως χρονική βάση δεδομένων):

```
CONSTRAINT S_FROM_ TO_ OK
    IS_EMPTY (S_FROM_TO WHERE TO < FROM);

CONSTRAINT SP_ FROM_ TO_ OK
    IS EMPTY (SP_ FROM_ TO WHERE TO < FROM);
```

Παρατηρήστε στο σχήμα 2.3 ότι έχουμε περιλάβει τις ιδιότητες FROM στο αρχικό κλειδί και για την σχέση S_FROM_TO και για την σχέση SP_ FROM_TO. Πράγματι. *το* αρχικό κλειδί για *την* S_FROM_TO (παραδείγματος χάριν) σαφώς δεν μπορεί να είναι μόνο το S#, επειδή εάν ήταν δεν θα ήμασταν σε θέση να εξετάσουμε έναν προμηθευτή όπως τον προμηθευτή S2 ο οποίος ήταν στο πλαίσιο της σύμβασης κατά τη διάρκεια δύο ή περισσότερων χωριστών διαστημάτων. Μια παρόμοια παρατήρηση ισχύει και για το SP_FROM_TO.

Εδώ είναι οι πιθανοί ορισμοί σχέσης, εκφραζόμενοι για άλλη μια φορά στο TUTORIAL D:

```
VAR S_FROM_TO RELATION {S# S#, FROM DATE, TO DATE}
    KEY {S#, FROM}
    KEY {S#, TO};

VAR SP_FROM_TO RELATION {S# S#, P# P#, FROM DATE, TO DATE}
    KEY {S#, P#, FROM}
    KEY {S#, P#, TO};
```

Εντούτοις, οι περιορισμοί που έχουμε συζητήσει μέχρι τώρα, τους δύο "_FROM_TO_OK" περιορισμούς και τους τέσσερις περιορισμούς κλειδιού, είμαστε ακόμα ανεπαρκείς να συλλάβουμε όλα όσα θα θέλαμε. Εξετάστε την σχέση S_FROM TO _, παραδείγματος χάριν. Προφανώς, εάν υπάρχει μία πλειάδα για τον προμηθευτή Sx σε αυτή την σχέση με τιμή FROM f *και* TO την τιμή t, τότε θέλουμε εκεί να μην υπάρχει πλειάδα για τον προμηθευτή Sx σε αυτή την ίδια σχέση, δείχνοντας ότι ο Sx ήταν στο πλαίσιο της σύμβασης την ημέρα αμέσως πριν από *την* f ή την ημέρα αμέσως μετά από *την* t. Εξετάστε τον προμηθευτή S1, για τον οποίο έχουμε μόνο μια πλειάδα S_FROM_TO, *με* FROM = d04 *και* TO = d10. Το γεγονός ότι {S#, FROM} είναι ένα πρωτεύον κλειδί για αυτή την σχέση, είναι σαφώς ανεπαρκές να αποτρέψει την εμφάνιση μιας πρόσθετης πλειάδας για τον S1 *με* FROM = d02 *και* TO = d06, το οποίο θα έδειχνε μεταξύ άλλων ότι ο S1 ήταν στο πλαίσιο της σύμβασης την ημέρα αμέσως πριν από την 4η ημέρα. Σαφώς, το τι θα θέλαμε είναι ότι αυτές οι δύο πλειάδες S1 μπορούν να συνδυαστούν σε μια ενιαία πλειάδα με FROM = d02 *και* TO = d10.

Ο μη συνδυασμός δύο πλειάδων στο προηγούμενο παράδειγμα θα ήταν σχεδόν κακός διότι επιτρέπει διπλότυπα (λέει το ίδιο πράγμα δύο φορές). Και αυτές οι δύο πλειάδες για τον προμηθευτή S1 με την επικάλυψη FROM - TO στα διαστήματα πράγματι "λένε το ίδιο πράγμα δύο φορές". Για να είμαστε συγκεκριμένοι, και οι δύο λένε ότι ο προμηθευτής S1 ήταν στο πλαίσιο της σύμβασης τις ημέρες 4, 5, και 6. Πράγματι, εάν αυτές οι δύο πλειάδες εμφανίζονταν και οι δύο, τότε η σχέση S_FROM _ *θα* ήταν παραβίαση του ορίσματός της.

Εδώ είναι ένας περιορισμός που απαγορεύει τέτοια επικάλυψη και κατάληξη:

```
CONSTRAINT XFT11
      IS_ EMPTY
        (((S_ FROM_ TO   RENAME (FROM AS F1, TO AS T1)) JOIN
          (S_FROM_TO   RENAME (FROM AS F2, TO AS T2)))
             WHERE (T1≥ F2 AND T2 ≥ F1)) OR
                    ( F2 = T1+1 OR  F1 = T2+1 ) ) ;
```

Με αυτό το παράδειγμα, αρχίζουμε να βλέπουμε το πρόβλημα. Αυτός ο περιορισμός είναι αρκετά περίπλοκος. Να αναφέρουμε το γεγονός ότι το T1 + 1 το χρησιμοποιούμε για να υποδείξουμε τον άμεσο διάδοχο της ημέρας που δείχνεται από το T1. Επιπλέον, παρατηρούμε ότι λαμβάνοντας υπόψη μια πλήρως χρονική βάση δεδομένων όπως αυτήν του σχήματος 2.3, θα πρέπει πιθανώς να δηλώσουμε πολλούς περιορισμούς της ίδιας γενικής φύσης με τον περιορισμό XFT1.

Σημειώστε ότι ο συνδυασμός ιδιοτήτων {S#, FROM} στην σχέση SP_ FROM_TO *δεν* είναι ένα ξένο κλειδί από αυτή τη σχέση, σε σχέση *με την* S_FROM_TO, ακόμα κι αν περιλαμβάνει τις ίδιες ιδιότητες με το πρωτεύον κλειδί της σχέσης S_FROM_TO. (Δεν είναι ξένο κλειδί, επειδή είναι δυνατό για μια τιμή {S#, FROM} να εμφανιστεί στη*ν* SP_ FROM_TO και όχι *στην* S_FROM_ TO.) Αλλά βεβαίως πρέπει να εξασφαλίσουμε ότι εάν ένας δεδομένος προμηθευτής αντιπροσωπεύεται στην σχέση SP_FROM_TO, τότε αυτός ο προμηθευτής αντιπροσωπεύεται στη σχέση S_FROM_TO επίσης:

```
CONSTRAINT XFT2
      SP_FROM_TO {S#} ⊆ S_FROM_TO {S#};
```

Αυτός ο περιορισμός είναι ένα παράδειγμα μιας εξάρτησης συνυπολογισμού. Οι εξαρτήσεις συνυπολογισμού μπορούν να θεωρηθούν ως γενίκευση των αναφερόμενων περιορισμών. Και πρέπει να είναι σαφές ότι οποιαδήποτε πλήρως χρονική βάση δεδομένων όπως αυτήν του σχήματος 2.3 είναι πιθανό να περιλάβει έναν μεγάλο αριθμό τέτοιων εξαρτήσεων, τουλάχιστον σιωπηρά.

Ο περιορισμός XFT2 δεν είναι ακόμα αρκετός, ωστόσο πρέπει να εξασφαλίσουμε ότι εάν η σχέση SP_FROM_TO παρουσιάζει κάποιο προμηθευτή να είναι σε θέση να *προμηθεύσει* κάποιο μέρος κατά τη διάρκεια κάποιου διαστήματος του χρόνου, τότε η σχέση S_FROM_TO δείχνει αυτόν τον ίδιο προμηθευτή να είναι στο πλαίσιο της σύμβασης σε όλο αυτό το διάστημα του χρόνου. Δύο προσπάθειες για τη διατύπωση

αυτού του περιορισμού (ο πρώτος του οποίου είναι ανακριβής) παρουσιάζονται παρακάτω.

Εδώ είναι μια πρώτη προσπάθεια:

```
CONSTRAINT XFT3                    /* προειδοποίηση--ανεπαρκής! */

IS_EMPTY
     ((S_FROM_TO   RENAME (FROM AS SF, TO AS ST)) JOIN

      (SP_FROM_TO RENAME (FROM AS SPF, TO AS SPT) ) )

      WHERE SPF < SF OR SPT > ST);
```

Εάν οι πλειάδες S και SP, από τις σχέσεις S_FROM_TO και SP_FROM_TO, αντίστοιχα, αντιστοιχούν στον ίδιο προμηθευτή, τότε το διάστημα FROM-TO *στο* S πρέπει να καλύψει αυτού *στην* SP. Η διατύπωση αυτή είναι ανακριβείς, ή τουλάχιστον ελλιπείς. Για να δείτε το γιατί, έστω η σχέση S_FROM_TO να είναι όπως φαίνεται στο σχήμα 2.3, κι έστω η σχέση SP_FROM_TO να περιλαμβάνει μια πλειάδα για τον προμηθευτή S2 *με* FROM = d03 *και* TO=d04. Μια τέτοια ρύθμιση είναι σαφώς συνεπής, και όμως ο περιορισμός XFT3 όπως δηλώνει θα την απαγόρευε (επειδή το αποτέλεσμα της συνένωσης(join) θα περιελάμβανε μια πλειάδα για τον προμηθευτή S2 *με* SF = d07, ST = d10, SPF = d03, *και* SPT = d04, με αυτόν τον τρόπο αναγκάζοντας την IS_EMPTY δοκιμή να δώσει ψευδή(false)).

Εδώ σε αντίθεση είναι μια σωστή διατύπωση:

```
CONSTRAINT XFT3
      COUNT (SP_ FROM_ TO {ALL BUT P#}) =
      COUNT (((SP_ FROM_ TO RENAME (FROM AS SPF, TO AS SPT ) )
                                                    {ALL BUT P#}
                JOIN
                   (S_FROM_TO RENAME (FROM AS SF, TO AS ST)))
                 WHERE SF ≤ SPF AND ST ≥ SPT);
```

Σύμφωνα μ' αυτήν την διατύπωση εάν η σχέση SP_ FROM_TO περιλαμβάνει μια πλειάδα που παρουσιάζει τον προμηθευτή Sx ικανό να προμηθεύσει κάποιο συγκεκριμένο μέρος από την ημέρα SPF στην ημέρα SPT, τότε η σχέση S_FROM_TO πρέπει να περιλάβει ακριβώς μια πλειάδα που παρουσιάζει τον προμηθευτή Sx να είναι στο πλαίσιο της σύμβασης σε όλο αυτό το διάστημα. Μια λεπτομερής εξήγηση ακολουθεί:

- Ο περιορισμός όπως δηλώνεται βεβαιώνει ότι τα δύο μέρη πρέπει να είναι ίσα.
- το πρώτο μέρος είναι μια αρίθμηση του αριθμού ευδιάκριτων προτάσεων της μορφής "Ο Sx παρέχει κάποιο μέρος από την ημέρα SPF στην ημέρα SPT "*που* υπονοείται από την σχέση SP _ FROM_TO. Έστω ότι η αρίθμηση είναι N.

Το δεύτερο μέρος είναι μια αρίθμηση του αριθμού πλειάδων που περιλαμβάνονται σε έναν ορισμένο περιορισμό μιας ορισμένης συνένωσης. Η εν λόγω συνένωση πρέπει να περιέχει τουλάχιστον μια πλειάδα για καθεμία από τις Ν προτάσεις της μορφής "Ο Sx προμηθεύει κάποιο μέρος από την ημέρα SPF ως την ημέρα SPT" που υπονοείται από την σχέση SP_FROM_TO.

Σημείωση: *Με την χρήση στον περιορισμό XFT3 των εκφράσεων της μορφής "ALL BUT... " διευκρινίζουμε μια προβολή κάποιας σχέσης σε όλες τις ιδιότητες εκτός από αυτές που διευκρινίζονται.*

Ερωτήματα (Queries)
(πλήρως χρονική βάση δεδομένων):

Εδώ είναι τα πλήρως χρονικά ανάλογα των ερωτημάτων Α και Β:

- **QUERY A:** πάρτε τις τριάδες S#-FROM-TO για τους προμηθευτές που ήταν σε θέση να προμηθεύσουν τουλάχιστον ένα μέρος κατά τη διάρκεια τουλάχιστον ενός διαστήματος του χρόνου, όπου FROM *και* TO μαζί υποδεικνύουνε ένα μέγιστο διάστημα κατά τη διάρκεια του οποίου ο προμηθευτής S# ήταν στην πραγματικότητα ικανός να προμηθεύσει τουλάχιστον ένα μέρος.

ΣΗΜΕΙΩΣΗ: Χρησιμοποιούμε τον όρο "μέγιστο" εδώ ως κατάλληλη στενογραφία για να σημάνουμε ότι ο προμηθευτής S# ήταν ανίκανος να παρέχει οποιοδήποτε μέρος την ημέρα αμέσως πριν την FROM ή αμέσως μετά από την TO. Σημειώστε επίσης ότι το αποτέλεσμα του ερωτήματος μπορεί να περιέχει διάφορες πλειάδες για τον ίδιο προμηθευτή (αλλά με διαφορετικά διαστήματα)

- **QUERY B:** πάρτε τις τριάδες S#-FROM-TO για τους προμηθευτές που ήταν ανίκανοι να προμηθεύσουν οποιαδήποτε μέρη κατά τη διάρκεια τουλάχιστον ενός διαστήματος του χρόνου, όπου FROM *και* TO μαζί υποδεικνύουνε ένα μέγιστο διάστημα κατά τη διάρκεια του οποίου ο προμηθευτής S# ήταν στην πραγματικότητα ανίκανος να προμηθεύσει οποιοδήποτε μέρος. (Πάλι το αποτέλεσμα μπορεί να περιέχει διάφορες πλειάδες για τον ίδιο προμηθευτή.)

Το πρόβλημα των χρονικών στοιχείων είναι ότι οδηγούν γρήγορα σε περιορισμούς και ερωτήματα που είναι αδικαιολόγητα σύνθετοι να εκφραστούν: αδικαιολόγητα σύνθετοι, δηλαδή, εκτός αν το σύστημα παρέχει μερικές καλά σχεδιασμένες στενογραφίες, οι οποίες είναι εμπορικά διαθέσιμες

ΚΕΦΑΛΑΙΟ 3: ΔΙΑΧΕΙΡΙΣΗ ΔΙΑΣΤΗΜΑΤΩΝ

3.1 Εισαγωγή

Είμαστε τώρα έτοιμοι να αρχίσουμε την ανάπτυξή ενός κατάλληλου συνόλου κατασκευασμάτων για την εξέταση των χρονικών δεδομένων. Το πρώτο και πιό θεμελιώδες βήμα είναι να αναγνωρίσουμε την ανάγκη να αντιμετωπίσουμε τα διαστήματα και να τα μεταχειριστούμε ως τιμές , αντί να τα μεταχείριστούμε ως ζευγάρια χωριστών τιμών όπως κάναμε στο προηγούμενο κεφάλαιο.

Τι ακριβώς είναι ένα διάστημα; Παρατηρήστε το σχήμα 2.3 στο κεφάλαιο 2. Σύμφωνα με αυτό το παράδειγμα, ο προμηθευτής S1 ήταν σε θέση να προμηθεύσει το μέρος P1 κατά τη διάρκεια του διαστήματος από την $4^{η}$ *ως την* $10^{η}$ ημέρα. Αλλά τι σημαίνει " από την $4^{η}$ *ως την* $10^{η}$ ημέρα"; Είναι σαφές ότι οι ημέρες 5 , 6, 7, 8 και 9 περιλαμβάνονται. Αλλά τι γίνεται με τις ημέρες 4 και 10; Λαμβάνοντας υπόψη ένα καθορισμένο διάστημα εκτεινόμενο "από *το ρ στο* q", θέλουμε μερικές φορές να θεωρήσουμε τα σημεία *ρ και* q ότι συμπεριλαμβάνονται στο διάστημα και μερικές φορές όχι. Εάν θέλουμε να περιλάβουμε το σημείο *ρ*, λέμε την έκφραση "από *το ρ στο* q" είναι κλειστή στην αρχή της, διαφορετικά λέμε ότι είναι ανοικτή στην αρχή της. Επιπλέον, εάν θέλουμε να περιλάβουμε το σημείο q, λέμε ότι η έκφραση είναι κλειστή στο τέλος της, διαφορετικά λέμε ότι είναι ανοικτή στο τέλος της.

Συμβατικά, δηλώνουμε ένα διάστημα ενός ζεύγους σημείων *ρ και* q που χωρίζονται από μια άνω κάτω τελεία, όπου προηγείται μία αγκύλη ή μια παρένθεση ανοίγματος και ακολουθείται από μία αγκύλη ή μια παρένθεση κλεισίματος. Μία αγκύλη χρησιμοποιείται όταν θέλουμε μια κλειστή ερμηνεία, ενώ μια παρένθεση όπου θέλουμε μια ανοικτή. Παραδείγματος χάριν, υπάρχουν τέσσερις ευδιάκριτοι τρόποι να δειχτεί το συγκεκριμένο διάστημα με σημείο έναρξης την $4^{η}$ ημέρα και σημείο λήξης την $10^{η}$ ημέρα:

[d04:d10]
[d04:d11)
(d03:d10]
(d03:d11)

Μπορεί να σκεφτείτε ότι είναι περίεργο στη χρήση, το να λέτε μία αγκύλη ανοίγματος με μια παρένθεση κλεισίματος: η αλήθεια είναι ότι υπάρχουν καλοί λόγοι να επιτραπούν και οι τέσσερις μορφές. Πράγματι, αυτό που αποκαλείται κλειστό-ανοικτό στυλ (κλειστό στο άνοιγμα και ανοικτό στο τέλος, όπως το [d04:d11)) είναι αυτό που χρησιμοποιείται συχνότερα στην πράξη. Αλλά το κλειστό- κλειστό στύλ όπως το [d04:d10], είναι σίγουρα το πιο διαισθητικό.

ΣΗΜΕΙΩΣΗ: *Για να δούμε γιατί το κλειστό-ανοικτό στύλ έχει ίσως περισσότερα πλεονεκτήματα, σκεφτείται τη λειτουργία να χωρίσουμε το διάστημα [d04:d11) αμέσως πριν, την 7η ημέρα. Το αποτέλεσμα είναι το ζευγάρι κατάληξης των διαστημάτων [d04:d07) και [d07:d11).*

Τώρα, δεδομένου ότι ένα διάστημα όπως το [d04:d10) μπορεί να θεωρηθεί ως τιμή, έχει σαφώς νόημα να συνδυάσει τις ιδιότητες FROM και TO για καθεμιά από τις σχέσεις στο σχήμα 2.3 σε μια ενιαία ιδιότητα DURING, της οποίας οι τιμές προέρχονται από κάποιο τύπο διαστήματος. Το σχήμα 3.1 παρουσιάζει τι συμβαίνει στο τρέχων παράδειγμά μας, εάν υιοθετήσουμε αυτήν την προσέγγιση (παρατηρήστε τις αλλαγές ονόματος των σχέσεων).

Παράδειγμα 3.1
Β.Δ. προμηθευτών-αποστολών (Πλήρως χρονική έκδοση με την χρήση διαστημάτων

S_DURING

S#	DURING
S1	[d04:d10]
S2	[d02:d04]
S2	[d07:d10]
S3	[d03:d10]
S4	[d04:d10]
S5	[d02:d10]

SP_DURING

S#	P#	DURING
S1	P1	[d04:d10]
S1	P2	[d05:d10]
S1	P3	[d09:d10]
S1	P4	[d05:d10]
S1	P5	[d04:d10]
S1	P6	[d06:d10]
S2	P1	[d02:d04]
S2	P1	[d08:d10]
S2	P2	[d03:d03]
S2	P2	[d09:d10]
S3	P2	[d08:d10]
S4	P2	[d06:d09]
S4	P4	[d04:d08]
S4	P5	[d05:d10]

Ο ορισμός για την σχέση S_DURING είναι:

Από την ημέρα που είναι το σημείο έναρξης της DURING (και όχι της ημέρας αμέσως πριν από αυτήν την ημέρα) ως την ημέρα που είναι το σημείο λήξης της DURING (και όχι της ημέρας αμέσως μετά από αυτήν την ημέρα), ο προμηθευτής S# ήταν στο πλαίσιο της σύμβασης.

Και ο ορισμός για την σχέση SP _DURING είναι:

Από την ημέρα που είναι το σημείο έναρξης της DURING (και όχι της ημέρας αμέσως πριν από αυτήν την ημέρα) ως την ημέρα που είναι το σημείο λήξης της DURING (και όχι της ημέρας αμέσως μετά από αυτήν την ημέρα), ο προμηθευτής S# ήταν σε θέση να προμηθεύσει το μέρος P#.

Η ιδέα της αντικατάστασης του ζεύγους των ιδιοτήτων FROM *και* TO από τις ενιαίες ιδιότητες DURING σε καθεμιά από τις δυο σχέσεις, φέρνει διάφορα άμεσα πλεονεκτήματα. Εδώ είναι μερικά από αυτά:

- Αποφεύγει το πρόβλημα του να πρέπει να κάνει μια αυθαίρετη επιλογή ως προς *το* ποιο από τα δύο υποψήφια κλειδιά πρέπει να θεωρηθεί ως πρωτεύον. Παραδείγματος χάριν, η σχέση S_FROM_TO είχε δύο υποψηφία κλειδιά, {S#, FROM}, *και* {S#, TO} , αλλά η σχέση S_DURING έχει μόνο ένα, το {S#, DURING}, το οποίο μπορούμε επομένως να το υποδείξουμε ως πρωτεύον (εάν επιθυμούμε) χωρίς οποιαδήποτε ανεπιθύμητη αυθαιρεσία. Ομοίως, η σχέση SP _ FROM_TO είχε επίσης δύο υποψηφία κλειδιά αλλά η σχέση SP _DURING έχει ακριβώς το ένα, {S#, P#, DURING}, το οποίο πάλι μπορούμε να το υποδείξουμε ως πρωτεύον εάν επιθυμούμε.

- Αποφεύγει επίσης το πρόβλημα να πρέπει να αποφασίσει κατά πόσον τα διαστήματα FROM-TO στην προηγούμενη έκδοση της βάσης δεδομένων πρόκειται να ερμηνευτούν ως κλειστά ή ανοιχτά όσον αφορά το FROM *και το* TO. *Στο* 2^o κεφάλαιο αυτά τα διαστήματα θεωρήθηκαν ως κλειστά. Αλλά τώρα, *τα* [d04:d10]. [d04:d11), (d03:d10], *και* (d03:d11) είναι τέσσερις ευδιάκριτες πιθανές αντιπροσωπεύσεις του ίδιου διαστήματος, και δεν έχουμε καμία ανάγκη να ξέρουμε ποιο, ενδεχομένως είναι η πραγματική φυσική αντιπροσώπευση.

- Ακόμα ένα πλεονέκτημα είναι ότι οι περιορισμοί ακεραιότητας "για να προστατεύσουν ενάντια στο παραλογισμό ενός ζεύγους FROM-TO *που* εμφανίζεται, στο οποίο η τιμή TO είναι μικρότερη από την τιμή FROM" δεν είναι *πλέον* απαραίτητοι, επειδή ο περιορισμός "FROM ≤TO" υπονοεί την ίδια την έννοια ενός τύπου διαστήματος. Δηλαδή οι περιορισμοί της μορφής "FROM≤TO"αντικαθίστανται αποτελεσματικά από έναν γενικό περιορισμό που ισχύει για κάθε μεμονωμένο τύπο διαστήματος. Αυτοί οι μεμονωμένοι τύποι διαστήματος καθορίζονται με τη βοήθεια των επικλήσεων της γεννήτριας τύπων διαστήματος. Ο γενικός περιορισμός μπορεί να θεωρηθεί ότι συνδέεται με αυτήν την γεννήτρια τύπων, όπως επίσης και οι γενικοί τελεστές διαστήματος μπορούν να θεωρηθούν συνδεμένοι με αυτήν την γεννήτρια τύπων.

- Υποθέστε ότι οι σχέσεις r1 *και* r2 επρόκειτο να περιλάβουν ξεχωριστές ιδιότητες FROM *και* TO (αν και με διαφορετικά ονόματα σε κάθε περίπτωση), αντί μιας ενιαίας ιδιότητας DURING, και να υποθέσουμε ότι εμείς επρόκειτο να συνενώσουμε τα r1 *και* r2 για να παραγάγουμε το r3. Τότε *η* r3 θα περιέχε δύο ζευγάρια ιδιοτήτων FROM-TO και θα είναι στην ευθύνη του χρήστη, και όχι του συστήματος, να ταιριάξει τις ιδιότητες FROM *και* TO κατάλληλα. Σαφώς, αυτό το πρόβλημα θα γίνει χειρότερο δεδομένου ότι ο αριθμός των συνενώσεων αυξάνεται, και θα μπορούσε να προκαλέσει σοβαρά ανθρώπινα λάθη. Οι δυσκολίες θα αυξάνονταν εάν επρόκειτο να απορρίψουμε μερικά από τα FROMs ή/και τα TOs με τη βοήθεια των προβολών. Τέτοια προβλήματα δεν προκύπτουν ή καλύτερα, είναι πολύ λιγότερο αυστηρά με τις ιδιότητες DURING.

Όσον αφορά τους περιορισμούς και τα ερωτήματα που συζητούνται στο προηγούμενο κεφάλαιο, πρέπει να είναι σαφές ότι τα άμεσα ανάλογα εκείνων των περιορισμών και ερωτήσεων μπορούν να διατυπωθούν ενάντια στη βάση δεδομένων του σχήματος 3.1, εφ' όσον έχουμε έναν τρόπο να έχουμε πρόσβαση στα σημεία έναρξης και λήξης οποιουδήποτε δεδομένου διαστήματος.

Ένα τελευταίο σημείο: Πρέπει να τονίσουμε το γεγονός ότι τα διαστήματα όπως συζητούνται σε αυτό το κεφάλαιο είναι κλιμακωτές τιμές που δεν έχουν κανένα ορατό στο χρήστη συστατικό. (τα σημεία έναρξης και λήξης είναι συστατικά των πιθανών αντιπροσωπεύσεων των διαστημάτων, κι όχι συστατικά των διαστημάτων υπό αυτήν τη μορφή.)

3.2 Εφαρμογές των διαστημάτων

Η έννοια του διαστήματος είναι το κλειδί στην εξέταση των προβλημάτων που θίγονται στο προηγούμενο κεφάλαιο. Με άλλα λόγια, τα διαστήματα είναι η θεμελιώδης αφαίρεση που χρειαζόμαστε για να εξετάσουμε τα χρονικά στοιχεία ικανοποιητικά. Προτού ερευνήσουμε με λεπτομέρειες τα χρονικά διαστήματα υπό αυτήν τη μορφή, πρέπει να καταστήσουμε σαφές ότι η έννοια του διαστήματος είναι πραγματικά πολύ ευρύτερη εφαρμογή δηλαδή υπάρχουν πολλές άλλες εφαρμογές για τα διαστήματα, εφαρμογές στις οποίες τα διαστήματα δεν είναι απαραιτήτως χρονικά.

Εδώ είναι μερικά παραδείγματα:

- οι φορολογικές ταξινομήσεις αντιπροσωπεύονται από κλίμακες στα φορολογήσιμα εισοδήματα, δηλαδή διαστήματα των οποίων τα σημεία έναρξης και λήξης (και όλα τα ενδιάμεσα σημεία φυσικά) είναι τιμές χρημάτων.

- οι μηχανές δημιουργούνται για να λειτουργήσουν μέσα σε ορισμένη θερμοκρασία και τάση. Με άλλα λόγια, διαστήματα των οποίων τα σημεία που περιέχονται είναι οι θερμοκρασίες και οι τάσεις, αντίστοιχα.

- τα ζώα ποικίλλουν στο φάσμα των συχνοτήτων των κυμάτων του φωτός και του ήχου στα οποία τα μάτια και τα αυτιά τους είναι δεκτικά.

- τα διάφορα φυσικά φαινόμενα εμφανίζονται και μπορούν να μετρηθούν σύμφωνα με τις διακυμάωσεις του βάθους του χώματος ή της θάλασσας ή του ύψους επάνω από τη στάθμη της θάλασσας.

Όλα τα διαστήματα που συζητούνται μέχρι τώρα μπορούν να θεωρηθούν ως μονοδιάστατα. Εντούτοις, μπορούμε να συνδυάσουμε δύο μονοδιάστατα διαστήματα για να διαμορφώσουμε ένα δισδιάστατο διάστημα. Παραδείγματος χάριν, μια ορθογώνια πλοκή του εδάφους μπορεί να θεωρηθεί ως δισδιάστατο διάστημα, επειδή είναι εξ ορισμού ένα αντικείμενο με το μήκος και το πλάτος, καθένα από τα οποία είναι ένα μονοδιάστατο διάστημα που μετριέται κατά μήκος κάποιου άξονα. Και φυσικά,

μπορούμε να επεκτείνουμε αυτήν την ιδέα σε οποιοδήποτε αριθμό διαστάσεων. Παραδείγματος χάριν, ένα κτίριο μπορεί να θεωρηθεί ως τρισδιάστατο διάστημα: Είναι ένα αντικείμενο με το μήκος, το πλάτος, και το ύψος, ή με άλλα λόγια ένα κυβοειδή.

3.3 Τύποι σημείων και τύποι διαστημάτων

Αρχίζουμε με την εξέταση της τιμής διαστήματος [d04:d10], όπου θα αναφερόμαστε σ' αυτήν με την τιμή διαστήματος ι, ή ως διάστημα ι για συντομία. Σύμφωνα με τον τρέχοντα ορισμό μας, τα σημεία που αποτελούν το διάστημα ι (συγκεκριμένα, είναι το d04, d05..., *και* d10) είναι όλα ημέρες. Χάριν του παραδείγματος, υποθέτουμε ότι αυτά τα σημεία είναι όλες τιμές τύπου DATE, όπου ο τύπος DATE αντιπροσωπεύει τις γρηγοριανές ημερομηνίες. Κατόπιν ο τύπος DATE λέγεται ότι είναι ο τύπος σημείου για το διάστημα *ι*.

Αλλά πόσο ακριβώς ξέρουμε ότι τα σημεία στο διάστημα ι είναι αυτά που είπαμε ότι ήταν (δηλ., d04, d05..., d10); Ξέρουμε σίγουρα ότι το ι περιλαμβάνει τα σημεία έναρξης και λήξης d04 *και* d10, εξ ορισμού. Επίσης ξέρουμε ότι το ι αποτελείται από ένα σύνολο σημείων που τακτοποιούνται σύμφωνα με κάποια συμφωνηθείσα αρχή. Έτσι εάν πρόκειται να καθορίσουμε το πλήρες σύνολο σημείων στο ι, πρέπει αρχικά να καθορίσουμε το σημείο, και για απλότητα αναφερόμαστε σ' αυτό ως d04+1. Αυτό το σημείο d04 +1 είναι ο διάδοχος *του* d04 σύμφωνα με αυτήν την διαταγή, και η λειτουργία από την οποία αυτός ο διάδοχος καθορίζεται είναι η αντίστοιχη λειτουργία διαδόχων. Στην συγκεκριμένη περίπτωση, ο τύπος σημείου είναι DATE, η λειτουργία διαδόχων είναι βασικά η "επόμενη ημέρα" ("προσθέτουμε μια ημέρα στη δεδομένη ημερομηνία") δηλαδή είναι μια λειτουργία, στην οποία δίνοντας στην DATE μια τιμή d, επιστρέφει την τιμή DATE που είναι ο άμεσος διάδοχος *του* d σύμφωνα με τους κανόνες του γρηγοριανού ημερολογίου.

ΣΗΜΕΙΩΣΗ: *Εάν το d+1 είναι ο διάδοχος του d σε κάποια διαταγή, τότε το d είναι ο προκάτοχος d+1 σε αυτήν την ίδια διαταγή. Για απλότητα, αναφερόμαστε μερικές φορές στον προκάτοχο του d ως d -1.*

Έχοντας αποφασίσει ότι *το* d04 + 1 είναι ο διάδοχος του d04, πρέπει έπειτα να καθορίσουμε εάν *το* d04 + 1 έρχεται ή όχι μετά από το τελικό σημείο d10 σύμφωνα με αυτήν την διαταγή. Εάν όχι, τότε το d04 + 1 είναι πράγματι ένα σημείο *ι* = [d04:d10], *και* πρέπει τώρα να εξετάσουμε το επόμενο σημείο, το d04+2. Επαναλαμβάνουμε αυτή την διαδικασία έως ότου ερχόμαστε στο πρώτο σημείο d04+n (στην πραγματικότητα d04+7, ή με άλλα λόγια d11=d10+1) που έρχεται μετά από *το* d10, έως ότου ερχόμαστε "στην τελευταία ημέρα".

Γενικότερα, το διάστημα *ι*=[b:e],όπου *το* b *και το* e είναι πάλι τιμές του τύπου DATE, και η ίδια λειτουργία διαδόχων "επόμενης ημέρας" ισχύει. Τότε υπάρχουν, μερικές ειδικές περιπτώσεις που εξετάζουμε:

- b = "η πρώτη ημέρα" (δηλ, το σημείο που αντιστοιχεί "στην αρχή του χρόνου" που δεν έχει κανέναν προκάτοχο). Η έκφραση b-1 είναι απροσδιόριστη σε αυτήν την περίπτωση.
- e = "η τελευταία ημέρα" (δηλ, το σημείο που αντιστοιχεί "στο τέλος του χρόνου" που δεν έχει κανέναν διάδοχο). Η έκφραση e + 1 είναι απροσδιόριστη σε αυτήν την περίπτωση.

Ένας δεδομένος τύπος T είναι χρησιμοποιήσιμος ως τύπος σημείου εάν ορίζονται τα παρακάτω για αυτόν:

- *μια* συνολική διαταγή, σύμφωνα με την οποία ο τελεστής ">" (μεγαλύτερος από) καθορίζεται για κάθε ζευγάρι των τιμών v1 *και* v2 του τύπου T. Εάν *τα* v1 *και* v2 είναι ευδιάκριτα, τότε μια από τις εκφράσεις "v1 > v2" *και* "v2 > v1" επιστρέφει true (αληθή τιμή) και η άλλη επιστρέφει false (ψευδή). Όπως ξέρουμε από το κεφάλαιο 1, ο τελεστής "=" καθορίζεται βεβαίως για το T. Δεδομένου ότι το ">" καθορίζεται επίσης, μπορούμε νόμιμα να υποθέσουμε ότι όλοι οι συνηθισμένοι τελεστές σύγκρισης "=", "#", ">", " ≥", "<", και "≤" είναι διαθέσιμοι για όλα τα ζευγάρια των τιμών του τύπου T.

- Niladic "πρώτοι" και "τελευταίοι" τελεστές, οι οποίοι επιστρέφουν τη μικρότερη *και μεγαλύτερη* τιμή του T αντίστοιχα, σύμφωνα με την προαναφερθείσα διαταγή.

- μοναδιαίοι "επόμενοι" και "προγενέστεροι" τελεστές, που επιστρέφουν το διάδοχο και τον προκάτοχο αντίστοιχα, οποιασδήποτε δεδομένης τιμής του τύπου T σύμφωνα με την προαναφερθείσα διαταγή. Φυσικά, ο "επόμενος" τελεστής είναι η λειτουργία διαδόχων. Όπως επισημαίνεται ήδη, οι "επόμενοι" και "προγενέστεροι" τελεστές είναι απροσδιόριστοι εάν η δεδομένη τιμή του τύπου T είναι στην πραγματικότητα η "τελευταία" ή η "πρώτη" τιμή, αντίστοιχα, αυτού του τύπου.

Παρατηρούμε ότι ο κενός κλιμακωτός τύπος (αποκαλούμενος Omega) ικανοποιεί τις προηγούμενες απαιτήσεις και είναι έτσι ένας έγκυρος τύπος σημείου. Εντούτοις, εάν ο τύπος σημείου είναι κενός, τότε ο αντίστοιχος τύπος διαστήματος είναι απαραιτήτως επίσης κενός. Φυσικά, ο "πρώτος", ο "τελευταίος", ο "έπόμενος" και ο "προγενέστερος" τελεστής θα είναι απροσδιόριστος εάν ο τύπος σημείου είναι κενός.

Εδώ είναι δύο παραδείγματα τύπου διαστήματος:

- **INTERVAL_INTEGER**

Ο τύπος σημείου εδώ είναι INTEGER και η λειτουργία διαδόχων είναι "ο επόμενος INTEGER"(*δηλ*, "προσθέστε ένα"), και οι τιμές αυτού του τύπου διαστήματος είναι διαστήματα της μορφής [b:e] όπου b *και* e είναι τιμές του τύπου INTEGER *και* b ≤e

✪ **INTERVAL_MONEY**

Τα χρήματα εδώ είναι ένας τύπος που αντιπροσωπεύει τα νομισματικά ποσά που μετριούνται σε δολάρια και σεντς. Η λειτουργία διαδόχων είναι "προσθέτει ένα cent". Οι τιμές αυτού του τύπου διαστήματος είναι διαστήματα της μορφής [b:e], όπου b *και* e είναι τιμές *του* τύπου MONEY *και* b ≤e

3.4 Ένα παράδειγμα έρευνας

Οι σχέσεις S_DURING *και* SP_DURING και οι δύο περιλαμβάνουν μόνο μια ιδιότητα διαστήματος, που καλείται DURING. Εντούτοις, είναι δυνατό για μια σχέση να περιλαμβάνει δύο ή περισσότερες τέτοιες ιδιότητες. Παραδείγματος χάριν, υποθέστε ότι υπάρχει ένα σύνολο παραγγελιών στους αριθμούς μερών, για παράδειγμα P1< P2 < P3 (κ.λπ.), και υποθέστε ότι επιθυμούμε η βάση δεδομένων μας να δείχνει ότι ορισμένοι προμηθευτές ήταν σε θέση να προμηθεύουν ορισμένες σειρές μερών κατά τη διάρκεια ορισμένων διαστημάτων του χρόνου. Τότε η σχέση SP_DURING μπορεί να έχει δύο ιδιότητες διαστήματος, DURING *και* PARTS, όπου η DURING είναι όπως *πριν*, *και τα* PARTS δείχνουν τις αντίστοιχες σειρές μερών. Για να αποφύγουμε τυχόν σύγχυση θα αναφερόμαστε σ'αυτήν την αναθεωρημένη έκδοση της σχέσης ως S_PARTS_DURING αντί SP _DURING.

Παράδειγμα 3.2
Μια σχέση με δυο Ιδιότητες διαστήματος

S_PARTS_DURING

S#	PARTS	DURING
S1	[P1:P3]	[d01:d04]
S1	[P2:P4]	[d07:d08]
S1	[P5:P6]	[d09:d09]
S2	[P1:P1]	[d08:d09]
S2	[P1:P2]	[d08:d08]
S2	[P3:P4]	[d07:d08]
S3	[P2:P4]	[d01:d04]
S3	[P3:P5]	[d01:d04]
S3	[P2:P4]	[d05:d06]
S3	[P2:P4]	[d06:d09]
S4	[P3:P4]	[d05:d08]

Παρατηρήστε το σχήμα 3.2

Σημειώστε ότι οι τιμές δειγμάτων στο παρακάτω παράδειγμα δεν αντιστοιχούν στις τιμές δειγμάτων για την σχέση SP_DURING του παραδείγματος 3.1. Πράγματι, μπορεί να

έχετε παρατηρήσει ότι οι τιμές δειγμάτων του παραδείγματος 3.2 υποφέρουν από ορισμένα προβλήματα. Για παράδειγμα μας λέει δύο φορές ότι ο προμηθευτής S3 ήταν σε θέση να προμηθεύσει το μέρος P4 *απ' την* $1^{η}$ *ως την* $4^{η}$ ημέρα.

Κάποιες παρατηρήσεις για να κλείσουμε αυτή την ενότητα:

- παρόλο που η σχέση S_PARTS_DURING περιλαμβάνει δύο ιδιότητες διαστήματος, μόνο μια από τις δύο περιλαμβάνει τα χρονικά διαστήματα. Εδώ σε αντίθεση σας παρουσιάζουμε μία σχέση με δύο ευδιάκριτες ιδιότητες διαστήματος, και οι δύο από τις οποίες αντιπροσωπεύουν τα χρονικά διαστήματα

EMP {EMP#, PRIMARY, SECONDARY}

Εδώ οι ιδιότητες PRIMARY *και* SECONDARY παρουσιάζουν τα διαστήματα του χρόνου κατά τη διάρκεια των οποίων ο υπάλληλος που προσδιορίζεται από το emp # έλαβε την πρωτοβάθμια και δευτεροβάθμια εκπαίδευση του.

- Δεύτερον, θα πρέπει να εξετάσουμε και τις σχέσεις που είναι τιμές σχέσεων *με* δύο ή περισσότερες ιδιότητες διαστήματος, ακόμα κι αν δεν είχαμε καμιά σχέση (όπως το emp ή S_PARTS_DURING) με δύο ή περισσότερες τέτοιες ιδιότητες. Παραδείγματος χάριν, μόλις συνενώσουμε τις δύο σχέσεις R1{A, B} *και* R2{A, C}, όπου B *και* C είναι ιδιότητες διαστήματος, λαμβάνουμε μια σχέση που έχει δύο ιδιότητες διαστήματος.

- Τρίτον, θα μπορούσαμε να επεκτείνουμε το παράδειγμα S_PARTS_DURING *για* να περιλάβουμε τρεις ιδιότητες διαστήματος με την αντικατάσταση της ιδιότητας S# από μια ιδιότητα SUPPLIERS που δείχνει τον αριθμό σειράς του προμηθευτή.

SUPPLIERS	PARTS	DURING
[S1:S2]	[P2:P3]	[d03:d04]
[S2:S3]	[P3:P4]	[d04:d05]

3.5 Τελεστές Διαστημάτων

Σε αυτό το σημείο εισάγουμε διάφορους χρήσιμους τελεστές που απευθύ*νονται στις* τιμές διαστημάτων. Έχουμε πάρει την "άδεια" να αλλάξουμε πολλά απ' τα ονόματά τους, σε ονόματα που βρίσκουμε καλύτερα για κάποιο λόγο.

Με σκοπό την εξήγηση των τελεστών, υιοθετούμε την ίδια άτυπη σημείωση:

- Πρώτα, έστω ότι T είναι ένα σημείο type, κι έστω ότι p είναι η τιμή του type T. Κατόπιν χρησιμοποιούμε τις εκφράσεις p+1, p+2 για να δείξουμε την τιμή που είναι διάδοχος *του* p, τη τιμή που είναι διάδοχος του p+1, και τα λοιπά. Φυσικά, η προηγούμενη σημείωση είναι άτυπη. Μια πραγματική γλώσσα θα έπρεπε να παρέχει ένα είδος σαφή "επόμενου" τελεστή. Όταν πρέπει να αναφερθούμε σε

εκείνο τον τελεστή, θα το καλέσουμε NEXT_T. Έτσι το NEXT_T(p) επιστρέφει p+1, το NEXT_T(NEXT_T(p)) επιστρέφει p+2, και τα λοιπά. Παρατηρήστε ότι ο επίσημος "επόμενος" τελεστής περιλαμβάνει έναν σαφή "_T" πιστοποιητή.

- χρησιμοποιούμε επίσης (πάλι ανεπίσημα) τις εκφράσεις p-1, p-2 κτλ, για να δείξουμε την τιμή της οποίας ο διάδοχος είναι p, η τιμή της οποίας ο διάδοχος είναι p-1 και τα λοιπά.
  Μια πραγματική γλώσσα θα έπρεπε να παρέχει έναν σαφή προγενέστερο (prior) τελεστή, τον οποίο θα καλέσουμε PRIOR_T. Το PRIOR_T (p) επιστρέφει p-1, PRIOR_T(PRIOR_T(p)) επιστρέφει p-2, και τα λοιπά.
- χρειαζόμαστε επίσης τους τελεστές FIRST_T και LAST_T. Οι τελεστές FIRST_TO και LAST_TO επιστρέφουν την "πρώτη" και την "τελευταία" τιμή του τύπου T αντίστοιχα.
- ο τύπος διαστήματος που αντιστοιχεί στο σημείο τύπου T είναι INTERVAL_T. Χρησιμοποιούμε την έκφραση [pl:pn] για να δηλώσουμε το διάστημα του οποίου περιείχε τα σημεία είναι p1, p1+1, pl+2... , pn (1 ≤n). Παρατηρήστε ότι η έκφραση [pl:pn] μπορεί να θεωρηθεί ως άτυπο παράδειγμα μιας επίκλησης επιλογέων διαστήματος. Μια πραγματική γλώσσα θα έπρεπε να παρέχει κάποιο είδος σαφής σύνταξης για τέτοιες επικλήσεις, όπως για παράδειγμα INTERVAL_T ([pl:pn]).
- χρησιμοποιούμε περιστασιακά τις άλλες άτυπες μορφές για τα διαστήματα (κλειστό-ανοικτό, ανοικτό-κλειστό και ανοικτό-ανοικτό). Έστω ότι το σημείο είναι INTEGER(ακέραιος), και θυμηθείτε τον αντίστοιχο τύπο διαστήματος INTERVAL_INTEGER. Κατόπιν οι εκφράσεις [3:5], [3:6), (2:5], και (2:6) δηλώνουν όλες ανεπίσημα σχεδόν το ίδιο διάστημα. Πιο συγκεκριμένα, αυτό το διάστημα του οποίου τα περιεχόμενα σημεία είναι ακριβώς 3, 4 και 5. Ανάλογα τα επίσημα αυτών των εκφράσεων είναι INTERVAL_INTEGER([3:5), INTERVAL_INTEGER([3:6)), INTERVAL_INTEGER((2:5]) και INTERVAL_INTEGER ((2:6)) αντίστοιχα.

Σας υπενθυμίζουμε επίσης τους τελεστές BEGIN, END και ∈ από το κεφάλαιο 2. Επαναλαμβάνουμε τους ορισμούς εδώ. Έστω ότι ι είναι το διάστημα [b:e] του τύπου INTERVAL_T κι έστω ότι p είναι μια τιμή του τύπου T. Κατόπιν:

- Το BEGIN(i) επιστρέφει b
- Το END(i) επιστρέφει e
- το P ∈ i επιστρέφει true εάν και μόνο εάν τα b ≤p και p ≤e επιστρέφουν και τα δυο true

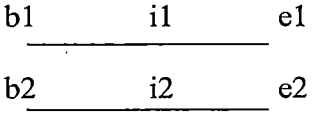
Επίσης ορίζουμε τους τελεστές PRO(i) και POST(i), οι οποίοι επιστρέφουν b-1 και e+1 αντίστοιχα, και το τελεστή i ∃ p (που διαβάζεται "το i περιέχει το p"), το οποίο επιστρέφει true εάν και μόνο εάν το p ∈ I επιστρέφει true.

3.6 Τελεστές σύγκρισης

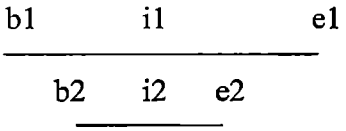
Ποικιλία τελεστών μπορεί να οριστεί για τη δοκιμή εάν δύο διαστήματα είναι ίσα ή όχι. Σε αυτή την ενότητα περιγράφουμε μερικούς από τους πιο χρήσιμους τελεστές αυτού του είδους, που δίνουν σε κάθε περίπτωση έναν επίσημο ορισμό μαζί με μια διαισθητική εικόνα για να επεξηγήσουμε τη λειτουργία. ΣΗΜΕΙΩΣΗ: Οι τελεστές που συζητούνται είναι γνωστοί συλλογικά ως τελεστές Allen, οι περισσότεροι εκ των οποίων είχαν αναφερθεί πρώτα απ' τον Allen.

Εδώ και σε όλο το υπόλοιπο αυτού του κεφαλαίου, παίρνουμε τα i1 *και* i2 *για να* δείχνουν τα διαστήματα [b1:e1] *και* [b2:e2] αντίστοιχα, και τα δύο του ίδιου τύπου INTERVAL_T. Σημειώστε ότι για να ορίζονται οι διάφοροι τελεστές, τα δύο διαστήματα πρέπει να είναι του ίδιου τύπου διαστήματα.

Equals (ίσα =): τα διαστήματα i1 *και* i2 είναι ίσα εάν και μόνο εάν τα b1=b2 *και* e1=e2 είναι ταυτόχρονα true

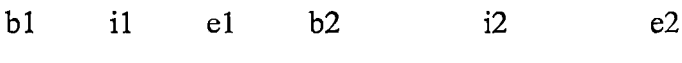


Includes (περιλαμβάνει ⊇) *και* **included in** (*περιλαμβάνεται σε* ⊆): *το* i1 ⊇ i2 είναι true εάν και μόνο εάν τα b1 ≤ b2 *και* e1 ≥ e2 είναι ταυτόχρονα true.

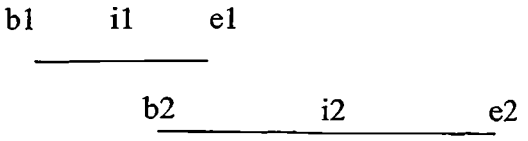


BEFORE (*πριν*) *και* **AFTER** (μετά): *το* i1 BEFORE i2 είναι true εάν και μόνο εάν το e1 < b2 είναι αληθινό. Το i2 AFTER i1 είναι true εάν *και* μόνο εάν *το* i1 BEFORE i2 είναι true

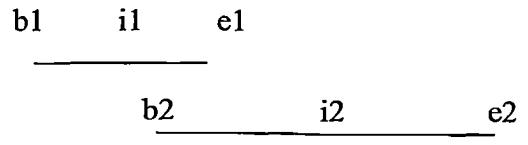
MEETS (συναντά): *το* i1 MEETS i2 είναι true εάν και μόνο εάν το b2=e1+1 είναι true ή *το* b1=e2+1 είναι true.



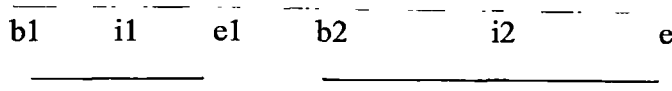
OVERLAPS (επικάλυψη): το i1 OVERLAPS i2 είναι true εάν και μόνο εάν τα b1 ≤ e2 *και* b2 ≤ e1 είναι ταυτόχρονα true



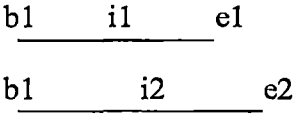
MERGES (συγχώνευση): *το* i1 MERGES i2 είναι true, εάν και μόνο εάν το i1 OVERLAPS i2 είναι true



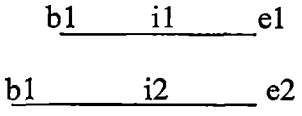
Ή



BEGINS: *το* i1 BEGINS i2 είναι true εάν *και* μόνο εάν *τα* b1=b2 *και* e1≤e2 είναι *ταυτόχρονα* true.



ENDS: *το* i1 ENDS i2 είναι true εάν και μόνο εάν τα e1=e2 *και* b1≥b2 είναι ταυτόχρονα true.



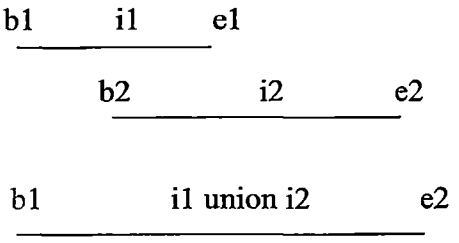
3.7 Άλλοι τελεστές

Ο τελεστής COUNT (i) επιστρέφει μια αρίθμηση του αριθμού σημείων στο διάστημα *ι*. Για παράδειγμα εάν *το ι* είναι το διάστημα [d03:d07] *του* τύπου INTERVAL_DATE, τότε το COUNT (i) είναι 5. Να σημειώσουμε ότι η λήξη DURATION πολλές φορές χρησιμοποιείται στη θέση του COUNT. Προτιμούμε το COUNT γιατί είναι πιο ουδέτερο.

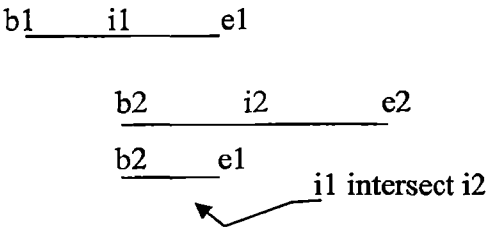
Έστω ότι p1 *και* p2 είναι τιμές του τύπου T. Τότε ορίζουμε ότι το max(pl, p2) επιστρέφει p2 εάν *το* p1<p2 είναι true αλλιώς επιστρέφει p2. Το min(p1, p2) επιστρέφει p1 εάν *το* p1<p2 είναι true αλλιώς επιστρέφει p2. Χρησιμοποιούμε αυτούς τους τελεστές για τον ορισμό ορισμένων χρήσιμων δυαδικών τελεστών στα διαστήματα (UNION, INTERSECT, MINUS). Καθένας από αυτούς τους τελεστές παίρνει δύο διαστήματα του

ίδιου τύπου με τους τελεστέους του και επιστρέφει ένα άλλο διάστημα του ίδιου τύπου σαν αποτέλεσμά του.

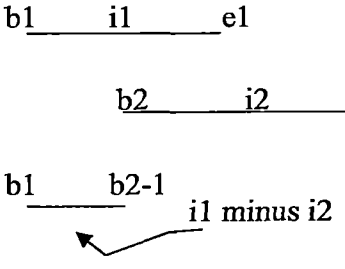
UNION (ένωση): *το* i1 UNION *ι2* επιστρέφει [MIN (b1, b2): MAX (e1, e2)] εάν i1 MERGES i2 είναι true *και* είναι απροσδιόριστο.



INTERSECT: *το* i1 INTERSECT i2 επιστρέφει [MAX (b1, b2): MIN (e1, e2)] εάν i1 OVERLAPS i2 είναι true και είναι απροσδιόριστο.



MINUS: *το* i1 MINUS i2 επιστρέφει [b1: MIN (b2-1, e1)] *αν τα* b1<b2 *και* e1≤e2 είναι ταυτόχρονα true, [MAX (e2+1), b1):e1] *αν τα* b1≥b2 και e1>e2 είναι ταυτόχρονα true



Στις προηγούμενες ενότητες αντιμετωπίσαμε μια ποικιλία από γενικούς κλιμακωτούς τελεστές, που ισχύουν για τα διαστήματα. Σ' αυτό το σημείο θα συναντήσουμε δύο ακόμα γενικούς τελεστές, τους οποίους καλούμε EXPAND(επέκταση) και COLLAPSE(συντόμευση). Αντίθετα από εκείνους τους πρώτους τελεστές, οι EXPAND *και* COLLAPSE δεν είναι κλιμακωτοί. Ισχύουν για τα σύνολα διαστημάτων, αντί των μεμονωμένων διαστημάτων και παράγουν ένα άλλο τέτοιο σύνολο διαστημάτων σαν

αποτέλεσμά τους. Πιο συγκεκριμένα, παίρνουν ένα σύνολο διαστημάτων του ίδιου τύπου με την είσοδό τους, και επιστρέφουν ένα άλλο σύνολο διαστημάτων του ίδιου τύπου, σαν αποτέλεσμά τους. Σε κάθε περίπτωση, αυτό το αποτέλεσμα μπορεί να θεωρηθεί ως κανονική μορφή για το εισαγόμενο σύνολο.

3.8 Η Μορφή EXPANDED

Όπως ήδη έχει λεχθεί, τα αντικείμενα που επιθυμούμε να μελετήσουμε είναι σύνολα διαστημάτων, όπου όλα τα εν λόγω διαστήματα είναι του ίδιου τύπου. Έστω ότι X1 *και* X2 είναι δύο τέτοια σύνολα. Κατόπιν καθορίζουμε την απαραίτητη έννοια της ισοδυναμίας ως εξής:

Τα X1 *και* X2 είναι ισοδύναμα εάν και μόνο εάν το σύνολο όλων των σημείων που περιλαμβάνονται στα διαστήματα X1 είναι ίσο με το σύνολο όλων των σημείων που περιλαμβάνονται στο διάστημα X2.

Έστω ότι X1 *και* X2 είναι τα σύνολα

{[d01:d01], [d03:d05], [d04:d06]}

και

{[d01:d01], [d03:d04], [d05:d05], [d05:d06]}

αντίστοιχα

Τα σύνολα X1 *και* X2 δεν είναι ίσα. Εντούτοις, είναι εύκολο να φανεί ότι είναι τουλάχιστον ισοδύναμα, επειδή το σύνολο όλων των σημείων p, όπου p είναι τα σημεία που περιλαμβάνονται στο διάστημα X1 είναι ίσο με το σύνολο των σημείων p, όπου p είναι τα σημεία που περιλαμβάνονται στο διάστημα X2. Το σύνολο των σημείων είναι

{d01, d03, d04, d05, d06}

Για λόγους που θα γίνουν σύντομα προφανείς, δεν ενδιαφερόμαστε τόσο για το σύνολο σημείων, αλλά για το αντίστοιχο σύνολο μονάδων διαστημάτων, έστω το X3 που βλέπουμε παρακάτω:

{[d01:d01], [d03:d03], [d04:d04], [d05:d05], [d06:d06]}

το X3 είναι ισοδύναμο με καθένα από τα X1 *και* X2. Λέμε ότι είναι η *επεκταθείσα* μορφή(expanded form) για καθένα απ' αυτά τα σύνολα. Γενικότερα, εάν το X είναι ένα σύνολο διαστημάτων του ίδιου τύπου, τότε *η επεκταθείσα μορφή* του X είναι ένα σύνολο από διαστήματα (ακριβέστερα, το σύνολο όλων των διαστημάτων μονάδων) της μορφής [p:p], όπου *το* p είναι ένα σημείο κάποιου διαστήματος X. Δίνοντας αυτό τον ορισμό, πρέπει να γίνει σαφές ότι:

(1)Δίνοντας κάθε σύνολο διαστημάτων X, όλα του ίδιου τύπου, η αντίστοιχη επεκταθείσα μορφή του X υπάρχει πάντα
(2)Η επεκταθείσα μορφή είναι ισοδύναμη με το X
(3)Η επεκταθείσα μορφή είναι μοναδική

Σημειώστε ότι εάν το X είναι κενό, κατόπιν η επεκταθείσα μορφή του X είναι επίσης κενή. Η επεκταθείσα μορφή του X είναι μια πιθανή κανονική μορφή για το X. Η επεκταθείσα μορφή του X επιτρέπει σε μας για να εστιάσουμε στις πληροφορίες που περιέχονται στο X σε ατομικό επίπεδο, χωρίς να πρέπει να ανησυχεί για τους πολλούς τρόπους με τους οποίους εκείνες οι πληροφορίες να συσσωρευτούν σε "μάζες".
Σημειώστε ότι η αρχή της επεκταθείσας μορφής μας επιτρέπει να επαναορίσουμε τον ορισμό της ισότητας πιο περιληπτικά:

Δυο σύνολα διαστημάτων είναι ίσα εάν και μόνο εάν έχουν την ίδια επεκταθείσα αρχή.

3.9 Η Μορφή COLLAPSE

Τα σύνολα X1, X2 *και* X3 που συζητήθηκαν στην προηγούμενη ενότητα, έχουν όλα διαφορετικό αριθμό στοιχείων συνόλου. Στην πραγματικότητα, συμβαίνει σ' εκείνο το συγκεκριμένο παράδειγμα το X3 (η επεκταθείσα μορφή) να είναι αυτό του οποίου ο αριθμός στοιχείων συνόλου να είναι ο μέγιστος. Εντούτοις, είναι εύκολο να βρούμε ένα άλλο σύνολο X4 που να έχει την ίδια επεκταθείσα μορφή, αλλά να έχει αριθμό στοιχείων συνόλου μεγαλύτερο από αυτόν του X3. Ένα τέτοιο πιθανό σύνολο X4 (όχι όμως και μοναδικό) είναι το παρακάτω:

{[d01:d01], [d03, d03], [d03, d04], [d03, d05], [d03, d06],
[d04, d04], [d04, d05], [d04, d06]}

Είναι επίσης εύκολο να βρεθεί ένα σύνολο X5 που έχει την ίδια επεκταθείσα μορφή και τον ελάχιστο πιθανό αριθμό στοιχείων συνόλου:

{[d01: d01], [d03: d06]}

Το X5 λέγεται ότι είναι η *συντομευμένη* μορφή του X1 (και επίσης του X2, X3, *και* X4). Γενικότερα, εάν το X είναι ένα σύνολο διαστημάτων όλα του ίδιου τύπου, τότε η *συντομευμένη* μορφή του X είναι το σύνολο Y των διαστημάτων του ίδιου τύπου έτσι ώστε:

- Το X *και το* Υ *να* έχουν την ίδια επεκταθείσα μορφή.
- Δε θα πρέπει να υπάρχουν δυο ευδιάκριτα διαστήματα i1 *και* i2 *στο* Υ ώστε το i1 MERGES i2 να είναι true.(θυμίζουμε ότι merges σημαίνει συγχώνευση. Ισοδύναμα, δε θα πρέπει να υπάρχουν δυο ευδιάκριτα διαστήματα i1 *και* i2 στο Y ώστε να ορίζεται *το* i1 UNION i2.

Προκύπτει από αυτό το τελευταίο σημείο ότι *το* Υ μπορεί να υπολογιστεί από το X, από διαδοχικές αντικαταστάσεις ζευγαριών διαστημάτων στο X, από την ένωσή τους έως ότου καμιά περαιτέρω αντικατάσταση να μην είναι δυνατή. Δε θα πρέπει να υπάρχουν δυο ευδιάκριτα διαστήματα i1 *και* i2 στο Y ώστε το i1 INTERSECT i2 να ορίζεται.
Η *συντομευμένη* μορφή του X είναι άλλη μια πιθανή κανονική μορφή για το X. Είναι το *μοναδικό* ισοδύναμο σύνολο που έχει τον ελάχιστο πιθανό αριθμό στοιχείων συνόλου.
Σημειώστε ότι πολλά ευδιάκριτα σύνολα μπορούν να έχουν την ίδια *συντομευμένη* μορφή. Επίσης, πρέπει να είναι σαφές ότι:

(1)Δίνοντας κάθε σύνολο διαστημάτων Χ, όλα του ίδιου τύπου, η αντίστοιχη *συντομευμένη* μορφή του Χ υπάρχει πάντα
(2)Η *συντομευμένη* μορφή είναι ισοδύναμη με το Χ
(3)Η *συντομευμένη* μορφή είναι μοναδική

Σημειώστε ότι εάν το Χ είναι κενό, κατόπιν η συντομευμένη μορφή του Χ είναι επίσης κενή. Η ακόλουθη πρόταση είναι επίσης σωστή:

Δυο σύνολα διαστημάτων είναι ίσα εάν και μόνο εάν έχουν την ίδια συντομευμένη αρχή.

3.10 Λειτουργίες στα σύνολα διαστημάτων

Τώρα μπορούμε να ορίσουμε τους τελεστές EXPAND *και* COLLAPSE. Έστω ότι Χ είναι ένα σύνολο διαστημάτων, όλα του ίδιου τύπου. Το EXPAND(X) επιστρέφει την επεκταθείσα μορφή του Χ και το COLLAPSE(X) επιστρέφει *την συντομευμένη μορφή* του Χ. Σημειώστε πιο συγκεκριμένα ότι:

α) αν ο αριθμός στοιχείων συνόλου του Χ είναι 0, τότε το αποτέλεσμα είναι 0 *και για το* EXPAND *και για το* COLLAPSE
β) αν ο αριθμός στοιχείων συνόλου του Χ είναι 1, το αποτέλεσμα είναι ίδιο *για το* COLLAPSE, αλλά όχι και για το EXPAND

Μην κάνετε το λάθος να πιστεύετε ότι οι τελεστές EXPAND *και* COLLAPSE είναι ο ένας αντίστροφος του άλλου. Για παράδειγμα έστω ότι Χ είναι το σύνολο

{[d01, d01], [d03, d05], [d04, d06]}

Αν επεκτείνουμε αυτό το σύνολο και μετά συντομεύσουμε το αποτέλεσμα, τότε *θα* έχουμε τη *συντομευμένη* μορφή Χ5:

{[d01, d01], [d03, d06]}

Αν συντομεύσουμε το αρχικό σύνολο Χ1 και μετά επεκτείνουμε το αποτέλεσμα, θα έχουμε την επεκταθείσα μορφή Χ3:

{[d01, d01], {[d03, d03], {[d04, d04], {[d05, d05], {[d06, d06]}

Με άλλα λόγια, ούτε το EXPAND(COLLAPSE(X)), αλλά ούτε το COLLAPSE(EXPAND(X)) είναι ταυτόσημα με το Χ. Βλέπουμε όμως ότι ισχύουν τα ακόλουθα:

EXPAND (COLLAPSE(X)) ≡ EXPAND(X)
COLLAPSE (EXPAND(X)) ≡ COLLAPSE(X)

Ας δώσουμε τώρα τους επίσημους ορισμούς των τελεστών EXPAND *και* COLLAPSE. Ας ξεκινήσουμε με τον τελεστή EXPAND.
Έστω ότι Χ είναι ένα σύνολο διαστημάτων όλα του ίδιου τύπου, του τύπου IT, κι έστω

ότι i *και* j είναι διαστήματα αυτού του τύπου (IT). Έστω επίσης ότι i= [b: e]. Τότε έχουμε:

EXPAND (X) = {i: b=e AND
(EXISTS j ∈ x) (b ∈ j)}

Με άλλα λόγια, το EXPAND(X) είναι ένα σύνολο διαστημάτων i του τύπου IT ώστε
α) τα αρχικά και τελικά σημεία του i είναι ίσα
β) το σημείο της ερώτησης περιέχεται σε τουλάχιστον ένα διάστημα στο X

Ο τελεστής COLLAPSE είναι πιο περίπλοκος. Έστω πάλι ότι X είναι ένα σύνολο διαστημάτων, όλα του ίδιου τύπου IT. Έστω ότι το βασικό σημείο τύπου είναι T, τα i, i1, i2, j, k, είναι διαστήματα του τύπου IT και το t είναι σημείο τύπου T. Έστω ακόμα ότι i=[b:e], i1=[b1:e1], i2=[b2:e2]. Τότε έχουμε:

```
0  COLLAPSE (X) =
1       {i: (EXISTS i1 ∈ x)
2          ((EXISTS i2 ∈ x)
3             (b = b1 AND e = e2 AND b1 ≤ b2 AND e1 ≤ e2 AND
4                (FORALL t < b)
5                   (NOT (EXISTS j ∈ x)
6                      (PRIOR_T (b) ∈ j))
7             AND
8                (FORALL t > e)
9                   (NOT (EXISTS j ∈ x)
10                     (NEXT_T (e) ∈ j))
11            AND
12                (FORALL t)
13                  (IF t > e1 AND t < b2 THEN
14                     (EXISTS k ∈ x) (t ∈ k)
15                        END IF)
16                )
17          )
18      }
```

ΕΞΗΓΗΣΗ: Σημειώστε πρώτα ότι αν το i εμφανίζεται στο COLLAPSE (X), τότε το αρχικό σημείο του i θα πρέπει να είναι το αρχικό σημείο τουλάχιστον ενός διαστήματος i1 στο X, και το τελικό σημείο του i θα πρέπει να είναι το τελικό σημείο τουλάχιστον ενός διαστήματος i1 στο X (δείτε τις γραμμές 1-3). Οι γραμμές 4-6 εγγυώνται ότι δεν υπάρχει κανένα διάστημα στο X που περιέχει το b-1 *του* BEGIN(i).
Επιπλέον, οι γραμμές 8 μέχρι 10 εγγυώνται ότι δεν υπάρχει κανένα διάστημα στο X που περιέχει το e+1 *του* END(i). Τέλος, οι γραμμές 12-15 εγγυώνται ότι αν υπάρχει ένα "κενό" ανάμεσα στο i1 *και* i2 (αν δηλαδή το i1 MEETS i2 είναι true), τότε κάθε σημείο σ' αυτό το κενό περιέχεται σε κάποιο διάστημα του X, και γι' αυτό το λόγο περιέχεται *στο* i. Ας δούμε τώρα κάποιες ερωτήσεις:

(1)Τι συμβαίνει σύμφωνα με τον ορισμό αν το X είναι άδειο;
Απ: το COLLAPSE(X) είναι ίσο με το X σ' αυτή τη περίπτωση
(2)Τι συμβαίνει εάν το X περιέχει μόνο ένα διάστημα;
Απ: το COLLAPSE(X) είναι ίσο με το X και σ' αυτή τη περίπτωση

Σ' αυτό το σημείο θα εισαχθούν και θα περιγραφούν ορισμένοι σχεσιακοί τελεστές που στηρίζονται στους τελεστές COLLAPSE(συντόμευση) και EXPAND(επέκταση) που συζητήθηκαν στο προηγούμενο κεφάλαιο. Οι εν λόγω τελεστές καλούνται PACK(συμπίεση) και UNPACK(αποσυμπίεση).
Τα ακόλουθα παραδείγματα θα σας βοηθήσουν να καταλάβετε τις λεπτομερείς συζητήσεις που εμφανίζονται στα επόμενα τμήματα. Υποθέστε ότι η σχέση r μοιάζει με αυτό:

S#	DURING
S2	[d02:d04]
S2	[d03:d05]
S4	[d02:d05]
S4	[d04:d06]
S4	[d09:d10]

Συμπιέζω αυτήν την σχέση "στο DURING" την οποία θα εκφράσουμε τυπικά ως PACK r ON DURING δίνει:

S#	DURING
S2	[d02:d05]
S4	[d02:d06]
S4	[d09:d10]

Ανεπίσημα, αυτό το αποτέλεσμα αντιπροσωπεύει τις ίδιες πληροφορίες με την αρχική σχέση r, αλλά συμπιεσμένο ή ρυθμισμένο εκ νέου κατά τέτοιο τρόπο ώστε δύο διαστήματα DURING για έναν δεδομένο προμηθευτή να μην συναντιούνται ούτε να επικαλύπτονται. Το αποτέλεσμα αυτής της συμπίεσης είναι να μας αφήσει να δούμε το περιεχόμενο των πληροφοριών του r σε μια συμπαγή μορφή, χωρίς να πρέπει να ανησυχούμε για την πιθανότητα ότι οι ευδιάκριτες μάζες μπορεί να συναντηθούν ή να επικαλυφθούν. Στην πραγματικότητα, η αρχική σχέση και η συμπιεσμένη έκδοση είναι ισοδύναμες από μία άποψη, αυτό θα το εξηγήσουμε στο τελικό τμήμα αυτού του κεφαλαίου.
Ανάλογα, αποσυμπιέζοντας την ίδια αρχική σχέση r "στο DURING" την οποία θα εκφράσουμε τυπικά ως UNPACK r ON DURING, δίνει:

S#	DURING
S2	[d02:d02]
S2	[d03:d03]
S2	[d04:d04]
S2	[d05:d05]
S4	[d02:d02]
S4	[d03:d03]
S4	[d04:d04]
S4	[d05:d05]
S4	[d06:d06]
S4	[d09:d09]
S4	[d10:d10]

Ανεπίσημα, αυτό το αποτέλεσμα αντιπροσωπεύει επίσης τις ίδιες πληροφορίες με *την* αρχική σχέση r, αλλά αποσυμπιεσμένες ή ρυθμισμένες εκ νέου κατά τέτοιο τρόπο ώστε κάθε τιμή DURING να είναι ένα διάστημα μοναδικό. Το αποτέλεσμα αυτής της αποσυμπίεσης είναι να δούμε το περιεχόμενο των πληροφοριών *του* r σε ατομικό επίπεδο, χωρίς να πρέπει να ανησυχούμε για τους πολλούς τρόπους με τους οποίους αυτές οι πληροφορίες μπορούν να συσσωρευτούν σε μάζες (η αρχική σχέση είναι ισοδύναμη με την αποσυμπιεσμένη έκδοση).

Στα επόμενα δύο τμήματα, θα εξηγήσουμε τους PACK *και* UNPACK τελεστές *λεπτομερώς*. Οι εξηγήσεις μας είναι βασισμένες στα παραδείγματα που είναι στη συνέχεια βασισμένα στα ερωτήματα A και B *που συναντήσαμε* στο τέλος του 2^{ου} κεφαλαίου. Για ευκολία, επαναλαμβάνουμε αυτά τα ερωτήματα:

- **QUERY A:** Πάρτε τις τριάδες S#-FROM-TO *για* τους προμηθευτές που ήταν σε θέση να προμηθεύσουν τουλάχιστον ένα μέρος κατά τη διάρκεια τουλάχιστον ενός διαστήματος του χρόνου, όπου το FROM *και* TO μαζί υποδεικνύουν ένα μέγιστο διάστημα κατά τη διάρκεια του οποίου ο προμηθευτής S# ήταν στην πραγματικότητα ικανός να προμηθεύσει τουλάχιστον ένα μέρος. ΣΗΜΕΙΩΣΗ: Χρησιμοποιούμε εδώ τον όρο "μέγιστο" ως κατάλληλη στενογραφία που σημαίνει ότι ο προμηθευτής S# ήταν ανίκανος να προμηθεύσει οποιοδήποτε μέρος την ημέρα αμέσως πριν την FROM ή αμέσως μετά *την* TO. Να σημειώσουμε επίσης ότι το αποτέλεσμα της ερώτησης μπορεί να περιέχει διάφορες πλειάδες για τον ίδιο προμηθευτή (αλλά με διαφορετικά διαστήματα, φυσικά, αυτά τα διαστήματα ούτε θα συνορεύουν ούτε θα επικαλύπτονται).

- **QUERY B:** Πάρτε τις τριάδες S#-FROM-TO για τους προμηθευτές που ήταν ανίκανοι να προμηθεύσουν οποιαδήποτε μέρη κατά τη διάρκεια τουλάχιστον ενός διαστήματος του χρόνου, όπου *το* FROM *και* TO μαζί υποδεικνύουν ένα μέγιστο διάστημα κατά τη διάρκεια του οποίου ο προμηθευτής S# ήταν στην *πραγματικότητα* ανίκανος να προμηθεύσει οποιοδήποτε μέρος. (Πάλι το αποτέλεσμα μπορεί να περιέχει διάφορες πλειάδες για τον ίδιο προμηθευτή).

3.11 Σχέσεις συμπίεσης

Επικεντρωνόμαστε πρώτα στο ερώτημα Α. Παρακάτω είναι μια επιβεβαίωση αυτού του ερωτήματος από την άποψη της βάσης δεδομένων του σχήματος 3.1

- **QUERY A:** Πάρτε τα ζευγάρια S#-DURING για τους προμηθευτές που ήταν σε θέση να προμηθεύσουν τουλάχιστον ένα μέρος κατά τη διάρκεια τουλάχιστον ενός διαστήματος του χρόνου, όπου το DURING υποδεικνύει ένα μέγιστο διάστημα κατά τη διάρκεια του οποίου ο προμηθευτής S# ήταν στην πραγματικότητα ικανός να προμηθεύσουν τουλάχιστον ένα μέρος.

Πιθανώς θα θυμάστε ότι σε μια προηγούμενη έκδοση αυτού του ερωτήματος (που συζητήθηκε επίσης στο κεφάλαιο 2) απαιτούσε τη χρήση ορισμένων διαδικασιών ομαδοποίησης. Πιο συγκεκριμένα, περιλάμβανε έναν σχεσιακό τελεστή SUMMARIZE. Πιθανώς δεν θα εκπλαγείτε να μάθετε, ότι αυτή η επαναδιατυπωμένη έκδοση πρόκειται επίσης να απαιτήσει ορισμένες διαδικασίες ομαδοποίησης. Εντούτοις, θα ενισχύσουμε τη διατύπωση αυτού του ερωτήματος με ένα μικρό βήμα σε έναν χρόνο, και η ομαδοποίηση αυτή καθ' εαυτή δεν θα μπει στην εικόνα μέχρι το δεύτερο βήμα. Εδώ είναι το πρώτο βήμα:

WITH SP_DURING {S#, DURING} AS T1:

(Υπάρχει και συνέχεια αυτής της έκφρασης, όπως η άνω και κάτω τελεία προτείνει.) Αυτό το βήμα παρουσιάζει μερικούς αριθμούς μερών οι οποίοι είναι άσχετοι με το υπό εξέταση ερώτημα και εισάγει ένα όνομα για το αποτέλεσμα αυτής της προβολής. Λαμβάνοντας υπόψη τις συνηθισμένες τιμές στοιχείων δειγμάτων μας, το T1 μοιάζει με αυτό:

S#	DURING
S1	[d04:d10]
S1	[d05:d10]
S1	[d09:d10]
S1	[d06:d10]
S2	[d02:d04]
S2	[d08:d10]
S2	[d03:d03]
S2	[d09:d10]
S3	[d08:d10]
S4	[d06:d09]
S4	[d04:d08]
S4	[d05:d10]

Παρατηρήστε τώρα ότι αυτή η σχέση περιέχει περιττές πληροφορίες, παραδείγματος χάριν, μας λέει πάνω από τρεις φορές ότι ο προμηθευτής S1 ήταν σε θέση να προμηθεύσει κάτι την 6η ημέρα. Το επιθυμητό αποτέλεσμα, που αποβάλλει όλο αυτόν τον πλεονασμό, είναι το εξής:

S#	DURING
S1	[d04:d10]
S2	[d02:d04]
S2	[d08:d10]
S3	[d08:d10]
S4	[d04:d10]

Καλούμε αυτό το αποτέλεσμα συμπιεσμένη μορφή του T1 *στο* DURING. Προσέξτε ότι μία τιμή DURING για έναν δεδομένο προμηθευτή σε αυτήν την συμπιεσμένη μορφή δεν χρειάζεται να υπάρξει ως ρητή τιμή DURING για αυτόν τον προμηθευτή στο T1 από την οποία η συμπιεσμένη μορφή παράγεται, στο παράδειγμά μας, αυτή η παρατήρηση ισχύει για τον προμηθευτή S4.
Τώρα, θα φθάσουμε τελικά σε ένα σημείο όπου μπορούμε να επιτύχουμε αυτό το αποτέλεσμα με τη βοήθεια μιας απλής έκφρασης της μορφής

PACK T1 ON DURING

Το επόμενο βήμα είναι το ακόλουθο:

WITH (T1 GROUP {DURING} AS X) AS T2

S#	X
S1	DURING [d04:d10] [d05:d10] [d09:d10] [d06:d10]
S2	DURING [d02:d04] [d08:dl0] [d03:d03] [d09:d10]
S3	DURING [d08:d10]
S4	DURING [d06:d09] [d04:d08] [d05:d10]

Η ιδιότητα X είναι τιμή σχέσης. Στη συνέχεια εφαρμόζουμε τον τελεστή COLLAPSE στις μοναδικές σχέσεις οι οποίες είναι τιμές της τιμής σχέσης της ιδιότητας X:

WITH (EXTEND T2 ADD COLLAPSE (X) AS Y)
{ALL BUT X} AS T3:

S#	Y
S1	DURING [d04:d10]
S2	DURING [d02:d04] [d08:d10]
S3	DURING [d08:d10]
S4	DURING [d04:d10]

Τέλος χωρίζουμε:

T3 UNGROUP Y

Παρουσιάζουμε τώρα όλα τα βήματα μαζί, όπου το αποτέλεσμα είναι το αποτέλεσμα της αξιολόγησης της ακόλουθης γενικής έκφρασης:

WITH SP_DURING {S#, DURING} AS T1,
(TI GROUP {DURING} AS X) AS T2,
(EXTEND T2 ADD COLLAPSE (X) AS Y) {ALL BUT X} AS T3:
T3 UNGROUP Y

Προφανώς, θα ήταν επιθυμητό να είναι σε θέση να πάρει από το αποτέλεσμα σε μια ενιαία λειτουργία. Με αυτό το στόχο, εισάγουμε (επιτέλους!) τον νέο τελεστή συμπίεσης (PACK), με τη σύνταξη ως εξής:

PACK R ON A

Εδώ το R είναι μια σχεσιακή έκφραση και το A είναι μια ιδιότητα διαστήματος της σχέσης r που δείχνεται από αυτήν την έκφραση. Η σημασιολογία καθορίζεται κατά προφανή γενίκευση της ομαδοποίησης, επέκτασης, προβολής, και της μη ομαδοποιημένης διαδικασίας, από τις οποίες επιτύχαμε το αποτέλεσμα από το T1:

```
PACK R ON A = WITH (R GROUP {A} AS X) AS R1,
                  (EXTEND R1 ADD COLLAPSE (X) AS Y}
                                    {ALL BUT X} AS R2:
               R2 UNGROUP Y
```

Το PACK είναι μια στενογραφία. Σημείωση: μπορεί να σας βοηθήσει να επισημάνουμε ότι συμπιέζοντας μια σχέση σε μερικές ιδιότητες A περιλαμβάνει την ομαδοποίηση όλων των ιδιοτήτων της εκτός από αυτήν την ιδιότητα A. Ωστόσο, σημειώστε ότι όσο το R GROUP {A}... εγγυάται να επιστρέψει ένα αποτέλεσμα με ακριβώς μία πλειάδα για κάθε ξεχωριστή τιμή του B (όπου το B είναι όλες οι ιδιότητες του R εκτός από την A), το PACK R ON A μπορεί να επιστρέψει ένα αποτέλεσμα με διάφορες πλειάδες για κάθε τιμή που δίνεται στο B.

Για να επιστρέψουμε στο ερώτημα A, μπορούμε τώρα να υποδείξουμε τα εξής ως συνοπτική διατύπωση αυτού του ερωτήματος:

```
PACK SP_DURING (S1, DURING) ON DURING
```

Η γενική λειτουργία που δείχνεται από αυτήν την έκφραση είναι ένα παράδειγμα αυτού που καλείται μερικές φορές χρονική προβολή. Συγκεκριμένα, είναι μια "χρονική προβολή" της SP _ DURING πάνω στην S # και DURING .

3.12 Σχέσεις Αποσυμπίεσης

Επιστέφουμε τώρα στο ερώτημα B. Παρακάτω είναι μια επιβεβαίωση αυτού του ερωτήματος, βάση του σχήματος 3.1:

- **Query B:** πάρτε το ζευγάρι S#-DURING για τους προμηθευτές που ήταν ανίκανοι να προμηθεύσουν οποιαδήποτε μέρη κατά τη διάρκεια τουλάχιστον ενός διαστήματος του χρόνου, όπου το DURING υποδεικνύει ένα μέγιστο διάστημα κατά τη διάρκεια του προμηθευτή S# που ήταν στην πραγματικότητα ανίκανος να προμηθεύσει οποιοδήποτε μέρος.

Θυμηθείτε τώρα ότι η αρχική έκδοση αυτού του ερωτήματος περιελάμβανε μια σχεσιακή λειτουργία διαφοράς. Κατά συνέπεια, εάν αναμένετε να δείτε κάτι που μπορεί να ονομάζεται χρονική διαφορά, τότε είστε σωστοί. Όπως επίσης, ενώ η χρονική προβολή περιλαμβάνει τον τελεστή σχέσης PACK, η χρονική διαφορά περιλαμβάνει τον τελεστή σχέσης UNPACK. (Στην πραγματικότητα περιλαμβάνει και τον τελεστή PACK, όπως θα δούμε παρακάτω).

Όπως και η κανονική λειτουργία διαφοράς, η χρονική διαφορά περιλαμβάνει δύο τελεστές σχέσης. Στο παράδειγμα, στην πραγματικότητα, πρέπει να είναι διαισθητικά σαφές το τι πρέπει να κάνουμε. Στην ουσία, ψάχνει για ζευγάρια S#-DURING που εμφανίζονται μέσα ή υπονοούνται από την S_DURING και δεν εμφανίζονται μέσα και δεν υπονοούνται από την SP_DURING. Αυτός ο συνοπτικός χαρακτηρισμός πρέπει να είναι επαρκής για να προτείνει (σωστά), ότι αυτό που πρέπει να κάνουμε είναι να εκτελούμε μερικές UNPACK (αποσυμπιεσμένες) διαδικασίες και έπειτα να παίρνουμε τη διαφορά μεταξύ των αποτελεσμάτων. Σας παρουσιάζουμε λοιπόν τον τελεστή UNPACK:

```
UNPACK R ON A = WITH (R GROUP {A} AS X) AS R1,
                    (EXTEND R1 ADD EXPAND (X) AS Y}
                                { ALL BUT X } AS R2 :
                R2 UNGROUP Y
```

Αυτός ο ορισμός είναι ίδιος με αυτόν για τον τελεστή PACK, εκτός από την εμφάνιση της EXPAND(επέκτασης) αντί της COLLAPSE στη δεύτερη γραμμή. Καλούμε το αποτέλεσμα της έκφρασης αποσυμπιεσμένη μορφή της R στο A.

Επιστρέφοντας στο ερώτημα B, μπορούμε τώρα να λάβουμε τον αριστερό τελεστέο που χρειαζόμαστε (δηλ., ζευγάρια S#-DURING που εμφανίζονται μέσα ή υπονοούνται απ' την S_DURING) ως εξής:

```
UNPACK S_DURING {S#, DURING} ON DURING
```

Η επεκταθείσα μορφή αυτής της έκφρασης:

```
WITH S_DURING {S#, DURING} AS T1,
(T1 GROUP {DURING} AS X} AS T2,
(EXTEND T2 ADD EXPAND (X) AS Y) {ALL BUT X} AS T3:
T3 UNGROUP Y
```

Λαμβάνοντας υπόψη τα στοιχεία δειγμάτων του σχήματος 5.1, το γενικό αποτέλεσμα- που το ονομάζουμε U1-μοιάζει μ' αυτό:

S#	DURING
S1	[d04:d04]
S1	[d05:d05]
S1	[d06:d06]
S1	[d07:d07]
S1	[d08:d08]
S1	[d09:d09]
S1	[d10:d10]
S2	[d02:d02]
S2	[d03:d03]
S2	[d04:d04]
S2	[d07:d07]
S2	[d08:d08]
S2	[d09:d09]
S2	[d10:d10]
S3	[d03:d03]
S3	[d04:d04]
S3	[d05:d05]
S3	[d06:d06]
S3	[d07:d07]
S3	[d08:d08]
S3	[d09:d09]
S3	[d10:d10]
S4	[d04:d04]
S4	[d05:d05]
S4	[d06:d06]
S4	[d07:d07]
S4	[d08:d08]
S4	[d09:d09]
S4	[d10:d10]
S5	[d02:d02]
S5	[d03:d03]
S5	[d04:d04]
S5	[d05:d05]
S5	[d06:d06]
S5	[d07:d07]
S5	[d08:d08]
S5	[d09:d09]
S5	[d10:d10]

Τώρα έχουμε τον αριστερό τελεστέο για τη λειτουργία διαφοράς που ενισχύουμε βαθμιαία. Φυσικά, ο σωστός τελεστέος (δηλ., τα S#-DURING ζευγάρια που εμφανίζονται μέσα ή υπονοούνται από την SP _ DURING) λαμβάνεται μέσα από αυτόν τον τρόπο:

UNPACK SP_DURING {S#, DURING} ON DURING

Ονομάζουμε το αποτέλεσμα αυτής της έκφρασης U2.Το αποτέλεσμα μοιάζει μ' αυτό:

S#	DURING
S1	[d04:d04]
S1	[d05:d05]
S1	[d06:d06]
S1	[d01:d01]
S1	[d08:d08]
S1	[d09:d09]
S1	[d10:d10]
S2	[d02:d02]
S2	[d03:d03]
S2	[d04:d04]
S2	[d08:d08]
S2	[d09:d09]
S2	[d10:d10]
S3	[d08:d08]
S3	[d09:d09]
S3	[d10:d10]
S4	[d04:d04]
S4	[d05:d05]
S4	[d06:d06]
S4	[d07:d07]
S4	[d08:d08]
S4	[d09:d09]
S4	[dl0:dl0]

Τώρα μπορούμε να εφαρμόσουμε (την κανονική) διαφορά σχέσης:

U1 MINUS U2

Το αποτέλεσμα αυτής της έκφρασης μοιάζει μ' αυτό:

S#	DURING
S2	[d07:d07]
S3	[d03:d03]
S3	[d04:d04]
S3	[d05:d05]
S3	[d06:d06]
S3	[d07:d07]
S5	[d02:d02]
S5	[d03:d03]
S5	[d04:d04]
S5	[d05:d05]
S5	[d06:d06]
S5	[d07:d07]
S5	[d08:d08]
S5	[d09:d09]
S5	[dl0:dl0]

Τέλος συμπιέζουμε το U3 *στο* DURING για να πάρουμε το επιθυμητό αποτέλεσμα:

PACK U3 ON DURING

Το τελικό αποτέλεσμα μοιάζει μ' αυτό:

S1	DURING
S2	[d01:d01]
S3	[d03:d07]
S5	[d02:d10]

Εδώ είναι μια διατύπωση του ερωτήματος ως ενιαία τοποθετημένη έκφραση:

```
PACK
        ((UNPACK S_DURING {S#, DURING} ON DURING)
          MINUS
        (UNPACK SP_DURING {S#, DURING} ON DURING))
        ON DURING
```

Όπως ήδη έχει λεχθεί, η γενική λειτουργία που δείχνεται από αυτήν την έκφραση είναι ένα παράδειγμα που καλείται μερικές φορές χρονική διαφορά. Ακριβέστερα, είναι μια χρονική διαφορά μεταξύ (1) της προβολής S_DURING πέρα από την S# *και* DURING και (2) της προβολής της SP_DURING πέρα από την S# *και* DURING. Θα έχουμε πολύ περισσότερο να πούμε για τέτοια θέματα στο επόμενο κεφάλαιο. Εδώ ικανοποιούμαστε με μερικές πρόσθετες παρατηρήσεις στη σημασιολογία των τελεστών PACK *και* UNPACK. Εδώ είναι μια επανάληψη του επίσημου ορισμού UNPACK:

```
UNPACK R ON A = WITH (R GROUP {A} AS X) AS R1.
        (EXTEND R1 ADD EXPAND (X) AS Y)
        {ALL BUT X} AS R2:
                R2 UNGROUP Y
```

Τώρα παρατηρείστε τα παρακάτω:

- ✓ UNPACKING R ON A (αποσυμπίεση του R στο A) περιλαμβάνει την ομαδοποίηση του R από όλες τις ιδιότητες *του* R εκτός από το A.
- ✓ Οι τελεστές PACK *και* UNPACK είναι στην πραγματικότητα στενογραφίες *και* καθορίζονται σε επίπεδο του τελεστή EXPAND
- ✓ Οι ακόλουθες ταυτότητες κρατούν τις αντίστοιχες τιμές:
  UNPACK R ON A ≡UNPACK (PACK R ON A) ON A
  PACK R ON A≡PACK (UNPACK R ON A) ON A

Ακολουθεί ότι η πρώτη λειτουργία συμπίεση–αποσυμπίεση ή η ακολουθία αποσυμπίεση–συμπίεση σε κάποια δεδομένη σχέση μπορεί απλά να αγνοηθεί, ένα γεγονός που θα μπορούσε να είναι χρήσιμο για λόγους βελτιστοποίησης (ειδικά όταν η πρώτη λειτουργία είναι UNPACK).

- ✓ Όπως τους τελεστές COLLAPSE *και* EXPAND στους οποίους είναι βασισμένοι, *οι* PACK *και* UNPACK δεν είναι αντίστροφοι.

Δηλαδή καμία από τις εκφράσεις UNPACK (PACK R ON A) στο A *και* PACK (UNPACK R ON A) στο A δεν εγγυώνται ότι θα επιστρέψουν ένα αποτέλεσμα που είναι ίσο με *το* R.

3.13 Ερωτήσεις δειγμάτων

Σε αυτό το τμήμα δίνουμε τα παραδείγματα της χρήσης του PACK *και* UNPACK *στη* διατύπωση των ερωτήσεων. Υποθέτουμε, αρκετά εύλογα, ότι το αποτέλεσμα απαιτείται με κατάλληλα συσκευασμένη μορφή σε κάθε περίπτωση.

Το πρώτο παράδειγμά μας σκόπιμα δεν είναι χρονικό. Υποθέστε ότι μας δίνεται μια σχέση NHW, με ιδιότητες το όνομα(NAME), το ύψος(HEIGHT) *και το* βάρος(WEIGHT), που δίνει το ύψος και το βάρος ορισμένων προσώπων. Εξετάστε το ερώτημα "Για κάθε βάρος που αντιπροσωπεύεται σε NHW, πάρτε κάθε σειρά των υψών έτσι ώστε για κάθε τέτοια σειρά r και για κάθε ύψος στο r υπάρχει τουλάχιστον ένα άτομο που αντιπροσωπεύεται σε NHW που είναι αυτού του ύψους και βάρους". Εδώ είναι μια πιθανή διατύπωση:

```
PACK
      ( ( EXTEND NHW {HEIGHT, WEIGHT}
          ADD INTERVAL_HEIGHT ([HEIGHT: HEIGHT] ) AS HR )
      { WEIGHT, HR } )
ON HR
```

ΕΞΗΓΗΣΗ: Αρχίζουμε με την προβολή NHW στο ύψος και το βάρος. Με αυτόν τον τρόπο λαμβάνουμε όλα τα ζευγάρια ύψους-βάρους στην αρχική σχέση (δηλαδή, όλα τα ζευγάρια ύψους-βάρους έτσι ώστε *να* υπάρχει τουλάχιστον ένα άτομο αυτού του ύψους και βάρους). Επεκτείνουμε έπειτα αυτήν την προβολή με την εισαγωγή μιας άλλης ιδιότητας, HR, της οποίας η τιμή σε οποιαδήποτε δεδομένη πλειάδα είναι ένα διάστημα μονάδων της μορφής [h:h], όπου *το* h είναι η τιμή του ύψους σε αυτή την ίδια πλειάδα. Προβάλλουμε έπειτα την ιδιότητα ύψους και συσκευάζουμε το αποτέλεσμα για την HR. Το τελικό αποτέλεσμα είναι μια σχέση με τις δύο ιδιότητες, το βάρος και HR , *και με το* ακόλουθο κατηγόρημα:

Για όλα τα ύψη h στο HR-αλλά όχι για το h= PRE (HR) ή h = POST(HR), υπάρχει τουλάχιστον ένα άτομο p έτσι ώστε το p να έχει βάρος WEIGHT και ύψος HEIGHT h.

Σημειώστε ότι αυτό το παράδειγμα είναι πράγματι, όπως προηγουμένως δηλώνεται, μη χρονικό. Τα διαστήματα αντιπροσωπεύουν τις σειρές των υψών, και είναι μη χρονικά διαστήματα (με άλλα λόγια, είναι τύπου INTERVAL_HEIGHT, όπου το ύψος είναι ο εφαρμόσιμος τύπος σημείου).

Μέσω ενός δεύτερου παραδείγματος, εξετάστε την σχέση SP_DURING άλλη μια φορά. Οποιαδήποτε στιγμή, εάν υπάρχουν οποιεσδήποτε αποστολές όλη αυτήν την περίοδο, τότε υπάρχει κάποιος αριθμός μερών pmax έτσι ώστε κανένας προμηθευτής δεν είναι σε θέση να προμηθεύσει οποιοδήποτε μέρος αυτήν την περίοδο με έναν αριθμό μερών μεγαλύτερο από pmax.

(Προφανώς υποθέτουμε εδώ ότι ο τελεστής ">" καθορίζεται για τις τιμές του τύπου P#). Έτσι εξετάστε το ερώτημα "για κάθε αριθμό μερών που υπάρχει πάντα μια τέτοια τιμή pmax, παίρνει αυτόν τον αριθμό μερών μαζί με το διάστημα (s) κατά τη διάρκεια του οποίου πραγματικά ήταν pmax τιμή". Εδώ είναι μια πιθανή διατύπωση:

```
WITH (UNPACK SP_DURING ON DURING) AS SP_UNPACKED,
        (SUMMARIZE SP UNPACKED
        PER SP_UNPACKED {DURING}
        ADD MAX (P#) AS PMAX) AS SUMMARY:
PACK SUMMARY ON DURING
```

Το τρίτο και τελευταίο παράδειγμά μας είναι βασισμένο στη σχέση S_PARTS_DURING από την παράγραφο 3.2 (το ένα με δύο ιδιότητες διαστήματος). Για ευκολία βλέπουμε κάποιες ενδεικτικές τιμές γι' αυτήν την σχέση στο παρακάτω παράδειγμα. Εξετάζουμε την ερώτηση "για κάθε μέρος που είναι πάντα σε θέση να προμηθευτεί από τον προμηθευτή S3, πάρτε τον αριθμό μερών και *τα* εφαρμόσιμα διαστήματα του χρόνου". Εδώ είναι μια πιθανή διατύπωση:

```
WITH (S_PARTS_DURING WHERE S# = S# ('S3')) AS T1,
        T1 {PARTS, DURING} AS T2,
        (UNPACK T2 ON PARTS) AS T3,
        (EXTEND T3 ADD POINT FROM PARTS AS P#) AS T4
        T4 {P#, DURING} AS T5 :
        PACK T5 ON DURING
```

Σημειώστε τη χρήση του POINT FROM σε αυτό το παράδειγμα για να εξαγάγετε το ενιαίο σημείο από ένα διάστημα μονάδων.

Η σχέση
S_PARTS_DURING
(ενδεικτικές τιμές)

S_PARTS_DURING

S#	PARTS	DURING
S1	[P1:P3]	[d01:d04]
S1	[P2:P4]	[d07:d08]
S1	[P5:P6]	[d09:d09]
S2	[P1:P1]	[d08:d09]
S2	[P1:P2]	[d08:d08]
S2	[P3:P4]	[d07:d08]
S3	[P2:P4]	[d01:d04]
S3	[P3:P5]	[d01:d04]
S3	[P2:P4]	[d05:d06]
S3	[P2:P4]	[d06:d09]
S4	[P3:P4]	[d05:d08]

Παράδειγμα 3.2

ενδεικτικά αποτελέσματα

P#	DURING
P2	[d01:d09]
P3	[d01:d09]
P4	[d01:d09]
P5	[d01:d04]

3.14 Συμπίεση και Αποσυμπίεση χωρίς ιδιότητες

Μέχρι αυτό το σημείο, έχουμε εξετάσει την συμπίεση και αποσυμπίεση των σχέσεων ακριβώς σε μια ενιαία ιδιότητα της εν λόγω σχέσης. Εντούτοις, είναι δυνατό να γενικευτούν οι τελεστές με τέτοιον τρόπο ώστε να μας επιτρέψει να γίνει η συμπίεση και αποσυμπίεση σε οποιοδήποτε υποσύνολο των ιδιοτήτων σχέσης, συμπεριλαμβανομένου του κενού συνόλου ιδιοτήτων (δεδομένου ότι το κενό σύνολο είναι ένα υποσύνολο κάθε συνόλου). Εξετάζουμε το κενό σύνολο ιδιοτήτων σε αυτό το τμήμα και άλλα σύνολα ιδιοτήτων στο επόμενο τμήμα.

Όπως θα δούμε και παρακάτω, η ικανότητα να συμπιέσουμε ή να αποσυμπιέσουμε μια σχέση πάνω σε καμία ιδιότητα καταλήγει να είναι πολύ σημαντική. Η σύνταξη είναι όπως ακολουθεί:

1) PACK R ON ()
2) UNPACK R ON ()

Καθορίζουμε το αποτέλεσμα και των δύο εκφράσεων να είναι, η σχέση r, που είναι το αποτέλεσμα της αξιολόγησης της διευκρινισμένης σχεσιακής έκφρασης R. Δικαιολογούμε αυτήν την θέση ως εξής. Κατ' αρχάς, PACK R ON () καθορίζεται, αρκετά εύλογα, να είναι στενογραφία για την έκφραση:

```
WITH (R GROUP { } AS X) AS R1,
      (EXTEND R1 ADD COLLAPSE (X) AS Y)
      {ALL BUT X}AS R2:
R2 UNGROUP Y
```

Αυτή η έκφραση είναι ίδια με την επέκταση PACK R ON A, εκτός από το ότι το πρώτο βήμα (το βήμα ομαδοποίησης) διευκρινίζει την "GROUP{}" αντί την "GROUP{A}". Η σημασιολογία είναι η ακόλουθη:

- Το βήμα ομαδοποίησης δίνει ένα ενδιάμεσο αποτέλεσμα R1 με τον ίδιο αριθμό στοιχείων συνόλου με το r και με τον ίδιο τίτλο με το r, εκτός από το ότι περιέχει μια πρόσθετη ιδιότητα X, η οποία είναι εκτιμημένη σχέση. (θυμηθείτε από το κεφάλαιο 1 ότι, γενικά, το αποτέλεσμα της R GROUP {A1,A2... , An} AS B έχει βαθμό n R-n + 1, όπου nR είναι ο βαθμός του R. Εάν n = 0, ενδεχομένως η πιθανότητα μέτρησης της επίδρασης είναι να προστεθεί μια ιδιότητα στον τίτλο.)

Οι σχέσεις που είναι τιμές του X έχουν το βαθμό μηδέν. Καθεμιά από αυτές τις σχέσεις είναι μηδενικές σχέσεις. Επιπλέον, καθεμιά από αυτές είναι TABLE_DEE και όχι TABLE_DUM. Επειδή κάθε πλειάδα t *στην* r περιλαμβάνει την μηδενική πλειάδα ως τιμή του για αυτήν την υπόπλειάδα του t *που* αντιστοιχεί στο κενό σύνολο ιδιοτήτων. Κατά συνέπεια, κάθε πλειάδα *στην* R1 αποτελείται από την αντίστοιχη πλειάδα από το r που επεκτείνεται με την X τιμή TABLE_DEE.

- Το επόμενο βήμα δίνει ένα ενδιάμεσο αποτέλεσμα R2 που είναι ίδιο με το R1, εκτός από το ότι η ιδιότητα X μετονομάζεται *σε* Y.

- Το τελικό βήμα έπειτα αντικαθιστά κάθε πλειάδα t *στην* R2 από την αλληλουχία του με μηδέν πλειάδες, απορρίπτει την ιδιότητα Y και έπειτα επιστρέφει την σχέση που λαμβάνεται ως γενικό αποτέλεσμα. Αλλά η σύνδεση οποιασδήποτε πλειάδας t με μηδέν πλειάδες παράγει την ίδια πλειάδα. Κατά συνέπεια, το τελικό αποτέλεσμα είναι ίδιο με τη σχέση r.

Επιστρέφοντας τώρα στην αποσυμπίεση UNPACK καθορίζουμε την UNPACK R ON () να είναι στενογραφία της έκφρασης

```
WITH (R GROUP { } AS X) AS RI.
        (EXTEND RI ADD EXPAND (X) AS Y)
        {ALL BUT X}AS R2:
R2 UNGROUP Y
```

Δύο προφανείς αλλά εκτεταμένες συνέπειες των προηγούμενων ορισμών είναι ότι (1) η συμπίεση r σε καμία ιδιότητα και μετά η αποσυμπίεση του αποτελέσματος , πάλι, σε καμία ιδιότητα επιστρέφει το r και (2) η αποσυμπίεση του r σε καμία ιδιότητα και μετά η συμπίεση του αποτελέσματος ,ξανά σε καμία ιδιότητα επιστρέφει επίσης το r.

3.15 Συμπίεση *και* Αποσυμπίεση με διάφορες ιδιότητες

Τώρα γυρίζουμε στο θέμα της συμπίεσης και αποσυμπίεσης σε σχέσεις με δύο ή περισσότερες ιδιότητες. Για λόγους που θα γίνουν σαφείς αργότερα, όταν επιδρούμε στην υποενότητα του PACK, πρέπει πρώτα να εξετάσουμε το UNPACK.

UNPACK

Αρχίζουμε με την εξέταση της περίπτωσης της αποσυμπίεσης σε δύο ιδιότητες. Έστω ότι r είναι μια σχέση με δύο ευδιάκριτες ιδιότητες διαστήματος A1 *και* A2. Υποθέστε ότι το r μοιάζει με αυτό:

A1	A2
[P1:P1]	[d08:d09]
[P1:P2]	[d08:08]
[P3:P4]	[d07:08]

Αυτή η σχέση είναι στην πραγματικότητα ο περιορισμός της σχέσης που παρουσιάζεται στο σχήμα 3.1 ακριβώς σε αυτές τις πλειάδες για τον προμηθευτή S2 που προβάλλονται πέρα από το PARTS και DURING εκτός από το ότι έχουμε μετονομάσει τις δύο ιδιότητες A1 και A2.

Τώρα εξετάζουμε την έκφραση

UNPACK (UNPACK r ON Al) ON A2

Η εσωτερική έκφραση UNPACK r ON Al αποφέρει:

A1	A2
[P1:P1]	[d08:d09]
[P1:P1]	[d08:d08]
[P2:P2]	[d08:d08]
[P3:P3]	[d07:d08]
[P4:P4]	[d07:d08]

Αποσυμπιέζοντας αυτήν την σχέση πάνω στο A2 αποφέρει:

A1	A2
[P1:P1]	[d08:d08]
[Pl:Pl]	[d09:d09]
[P2:P2]	[d08:d08]
[P3:P3]	[d07: d07]
[P3:P3]	[d08:d08]
[P4:P4]	[d07: d07]
[P4:P4]	[d08:d08]

Τώρα εξετάζουμε τι συμβαίνει εάν κάνουμε τις δύο αποσυμπιέσεις με την αντίθετη σειρά, εξετάζουμε την έκφραση

UNPACK (UNPACK r ON A2) ON Al

Η εσωτερική έκφραση UNPACK r ON A2 αποφέρει:

A1	A2
[P1:P1]	[d08:d08]
[P1:P1]	[d09:d09]
[P1:P2]	[d08:d08]
[P3:P4]	[d07:d07]
[P3:P4]	[d08:d08]

Αποσυμπιέζοντας αυτήν την σχέση πάνω στο A1 αποφέρει:

A1	A2
[P1:P1]	[d08:d08]
[P1:P1]	[d09:d09]
[P2:P2]	[d08:d08]
[P3:P3]	[d07:d07]
[P4:P4]	[d07:d07]
[P3:P3]	[d08:d08]
[P4:P4]	[d08:d08]

Και αυτό το γενικό αποτέλεσμα φαίνεται να είναι το ίδιο με πριν. Στην πραγματικότητα, είναι εύκολο να φανεί ότι, εάν το r είναι οποιαδήποτε σχέση με ιδιότητες διαστήματος *το* A1 *και το* A2, τότε

UNPACK (UNPACK r ON A1) ON A2 = UNPACK (UNPACK r ON A2) ON A1

Στην πραγματικότητα, είναι εύκολο να φανεί ότι, εάν το r έχει ιδιότητες διαστήματος A1, A2...,An τότε αποσυμπιέζοντας το r σε αυτές τις ιδιότητες σε οποιαδήποτε σειρά *θα* παράγει πάντα το ίδιο γενικό αποτέλεσμα. Επομένως προτείνουμε την παρακάτω σύνταξη στενογραφίας:

<unpack>
::= UNPACK <relation exp>
ON (<attribute name commalist>)

Η σημασιολογία είναι η ακόλουθη:

UNPACK r ON (A1, A2, ..., An) =
UNPACK (... (UNPACK (UNPACK r ON B1) ON B2) ...) ON Bn

όπου η ακολουθία των ιδιοτήτων το B1, B2...,Bn είναι κάποια αυθαίρετη μεταλλαγή της διευκρινισμένης ακολουθίας ιδιοτήτων A1, A2,....An. Οι παρενθέσεις μπορούν να παραλειφθούν εάν η διευκρινισμένη ακολουθία περιέχει μόνο ένα όνομα ιδιοτήτων.

ΣΗΜΕΙΩΣΗ: *Στην πραγματικότητα*, η διευκρινισμένη ακολουθία Al,A2...An, δείχνει πραγματικά ένα σύνολο ονομάτων ιδιοτήτων *στο* tutorial D. Επομένως, θα το εσωκλείαμε κανονικά σε αγκύλες, κι όχι παρενθέσεις. Όπως θα δούμε στην επόμενη υποενότητα, οι παρενθέσεις απαιτούνται *στην* περίπτωση συμπίεσης γιατί η ακολουθία στην οποία τα ονόματα ιδιοτήτων διευκρινίζονται είναι σημαντικά *για το* PACK (ακόμα κι αν είναι η περίπτωση που καμία ιδιότητα δεν μπορεί να διευκρινιστεί περισσότερο από μία φορά). Αισθανθήκαμε έτσι ότι θα ήταν φιλικότερο προς το χρήστη να χρησιμοποιήσει τις παρενθέσεις αντί τις αγκύλες, αλλά μετά για να δηλώσει ρητά ότι η ακολουθία είναι αυθαίρετη, όπως επίσης και στην περίπτωση UNPACK.

PACK (συμπίεση)

Γυρίζουμε τώρα στη PACK. Εδώ προχωράμε κάπως διαφορετικά, αρχίζουμε με τον ορισμό ενός τελεστή στον οποίο θα αναφερθούμε προς το παρόν ως PACK+ που λειτουργεί ως εξής. Πρώτα αποσυμπιέζει τη διευκρινισμένη σχέση σε όλες τις διευκρινισμένες ιδιότητες, και έπειτα ξανασυμπιέζει αυτήν την αποσυμπιεσμένη σχέση σε αυτές τις ίδιες ιδιότητες στη σειρά με την οποία διευκρινίζονται. Κατά συνέπεια, η έκφραση

PACK+ r ON (A1, A2)

Ορίζεται να είναι στενογραφία για την έκφραση

PACK (PACK (UNPACK r ON (A1, A2)) ON A1) ON A2
Με άλλα λόγια, η αρχική PACK + επίκληση PACK + r ON (A1, A2) αξιολογείται από την πρώτη σχέση r *στο* A1 *και* A2, μετά συσκευάζει το αποτέλεσμα για το A1 και τέλος συσκευάζει το αποτέλεσμα για το A2.

Σαν βάση για αυτόν τον ορισμό, υποθέστε ότι η σχέση r μοιάζει με αυτό

A1	A2
[P2:P4]	[d01:d04]
[P3:P5]	[d01:d04]
[P2:P4]	[d05:d06]
[P2:P4]	[d06:d09]

Αυτή η σχέση είναι στην πραγματικότητα ο περιορισμός της σχέσης που παρουσιάζεται στο σχήμα 3.1, ακριβώς σε αυτές τις πλειάδες για τον προμηθευτή S3, που προβάλλονται πέρα από τα PARTS *και* DURING, εκτός από το ότι έχουμε μετονομάσει τις δύο ιδιότητες A1 *και* A2, αντίστοιχα.

Εξετάζουμε τώρα το PACK+ r ON (A1, A2). Κατ' αρχάς, εδώ είναι το αποτέλεσμα της αξιολόγησης του υπονοούμενου UNPACK r ON (A1, A2). Αυτό αποτελεί τον πυρήνα *της* επέκτασης αυτής της έκφρασης:

A1	A2
[P2:P2]	[d01:d01]
[P2:P2]	[d02:d02]
[P2:P2]	[d03:d03]
[P2:P2]	[d04:d04]
[P3:P3]	[d01:d01]
[P3:P3]	[d02:d02]
[P3:P3]	[d03:d03]
[P3:P3]	[d04:d04]
[P4:P4]	[d01:d01]
[P4:P4]	[d02:d02]
[P4:P4]	[d03:d03]
[P4:P4]	[d04:d04]
[P5:P5]	[d01:d01]
[P5:P5]	[d02:d02]
[P5:P5]	[d03:d03]
[P5:P5]	[d04:d04]
[P2:P2]	[d05:d05]
[P2:P2]	[d06:d06]
[P3:P3]	[d05:d05]
[P3:P3]	[d06:d06]
[P4:P4]	[d05:d05]
[P4:P4]	[d06:d06]
[P2:P2]	[d07:d07]
[P2:P2]	[d08:d08]
[P2:P2]	[d09:d09]
[P3:P3]	[d07: d07]
[P3:P3]	[d08:d08]
[P3:P3]	[d09:d09]
[P4:P4]	[d07:d07]
[P4:P4]	[d08:d08]
[P4:P4]	[d09:d09]

Συμπιέζοντας αυτήν την σχέση στο A1 αποφέρει:

A1	A2
[P2:P5]	[d01:d01]
[P2:P5]	[d02:d02]
[P2:P5]	[d03:d03]
[P2:P5]	[d04:d04]
[P2:P4]	[d05:d05]
[P2:P4]	[d06:d06]
[P2:P4]	[d07:d07]
[P2:P4]	[d08:d08]
[P2:P4]	[d09:d09]

Συμπιέζοντας αυτήν την σχέση στο A2 τότε αποφέρει:

A1	A2
[P2:P5]	[d01:d04]
[P2:P4]	[d05:d09]

Τώρα εξετάζουμε τι συμβαίνει εάν κάνουμε τις δύο συμπιέσεις στην αντίθετη σειρά, δηλαδή εξετάζουμε το PACK+ r ON (A1, A2). Κατ' αρχάς, το υπονοούμενο UNPACK τις αποφέρει την ίδια σχέση με πριν. Συμπιέζοντας αυτήν *την* σχέση *στο* A2 παράγει:

A1	A2
[P2:P2)	[d0l:d09]
[P3:P3)	[d0l:d09)
[P4:P4]	[d0l:d09]
[P5:P5]	[d0l:d04]

Συμπιέζοντας αυτήν την σχέση στο A1 αποφέρει:

A1	A2
[P2:P4]	[d0l:d09)
[P5:P5]	[d0l:d04]

Αυτό το γενικό αποτέλεσμα είναι σαφώς διαφορετικό από πριν. Ως εκ τούτου, έχουμε ότι εάν το r είναι μια σχέση με ιδιότητες διαστήματος A1 *και* A2, τότε

PACK+ r ON (Al, A2) # PACK+ r ON (A2, Al)

Με άλλα λόγια, αν και προτείνουμε την ακόλουθη σύνταξη στενογραφίας :

<Pack+>

::= PACK+ <relation exp>
ON (<attribute name commalist>)

είναι σημαντικό να γίνει κατανοητό ότι (1) μία προκαταρκτική αποσυμπίεση σε όλες τις διευκρινισμένες ιδιότητες απαιτείται προτού να γίνει οποιαδήποτε συμπίεση, και (2) η μεμονωμένη συμπίεση πρέπει να γίνει στην ακολουθία που διευκρινίζεται από τη λίστα των ονομάτων ιδιοτήτων. (οι παρενθέσεις μπορούν να παραλειφθούν εάν η λίστα περιέχει μόνο ένα όνομα ιδιοτήτων). Κατά συνέπεια, η σημασιολογία είναι η ακόλουθη:

PACK+ r ON (AI, A2, ...An) =
PACK (... (PACK (PACK r' ON Al) ON A2) ...) ON An

Where r' is

UNPACK r ON (A1, A2, ..., An)

Σημειώστε σαφώς ότι αυτός ο ορισμός για το PACK + στις ιδιότητες n στηρίζεται ρητά στον ορισμό του PACK. Τώρα εξετάστε τις ειδικές υποθέσεις n = 1 *και* n = 0. Εάν n = 1, εκτελούμε την PACK + λειτουργία σε μια ενιαία ιδιότητα A, και έχουμε:

PACK+ r ON A = PACK (UNPACK r ON A) ON A
= PACK r ON A

Ομοίως, εάν n = 0, εκτελούμε την PACK + λειτουργία σε καμία ιδιότητα, και έχουμε:

PACK+ r ON () =PACK (UNPACK r ON ()) ON ()
= PACK r ON ()

Κατά συνέπεια, το PACK + στις ιδιότητες n *για* n > 1 είναι μια απλή γενίκευση του PACK όπως πριν καθορίστηκε για n = 0 και ν = 1, κατά συνέπεια, μπορούμε να μετονομάσουμε *το* PACK + ως απλά PACK (δηλ., μπορούμε να αφαιρέσουμε το "+") χωρίς κίνδυνο ασάφειας.

ΚΕΦΑΛΑΙΟ 4: ΠΕΙΡΑΜΑΤΙΚΗ ΠΛΑΤΦΟΡΜΑ

4.1 Εισαγωγή

Μέχρι αυτό το σημείο του βιβλίου, τα παραδείγματα S_DURING και SP_DURING μας έχουν βοηθήσει αρκετά, αποδεικνύοντας την ανάγκη για τύπους διαστημάτων και την επιθυμία για ειδικούς τελεστές για την αντιμετώπιση των δεδομένων διαστημάτων. Είναι φανερό ότι αυτές οι δυο μεταβλητές σχέσης είναι εύκολο να δομηθούν.
Ας δούμε τώρα τις κανονικές μη-χρονικές σχέσης S και SP. Εδώ βλέπουμε τους ορισμούς:

```
VAR S RELATION
    {S#         S#,
     SNAME   NAME,
     STATUS INTEGER,
     CITY      CHAR}
    KEY {S#};
```

```
VAR SP RELATION
        {S# S#,
         P# P#}
       KEY {S#, P#}
       FOREIGN KEY {S#}
            REFERENCES S;
```

Ας εστιάσουμε στους προμηθευτές αυτή τη στιγμή. Θυμίζουμε ότι η δήλωση για τη μεταβλητή σχέσης των προμηθευτών (σχέση S) είναι:

Ο προμηθευτής S# έχει συμβόλαιο, ονομάζεται SNAME, βρίσκεται στην κατάσταση STATUS και είναι στην πόλη CITY.

Σε μια προσπάθεια να αποφύγουμε κάθε πιθανή παρερμηνεία θα ορίσουμε την παραπάνω έκφραση πιο συγκεκριμένα:

Σε μια δεδομένη στιγμή, ο μοναδικός προμηθευτής με ταυτότητα S# έχει ένα και μοναδικό συμβόλαιο, έχει ακριβώς ένα όνομα SNAME, βρίσκεται στην μοναδική κατάσταση STATUS και είναι σε ακριβώς μια πόλη CITY.

Ας υποθέσουμε τώρα ότι θέλουμε να σχεδιάσουμε μια ανάλογη χρονική απεικόνιση αυτής της μη-χρονικής μεταβλητής σχέσης. Πώς όμως μπορούμε να το κάνουμε; Η πιο φανερή προσέγγιση, είναι να προσθέσουμε μια κατάλληλη χρονική ιδιότητα. Την ιδιότητα "since" αν θέλουμε να την σχεδιάσουμε ημι-χρονικά, ή την ιδιότητα "during" αν θέλουμε να την σχεδιάσουμε πλήρως-χρονικά. Αυτή εδώ είναι η ημι-χρονική έκδοση:

```
VAR SSSC_SINCE RELATION
    {S#     S#,
     SNAME NAME,
     STATUS INTEGER,
     CITY CHAR,
     SINCE DATE}
    KEY {S#};
```

Κι αυτή εδώ είναι η πλήρως χρονική έκδοση:

```
VAR SSSC_DURING RELATION
    {S#     S#,
     SNAME NAME,
     STATUS INTEGER,
     CITY CHAR,
     DURING INTERVAL_DATE}
    KEY {S#, DURING};
```

Αυτό που πρέπει να ρωτήσουμε τώρα είναι αν οι μεταβλητές σχέσης SSSC_SINCE και SSSC_DURING είναι καλά παραδείγματα με αυτό που οι χρονικές μεταβλητές σχέσης δείχνουν στην πράξη. Είναι καλά σχεδιασμένες; Αν όχι, πως μπορούν να βελτιωθούν;
Να τονίσουμε ότι σχεδιάσαμε αυτές τις μεταβλητές σχέσης απλά προσθέτοντας μια *χρονική* ιδιότητα στα μη χρονικά μέρη. Το ερώτημα που προκύπτει είναι αν είναι αυτή η κατάλληλη μέθοδος χρονικού σχεδιασμού.
Η απάντηση είναι αρνητική. Και οι δυο μεταβλητές σχέσης είναι κακά σχεδιασμένες. Το πρόβλημα με το SSSC_SINCE φαίνεται να έχει εύκολη λύση, αλλά το πρόβλημα με το SSSC_DURING απαιτεί περισσότερη σκέψη και πιο δραστική λύση. Χρειάζεται θα λέγαμε να αναλύθει σε συνθετικά μέρη. Στην πραγματικότητα, θα προτείναμε τα ακόλουθα:

- Επιπλέον κάθετη ανάλυση, ώστε να ανιμετωπιστεί το γεγονός ότι οι ευδιάκριτες ιδιότητες της ίδιας οντότητας, μεταβάλλονται σε διαφορετικές αξίες
- Επιπλέον οριζόντια ανάλυση, ώστε να αντιμετωπιστεί η διάκριση ανάμεσα σε τωρινές και προηγούμενες πληροφορίες

Στην ενότητα 4.2 μελετούμε το SSSC_SINCE που περιέχει παρούσες πληροφορίες μόνο, και κάνουμε κάποιες χρονικές υποδείξεις. Στην ενότητα 4.3 μελετούμε το SSSC_DURING που περιέχει ιστορικές πληροφορίες. Αυτή η συζήτηση οδηγεί στην κάθετη ανάλυση(τη βλέπουμε σε βάθος στην ενότητα 4.4). Στην ενότητα 4.5 συζητούμε το ειδικό χρονικό πρόβλημα που προκύπτει απ'την σύνδεση με ιστορικές μεταβλητές σχέσης όπως την SSSC_DURING, ακόμα κι αν προχωρήσουμε στην κάθετη ανάλυση. Τέλος, στην ενότητα 4.6 βλέπουμε την οριζόντια ανάλυση.

4.2 Παρούσες Μεταβλητές Σχέσης

Χρησιμοποιούμε τον όρο παρούσες μεταβλητές για να δηλώσουμε μεταβλητές που είναι μερικώς ημι-χρονικές κι έτσι περιέχουν μόνο παρούσες πληροφορίες. Στην πραγματικότητα μια παρούσα μεταβλητή περιέχει πληροφορίες για το παρελθόν, αλλά μπορεί να έχει και κάποιες πληροφορίες για το μέλλον. Αυτό που λέμε μόνο για παρούσες πληροφορίες είναι μόνο μια διαδικασία προσέγγισης της αληθινής κατάστασης. Πιο κάτω βλέπουμε τον ορισμό της μεταβλητής σχέσης SSSC_SINCE *της* προηγούμενης ενότητας:

SSSC_SINCE {S#, SNAME, STATUS, CITY, SINCE}
KEY {S#}

Αυτό που ισχύει είναι το ακόλουθο:

- Ο προμηθευτής S# έχει κάποιο συμβόλαιο
- Ο προμηθευτής S# έχει το όνομα SNAME
- Ο προμηθευτής S# είναι σε μια κατάσταση STATUS
- Ο προμηθευτής S# βρίσκεται σε μια πόλη CITY

Θα πρέπει να γίνει σαφές απ' αυτή τη διατύπωση ότι η μεταβλητή σχέσης δεν είναι καλά σχεδιασμένη. Το γιατί, υποθέστε ότι περιλαμβάνει την παρακάτω πλειάδα:

S#	SNAME	STATUS	CITY	SINCE
S1	Smith	20	London	d04

Υποθέστε επίσης ότι σήμερα είναι η $10^{η}$ μέρα και η κατάσταση του προμηθευτή S1 πρόκειται να αλλάξει σε 30. Γι' αυτό αντικαθιστούμε την πλειάδα όπως φαίνεται παρακάτω:

S#	SNAME	STATUS	CITY	SINCE
S1	Smith	30	London	d10

Τώρα όμως έχουμε χάσει ανάμεσα σε άλλα και την πληροφορία ότι ο προμηθευτής S1 ήταν στο Λονδίνο απ' *την* $4^{η}$ μέρα.
Πρέπει να γίνει σαφές απ'αυτό το παράδειγμα ότι η μεταβλητή σχέση SSSC_SINCE όπως έχει τώρα σχεδιαστεί, δε μπορεί να αναπαραστήσει καμία πληροφορία για τον παρόντα προμηθευτή που προηγείται χρονικός, του χρόνου της πιο πρόσφατης ενημέρωσης γι'αυτό τον προμηθευτή. Αυτό μπορεί να διορθωθεί απλά αντικαθιστώντας την υπάρχουσα ιδιότητα "since" από τέσσερις ιδιότητες, μια για κάθε άλλη ιδιότητα, ώστε:

SSSC_SINCE {S#, S#_SINCE,
SNAME, SNAME_SINCE,
STATUS, STATUS_SINCE,
CITY, CITY_SINCE}
KEY {S#}

Αυτό που αντιστοιχεί η αναθεωρημένη έκδοση του SSSC_SINCE είναι:

Ο προμηθευτής S# είχε συμβόλαιο ως το S#_SINCE, είχε ένα όνομα SNAME *ως* SNAME_SINCE, είχε μια κατάσταση STATUS *ως την* STATUS_SINCE και βρισκόταν σε μια πόλη CITY *ως την* CITY_SINCE.

Εδώ βλέπουμε μια τυπική πλειάδα για τον αναθεωρημένο σχεδιασμό:

S#	S#_SINCE	SNAME	SNAME_SINCE	STATUS	STATUS_SINCE	CITY	CITY_SINCE
S1	d04	Smith	d04	30	d10	London	d04

Αυτή η πλειάδα δείχνει ανάμεσα σε άλλα ότι ο προμηθευτής S1 βρισκόταν στην κατάσταση 30 ως την $10^{η}$ μέρα, αλλά βρισκόταν στο Λονδίνο μέχρι ως την $4^{η}$ μέρα, οπότε έχουμε λύση και στο πρόβλημά μας.
Αυτή όμως η περίπτωση αναπαριστά παρούσες πληροφορίες μόνο. Η μεταβλητή σχέσης είναι ακόμα παρούσα και η βάση δεδομένων είναι ακόμα ημι-χρονική. Για παράδειγμα, *αν την* $10^{η}$ *μέρα* ο προμηθευτής S1 πάει στο Παρίσι, και αντικαταστήσουμε την πλειάδα απλά δείχνοντας με μια άλλη ότι CITY=Paris *και* CITY_SINCE= d10, τότε χάνουμε την πληροφορία ότι ο προμηθευτής S1 ήταν στο Λονδίνο. Απ' αυτό συμπεραίνουμε ότι αυτός ο ημι-χρονικός σχεδιασμός δεν αναπαριστά ιστορικές πληροφορίες.

4.3 Ιστορικές Μεταβλητές Σχέσεις

Ας δώσουμε τώρα την προσοχή μας στην ιστορική ή πλήρως χρονική μεταβλητή σχέση SSSC_DURING. Είναι επίσης πιθανό για μια ιστορική σχέση να περιέχει πληροφορίες που αναφέρονται στο μέλλον. Για παράδειγμα η σχέση SSSC_DURING μπορεί να περιέχει μια πλειάδα που δηλώνει ότι το συμβόλαιο του προμηθευτή Sx εκτείνεται από di ως dj, όπου το dj είναι μια ημερομηνία στο μέλλον, πράγμα που μπορεί να συμβαίνει και *με το* di.
Ας υποθέσουμε τώρα την μεταβλητή σχέση VACATION {EMP#, DURING} με την εξής δήλωση:

Ο υπάλληλος EMP# ήταν, είναι ή θα είναι σε άδεια κατά το διάστημα DURING.

Πιο κάτω βλέπουμε τον ορισμό για την μεταβλητή σχέση SSSC_DURING.

SSSC_DURING {S#, SNAME, STATUS, CITY, DURING}
KEY {S#, DURING}

Ο ορισμός γι'αυτή τη σχέση είναι:

Απ'την μέρα που είναι το αρχικό σημείο του DURING ως την μέρα που είναι το τελικό σημείο του DURING, ισχύουν τα ακόλουθα:

- Ο προμηθευτής S# έχει κάποιο συμβόλαιο
- Ο προμηθευτής S# έχει το όνομα SNAME
- Ο προμηθευτής S# είναι σε μια κατάσταση STATUS
- Ο προμηθευτής S# βρίσκεται σε μια πόλη CITY

Όπως είναι φανερό, όπως και στην περίπτωση SSSC_SINCE, ο ορισμός αυτός δεν είναι καλά σχεδιασμένος. Το γιατί, υποθέστε ότι περιλαμβάνει την ακόλουθη πλειάδα:

S#	SNAME	STATUS	CITY	DURING
S2	Jones	10	Paris	[d02:d04]

Υποθέστε ότι μόλις μάθαμε ότι η κατάσταση του προμηθευτή S2 ήταν 10 την $2^{η}$ *και την* $3^{η}$ μέρα, αλλά έγινε 15 την $4^{η}$ μέρα. Τότε πρέπει να κάνουμε δύο αλλαγές στη σχέση μας προκειμένου να δείξουμε την αλλαγή αυτή. Πιο συγκεκριμένα πρέπει να διαγράψουμε την υπάρχουσα πλειάδα και να εισάγουμε δύο νέες, το οποίο μοιάζει με το εξής:

S#	SNAME	STATUS	CITY	DURING
S2	Jones	10	Paris	[d02:d03]

S#	SNAME	STATUS	CITY	DURING
S2	Jones	15	Paris	[d04:d04]

Το πρόβλημα είναι ότι η χρονική ιδιότητα DURING προσδιορίζεται πολύ έντονα. Προσδιορίζει ένα συνδυασμό τεσσάρων ξεχωριστών προτάσεων, αντί για μια μόνο πρόταση. Το ερώτημα είναι γιατί δεν αντικαθιστούμε την αρχική σχέση S_DURING από τέσσερις χωριστές σχέσεις, καθεμιά με το δικό της χρονικό προσδιορισμό; Οι σχέσεις αυτές θα έμοιαζαν με τις παρακάτω:

S_DURING {S#, DURING}
KEY {S#, DURING}

S_NAME_DURING {S#, SNAME, DURING}
KEY {S#, DURING}

S STATUS DURING {S#, STATUS, DURING}
KEY {S#, DURING}

S_CITY_DURING {S#, CITY, DURING}
KEY {S#, DURING}

Το παράδειγμα αυτό δείχνει την κάθετη ολοκλήρωση όπως χρησιμοποιείται στη σχέση SSSC_DURING. Η σχέση SSSC_DURING δείχνει ποιοι προμηθευτές είχαν συμβόλαιο και πότε, η σχέση S_NAME_DURING δείχνει ποιοι προμηθευτές είχαν ποιο όνομα και πότε, η σχέση S_STATUS_DURING δείχνει ποια κατάσταση είχαν οι προμηθευτές και πότε, και η σχέση S_CITY_DURING ποιοι προμηθευτές βρισκόταν σε ποια πόλη και πότε. Παρακάτω βλέπουμε μερικές τέτοιες πλειάδες σαν παράδειγμα:

S#	SNAME	DURING
S2	Jones	[d02:d04]

S#	DURING
S2	[d02:d04]

S#	STATUS	DURING
S2	10	[d02:d03]

S#	STATUS	DURING
S2	15	[d04:d04]

S#	CITY	DURING
S2	Paris	[d02:d04]

4.4 Η 6η Κανονική Μορφή

Η κάθετη ολοκλήρωση είναι αυτό με το οποίο η θεωρία της κανονικοποίησης έχει ασχοληθεί. Ο τελεστής της ανάλυσης σ' αυτή τη θεωρία είναι η προβολή και ο αντίστοιχος τελεστής αναδιάταξης είναι η συνένωση. Η βασική κανονική μορφή, η 5η, μερικές φορές καλείται κανονική μορφή προβολής/συνένωσης. Η 5η κανονική μορφή βασίζεται στην εξαρτημένη συνένωση (join dependency), η οποία ορίζεται ως εξής:

Έστω ότι R είναι μια σχέση και τα A, B, ..., Z είναι υποσύνολα των ιδιοτήτων R. Τότε λέμε ότι η R ικανοποιεί την εξαρτημένη συνένωση,

* {A, B, ..., Z} (διαβάζεται "αστεράκι A, B, ..., Z")

εάν και μόνο εάν κάθε λογική τιμή του R είναι ίση με τη συνένωση της προβολής των A, B, ..., Z.

Για παράδειγμα θυμηθείτε την μη-χρονική σχέση S των προμηθευτών, με ιδιότητες S#, SNAME, STATUS *και* CITY. Αν συμφωνήσουμε να χρησιμοποιούμε το όνομα "SN" *για* να αναφερόμαστε στο υποσύνολο {S#, SNAME} σαν σύνολο των ιδιοτήτων του S, *και* παρόμοια για τα "ST" *και* "SC", τότε μπορούμε να πούμε ότι η σχέση S ικανοποιεί την εξαρτημένη συνένωση *{ SN, ST, SC}.

Τώρα μπορούμε να ορίσουμε την 5η κανονική μορφή:

Μια σχέση R είναι σε **5η κανονική μορφή** (5NF), εάν και μόνο εάν κάθε μη-τετριμένη εξαρτημένη συνένωση που ικανοποιείται απ'την R υπονοείται απ' το πρωτεύον κλειδί *της* R, όπου:

- Η εξαρτημένη συνένωση *{ SN, ST, SC} *της* R είναι τετριμένη εάν και μόνο εάν τουλάχιστον ένα υποσύνολο A, B, ..., Z είναι το σύνολο όλων των ιδιοτήτων της R
- Η εξαρτημένη συνένωση *{ SN, ST, SC} *της* R υπονοείται απ' το πρωτεύον κλειδί της R εάν και μόνο εάν κάθε ένα απ' τα A, B, ..., Z είναι υπερκλειδί της R (θυμίζουμε ότι όλα τα πρωτεύοντα κλειδιά είναι υπερκλειδιά, αλλά μερικά υπερκλειδιά δεν είναι πρωτεύοντα)

Τώρα μπορούμε να ορίσουμε μια νέα κανονική μορφή, την οποία θα καλούμε $6^{η}$ κανονική μορφή:

Μια σχέση R είναι σε **$6^{η}$ κανονική μορφή** (6NF), εάν και μόνο εάν δεν ικανοποιεί καμιά *μη*-τετριμένη εξαρτημένη συνένωση.

Αυτό που προκύπτει απ'τον ορισμό είναι ότι κάθε σχέση που είναι σε έκτη κανονική μορφή, είναι και σε πέμπτη. Τώρα η σχέση SSSC_DURING δεν είναι σε 6NF επειδή:

- ικανοποιεί την εξαρτημένη συνένωση USING DURING *{SND, STD, SCD}
- η εξαρτημένη συνένωση είναι μη-τετριμένη

Αντίθετα, η σχέση SSSC_DURING είναι σε 5NF αφού η μόνη μη-τετριμένη εξαρτημένη συνένωση που ικανοποιείται, είναι αυτή που υποδηλώνεται απ'το μοναδικό πρωτεύων κλειδί.
Μια σχέση R που δεν είναι σε 6NF "μπορεί να επωφεληθεί απ'την ανάλυση σ'ένα σύνολο από προβολές σε 6NF". Πιο συγκεκριμένα, τα οφέλη συμβαίνουν εάν και μόνο εάν:

- *η* R έχει τουλάχιστον μια ιδιότητα διαστήματος I
- *η* R έχει τουλάχιστον ένα πρωτεύον κλειδί που περιέχει το I
- *η* R έχει τουλάχιστον δύο πρόσθετες ιδιότητες, επιπλέον αυτών που περιέχονται στο {K}
- *η* R είναι αντικείμενο στον περιορισμό PACKED ON I, κι επιπλέον αυτός ο περιορισμός είναι μη-τετριμένος

4.5 Η Χρονική Στιγμή "ΤΩΡΑ" (NOW)

Μέχρι στιγμή είχαμε υποθέσει ότι τα χρονικά διαστήματα στις ιστορικές σχέσεις εκτείνονται από ένα σημείο b, μέχρι ένα σημείο e, όπου b ≤ e *και το* e ≤ της παρούσας στιγμής. Επίσης είχαμε υποθέσει ότι η παρούσα στιγμή αναπαριστούνταν από μια τιμή (*π.χ.*d10) αλλά αυτή η υπόθεση δεν είναι και η πιο λογική. Πιο συγκεκριμένα, προτείνει ότι όποτε προχωράει ο χρόνος, η βάση δεδομένων πρέπει να ενημερώνεται ανάλογα.
Σκέψεις σαν αυτές έχουν οδηγήσει μερικούς συγγραφείς να προτείνουν τη χρήση ενός ειδικού δείκτη, τον οποίο ονομάζουμε **NOW**. Η βασική ιδέα είναι να επιτρέπει σ'αυτό το δείκτη να εμφανίζεται όπου μια τιμή του κατάλληλου σημείου επιτρέπεται. Έτσι για παράδειγμα η σχέση S_DURING μπορεί να περιλαμβάνει μια πλειάδα για τον προμηθευτή S1 με μια τιμή DURING [d04:NOW], αντί την τιμή [d04:d10].
Σημειώνουμε ότι αν η τιμή DURING στην μοναδική πλειάδα για τον προμηθευτή S1 *στη* σχέση είναι [d04:NOW], τότε το αποτέλεσμα του ερωτήματος "πότε τελειώνει το συμβόλαιο του προμηθευτή S1 " είναι NOW(=until further notice).
Αν το σύστημα αξιολογήσει ότι το NOW τη στιγμή του ερωτήματος εκτελείται και απαντά με την τιμή d10, τότε αυτή η απάντηση είναι λάθος, αφού το συμβόλαιο του προμηθευτή S1 δεν έχει τερματιστεί ακόμα. Αν το αποτέλεσμα είναι NOW, τότε αυτό το NOW πρέπει να διερμηνεύεται ως "κάποιο αόριστο διάστημα στο μέλλον". Ακόμα, αν το

ερώτημα εξέρχεται από κάποια εφαρμογή προγράμματος, τότε το NOW πρέπει να επιστρέφεται σε μια μεταβλητή του προγράμματος. Ποια θα είναι όμως η τιμή αυτής της μεταβλητής μετά τον προσδιορισμό του NOW; Τι τύπος δεδομένων θα πρέπει να έχει αυτή η μεταβλητή;
Εάν περιορίζουμε το σχεδιασμό μας στις ιστορικές σχέσεις μόνο, θα πρέπει να προσθέσουμε "κάτι" στη βάση δεδομένων που θα αναπαριστά το "μέχρι νεώτερη πληροφορία(until further notice)", όταν το "μέχρι νεώτερη πληροφορία" είναι αυτό που πραγματικά εννοούμε. Ας θυμηθούμε για άλλη μια φορά την περίπτωση του προμηθευτή του οποίου το συμβόλαιο δεν έχει τελειώσει ακόμα. Όπως εξηγήσαμε και νωρίτερα, αυτός ο προμηθευτής μπορεί να θεωρηθεί ότι έχει συμβόλαιο το οποίο προσεχώς τελειώνει. Μπορούμε να ορίσουμε την τιμή END(DURING) γι' αυτό το προμηθευτή σαν *τελευταία* μέρα, και μετά να αντικαταστήσουμε την τεχνητή αυτή τιμή με την αληθινή, όταν η αληθινή τιμή γίνει αργότερα γνωστή.

Κλείνοντας αυτή την ενότητα, θα δώσουμε έμφαση στο ότι αυτή η παράγραφος περιγράφει μια πιθανή λύση για το πρόβλημα του "τώρα". Η περιγραφή δεν είναι ίδια με την συμβουλή. Δεν είναι και η καλύτερη ιδέα να προσθέσουμε πληροφορίες στην βάση δεδομένων που δεν είναι σωστές.

4.6 Ιστορικές και Παρούσες Σχέσεις Μαζί

Σε προηγούμενη ενότητα περιγράψαμε μια προσέγγιση βασισμένη σε παρούσες σχέσεις μόνο. Το κύριο πρόβλημα μ'αυτή τη προσέγγιση είναι ότι η βάση δεδομένων δεν μπορεί να κρατήσει ιστορικά στοιχεία/εγγραφές(είναι δηλαδή ημι-χρονική). Περιγράψαμε επίσης μια προσέγγιση βασισμένη σε ιστορικές σχέσεις μόνο. Το μεγάλο πρόβλημα μ'αυτή τη προσέγγιση έχει να κάνει με το πρόβλημα του σημείου "NOW". Σ' αυτό το σημείο θα προσπαθήσουμε να χρησιμοποιήσουμε ένα συνδυασμό των δυο προηγούμενων προσεγγίσεων. Αυτός ο συνδυασμός μας επιτρέπει να κρατήσουμε ιστορικές εγγραφές, ενώ θα αποφύγουμε τις λύσεις για τη στιγμή "NOW".

Θα ξεκινήσουμε με την ακόλουθη παρατήρηση:
Υπάρχει μια σημαντική λογική διαφορά ανάμεσα σε παρούσες και ιστορικές πληροφορίες:

1) για τις ιστορικές πληροφορίες γνωρίζουμε την αρχική και τελική χρονική στιγμή ταυτόχρονα,
2) σε αντίθεση με τις παρούσες που γνωρίζουμε την αρχική χρονική στιγμή, αλλά όχι και την τελική

Η διαφορά αυτή προτείνει ότι θα πρέπει να υπάρχουν δύο σύνολα σχέσεων, ένα για την παρούσα κατάσταση, κι ένα για την ιστορική. Στο σχήμα 4.1 βλέπουμε τα ακόλουθα:

- η σχέση S_SINCE είναι η μοναδική παρούσα σχέση. Είναι ακριβώς η ίδια με τη σχέση SSSC_SINCE, με τη μόνη διαφορά ότι για λόγους απλότητας έχουμε συντομεύση το όνομα SSSC_SINCE *σε* S_SINCE

Παράδειγμα 4.1
ιστορικές σχέσεις για
τους προμηθευτές

```
S_SINCE {S#, S#_SINCE,
         SNAME, SNAME_SINCE,
         STATUS, STATUS_SINCE,
         CITY, CITY_SINCE}
        KEY{S#}

S_DURING {S#, DURING}
         KEY{S#, DURING}

S_NAME_DURING {S#, SNAME, DURING}
         KEY{S#, DURING}

S_STATUS_DURING {S#, STATUS, DURING}
         KEY {S#, DURING}

S_CITY_DURING {S#, CITY, DURING}
         KEY {S#, DURING}
```

- οι σχέσεις με το DURING στο όνομα τους είναι οι ιστορικές. Δεν περιλαμβάνουν καμία πλειάδα που να αναφέρεται στην παρούσα ή μελλοντική κατάσταση. Όλες οι πληροφορίες που θα αναπαριστούνταν από τέτοιες πλειάδες στην ενότητα 4.3, αναπαριστώνται τώρα από πλειάδες της παρούσας σχέσης S_SINCE. Αυτές οι ιστορικές σχέσεις δεν περιλαμβάνουν καμιά πλειάδα με τεχνητή τελική χρονική στιγμή. Για παράδειγμα, το όρισμα για την σχέση S_DURING ταμοιάζει τώρα με το εξής:

  Απ' τη μέρα που είναι το αρχικό σημείο του DURING, μέχρι τη μέρα που έιναι το τελικό σημείο του DURING, ο προμηθευτής S# είχε συμβόλαιο. Η μέρα που είναι το τελικό σημείο του DURING ανήκει στο παρελθόν

Η διαδικασία χωρισμού των παρούσων πληροφοριών απ'τις ιστορικές, είναι η διαδικασία της οριζόντιας ανάλυσης. Υποθέστε ότι ξεκινούμε με την πλήρως χρονική σχέση SSSC_DURING, *η* οποία υποθέτουμε ότι περιλαμβάνει πληροφορίες για το μέλλον, όπως και για το παρόν. Τότε:

- πρώτα εισάγουμε την παρούσα σχέση S_SINCE με μια ιδιότητα για κάθε ιδιότητα του SSSC_DURING, εκτός απ'την ιδιότητα του DURING
- δεύτερον, για κάθε ιδιότητα που εμφανίζεται *στην* S_SINCE απ'το πρώτο βήμα, προσθέτουμε την ιδιότητα "since" σ'αυτή τη σχέση
- τρίτον, για τη σχέση SSSC_DURING καταλαβαίνουμε τώρα ότι περιέχει ιστορικές πληροφορίες μόνο

Ας δούμε τώρα τις δύο προτάσεις που αναφερθήκαμε προηγουμένως. Η πρώτη είναι ότι η αρχική και τελική στιγμή είναι γνωστές στις ιστορικές πληροφορίες. Υποθέστε ότι

ξέρουμε ότι ο προμηθευτής S1 είχε συμβόλαιο, αλλά δεν ξέρουμε πότε. Αυτό είναι το πρόβλημα της χαμένης πληροφορίας.
Η δεύτερη είναι ότι γνωρίζουμε την αρχική στιγμή, όχι όμως και την τελική. Πολλές φορές ξέρουμε και την τελική στιγμή. Σ' αυτή την περίπτωση μπορούμε να υιοθετήσουμε την προσέγγιση που κρατά τις ιστορικές σχέσεις.
Στο ερώτημα του ποια προσέγγιση είναι η καλύτερη σε μια δεδομένη κατάσταση, αυτή εξαρτάται απ'τις καταστάσεις. Αισθανόμαστε υποχρεωμένοι να πούμε ότι η συνδυασμένη προσέγγιση, ιστορικών και παρόντων σχέσεων μαζί, είναι η προτιμότερη.

ΚΕΦΑΛΑΙΟ 5: ΤΕΛΕΣΤΕΣ ΣΤΟ ΓΕΝΙΚΕΥΜΕΝΟ ΜΟΝΤΕΛΟ

5.1 Ένωση, τομή και διαφορά

Παρακάτω βλέπουμε την εξής συντομογραφία:

USING (ACL) ◄R1 MINUS R2►

Τα R1 *και* R2 είναι σχεσιάκες εκφράσεις που δηλώνουν τις σχέσεις r1 *και* r2 του ίδιου σχεσιακού τύπου, και το ACL είναι μια λίστα ονομάτων ιδιοτήτων στα οποία κάθε ιδιότητα αναφέρεται: (1) ως κάποιου τύπου διαστήματος και (2) εμφανίζεται και στις δυο σχέσεις. Η γενική έκφραση ορίζεται να είναι σημασιολογικά ίση με την επόμενη έκφραση:

```
PACK
    ((UNPACK R1 ON (ACL))
    MINUS
        (UNPACK R2 ON (ACL)))
ON (ACL)
```

Σημεία που προκύπτουν:

1) Αν και έχουμε αναφερθεί στο τελεστή που ορίστηκε ως "χρονική διαφορά" πρέπει να πούμε ότι δεν ενδιαφερόμαστε για αυτήν την ορολογία, επειδή ο τελεστής δεν είναι ιδιαίτερος στα χρονικά διαστήματα. Θα αναφερόμαστε σ'αυτό ως U_difference ή πιο απλά U_MINUS.
2) Ως συνήθως, οι παρενθέσεις που περιβάλλουν τη λίστα των ονομάτων ιδιοτήτων στον όρο USING, μπορούν να παραλείπονται εάν η λίστα περιέχει μόνο ένα όνομα ιδιοτήτων.
3) Στα υπόλοιπα τμήματα αυτού του κεφαλαίου, θα καθορίσουμε διάφορες εκφράσεις στις οποίες μπορεί να εμφανιστεί ο όρος USING. Σε όλες εκείνες τις εκφράσεις, τα βέλη ◄ *και* ►, όπως στην *περίπτωση* U_MINUS, χρησιμοποιούνται για να οριοθετήσουν τη λειτουργική έκφραση για την οποία ο όρος USING απευθύνεται. Η λειτουργική αυτή έκφραση αντιπροσωπεύει είτε μια επίκληση κάποιου τελεστή σχεσιακής άλγεβρας, είτε μια σχεσιακή σύγκριση.

Στο παρακάτω παράδειγμα βλέπουμε:

```
USING DURING ◄ S_DURING {S#, DURING}
               MINUS
               SP_DURING {S#, DURING} ►
```

Αρχικά, οι προβολές των S_DURING *και* SP_DURING υπολογίζονται πάνω στο {S#, DURING}. Οι δυο αυτές προβολές μετά αποσυμπιέζονται στο DURING, υπολογίζεται η διαφορά τους, και το αποτέλεσμα συμπιέζεται ξανά στο DURING.

Αξίζει να σημειώσουμε ότι το U_MINUS μπορεί να παράγει αποτέλεσμα του οποίου ο αριθμός στοιχείων συνόλου να είναι μεγαλύτερος από εκείνο του αριστερού του τελεστή. Για παράδειγμα, έστω ότι τα R1 *και* R2 είναι:

R1

A
[d02:d04]

R2

A
[d03:d03]

Τότε το USING A ◄R1 MINUS R2► δίνει:

A
[d02:d02]
[d04:d04]

Γυρίζοντας τώρα στην **ένωση(union)** ορίζουμε την έκφραση "U_UNION"

USING (ACL) ◄R1 UNION R2►

σαν συντομογραφία του:

```
PACK
        ((UNPACK R1 ON (ACL))
        UNION
        (UNPACK R2 ON (ACL)))
ON (ACL)
```

Για τα R1, R2 *και* ACL ισχύουν τα ίδια όπως και στο U_MINUS. Τα R1, R2 θα πρέπει να είναι του ίδιου σχεσιακού τύπου και κάθε ιδιότητα που αναφέρεται στο ACL θα πρέπει να είναι ίδιου τύπου διαστήματος και πρέπει να εμφανίζεται ταυτόχρονα στα R1 *και* R2.

Περνώντας τώρα στην **τομή(intersect)**, η έκφραση U_INTERSECT είναι:

USING (ACL) ◄R1 INTERSECT R2►

όπου R1, R2 *και* ACL είναι όπως τα U_MINUS *και* U_UNION, ενώ ορίζονται σαν συντομογραφία του

```
PACK
        ((UNPACK R1 ON (ACL))
        INTERSECT
        (UNPACK R2 ON (ACL)))
ON (ACL)
```

Για παράδειγμα, έστω ότι R1 *και* R2 είναι τα ακόλουθα:

R1

A
[d01:d07]

R2

A
[d02:d02]
[d04:d08]

Τότε *το* USING (ACL) ◄R1 INTERSECT R2► δίνει:

A
[d02:d02]
[d04:d07]

Προσέξτε ότι ο αριθμός στοιχείων συνόλου αυτού του αποτελέσματος είναι μεγαλύτερος απ'αυτόν του R1. Στην πραγματικότητα, όπως ίσως να έχετε προσέξει, το αποτέλεσμα μπορεί να έχει αριθμό στοιχείων συνόλου μεγαλύτερο από αυτόν καθενός τελεστέου.

Βλέποντας τέλος τη **διαφορά(minus)**, η έκφραση

USING (ACL) ◄R1 MINUS R2►

θα ήταν καλύτερα (μάλλον και πιο λογικά) αν έμοιαζε με κάτι σαν αυτό:

R1 U_MINUS (ACL) R2

Μετά απ'όλα αυτά, το U_MINUS είναι βασικά ένας δυαδικός σχεσιακός τελεστής. Αφού χρησιμοποιούμε τη παραπάνω σημείωση για το τελεστή MINUS, θα ήταν πιο συνεπές να κάνουμε το ίδιο και για το U_MINUS. Παρολ'αυτά, δυο δυσκολίες είναι αυτές που δε μας το επιτρέπουν:

- Κάθε μια απ'τις εκδόσεις "U_" *των* MINUS, INTERSECT *και* UNION περιλαμβάνει αυτό που λέμε τρίτος τελεστής, αλλά οι "τρίτοι τελεστές" δεν ταιριάζουν καλά με το συντακτικό στυλ
- Στην περίπτωση των μοναδιαίων τελεστών restrict *και* project και το μοναδιαίο τελεστή EXTEND , είναι δύσκολο να προσαρμοστούν συντακτικά με τις εκδόσεις "U_" που είναι λογικοί και σύμφωνοι με τη σύνταξη των κανονικών εκδόσεων

5.2 Οι Τελεστές RESTRICT ΚΑΙ PROJECT

Ας δούμε τώρα τους μοναδιαίους τελεστές restrict *και* project. Γι' αυτούς τους τελεστές, υπάρχει εξ'ορισμού μια μόνο σχέση που "ξεσυμπιέζεται" στο αρχικό βήμα. Στην περίπτωση του restrict για παράδειγμα, ορίζουμε την έκφραση

USING (ACL) ◄R WHERE P►

που είναι η συντομογραφία του

PACK ((UNPACK R ON (ACL)) WHERE P) ON (ACL)

Κάθε ιδιότητα που αναφέρεται στο ACL πρέπει να είναι ιδιότητα του R και πρέπει να είναι κάποιου τύπου διαστήματος. Παρακάτω βλέπουμε ένα παράδειγμα που διευκρινίζει τη χρήση του U_RESTRICT:

USING DURING ◄S_DURING WHERE
INTERVAL_DATE ([d04:d04]) = DURING►

Ας δούμε τη διαφορά μεταξύ του U_RESTRICT και του ακόλουθου κανονικού τελεστή restrict:

S_DURING WHERE INTERVAL_DATE ([d04:d04]) = DURING

Υποθέστε ότι η μεταβλητή σχέσης S_DURING περιέχει μόνο τις δυο ακόλουθες πλειάδες:

S#	DURING
S2	[d02:d04]
S2	[d07:d10]

Τότε ο κανονικός τελεστής restrict θα επιστρέφει ένα αποτέλεσμα με αριθμό στοιχείων συνόλου μηδέν, ενώ το U_RESTRICT θα επιστρέφει ένα αποτέλεσμα με αριθμό στοιχείων συνόλου ένα.

Για να δούμε τώρα τον τελεστή project, ορίζουμε την έκφραση:

USING (ACL) ◄R {BCL} ►

που είναι η συντομογραφία του

PACK ((UNPACK R ON (ACL)) {BCL}) ON (ACL)

Κάθε ιδιότητα που αναφέρεται στο ACL πρέπει να είναι κάποιου τύπου διάστημα και πρέπει να αναφέρεται στο BCL. Ας δούμε τώρα το παρακάτω ερώτημα:

Query A: πάρτε τα ζευγάρια S#-DURING απ' τους προμηθευτές που μπορούν να εφοδιάσουν τουλάχιστον ένα μέρος κατά τη διάρκεια ενός τουλάχιστον χρονικού διαστήματος, όπου το DURING καθορίζει ένα μέγιστο διάστημα στο οποίο κάποιος προμηθευτής S# μπορούσε στην πραγματικότητα να εφοδιάσει τουλάχιστο ένα μέρος.

Εδώ βλέπουμε τη διατύπωση "U_project" γι'αυτό το ερώτημα:

USING DURING ◄SP_DURING {S#, DURING} ►

5.3 Ο Τελεστής JOIN(συνένωση)
Ορίζουμε την έκφραση "U_ JOIN"

USING (ACL) ◄R1 JOIN R2►

σαν συντομογραφία του

PACK
((UNPACK R1 ON (ACL))
JOIN
(UNPACK R2 ON (ACL)))
ON (ACL)

Κάθε ιδιότητα που αναφέρεται στο ACL θα πρέπει να είναι τύπου διάστημα και πρέπει να εμφανίζονται και στο R1 *και στο* R2. *Σημειώστε* ότι αν τα R1 *και* R2 είναι σχέση του ίδιου τύπου, τότε το U_ JOIN "καταλήγει", όπως θα αναμενόταν, στο U_INTERSECT.

5.4 Οι Τελεστές EXTEND ΚΑΙ SUMMARIZE
Ορίζουμε την έκφραση "U_EXTEND"

USING (ACL) ◄EXTEND R ADD exp AS B►

σαν συντομογραφία του

PACK
(EXTEND (UNPACK R ON (ACL)) ADD exp AS B)
ON (ACL)

Κάθε ιδιότητα που αναφέρεται στο ACL πρέπει να είναι του ίδιου τύπου διάστημα και πρέπει να είναι ιδιότητα του R.
Αντίθετα με τον τελεστή EXTEND, *ο* U_EXTEND μπορεί να επιστρέφει αποτέλεσμα με *αριθμό* στοιχείων συνόλου μεγαλύτερο ή μικρότερο από αυτόν της εισόδου τους. Υποθέστε ότι η μεταβλητή σχέση S_DURING περιέχει ακριβώς τις δυο παρακάτω πλιάδες:

S#	DURING
S2	[d01:d05]
S2	[d03:d04]

Τότε το πρώτο απ'τα παρακάτω U_EXTEND θα επιστρέφει μια σχέση με αριθμό στοιχείων συνόλου πέντε και η δεύτερη θα επιστρέφει μια σχέση με αριθμό στοιχείων συνόλου ένα:

1. USING DURING ◄EXTEND S_DURING ADD BEGIN (DURING) AS X►
2. USING DURING ◄EXTEND S_DURING ADD COUNT (DURING) AS X►

Ορίζουμε τον τελεστή "U_SUMMARIZE"

USING (ACL) ◄SUMMARIZE R1 PER R2 ADD summary AS B►

σαν συντομογραφία του

```
PACK
        (SUMMARIZE (UNPACK R1 ON (ACL))
        PER (UNPACK R2 ON (ACL'))
        ADD summary AS B))
ON (ACL')
```

Κάθε ιδιότητα που αναφέρεται στο ACL πρέπει να είναι τύπου διαστήματος, και πρέπει να είναι ιδιότητα του R1. Το ACL' είναι το ίδιο με το ACL, εξαιρώντας ότι κάθε ιδιότητα του ACL που δεν εμφανίζεται στο R2, απλά αγνοείται.

5.5 Οι Τελεστές GROUP ΚΑΙ UNGROUP

Θα ορίσουμε την παρακάτω έκφραση "U_GROUP"

USING (ACL) ◄R GROUP (BCL) AS C►

σαν συντομογραφία του

```
PACK
      (((UNPACK R ON (ACL)) GROUP (BCL) AS C)
ON (ACL)
```

Κάθε ιδιότητα που αναφέρεται στο ACL πρέπει να είναι διάστημα κάποιου τύπου, πρέπει να είναι ιδιότητα του R και δε θα πρέπει να αναφέρεται στο BCL. Για παράδειγμα υποθέστε η παρακάτω μεταβλητή σχέση SP_DURING είναι ως εξής:

S#	P#	DURING
S2	P1	[d08:d10]
S2	P2	[d09:d10]
S4	P2	[d07:d09]
S4	P4	[d07:d08]

Ας δούμε την έκφραση:

USING DURING ◄SP_DURING GROUP {P#} AS P#_REL►

Αυτό είναι το αποτέλεσμα *για το* UNPACK:

S#	P#	DURING
S2	P1	[d08:d08]
S2	P1	[d09:d09]
S2	P1	[d10:dl0]
S2	P2	[d09:d09]
S2	P2	[d10:dl0]
S4	P2	[d07:d07]
S4	P2	[d08:d08]
S4	P2	[d09:d09]
S4	P4	[d07:d07]
S4	P4	[d08:d08]

Κι' αυτό είναι το αποτέλεσμα για το GROUP:

S#	P#_REL	DURING
S2	P#	[d08:d08]
	P1	
S2	P#	[d09:d09]
	P1	
	P2	
S2	P#	[d10:d10]
	P1	
	P2	
S4	P#	[d07:d07]
	P2	
	P4	
S4	P#	[d08:d08]
	P2	
	P4	
S4	P#	[d09:d09]
	P2	

Τέλος, το αποτέλεσμα για το PACK είναι:

S#	P#_REL	DURING
S2	P#	[d08:d08]
	P1	
S2	P#	[d09:d10]
	P1	
	P2	
S4	P#	[d07:d08]
	P2	
	P4	
S4	P#	[d09:d09]
	P2	

Θα ορίσουμε τώρα την έκφραση "U_GROUP"

USING (ACL) ◄R UNGROUP C►

σαν τη συντομογραφία του

PACK
((UNPACK R ON (ACL)) UNGROUP C)
ON (ACL)

Κάθε ιδιότητα που αναφέρεται στο ACL πρέπει να είναι κάποιου τύπου διάστημα και πρέπει να είναι ιδιότητα του R. Το ACL δεν θα πρέπει να αναφέρει το C.

5.6 Σχεσιακές συγκρίσεις

Σε προηγούμενο κεφάλαιο παρουσιάσαμε κάποιες σχεσιάκες συγκρίσεις σαν μια ξεχωριστή περίπτωση εκφράσεων που λαμβάνουν λογικές τιμές, 0 ή 1(boolean), *τις* οποίες όμως δεν έχουμε χρησιμοποιήσει μέχρι αυτό το σημείο. Παρακάτω σας θυμίζουμε την σύνταξη τους:

<relation comp>
::= <relation exp> <relation comp op> <relation exp>

<relation comp op>
::= = | ≠| ⊆ | ⊂ | ⊇ | ⊃

Οι σχέσεις που δηλώνονται απ'τις δύο σχεσιάκες εκφράσεις, θα πρέπει να είναι σχέσεις του ίδιου τύπου.
Τώρα, οι σχεσιακές συγκρίσεις δεν είναι σχεσιακές διαδικασίες υπό αυτήν τη μορφή, επειδή επιστρέφουν αληθή τιμή, κι όχι μια σχέση. Εντούτοις, μπορούμε να τις υποβάλλουμε στο ίδιο είδος χρήσης που έχουμε απευθυνθεί τους σχεσιακούς τελεστές.

Το σημαντικό είναι, όταν οι σχέσεις στο ερώτημα περιλαμβάνουν τις ιδιότητες διαστήματος, κι αυτό που θέλουμε να κάνουμε είναι να συγκρίνει αποσυμπιεσμένες ισοδύναμες εκφράσεις αυτών των σχέσεων. Με αυτό το στόχο, εισάγουμε πρώτα ένα "U_" αντίστοιχο με την κανονική σύγκριση "ισότητας σχέσης". Για να είμαστε πιο συγκεκριμένοι, ορίζουμε την έκφραση

USING (ACL) ◄R1 = R2►

σαν συντομογραφία του

(UNPACK R1 ON (ACL)) = (UNPACK R2 ON (ACL))

Κάθε ιδιότητα που αναφέρεται στο ACL πρέπει να είναι ιδιότητα διαστήματος και πρέπει να εμφανίζεται μαζί στα R1 *και* R2.
Έστω για παράδειγμα ότι τα R1 *και* R2 είναι όπως τα παρακάτω:

R1

A
[d01:d03]
[d02:d05]
[d04:d04]

R2

A
[d01:d02]
[d03:d05]

Τότε το R1= R2 είναι ψευδές, ενώ το ◄R1 = R2► είναι αληθές. Αξίζει να δούμε το παρακάτω:
Έστω ότι R1 είναι μια σχέση, κι έστω η συμπιεσμένη(packed) και αποσυμπιεσμένη (unpacked) μορφή του R1 ότι είναι το σύνολο ιδιοτήτων PR1 *και* UR1 αντίστοιχα. Ομοίως, έστω ότι η R2 είναι σχέση του ίδιου τύπου με την R1, κι έστω η συμπιεσμένη(packed) και αποσυμπιεσμένη (unpacked) μορφή του R2 είναι το ίδιο σύνολο ιδιοτήτων, με PR2 *και* UR2 αντίστοιχα. Τέλος, έστω ότι το UR1 ⊂ UR2 είναι αληθές. Αυτό δεν σημαίνει ότι και το PR1 ⊂ PR2 είναι αληθές.

Σε αυτό το κομμάτι στρέφουμε την προσοχή μας στο κρίσιμο θέμα των περιορισμών ακεραιότητας που μπορεί να ισχύσουν για τα χρονικά στοιχεία.
Αρχίζουμε υποθέτοντας ότι η βάση δεδομένων έχει σχεδιαστεί ως συνδυασμός τρεχόντων και ιστορικών σχέσεων και επομένως αποτελείται απ' τις ακόλουθες επτά σχέσεις:

S_SINCE
S_DURING
S_NAME_DURING
S_STATUS_DURING
S_CITY_DURING
SP _SINCE
SP_ DURING

Πρέπει να γίνει σαφές ότι:

1. Μέσα *στην* τρέχουσα σχέση S_S1NCE *τα* ζευγάρια ιδιοτήτων {SNAME SNAME_SINCE}, {STATUS, STATUS_SINCE} *και*{ CITY, CITY_SINCE} *θα* εκθέσουν όλα *την* παρόμοια συμπεριφορά.
2. Οι ιστορικές σχέσεις S_NAME_DURING, S_STATUS_DURING *και* S_CITY_DURING θα εκθέσουν όλες μια παρόμοια συμπεριφορά

Μπορούμε να αγνοήσουμε και τα δύο:

- Τα ζευγάρια ιδιοτήτων {SNAME SNAME_SINCE} *και* {CITY, CITY_SINCE}στη σχέση S_S1NCE *και*
- Οι σχέσεις S_NAME_DURING *και* S_CITY_DURING

Για τους σκοπούς του υπολοίπου κεφαλαίου, μπορούμε να απλοποιήσουμε τη βάση δεδομένων μας. Για να είμαστε συγκεκριμένοι, μπορούμε να πάρουμε τη βάση δεδομένων μας όπως παρουσιάζεται στο παράδειγμα 5.1

Παράδειγμα 5.1

```
S SINCE {S#, S#_SINCE, STATUS, STATUS_SINCE}
          KEY {S#}
SP_SINCE {S#, P#, SINCE}
          KEY {S#, P#}
          FOREIGN KEY {S#} REFERENCES S_SINCE
S_DURING {S#, DURING}
          KEY {S#, DURING}
S_STATUS_DURING {S#, STATUS, DURING}
          KEY {S#, DURING}
SP DURING {S#, P# DURING}
       KEY {S#, P# DURING}
```

Τα παραδείγματα 5.2 και 5.3 δείχνουν ένα σύνολο από τιμές δειγμάτων γι' αυτήν την βάση δεδομένων.
Σημείωση:οι συγκεκριμένες τιμές σ' αυτά τα παραδείγματα δεν ανταποκρίνονται ακριβώς στις συνηθισμένες τιμές δειγμάτων του παραδείγματος 5.1.

Παράδειγμα 5.2

S_SINCE

S#	S#_SINCE	STATUS	STATUS_SINCE
S1	d04	20	d06
S2	d01	10	d01
S3	d03	30	d03
S4	d04	20	d08
S5	d02	30	d02

SP_SINCE

S#	P#	SINCE
S1	P1	d04
S1	P2	d05
S1	P3	d09
S1	P4	d05
S1	P5	d04
S1	P6	d06
S2	P1	d08
S2	P2	d09
S3	P2	d08
S4	P5	d05

Παράδειγμα 5.3
Ενδεικτικές τιμές για τις σχέσεις
S_DURING, SP_DURING
Και S_STATUS_DURING

S_DURING

S#	DURING
S2	[d02:d04]
S6	[d03:d05]

SP_DURING

S#	P#	DURING
S2	P10	[d02:d04]
S2	P2	[d03:d03]
S3	P5	[d05:d01]
S4	P2	[d06:d09]
S4	P4	[d04:d08]
S6	P3	[d03:d03]
56	P3	[d05:d05]

S_STATUS_DURING

S#	STATUS	DURING
S1	15	[d04:d05]
S2	5	[d02:d02]
S2	10	[d03:d04]
S4	10	[d04:d04]
S4	25	[d05:d07]
S6	5	[d03:d04]
S6	7	[d05:d05]

Κατηγορήματα

Εδώ είναι ανεπίσημα τα κατηγορήματα για τις σχέσεις του παραδείγματος 5.1. Πρώτα οι τρέχουσες σχέσεις:

- S_SINCE: ο προμηθευτής S# ήταν στο πλαίσιο της σύμβασης από S#_SINCE (και όχι την ημέρα αμέσως πριν την S#_SINCE) και είχε την θέση STATUS από *την* STATUS_SINCE (και όχι την ημέρα αμέσως πριν την STATUS_SINCE).
- SP_SINCE: ο προμηθευτής S# ήταν ικανός να προμηθεύσει το μέρος P# από την SINCE (και όχι την ημέρα αμέσως πριν την SINCE).

Οι ιστορικές σχέσεις (εδώ χρησιμοποιούμε το b *και* e *για να* δηλώσουμε την BEGIN(DURING) *και* END(DURING) αντίστοιχα):

- S_DURING: ο προμηθευτής S# ήταν στο πλαίσιο της σύμβασης από την ημέρα b μέχρι την ημέρα e (και όχι την ημέρα αμέσως πριν την b ή μετά την e)
- S_STATUS_DURING: ο προμηθευτής S# έχει την θέση STATUS από την ημέρα b μέχρι την ημέρα e (και όχι την ημέρα αμέσως πριν την b ή μετά την e)
- SP_DURING: ο προμηθευτής S# ήταν ικανός να προμηθεύσει το μέρος P# από την ημέρα b μέχρι την ημέρα e (και όχι την ημέρα αμέσως πριν την b ή μετά την e)

Θυμηθείτε ότι όταν λέμε ένας δεδομένος προμηθευτής έχει κάποια θέση μια δεδομένη ημέρα, εννοούμε ότι ο εν λόγω προμηθευτής έχει ακριβώς μια θέση αυτήν την ημέρα Επιπλέον, σύμφωνα με τις συζητήσεις στην παράγραφο 4.6, υποθέτουμε ότι η σχέση since περιέχει τρέχοντα δεδομένα μόνο, και οι σχέσεις during περιέχουν ιστορικά δεδομένα μόνο. Κάθε τιμή END(DURING) είναι μικρότερη από την σημερινή ημερομηνία. Εάν οι τιμές δειγμάτων που φαίνονται στα παραδείγματα 5.2 και 5.3 ισχύουν, σήμερα πρέπει να είναι το λιγότερο η 10^7 ημέρα.

Πρωτεύοντα και ξένα κλειδιά

Τα υποψήφια κλειδιά που ισχύουν στην βάση δεδομένων του παραδείγματος 5.1 είναι αυτεξήγητα και στην περίπτωση των σχέσεων since υπάρχουν λίγα να πούμε γι' αυτά. Σε αντίθεση υπάρχουν περισσότερα να πούμε στην περίπτωση των σχέσεων during.
Όσον αφορά τα ξένα κλειδιά, υπάρχει στην πραγματικότητα μόνο ένα στην απλοποιημένη βάση δεδομένων μας και αυτό είναι {S#} στην τρέχουσα σχέση SP_SINCE, το οποίο είναι ξένο κλειδί που αναφέρεται στο μόνο υποψήφιο κλειδί, στην πραγματικότητα στο πρωτεύον κλειδί.

FOREIGN KEY {S#} REFERENCES S_SINCE

Αυτός ο περιορισμός, που εμφανίζεται ως τμήμα του καθορισμού της σχέσης SP_ SINCE, απεικονίζει το γεγονός ότι οποιοσδήποτε προμηθευτής αυτήν την περίοδο που είναι ικανός να προμηθεύσει κάποιο μέρος πρέπει να είναι αυτήν την περίοδο στο πλαίσιο της σύμβασης. Όπως παρατηρήσαμε σε προηγούμενο κεφάλαιο, αυτός ο ξένος

βασικός περιορισμός κλειδιού από μόνος του δεν είναι ικανοποιητικός. Αυτό που χρειαζόμαστε πραγματικά είναι κάτι γενικότερο προκειμένου να επιβληθεί ο περιορισμός ότι όποτε ένας δεδομένος προμηθευτής είναι, ήταν, ή θα είναι σε θέση να προμηθεύσει κάποιο μέρος, τότε ο προμηθευτής αυτός είναι, ήταν, ή θα είναι στο πλαίσιο της σύμβασης σε αυτόν τον ίδιο χρόνο.

Παρακάτω εξετάζουμε, τρία γενικά προβλήματα που μπορούν να εμφανιστούν με τις χρονικές βάσεις δεδομένων. Αναφερόμαστε σε αυτά τα προβλήματα ως πρόβλημα πλεονασμού, πρόβλημα περίφρασης, *και* πρόβλημα αντίφασης, αντίστοιχα.

5.7 Το πρόβλημα πλεονασμού

Αρχίζουμε εξετάζοντας τη σχέση S_STATUS_DURING. Δεδομένου ότι το {S#, DURING} είναι ένα υποψήφιο κλειδί γι' αυτό τη σχέση, ένας ορισμός για το tutorial D μπορεί να μοιάζει μ' αυτό που ακολουθεί:

```
VAR S_STATUS_DURING RELATION
        {S# S#, STATUS INTEGER, DURING INTERVAL_DATE}
        KEY {S#, DURING};          /* προειδοποίηση--ανεπαρκής! */
```

Όπως προτείνει το σχόλιο, ο βασικός περιορισμός, αν και λογικά σωστός, είναι ανεπαρκής. Είναι ανεπαρκής επειδή αποτυγχάνει να αποτρέψει τη σχέση να περιέχει και τις δύο από τις ακόλουθες πλειάδες συγχρόνως:

S#	STATUS	DURING
S4	25	[d05:d06]

S#	STATUS	DURING
S4	25	[d06:d07]

Όπως μπορείτε να δείτε, αυτές οι δύο πλειάδες επιδεικνύουν έναν πλεονασμό, δεδομένου ότι η θέση για τον προμηθευτή S4 *την* $6^{η}$ ημέρα δηλώνεται δύο φορές. Σαφώς, θα ήταν καλύτερο εάν αντικαταστήσουμε τις δύο πλειάδες που παρουσιάστηκαν με την ακόλουθη ενιαία πλειάδα:

S#	STATUS	DURING
S4	25	[d05:d07]

Παρατηρήστε τώρα ότι εάν οι δύο αρχικές πλειάδες ήταν οι μόνες πλειάδες σε κάποια σχέση με δυο πλειάδες και συμπιέζαμε αυτήν την σχέση στη DURING, θα τερματίζαμε με μια σχέση μιας πλειάδας που περιέχει μια ενιαία πλειάδα. Μπορούμε να πούμε ότι η πλειάδα που παρουσιάστηκε είναι μια "συμπιεσμένη" πλειάδα, που λαμβάνεται *με τη* συμπίεση των δύο αρχικών πλειάδων στις ιδιότητες DURING. Εξετάστε τη δεξιά πλειάδα. Αυτή λέει μεταξύ άλλων ότι ο προμηθευτής S4 δεν είχε τη θέση 25 την ημέρα αμέσως πριν από την $6^{η}$ ημέρα, αλλά η άλλη πλειάδα (στα αριστερά) λέει ότι ο *προμηθευτής* S4 είχε τη θέση 25 την $5^{η}$ ημέρα, και φυσικά η $5^{η}$ ημέρα είναι η ημέρα αμέσως πριν από την $6^{η}$ ημέρα.

5.8 Το πρόβλημα περίφρασης

Ο βασικός περιορισμός για τη σχέση S_STATUS_DURING είναι ανεπαρκής και για άλλο λόγο. Για να είμαστε συγκεκριμένοι αποτυγχάνει να αποτρέψει τη σχέση να περιέχει και τις δύο ακόλουθες πλειάδες συγχρόνως:

S#	STATUS	DURING
S4	25	[d05:d05]

S#	STATUS	DURING
S4	25	[d06:d07]

Εδώ δεν υπάρχει κανένας πλεονασμός υπό αυτήν τη μορφή, αλλά υπάρχει μια ορισμένη περίφραση, δεδομένου ότι παίρνουμε δύο πλειάδες για να πούμε κάτι το οποίο θα μπορούσε να ειπωθεί καλύτερα με ακριβώς μία ενιαία συμπιεσμένη πλειάδα.

S#	STATUS	DURING
S4	25	[d05:d07]

Πράγματι, μην αντικαθιστώντας τις δύο αρχικές πλειάδες από αυτήν την συμπιεσμένη πλειάδα θα σήμαινε, πάλι, ότι η σχέση θα ήταν παραβίαση του κατηγορήματός του, όπως μπορεί εύκολα να επιβεβαιωθεί.

Πώς να διορθώσουμε το πρόβλημα πλεονασμού και περίφρασης

Τώρα, πρέπει να είναι σαφές ότι, προκειμένου να αποφευχθούν οι πλεονασμοί και οι περιφράσεις, αυτό που πρέπει να κάνουμε είναι να επιβάλλουμε κάποιον περιορισμό, τον οποίο ονομάζουμε περιορισμό A(constraint A) κατά μήκος των ακόλουθων γραμμών:

Περιορισμός Α: Εάν οποιαδήποτε στιγμή η σχέση S_STATUS_DURING περιέχει δύο ευδιάκριτες πλειάδες που είναι ίδιες εκτός από τις τιμές DURING i1 *και* i2, τότε η i1 MERGES i2 πρέπει να είναι ψευδής.

Θυμηθείτε απ' το 3° κεφάλαιο ότι, MERGES (συγχώνευση) είναι το λογικό OR *των* OVERLAPS (επικαλύψεων) και MEETS (συναντήσεων). Η αντικατάσταση της συγχώνευσης (MERGES) από την επικάλυψη (OVERLAPS) στον περιορισμό Α δίνει τον περιορισμό που πρέπει να επιβάλουμε προκειμένου να αποφευχθεί το πρόβλημα πλεονασμού. Αντικαθιστώντας αυτόν με την συνάντηση (MEETS) δίνει τον περιορισμό που πρέπει να επιβάλουμε προκειμένου να αποφευχθεί το πρόβλημα περίφρασης.

Πρέπει να είναι σαφές επίσης ότι υπάρχει ένας πολύ απλός τρόπος να επιβληθεί ο περιορισμός Α: δηλαδή, με την κράτηση της συμπιεσμένης σχέσης πάντα στις ιδιότητες DURING. Δημιουργούμε λοιπόν έναν νέο περιορισμό που τον ονομάζουμε PACKED ON, που μπορεί να εμφανιστεί σε μια καθορισμένη σχέση, όπως εδώ:

```
VAR S_STATUS_DURING RELATION
    {S# S#, STATUS INTEGER, DURING INTERVAL_DATE}
    PACKED ON DURING                /*προειδοποίηση-είναι ακόμη
    KEY {S#, DURING};                ανεπαρκής*/
```

Ο PACKED ON DURING εδώ είναι περιορισμός, περιορισμός μιας σχέσης από την άποψη του σχεδίου ταξινόμησης που περιγράφεται στο κεφάλαιο 1 στη σχέση S_STATUS_DURING. Ερμηνεύεται ως εξής: Η σχέση S_STATUS_DURING πρέπει πάντα να κρατηθεί συμπιεσμένη σε DURING; με άλλα λόγια, πρέπει πάντα να είναι ίδιο με το αποτέλεσμα του PACK S_STATUS_DURING ON DURING (υπονοώντας μεταξύ άλλων ότι αυτή η τελευταία έκφραση μπορεί πάντα να αντικατασταθεί από το απλούστερο S_STATUS_DURING).

Αυτή η ειδική σύνταξη αρκεί έτσι ώστε να λύσει τα προβλήματα πλεονασμού και περίφρασης. Με άλλα λόγια, λύνει το πρόβλημα που εξηγείται από τον περιορισμό που *αναφερθ*ήκαμε ως περιορισμός XFTI στο κεφάλαιο 2.

Παρατηρούμε ότι ένα επιχείρημα μπορεί να προβληθεί για την παροχή της ειδικής σύνταξης για το πρόβλημα του πλεονασμού και όχι το πρόβλημα της περίφρασης. Εντούτοις, πρέπει να δούμε ακόμα ένα αληθινά πειστικό επιχείρημα για μια τέτοια θέση, επομένως, η προτίμησή μας είναι να πετύχουμε το στόχο μας και να αποφύγουμε και τα δύο προβλήματα αμέσως.

Σημειώστε ότι ένας περιορισμός PACKED ON πρέπει να επιτραπεί για να διευκρινίσει δύο ή περισσότερες ιδιότητες, κατά συνέπεια:

PACKED ON (ACL)

όπου το ACL είναι η λίστα των ονομάτων ιδιοτήτων. (Εσωκλείουμε αυτήν την λίστα σε παρενθέσεις και όχι σε αγκύλες επειδή η ακολουθία ονομάτων ιδιοτήτων είναι σημαντική. Όπως συνηθίζεται, οι παρενθέσεις μπορούν να παραλειφθούν εάν η διευκρινισμένη ακολουθία περιέχει μόνο ένα όνομα ιδιοτήτων.)

5.9 Το πρόβλημα της αντίφασης

Συνεχίζουμε να συζητάμε για τον ορισμό της σχέσης S_STATUS_DURING. Δυστυχώς, ο PACKED ON και οι βασικοί περιορισμοί σε αυτόν τον ορισμό δεν είναι ακόμη αρκετά επαρκής, ακόμα και όταν τα παίρνουμε μαζί, αφού *απο*τυγχάνουν να αποτρέψουν τη σχέση να περιέχει και τις δύο ακόλουθες πλειάδες συγχρόνως:

S#	STATUS	DURING
S4	10	[d04:d06]

S#	STATUS	DURING
S4	25	[d05:d07]

Εδώ ο προμηθευτής S4 παρουσιάζεται να έχει μια θέση 10 και 25 τις ημέρες 5 και 6 και μια αδύνατη παρούσα κατάσταση. Με άλλα λόγια, έχουμε μια αντίφαση. *Στην* πραγματικότητα, η σχέση αποτελεί παραβίαση του κατηγορήματός του άλλη μια φορά, επειδή κάθε προμηθευτής υποτίθεται ότι έχει ακριβώς μια θέση οποιαδήποτε δεδομένη ημέρα.

ΣΗΜΕΙΩΣΗ: Για να πούμε ότι ο κάθε προμηθευτής υποτίθεται ότι έχει ακριβώς μια θέση οποιαδήποτε δεδομένη ημέρα μπορούμε να πούμε, τυπικότερα, ότι εάν επρόκειτο να αποσυμπιέσουμε τη σχέση S_STATUS_DURING ON DURING, με αυτόν τον τρόπο παράγουμε ένα αποτέλεσμα στο οποίο κάθε τιμή DURING συνίσταται ενός διαστήματος μονάδων, η λειτουργική εξάρτηση {S#,DURING}>{STATUS} θα κρατούσε αυτό το αποτέλεσμα.

Πώς να διορθώσουμε το πρόβλημα της αντίφασης

Πρέπει να είναι σαφές ότι, προκειμένου να αποφευχθούν οι αντιφάσεις πρέπει να επιβάλουμε κάποιον περιορισμό, τον οποίο ονομάζουμε περιορισμό Β, κατά μήκος των ακόλουθων γραμμών:

Περιορισμός Β: Εάν οποιαδήποτε στιγμή η σχέση S_STATUS_DURING περιέχει δύο πλειάδες που έχουν την ίδια τιμή S#, αλλά διαφέρουν στην τιμή θέσης τους(STATUS), τότε οι τιμές DURING i1 *και* i2 πρέπει να είναι έτσι ώστε το i1 OVERLAPS i2 να είναι ψευδή.

Σημειώστε προσεκτικά ότι, όπως έχουμε ήδη δει, ο περιορισμός Β προφανώς δεν επιβάλλεται από το μόνο γεγονός ότι η σχέση κρατιέται συμπιεσμένη(packed) *στο* DURING. Ακόμα προφανέστερο, είναι ότι δεν επιβάλλεται από το γεγονός ότι το {S#, DURING} είναι υποψήφιο κλειδί. Υποθέστε ότι η σχέση κρατήθηκε αποσυμπιεσμένη πάντα στις ιδιότητες DURING (αγνοούμε προς το παρόν το γεγονός ότι αυτή η υπόθεση είναι αδύνατη, δεδομένου ότι η σχέση πρόκειται στην πραγματικότητα να κρατηθεί συμπιεσμένο στο DURING). Τότε:

- Όπως δηλώθηκε νωρίτερα, όλες οι τιμές DURING σε αυτή *την αποσυμπιεσμένη* μορφή θα ήταν μοναδιαία διαστήματα και έτσι αποτελεσματικά θα αντιστοιχούσαν στα μεμονωμένα χρονικά σημεία.

- Το μοναδικό υποψήφιο κλειδί γι' αυτήν την αποσυμπιεσμένη μορφή εξακολουθεί να είναι το {S#, DURING}, επειδή οποιοσδήποτε προμηθευτής που είναι στο πλαίσιο της σύμβασης σε κάθε δεδομένη στιγμή έχει μόνο μία θέση STATUS αυτήν την στιγμή. (Δεν θα μπορούσαν δύο ευδιάκριτες πλειάδες με αυτήν την αποσυμπιεσμένη μορφή να έχουν την ίδια τιμή S# και να "επικαλύπτουν" διαστήματα μονάδων, επειδή ο μόνος τρόπος να γίνει μια τέτοια επικάλυψη θα ήταν εάν τα δύο διαστήματα μονάδων ήταν ένα, που σημαίνει ότι οι δύο πλειάδες θα ήταν αντίγραφα του ενός ή του άλλου, και ως εκ τούτου στην πραγματικότητα η ίδια πλειάδα.)

Εάν επρόκειτο να επιβάλουμε τον περιορισμό ότι το {S#, DURING}είναι υποψήφιο κλειδί για την αποσυμπιεσμένη μορφή UNPACK S_STATUS_DURING ON DURING τότε θα επιβάλουμε τον περιορισμό Β. Ας επινοήσουμε έναν νέο περιορισμό τον WHEN/THEN ο οποίος μπορεί να εμφανιστεί σε ένα καθορισμένο διάστημα οπού ένας απλός βασικός περιορισμός μπορεί να εμφανιστεί, όπως εδώ:

```
VAR S_STATUS_DURING RELATION
   {S# S#,  STATUS INTEGER, DURING INTERVAL_DATE}
   PACKED ON DURING
  WHEN UNPACKED ON DURING THEN KEY {S#, DURING)
  KEY {S#, DURING);
```

Το WHEN UNPACKED ON DURING THEN KEY {S#, DURING) εδώ είναι ένας περιορισμός, ένας περιορισμός σχέσης. Ερμηνεύεται ως εξής: η σχέση S_STATUS_DURING πρέπει να είναι πάντα έτσι ώστε καμία από τις δύο πλειάδες στο αποτέλεσμα της έκφρασης UNPACK S_STATUS_DURING να έχει την ίδια τιμή για την συνδυασμένη ιδιότητα {S#, DURING}. Το {S#, DURING} είναι υποψήφιο κλειδί *για* UNPACK S_STATUS_DURING. Αυτή η ειδική σύνταξη τείνει να λύσει το πρόβλημα της αντίφασης.

Προκύπτει από την προηγούμενη συζήτηση ότι το WHEN/THEN, PACKED ON *και οι* βασικές προδιαγραφές μπορούν να διορθώσουν όλα τα προβλήματα ακεραιότητας που έχουμε συζητήσει σε αυτό το κεφάλαιο μέχρι τώρα. Ένα τελικό σημείο για να κλείσει το παρόν τμήμα: Φυσικά, ένας περιορισμός WHEN/THEN πρέπει να επιτραπεί για να διευκρινίσει την αποσυμπίεση σε δύο ή περισσότερες ιδιότητες, κατά συνέπεια:

```
WHEN UNPACKED ON (ACL) THEN KEY {BCL}
```

(όπου το ACL *και* BCL είναι και οι δύο λίστες ονομάτων ιδιοτήτων).

ΣΗΜΕΙΩΣΗ: Εσωκλείουμε την λίστα ACL σε παρένθεση και όχι σε αγκύλες ακόμα κι αν η ακολουθία ονομάτων ιδιοτήτων είναι ασήμαντη, για λόγους που εξηγούνται στο κεφάλαιο 3. Όπως συνηθίζεται οι παρενθέσεις μπορούν να παραλειφθούν εάν η διευκρινισμένη ακολουθία περιέχει μόνο ένα όνομα ιδιοτήτων. Σημειώστε ότι σε κάθε περιορισμό WHEN/THEN, κάθε ιδιότητα που αναφέρεται στο ACL πρέπει να αναφερθεί και στο BCL (*δηλ., το* ACL πρέπει να είναι ένα υποσύνολο του BCL). Κάθε WHEN/THEN περιορισμός της μορφής WHEN... THEN KEY {K+} υπονοεί μια συμβατική βασική προδιαγραφή κλειδιού *της μορφής* KEY{K}, όπου το K είναι κάποιο υποσύνολο του K +.

5.10 Συνδυασμός των προδιαγραφών

Έχουμε συναντήσει τρία είδη περιορισμών που μπορούν να εμφανιστούν σε έναν ορισμό σχέσης: KEY, PACKED ON και WHEN/THEN περιορισμούς. Εκ πρώτης όψεως, υπάρχουν οκτώ πιθανοί συνδυασμοί περιορισμών που μπορούν να γίνουν για οποιοδήποτε δεδομένη σχέση. Αλλά ποιος απ' αυτούς πραγματικά κάνει;
Εάν ορίσουμε ότι τουλάχιστον ένας ρητός περιορισμός κλειδιού απαιτείται πάντα, τότε αυτό μειώνει αμέσως τις οκτώ δυνατότητες σε τέσσερις.

Έστω ότι R είναι μια σχέση με μια ιδιότητα διαστήματος. Ξέρουμε από τα προηγούμενα δύο τμήματα ότι το R μπορεί να χρειαστεί δύο περιορισμούς τον PACKED ON και τον WHEN/THEN. Έτσι οι δυνατότητες που πρέπει να εξετάσουμε είναι:

- Το R έχει έναν περιορισμό PACKED ON αλλά δεν έχει περιορισμό WHEN/THEN.
- Το R έχει έναν περιορισμό WHEN/THEN, αλλά δεν έχει περιορισμό PACKED ON.
- Το R δεν έχει ούτε PACKED ON περιορισμό ούτε WHEN/THEN περιορισμό.

Αυτές οι τρεις δυνατότητες αποτελούν το αντικείμενο των επόμενων τριών τμημάτων.

5.11 PACKED ON χωρίς τον περιορισμό WHEN/THEN

Εξετάστε την ιστορική σχέση S_DURING, με τις ιδιότητες S# και DURING. Πρέπει να είναι προφανές ότι το S_DURING είναι ευαίσθητο στα προβλήματα του πλεονασμού και της περίφρασης ανάλογων με αυτά που συζητήθηκαν για τη σχέση S_STATUS_DURING στις παραγράφους 5.2 και 5.3 αντίστοιχα. Εντούτοις, δεν είναι ευαίσθητο στο πρόβλημα αντίφασης που συζητείται στην παράγραφο 5.4. (**ΕΡΩΤΗΣΗ**: γιατί δεν είναι; **ΑΠΑΝΤΗΣΗ**: Επειδή είναι το μόνο υποψήφιο κλειδί του είναι ο συνδυασμός ιδιοτήτων {S#, DURING) και έτσι αυτό δεν μπορεί να περιέχει δύο πλειάδες που έρχονται σε αντίθεση η μία με την άλλη.) Έτσι, ο περιορισμός PACKED ON DURING ισχύει, αλλά ο περιορισμός WHEN UNPACKED ON DURING THEN KEY {S#, DURING) δεν είναι απαραίτητος.

Τα παρακάτω χρησιμεύουν ως ένας επαρκής καθορισμός για αυτή τη σχέση:

```
VAR S_DURING RELATION
    {S# S#, DURING INTERVAL_DATE}
  PACKED ON DURING
  KEY {S#, DURING};
```

Επαναλαμβάνουμε, ότι ένας περιορισμός WHEN/THEN είναι περιττός για αυτή τη σχέση. Εντούτοις, δεν θα ήταν λογικά λανθασμένο να διευκρινιστεί το ένα, όπως εδώ:

```
VAR S DURING RELATION
    {S# S#, DURING INTERVAL_DATE}
     PACKED ON DURING
    WHEN UNPACKED ON DURING THEN KEY {S#, DURING}
    KEY {S#, DURING};
```

Πάλι, εδώ ο περιορισμός WHEN/THEN δεν είναι λάθος, αλλά μπορεί να οδηγήσει σε κάποια ανεπάρκεια στην εφαρμογή εάν το σύστημα προσπαθήσει τυφλά να το επιβάλει. Ο ακόλουθος είναι ένας πιθανός καθορισμός για αυτή τη σχέση:

```
VAR SP_DURING RELATION
   {S# S#, P# P#, DURING INTERVAL_DATE}
    PACKED ON DURING
    KEY {S#, P#, DURING};
```

Από αυτά τα παραδείγματα, βλέπουμε ότι εάν η σχέση είναι "all key" έπειτα κανένας WHEN/THEN περιορισμός δεν απαιτείται. Αλλά δεν ισχύει ότι εάν κανένας WHEN/THEN περιορισμός δεν απαιτείται, τότε η σχέση είναι "all key".

5.12 WHEN/THEN χωρίς τον περιορισμό PACKED ON

Υποθέστε ότι μας δίνεται μια σχέση TERM που αντιπροσωπεύει τους Αμερικανικούς προεδρικούς όρους, με τις ιδιότητες DURING (διάρκεια) και PRESIDENT (πρόεδρος) και το μόνο κλειδί υποψηφίων {DURING}. Μια αξία δειγμάτων παρουσιάζεται στο σχήμα 5.4.

Παράδειγμα 5.4
Ενδεικτικές τιμές
Για την σχέση
TERM

TERM

DURING	PRESIDENT
[1974:1976]	Ford
[1977:1980]	Carter
[1981:1984]	Reagan
[1985:1988]	Reagan
[1993:1996]	Clinton
[1997:2000]	Clinton

Αυτό το παράδειγμα θίγει ορισμένα σημαντικά σημεία:

1. Πρώτον, οι προεδρικοί όροι δηλώνονται συνήθως υπό την μορφή επικαλυπτόμενων *διαστημάτων*. Η αλήθεια είναι, ότι η "κοκκοποίηση" αυτών των διαστημάτων όπως δηλώνονται συνήθως είναι λάθος, οι προεδρικοί όροι εκτείνονται από μια ημέρα εγκαινίασης (μια ημέρα τον Ιανουάριο) στην επόμενη, και έτσι, παραδείγματος χάριν, ο Ford ήταν Πρόεδρος για το πρώτο μέρος του 1977 *και* ο Carter ήταν Πρόεδρος για το υπόλοιπο αυτού του έτους.

2. Δεύτερον, τα μικρά ονόματα των Προέδρων δεν είναι απαραίτητα μοναδικά.
3. Τρίτον, εκτός από τους περιορισμούς κλειδιού που είναι η αρχική εστίαση αυτού του κεφαλαίου, υπάρχουν ορισμένοι πρόσθετοι περιορισμοί ακεραιότητας που ισχύουν και πρέπει να δηλωθούν και να επιβληθούν για την σχέση TERM. Εδώ είναι μερικοί από αυτούς:

    a) Υπάρχει ακριβώς ένας πρόεδρος οποιαδήποτε στιγμή
    b) Κανένας δεν επιτρέπεται να υπηρετήσει ως πρόεδρος για περισσότερες από δύο περιόδους
    c) Καμία περίοδος δεν επιτρέπεται να υπερβεί τα τέσσερα χρόνια

Πρέπει να είναι σαφές ότι ο περιορισμός PACKED ON DURING δεν πρέπει να διευκρινιστεί για την σχέση TERM, επειδή ένας τέτοιος περιορισμός θα ανάγκαζε δύο πλειάδες Reagan "να συμπιεστούν" σε μία και ομοίως τις δύο πλειάδες Clinton. Πρέπει να είναι σαφές ότι απαιτείται ένας περιορισμός WHEN/THEN προκειμένου να αποφευχθεί η δυνατότητα των σχέσεων να περιέχουν και τις δύο πλειάδες συγχρόνως:

DURING	PRESIDENT
[1985:1994]	Reagan

DURING	PRESIDENT
[1993:1996]	Clinton

Χωρίς έναν τέτοιο περιορισμό, η σχέση TERM θα ήταν σαφώς ευαίσθητο στο πρόβλημα αντίφασης. Επομένως, παρακάτω μπορεί να είναι ένας κατάλληλος καθορισμός γι' αυτή την σχέση:

```
VAR TERM RELATION
        {DURING INTERVAL_..., PRESIDENT NAME}
        WHEN UNPACKED ON DURING THEN KEY {DURING}
        KEY {DURING};
```

Υπάρχουν περαιτέρω θέματα που θίγονται σχετικά με αυτό το παράδειγμα. Ωστόσο, θέλουμε να αποφύγουμε τη δυνατότητα η σχέση να περιέχει και τις ακόλουθες πλειάδες συγχρόνως:

DURING	PRESIDENT
[1993:1995]	Clinton

DURING	PRESIDENT
[1994:1996]	Clinton

(ένα παράδειγμα του προβλήματος πλεονασμού). Ωστόσο, σίγουρα δεν θέλουμε να αποφύγουμε τη δυνατότητα της σχέσης να περιέχει και τις ακόλουθες πλειάδες συγχρόνως:

DURING	PRESIDENT
[1993:1996]	Clinton

DURING	PRESIDENT
[1997:2000]	Clinton

Πρέπει να πούμε ότι κατά την άποψή μας το παράδειγμα αποτυγχάνει. Το σημείο είναι, φυσικά, ότι δεν υπάρχει καμία περίφραση που περιλαμβάνει τις δύο πλειάδες που παρουσιάστηκαν. Στην πραγματικότητα, οι πληροφορίες θα χάνονταν εάν αυτές οι πλειάδες επρόκειτο να αντικατασταθούν από μία ενιαία συμπιεσμένη πλειάδα. Συγκεκριμένα, η πληροφορία ότι ένα από τα DURING διαστήματα αντιστοιχούν στην πρώτη περίοδο του Clinton και άλλο στη δεύτερη του. Το πραγματικό πρόβλημα είναι ότι ο "αριθμός περιόδου" δεν έχει αντιπροσωπευθεί ως ρητή ιδιότητα στη σχέση. Εάν προσθέσουμε μια τέτοια ιδιότητα όπως υποδεικνύεται στο σχήμα 5.5, το "πρόβλημα περίφρασης" πηγαίνει μακριά. (η σχέση μπορεί ακόμα να περιλάβει ένα ζευγάρι πλειάδας που θα αντικαθίσταται καλύτερα από μία ενιαία πλειάδα, παραδείγματος χάριν, να περιλάβει δύο πλειάδες για τον Carter, μία με τιμή DURING [1977:1978] και μία άλλη με μια τιμή DURING [1979:1980].

Αυτή η παρούσα κατάσταση πραγματικά είναι ένα παράδειγμα του προβλήματος περίφρασης. Αλλά ακριβώς η ίδια κατάσταση θα μπορούσε να εμφανιστεί προτού προσθέσουμε την ιδιότητα TERM#! Το πρόβλημα είναι, ότι προτού προσθέσουμε αυτήν την ιδιότητα, το σύστημα δεν θα μπορούσε να πει τη διαφορά μεταξύ αυτού του γνήσιου παραδείγματος της περίφρασης και του προηγούμενου παράδειγμα που θα μπορούσε να ονομαστεί "ψευδή περίφραση.")

Παράδειγμα 5.5
Η σχέση TERM
Με μια ιδιότητα
TERM#
Ενδεικτικές τιμές

TERM

DURING	PRESIDENT	TERM#
[1974:1976]	Ford	1
[1977:1980]	Carter	1
[1981:1984]	Reagan	1
[1985:1988]	Reagan	2
[1993:1996]	Clinton	1
[1997:2000]	Clinton	2

Φυσικά, αυτή η αναθεωρημένη έκδοση της σχέσης TERM *ικανοποιεί* ακόμα τον περιορισμό υποψήφιου κλειδιού KEY{DURING}. Όπως μπορείτε πιθανώς να δείτε, η σχέση είναι τώρα ευαίσθητη και στα τρία από τα συνηθισμένα προβλήματά μας (πλεονασμός, περίφραση, και αντίφαση). Ο πλεονασμός θα εμφανιστεί εάν η σχέση *περιέχει*, παραδείγματος χάριν, και τις δύο ακόλουθες πλειάδες συγχρόνως:

DURING	PRESIDENT	TERM#
[1977:1979]	Carter	1

DURING	PRESIDENT	TERM#
[1978:1980]	Carter	1

Όπως σημειώνεται ήδη, η περίφραση θα εμφανιστεί εάν η σχέση περιέχει, παραδείγματος χάριν, και τις δύο ακόλουθες πλειάδες συγχρόνως:

DURING	PRESIDENT	TERM#
[1977:1978]	Carter	1

DURING	PRESIDENT	TERM#
[1979:1980]	Carter	1

Και η αντίφαση θα εμφανιστεί εάν η σχέση περιέχει, παραδείγματος χάριν, και τις δύο ακόλουθες πλειάδες συγχρόνως:

DURING	PRESIDENT	TERM#
[1977:1980]	Carter	1

DURING	PRESIDENT	TERM#
[1974:1977]	Ford	1

Τώρα, μπορούμε να αποφύγουμε τα προβλήματα πλεονασμού και περίφρασης με τη διευκρίνιση του περιορισμού

PACKED ON DURING

Μπορούμε να αποφύγουμε το πρόβλημα αντίφασης με τη διευκρίνιση του περιορισμού

WHEN UNPACKED ON DURING THEN KEY {DURING}

Έτσι ένας πιθανός ορισμός της σχέσης για να αποφύγουμε και τα τρία προβλήματα είναι ο ακόλουθος:

```
VAR TERM RELATION
  {DURING INTERVAL_... PRESIDENT NAME, TERM# INTEGER}
  PACKED ON DURING
 WHEN UNPACKED ON DURING THEN KEY {DURING}
 KEY {DURING};
```

Στην πραγματικότητα, αυτή η σχέση ικανοποιεί πρόσθετα τον περιορισμό

KEY {PRESIDENT, TERM#}

(δηλ., ο συνδυασμός {PRESIDENT, TERM#} είναι υποψήφιο κλειδί, καμία από τις δύο πλειάδες δεν είχαν ποτέ τις ίδιες τιμές PRESIDENT *και* TERM. Και αυτός ο πρόσθετος περιορισμός κλειδιού κάνει τον περιορισμό PACKED ON περιττό. Ο λόγος είναι ότι από *τη* στιγμή που το {PRESIDENT, TERM#} είναι ένα κλειδί υποψηφίων, η συμπίεση της σχέσης στο DURING δεν μπορεί ενδεχομένως να έχει οποιαδήποτε επίδραση.

Εδώ είναι ο τελικός ορισμός για τη σχέση TERM:

```
VAR TERM RELATION
   {DURING INTERVAL_ ..., PRESIDENT NAME, TERM# INTEGER}
   WHEN UNPACKED ON DURING THEN KEY {DURING}
   KEY {DURING}
   KEY {PRESIDENT, TERM#};
```

Η εισαγωγή της ιδιότητας TERM#, ενώ λύνει μερικά προβλήματα, εισάγει επίσης άλλα: το πρόβλημα της μοναδικότητας ενός δεύτερου κλειδιού υποψηφίων. Γι' αυτόν τον λόγο, μπορεί να υποστηριχτεί ότι το αρχικό σχέδιο (χωρίς TERM#) είναι προτιμητέο.

Εξετάζουμε ένα άλλο ζήτημα που προκύπτει από το ίδιο παράδειγμα. Στον καθορισμό *του* TERM που παρουσιάζεται ανωτέρω ο συνδυασμός των περιορισμών

WHEN UNPACKED ON DURING THEN KEY {DURING}

and

KEY {DURING}

φαίνεται σαν να περιλαμβάνει κάποια περιττή προδιαγραφή δηλαδή σαν να λέει το ίδιο πράγμα δύο φορές.

Η προδιαγραφή WHEN/THEN σημαίνει ότι εάν η σχέση TERM κρατήθηκε αποσυμπιεσμένη *στο* DURING, τότε το {DURING} θα ήταν ακόμα ένα υποψήφιο κλειδί. Αλλά εάν η σχέση TERM κρατήθηκε αποσυμπιεσμένη στο DURING, τότε κάθε τιμή DURING θα ήταν ένα διάστημα μονάδων, κάθε τέτοιο διάστημα θα εμφανιζόταν με αυτήν την αποσυμπιεσμένη μορφή σε ακριβώς μία πλειάδα, που συνδέεται με ακριβώς έναν συνδυασμό τιμών PRESIDENT *και* TERM #. Εάν επρόκειτο τώρα να συμπιέσουμε αυτή την σχέση στο DURING τότε κάθε τιμή DURING στο αποτέλεσμα, ανεξάρτητα απ' το εάν είναι ένα διάστημα μονάδων ή όχι, θα εμφανίζεται επίσης σε ακριβώς μία πλειάδα και θα συνδέεται με ακριβώς έναν συνδυασμό PRESIDENT *και* TERM # τιμών.

Πριν προσπαθήσουμε να βγάλουμε οποιαδήποτε συμπεράσματα από αυτήν την συζήτηση, ας εξετάσουμε την σχέση S_STATUS_DURING άλλη μια φορά, με τον καθορισμό ως εξής:

VAR S_STATUS_DURING RELATION
(S# S#, STATUS INTEGER, DURING INTERVAL_DATE}
PACKED ON DURING
WHEN UNPACKED ON DURING THEN KEY {S#, DURING}
KEY {S#, DURING};

Η προδιαγραφή WHEN/THEN υπονοεί ότι κάθε τιμή {S#, DURING} που εμφανίζεται στην σχέση S_STATUS_DURING εμφανίζεται σε ακριβώς μία πλειάδα και σχετίζεται με ακριβώς μια τιμή STATUS. Δεν μπορούμε όμως να συμπεράνουμε απ' αυτό το γεγονός ότι το{S#, DURING} είναι υποψήφιο κλειδί για το S_STATUS_DURING μπορούμε να καταλήξουμε μόνο στο ότι είναι ένα υπερκλειδί γι' αυτή την σχέση.

Προκειμένου να διευκρινίσουμε το προηγούμενο παράδειγμα (όσον αφορά το π.χ για τους προμηθευτές), υποθέτουμε ότι:

a) Δεν αλλάζει ποτέ η θέση του προμηθευτή
b) Κανένας προμηθευτής δεν επιτρέπεται να είναι στο πλαίσιο της σύμβασης κατά *τη* διάρκεια δύο διαφορετικών διαστημάτων.

Ο καθορισμός της σχέσης μπορεί να απλοποιηθεί ως εξής:

VAR S_STATUS_DURING RELATION

{S# S#, STATUS INTEGER, DURING INTERVAL_DATE}
KEY {S#};

Το μόνο κλειδί υποψηφίων για το S_STATUS_DURING είναι τώρα *το* {S#}, επειδή για κάθε προμηθευτή που αντιπροσωπεύεται σε αυτή την σχέση υπάρχει μόνο μία αντίστοιχη θέση (status) και ένα αντίστοιχο διάστημα κατά τη διάρκεια του οποίου αυτός ο προμηθευτής ήταν στο πλαίσιο της σύμβασης. Συγχρόνως, το μόνο κλειδί υποψηφίων για την αποσυμπιεσμένη μορφή είναι ο συνδυασμός {S#, DURING} επειδή πολλοί προμηθευτές μπορούν να είναι στο πλαίσιο της σύμβασης συγχρόνως.

5.13 Ούτε PACKED ON ούτε WHEN/THEN περιορισμός

Υποθέστε ότι μας δίνεται μια σχέση INFLATION *που αντιπροσωπεύει* το ποσοστό πληθωρισμού για μια συγκεκριμένη χώρα κατά τη διάρκεια συγκεκριμένων χρονικών διαστημάτων. Οι ιδιότητες είναι DURING *και* PERCENTAGE, και το μόνο υποψήφιο κλειδί είναι το {DURING}. Μια τιμή δειγμάτων δίνεται στο σχήμα 5.6, που δείχνει *ότι* το ποσοστό πληθωρισμού ήταν 18% για τους πρώτους τρεις μήνες του έτους, ανέβηκε σε 20% για τους επόμενους τρεις μήνες, έμεινε σε 20% πάλι για τους επόμενους τρεις μήνες (αλλά ανέβηκε σε 25% στον $7^{ο}$ μήνα), και υπολόγισε κατά μέσο όρο σε 20% για το έτος συνολικά.

σχήμα 5.6

DURING	PERCENTAGE
[m01:m03]	18
[m04:m06]	20
[m07:m09]	20
[m07:m07]	25
.....	..
[m01:m12]	20

Πρέπει να είναι σαφές ότι ο περιορισμός PACKED ON DURING δεν πρέπει να διευκρινιστεί για την σχέση INFLATION, επειδή ένας τέτοιος περιορισμός θα προκαλούσε τρεις πλειάδες με PERCENTAGE = 20 για να συμπιεστεί σε μία, και θα χάναμε τις πληροφορίες ότι το ποσοστό πληθωρισμού για τους μήνες 4 μέχρι 6 και τους μήνες 7 μέχρι 9, καθώς επίσης και για το έτος συνολικά ήταν 20%. Πρέπει να είναι σαφές ότι ο μόνος πιθανός περιορισμός είναι ο WHEN/THEN που θα μπορούσε λογικά να διευκρινιστεί γι' αυτή την σχέση.

WHEN UNPACKED ON DURING THEN KEY {DURING, PERCENTAGE}

Αυτή η διευκρίνιση μας λέει πολλά περισσότερα από το ότι οι πλειάδες είναι μοναδικές στο αποτέλεσμα του UNPACK INFLATION ON DURING. Έτσι μπορούμε να πούμε ότι η σχέση INFLATION δεν φαίνεται να είναι θέμα για κανέναν περιορισμό WHEN/ THEN. Ένας προσδιορισμός Tutorial D είναι:

VAR INFLATION RELATION
{DURING INTERVAL_..... , PERCENTAGE INTEGER}
KEY {DURING};

Οι σχέσεις INFLATION *και* TERM φαίνεται να μην χρειάζονται έναν περιορισμό PACKED ON. Με ποιον τρόπο διαφέρουν αυτές οι δυο σχέσεις η μια απ' την άλλη; Διαφέρουν σημαντικά;

Οι δυο σχέσεις στην πραγματικότητα μοιάζουν η μια με την άλλη δεδομένου ότι η λειτουργική εξάρτηση {DURING} ->{PRESIDENT,TERM#} κρατά στην σχέση TERM. Υποθέτουμε την έκδοση της σχέσης που περιλαμβάνει μία ρητή ιδιότητα TERM# και την λειτουργική εξάρτηση {DURING} ->{PERCENTAGE} που κρατά στην σχέση INFLATION. Εντούτοις, διαφέρουν από μια σημαντική άποψη, δηλαδή ως εξής: Εάν επρόκειτο να αποσυμπιέσουμε κάθε σχέση στο DURING, τότε:

- Η λειτουργική εξάρτηση {DURING} ->{PRESIDENT,TERM#}θα εξακολουθεί να διατηρείται στην αποσυμπιεσμένη μορφή του TERM.
- Σε αντίθεση, η λειτουργική εξάρτηση {DURING} ->{PERCENTAGE} *δεν θα* εξακολουθεί να διατηρείται στην αποσυμπιεσμένη μορφή του INFLATION

Μια δεδομένη τιμή PERCENTAGE στην σχέση INFLATION είναι μια ιδιότητα του διαστήματος DURING που λαμβάνεται ολόκληρη και δεν είναι μια ιδιότητα των μεμονωμένων χρονικών σημείων που αποτελούν αυτό το διάστημα. Για *να το* τοποθετήσουμε διαφορετικά, ακριβώς επειδή το ποσοστό πληθωρισμού, για παράδειγμα, το διάστημα [m07:m09] ήταν 20%, δεν μπορούμε να συμπεράνουμε ότι το ποσοστό πληθωρισμού, για παράδειγμα, για το διάστημα [m07:m07] ήταν επίσης 20%. Και πράγματι δεν ήταν, όπως δείχνει το σχήμα 5.6.

Εάν θέλουμε να αντιπροσωπεύσουμε ορισμένες ιδιότητες δηλαδή, τα ποσοστά για ορισμένες οντότητες, δηλαδή τα χρονικά διαστήματα, τότε αυτά τα χρονικά διαστήματα είναι πράγματι ευδιάκριτες οντότητες και πρέπει να δοθούν ευδιάκριτες IDs που μπορεί να χρησιμοποιηθούν αλλού στη βάση δεδομένων ως αναφορές στις εν λόγω οντότητες. Το σχήμα 5.7 είναι μια αναθεωρημένη έκδοση του σχήματος 5.6, που παρουσιάζει τι συμβαίνει στην σχέση INFLATION εάν αυτή η προσέγγιση υιοθετηθεί.

Όπως μπορείτε να δείτε από αυτό το παράδειγμα, {DURING}, αν και φυσικά είναι ακόμα ένα κλειδί υποψηφίων, δεν είναι πλέον μοναδικό, και η ιδέα της διευκρίνισης του περιορισμού PACKED ON DURING τώρα σαφώς δεν έχει κανένα νόημα.

Η προβολή της σχέσης INFLATION *στο* DURING *και* PERCENTAGE

INFLATION {DURING, PERCENTAGE}

Παράδειγμα 5.7
Η σχέση INFLATION
Με μια ιδιότητα ID

ID	DURING	PERCENTAGE
h3	[m01:m03]	18
f7	[m04:m06]	20
x4	[m07:m09]	20
Z0	[m07:m07]	25
..	.....	....
q8	[m01:m12]	20

5.14 Νέα υποψήφια κλειδιά

Παρά τις συζητήσεις των τελευταίων τριών τμημάτων, φαίνεται στην πράξη ότι οι σχέσεις με ιδιότητες διαστήματος θα είναι στην συνέχεια θέμα για τους περιορισμούς WHEN/THEN *και* PACKED ON. Για αυτόν τον λόγο, φαίνεται επιθυμητό να βρούμε κάποια σύνταξη που περιλαμβάνει τη λειτουργία και των τριών. Επομένως προτείνουμε μια στενογραφία και για να είμαστε συγκεκριμένοι, προτείνουμε ότι ο ορισμός οποιουδήποτε δεδομένης σχέσης R πρέπει να επιτραπεί για να περιλάβει μια προδιαγραφή της μορφής

USING (ACL) KEY {K}

Εδώ:

1. Το ACL και το K είναι και τα δύο λίστες ονομάτων ιδιοτήτων, και κάθε ιδιότητα που αναφέρεται στο ACL πρέπει επίσης να αναφερθεί και στο K.
2. Όπως συνηθίζεται, οι παρενθέσεις μπορούν να παραλειφθούν εάν το ACL περιέχει μόνο ένα όνομα ιδιοτήτων.
3. Η προδιαγραφή καθορίζεται να είναι στενογραφία για το συνδυασμό του ακόλουθων τριών περιορισμών:

PACKED ON (ACL)
WHEN UNPACKED ON (ACL) THEN KEY {K}
KEY {K}

Υποθέστε τώρα ότι η λίστα των ονομάτων ιδιοτήτων ACL είναι άδεια. Τότε

USING ( ) KEY {K}

αυτό είναι στενογραφία για τον συνδυασμό των περιορισμών

PACKED ON ( )
WHEN UNPACKED ON ( ) THEN KEY {K}
KEY {K}

Έτσι:

- Πρώτον η σχετική σχέση δεν πρέπει να κρατηθεί συμπιεσμένη σε καμία ιδιότητα. Αλλά η συμπίεση μιας σχέσης r σε καμία ιδιότητα επιστρέφει απλά το r, έτσι το υπονοούμενο PACKED ON στην προδιαγραφή δεν έχει καμία επίδραση.
- Δεύτερον, η σχετική σχέση πρέπει να είναι έτσι ώστε εάν αποσυμπιέζεται σε κάποια ιδιότητα, τότε το {K} είναι ένα υποψήφιο κλειδί για το αποτέλεσμα. Αλλά αποσυμπιέζοντας μια σχέση r, σε καμία ιδιότητα δεν επιστρέφει απλά το r. Έτσι, η υπονοούμενη προδιαγραφή WHEN/THEN απλά σημαίνει ότι το K είναι ένα υποψήφιο κλειδί για την σχετική σχέση, και ο υπονοούμενος κανονικός περιορισμός κλειδιού είναι έτσι περιττός.

Ακολουθεί ότι μπορούμε να πάρουμε έναν κανονικό περιορισμό κλειδιού της μορφής K να είναι στενογραφία για ένα ορισμένο U_key. Συγκεκριμένα, μια από τη μορφή USING () KEY {K}. Με άλλα λόγια, οι κανονικοί περιορισμοί κλειδιού είναι ουσιαστικά ακριβώς μια ειδική περίπτωση της προτεινόμενης νέας σύνταξής μας! Έτσι, εάν επαναπροσδιορίζουμε τη σύνταξη ενός κανονικού περιορισμού κλειδιού (α < candidate key def >) έτσι

<candidate key def>
::= [USING (ACL)] KEY {K}

Είναι χρήσιμο να αναφερόμαστε στους περιορισμούς PACKED ON *και* WHEN/THEN που δεν έχουν καμία λογική επίδραση ως τετριμμένοι περιορισμοί. Ειδικότερα, οι περιορισμοί PACKED ON () *και* WHEN UNPACKED ON () THEN είναι και οι δύο από αυτή την άποψη, επίσης είναι

1. *ο* περιορισμός PACKED ON (ACL), εάν το σύνολο ιδιοτήτων που δεν περιλαμβάνονται *στο* ACL είναι ένα υπερκλειδί για την εν λόγω σχέση, και
2. *ο* περιορισμός WHEN ...THEN KEY {K}, εάν το K είναι το σύνολο όλων των ιδιοτήτων της εν λόγω σχέσης.

Κλείνουμε αυτό το κεφάλαιο με την παρουσίαση της γενικής επίδρασης των προηγούμενων συντακτικών απλοποιήσεων στο τρέχων παράδειγμά μας.

Παράδειγμα 5.8

S_SINCE {S#, S#_SINCE, STATUS, STATUS_SINCE}
KEY {S#}

SP_SINCE {S#, P#, SINCE}
KEY {S#, P#}
FOREIGN KEY {S#} REFERENCES S_SINCE

S_DURING {S#, DURING}
USING DURING KEY {S#, DURING}

S_STATUS_DURING {S#, STATUS, DURING}
USING DURING KEY {S#, DURING}

SP_DURING {S#, P#, DURING}
USING DURING KEY {S#, P#, DURING}

ΚΕΦΑΛΑΙΟ 6:ΓΕΝΙΚΟΙ ΠΕΡΙΟΡΙΣΜΟΙ ΚΑΙ ΠΕΡΙΟΡΙΣΜΟΙ ΑΚΕΡΑΙΟΤΗΤΑΣ

6.1 Εισαγωγή

Στο προηγούμενο κεφάλαιο χρησιμοποιήσαμε τις ιστορικές σχέσεις για να επεξηγήσουμε την ανάγκη για τη λειτουργία των PACKED ON *και* WHEN/THEN περιορισμών, που ενισχύουν την U_key *στενογραφία* και το γεγονός ότι οι (βασικοί) περιορισμοί κλειδιού μας μπορούν να θεωρηθούν ως ειδική περίπτωση αυτής της στενογραφίας. Σε αυτό το κεφάλαιο, υιοθετούμε μια διαφορετική στρατηγική. Για να είμαστε συγκεκριμένοι, γυρίζουμε πίσω μια στιγμή στο σχήμα 6.1 και εξετάζουμε, σε πολύ γενικούς όρους, όλους τους περιορισμούς που μπορεί να θελήσουμε σε μια χαρακτηριστική χρονική βάση δεδομένων, όπως το τρέχων παράδειγμά μας που περιλαμβάνει τους προμηθευτές και τις αποστολές. Αυτή η συζήτηση εμφανίζεται στην παράγραφο 6.2. Κατόπιν εξετάζουμε, στις παραγράφους 6.3, 6.4, και 6.5, αντίστοιχα, τι συμβαίνει σε αυτούς τους περιορισμούς εάν η βάση δεδομένων περιέχει (1) τρέχουσες ("since") σχέσεις μόνο, (2) ιστορικές ("during") σχέσεις μόνο, ή (3) ένα μίγμα και των δύο.

Σημείωση: Πρέπει να πούμε ότι, όπως στο κεφάλαιο 4, αυτό το κεφάλαιο ασχολείται περισσότερο με τα χρονικά στοιχεία. Ο κύριος λόγος είναι ότι το χρονικό στοιχείο, από την ίδια την φύση του, πρέπει συχνά να ικανοποιήσει μια ορισμένη «πυκνότητα» περιορισμών, που σημαίνει, ότι ορισμένοι όροι πρέπει να ικανοποιηθούν σε κάθε σημείο μέσα σε ορισμένα διαστήματα. Παραδείγματος χάριν, εάν η βάση δεδομένων που παρουσιάζει τον προμηθευτή S1 να είναι στο πλαίσιο της σύμβασης από την $4^{η}$ ημέρα, πρέπει επίσης να παρουσιάζει τον προμηθευτή S1 να έχει κάποια θέση από την $4^{η}$ ημέρα. Τέτοιοι περιορισμοί "πυκνότητας" δεν ισχύουν απαραιτήτως για άλλα είδη στοιχείων για τα οποία οι έννοιες των προηγούμενων κεφαλαίων, όπως η έννοια διαστήματος, ισχύουν.

6.2 Οι εννέα απαιτήσεις

Υπάρχουν εννέα γενικές απαιτήσεις που θέλουμε να εξετάσουμε. Εμπίπτουν σε τρεις ομάδες των τριών. Τα πρώτα τρία είναι όλα της μορφή "εάν η βάση δεδομένων παρουσιάζει έναν δεδομένο προμηθευτή που είναι στο πλαίσιο της σύμβασης μια δεδομένη ημέρα ή σε ζευγάρι διαδοχικών ημερών, τότε κάποιος άλλος όρος πρέπει επίσης να ικανοποιηθεί":

Απαίτηση R1: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να είναι στο πλαίσιο της σύμβασης την ημέρα d, τότε αυτό πρέπει να περιέχει ακριβώς μία πλειάδα που να δείχνει αυτό το γεγονός.

Απαίτηση R2: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να είναι στο πλαίσιο της σύμβασης τις ημέρες d *και* d + 1, τότε αυτό πρέπει να περιέχει ακριβώς μία πλειάδα που να δείχνει αυτό το γεγονός.

Απαίτηση R3: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να είναι στο πλαίσιο της σύμβασης την ημέρα d, τότε πρέπει επίσης να παρουσιάζει τον προμηθευτή Sx να έχει κάποια θέση την ημέρα d .

Παρατηρήστε ότι η απαίτηση R1 έχει να κάνει με την αποφυγή του πλεονασμού και η απαίτηση R2 με την αποφυγή της περίφρασης. Η απαίτηση R3 έχει να κάνει με αυτό που αναφέραμε στην παράγραφο 6.1 ως "πυκνότητα".
Οι επόμενες τρεις απαιτήσεις είναι όλες της μορφής "εάν η βάση δεδομένων παρουσιάζει έναν δεδομένο προμηθευτή να έχει μια δεδομένη θέση σε δεδομένη ημέρα ή σε ζευγάρι διαδοχικών ημερών, τότε κάποιος άλλος όρος πρέπει επίσης να ικανοποιηθεί". Έχουν μια ομοιότητα με τις απαιτήσεις R1-R3, όπως θα διαπιστώσετε.

Απαίτηση R4: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να έχει κάποια θέση την ημέρα d, τότε πρέπει να περιέχει ακριβώς μία πλειάδα που να δείχνει αυτό το γεγονός.

Απαίτηση R5: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να έχει την ίδια θέση(status) *τις* ημέρες d *και* d + 1, τότε αυτό πρέπει να περιέχει ακριβώς μία πλειάδα που να δείχνει αυτό το γεγονός.

Απαίτηση R6: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να έχει κάποια θέση την ημέρα d, τότε πρέπει επίσης να παρουσιάζει τον προμηθευτή Sx να είναι στο πλαίσιο της σύμβασης την ημέρα d .

Η απαίτηση R4 έχει να κάνει με την αποφυγή του πλεονασμού και επίσης με την αποφυγή της αντίφασης. Η απαίτηση R5 έχει να κάνει με την αποφυγή της περίφρασης και η απαίτηση R6 έχει να κάνει με την "πυκνότητα".

Οι τελικές τρεις απαιτήσεις είναι όλες της μορφής "εάν η βάση δεδομένων παρουσιάζει τον δεδομένο προμηθευτή ικανό να προμηθεύσει ένα δεδομένο μέρος σε κάποια δεδομένη ημέρα ή σε ζευγάρι διαδοχικών ημερών, τότε κάποιος άλλος όρος πρέπει επίσης να ικανοποιηθεί":

Απαίτηση R7: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx ικανό να προμηθεύσει κάποιο συγκεκριμένο μέρος Py την ημέρα d, τότε πρέπει να περιέχει ακριβώς μία πλειάδα που να δείχνει αυτό το γεγονός.

Απαίτηση R8: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx ικανό να προμηθεύσει το ίδιο μέρος Py τις ημέρες d *και* d + 1, τότε πρέπει να περιέχει ακριβώς μια πλειάδα που να δείχνει αυτό το γεγονός.

Απαίτηση R9: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx ικανό να προμηθεύσει κάποιο μέρος Py την ημέρα d, τότε πρέπει επίσης να παρουσιάζει τον προμηθευτή Sx να είναι στο πλαίσιο της σύμβασης την ημέρα d.

Η απαίτηση R7 έχει να κάνει με την αποφυγή του πλεονασμού, η απαίτηση R8 *με την* αποφυγή της περίφρασης, και η απαίτηση R9 με την "πυκνότητα".

Προσέξτε ότι:

- Η απαίτηση R1 υπονοεί ότι κανένας προμηθευτής δεν μπορεί να είναι κάτω από δύο ευδιάκριτες συμβάσεις συγχρόνως.
- Η απαίτηση R4 υπονοεί ότι κανένας προμηθευτής δεν μπορεί να έχει δύο ευδιάκριτες τιμές θέσης συγχρόνως.
- Η απαίτηση R7 υπονοεί ότι κανένας προμηθευτής δεν μπορεί να έχει δύο ευδιάκριτες "δυνατότητα να προμηθεύσει κάποιο συγκεκριμένο μέρος" συγχρόνως.

6.3 Μόνο Τρέχουσες Σχέσεις

Εξετάζουμε τώρα μια χρονική βάση δεδομένων που περιέχει τις τρέχοντες σχέσεις μόνο. Η βάση δεδομένων (δείτε το σχήμα 6.1) αποτελείται ουσιαστικά από τα δύο σχέσεις "since" του σχήματος 5.1. Προσέξτε ότι αυτές οι δυο σχέσεις, δεδομένου ότι δεν περιλαμβάνουν καμία ιδιότητα διαστήματος, δεν περιλαμβάνουν και κανέναν packed on ή when/then περιορισμό.

Παράδειγμα 6.1
Παρούσες σχέσεις μόνο

```
S _SINCE {S#, S#_SINCE, STATUS, STATUS_SINCE}
        KEY {S#}

SP_SINCE {S#, P#, SINCE}
        KEY{S#, P#}
        FOREIGN KEY {S#} REFERENCES S_SINCE
```

Παράδειγμα 6.2
Παρούσες σχέσεις μόνο
Ενδεικτικές τιμές

S#	S#_SINCE	STATUS	STATUS_SINCE
S1	d04	20	d06
S2	d01	10	d01
S3	d03	30	d03
S4	d14	20	d14
S5	d02	30	d02

S#	P#	SINCE
S1	P1	d04
S1	P2	d05
S1	P3	d09
S1	P4	d05
S1	P5	d04
S1	P6	d06
S2	P1	d08
S2	P2	d09
S3	P2	d08
S4	P5	d14

Φυσικά αυτήν η βάση δεδομένων είναι ημιχρονική: δεν μπορεί να αναπαραστήσει ιστορικές πληροφορίες (εκτός από αυτό που μπορεί να προκύψει από τις "since" τιμές). Σε αντίθεση, μπορεί να αντιπροσωπεύσει τις μελλοντικές πληροφορίες. Για να είμαστε συγκεκριμένοι:

- Οι σχέσεις S_SINCE *και* SP_SINCE και οι δύο περιέχουν τις υπονοούμενες πληροφορίες σχετικά με το μέλλον. Θυμηθείτε το κεφάλαιο 4 στο οποίο είπαμε, ότι εάν, παραδείγματος χάριν, μία πλειάδα S_SINCE παρουσιάζει τον προμηθευτή Sx να είναι κάτω από το πλαίσιο της σύμβασης από την ημέρα d, τότε αυτό σημαίνει ότι ο προμηθευτής Sx ήταν, είναι, ή θα είναι στο πλαίσιο της σύμβασής κάθε μέρα από την ημέρα d μέχρι "την τελευταία ημέρα"

- Αυτές οι σχέσεις μπορεί να περιέχουν ρητές πληροφορίες σχετικές με το μέλλον. Θυμηθείτε ξανά το κεφάλαιο 4 στο οποίο είπαμε ότι για παράδειγμα η τιμή S#_SINCE για κάποιον προμηθευτή Sx μπορεί να είναι κάποια μελλοντική ημέρα d, που σημαίνει ότι ο ενδεικνυόμενος προμηθευτής Sx μπορεί να τοποθετηθεί κάτω από το πλαίσιο της σύμβασης σ' αυτήν την μελλοντική ημέρα. Ο προμηθευτής S4 είναι η προκείμενη περίπτωση στο σχήμα 6.2 (άλλη μια φορά υποθέτουμε ότι σήμερα είναι η $10^{η}$ ημέρα).

Απαίτηση Rl:Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να είναι *στο* πλαίσιο της σύμβασης την ημέρα d τότε πρέπει να περιέχει ακριβώς μία πλειάδα που να δείχνει αυτό το γεγονός.
Ο βασικός περιορισμός κλειδιού στην σχέση S_SINCE φροντίζει αυτήν την απαίτηση. Χωρίς αυτή την σχέση μπορεί να περιέχει, παραδείγματος χάριν, και τις δύο ακόλουθες πλειάδες συγχρόνως:

S#	S#_SINCE	.....
S1	d04	.....

S#	S#_SINCE	.....
S1	d06	.....

Και οι δύο πλειάδες παρουσιάζουν μεταξύ άλλων ότι ο προμηθευτής s1 ήταν στο πλαίσιο της σύμβασης την $7^{η}$ ημέρα (που υποθέτει, φυσικά, ότι είναι σήμερα τουλάχιστον η $7^{η}$ ημέρα) εάν και τα δύο εμφανίζονταν , η απαίτηση R1 θα παραβιαζόταν.

Απαίτηση R2: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να είναι *στο* πλαίσιο της σύμβασης τις ημέρες d *και* d+1, κατόπιν πρέπει να περιέχει ακριβώς μια πλειάδα που δείχνει αυτό το γεγονός.

Ο βασικός περιορισμός κλειδιού στην σχέση S_SINCE φροντίζει αυτήν την απαίτηση. Χωρίς αυτή την σχέση μπορεί να περιέχει, παραδείγματος χάριν, και τις δύο ακόλουθες πλειάδες συγχρόνως:

S#	S#_SINCE	.....
S1	d04	.....

S#	S#_SINCE	.....
S1	d05	.....

Απαίτηση R3: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να είναι στο πλαίσιο της σύμβασης την ημέρα d, τότε πρέπει να παρουσιάζει τον προμηθευτή Sx *να* έχει κάποια θέση την ημέρα d.
Αυτή η απαίτηση δεν μπορεί να επιβληθεί σε μια βάση δεδομένων που είναι ημιχρονική. Το κατηγόρημα για την S_SINCE (ελαφρώς απλουστευμένο) είναι:

Ο προμηθευτής S# ήταν στο πλαίσιο της σύμβασης από S#_SINCE και είχε τη θέση STATUS από STATUS_SINCE.

Είναι λογικό για την S_SINCE να περιλαμβάνει μία πλειάδα στην οποία η τιμή STATUS_SINCE d' είναι μεγαλύτερη από τη τιμή d *της* S#_SINCE (βλ., μία πλειάδα S_SINCE *για τον* προμηθευτή s1 στο σχήμα 6.2). Και εάν ο προμηθευτής Sx είναι ο προμηθευτής που αντιπροσωπεύεται από αυτή την πλειάδα, τότε η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να είναι στο πλαίσιο της σύμβασης όλο το διάστημα [d:d'-1], αλλά δεν παρουσιάζει τον προμηθευτή Sx να έχει κάποια θέση σε όλο αυτό το διάστημα. Αυτά τα τελευταία είναι ιστορικές πληροφορίες που δεν μπορούν να αντιπροσωπευθούν σε αυτήν την βάση δεδομένων.

Η παρούσα κατάσταση που διευκρινίστηκε είναι αρκετά χαρακτηριστική του *πραγματικού* κόσμου των «χρονικών» βάσεων δεδομένων (π.χ βάσεις δεδομένων που περιλαμβάνουν μερικές χρονικές πληροφορίες αλλά εφαρμόζονται χωρίς άμεση χρονική υποστήριξη από το σύστημα). Για παράδειγμα δεν θα ήταν καθόλου ασυνήθιστο να βρείτε μια σχέση σαν αυτή σε μια τέτοια βάση δεδομένων:

EMP {EMP#, DATE_Of_HIRE, SALARY, DATE_Of_LAST_INCREASE}

Σε μια τέτοια σχέση, θα υπάρξουν πιθανώς πολλές πλειάδες στις οποίες η τιμή DATE_OF_LAST_INCREASE είναι μεγαλύτερη από την τιμή DATE_OF_HIRE.

Απαίτηση R4: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να έχει κάποια θέση την ημέρα d, τότε πρέπει να περιέχει ακριβώς μία πλειάδα που να δείχνει αυτό το γεγονός.
Ο βασικός περιορισμός κλειδιού της σχέσης S_SINCE φροντίζει αυτήν την απαίτηση. Χωρίς αυτό, η σχέση μπορεί να περιέχει, παραδείγματος χάριν, και τις δύο από τις ακόλουθες πλειάδες συγχρόνως:

S#	....	STATUS	STATUS_SINCE
S1	....	20	d06

S#	....	STATUS	STATUS_SINCE
S1	....	30	d06

Απαίτηση R5: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να έχει την ίδια θέση τις ημέρες d *και* d + 1, τότε πρέπει να περιέχει ακριβώς μία πλειάδα που να δείχνει αυτό το γεγονός.
Ο βασικός περιορισμός της σχέσης S_SINCE φροντίζει αυτήν την απαίτηση. Χωρίς αυτό, η σχέση μπορεί να περιέχει, παραδείγματος χάριν, και τις δύο από τις ακόλουθες πλειάδες συγχρόνως:

S#	....	STATUS	STATUS_SINCE
S1	....	20	d06

S#	....	STATUS	STATUS_SINCE
S1	....	20	d07

Απαίτηση R6: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να έχει κάποια θέση την ημέρα d, τότε πρέπει επίσης να παρουσιάζει τον προμηθευτή Sx να είναι στο πλαίσιο της σύμβασης την ημέρα d.
Έχουμε αντιμετωπίσει αυτήν την απαίτηση πριν, κοντά στο τέλος της παραγράφου 4.2, όπου παρατηρήσαμε ότι εάν οι τιμές S#_SINCE *και* STATUS_SINCE σε κάποια πλειάδα S_SINCE είναι d *και* d' αντίστοιχα, τότε πρέπει να έχουμε *το* d' >= d. Εδώ είναι η επίσημη δήλωση:

```
CONSTRAINT CR6 IS_EMPTY
(S_SINCE WHERE STATUS_SINCE < S#_SINCE);
```

Χωρίς αυτόν τον περιορισμό, η σχέση S_SINCE μπορεί να περιέχει, παραδείγματος χάριν, την ακόλουθη πλειάδα:

S#	S#_SINCE	STATUS	STATUS_SINCE
S#	d04	20	d02

Αυτή η πλειάδα παρουσιάζει σαφώς μεταξύ άλλων ότι ο προμηθευτής S1 είχε τη θέση 20 *την* $2^{η}$ ημέρα, επειδή η βάση δεδομένων περιέχει τις τρέχουσες σχέσεις μόνο (δηλ., δεν υπάρχει κανένα ιστορικό αρχείο). Αυτό δεν δείχνει ότι ο προμηθευτής S1 ήταν στο πλαίσιο της σύμβασης την $2^{η}$ ημέρα, με αυτόν τον τρόπο παραβιάζοντας την απαίτηση R6.

Απαίτηση R7: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να είναι ικανός να προμηθεύσει κάποιο συγκεκριμένο μέρος Py την ημέρα d, τότε πρέπει να περιέχει ακριβώς μία πλειάδα που να δείχνει αυτό το γεγονός.
Ο βασικός περιορισμός κλειδιού της σχέσης SP_SINCE φροντίζει αυτήν την απαίτηση. Χωρίς αυτόν, η σχέση μπορεί να περιέχει, παραδείγματος χάριν, και τις δύο από τις ακόλουθες πλειάδες συγχρόνως:

S#	P#	SINCE
S1	P1	d04

S#	P#	SINCE
S1	P1	d02

Απαίτηση R8: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx ικανό να προμηθεύσει το ίδιο μέρος Py τις ημέρες d και d+1 τότε πρέπει να περιέχει ακριβώς μία πλειάδα που να δείχνει αυτό το γεγονός.
Ο βασικός περιορισμός της σχέσης SP_SINCE φροντίζει αυτήν την απαίτηση. Χωρίς αυτόν, η σχέση μπορεί να περιέχει, παραδείγματος χάριν, και τις δύο από τις ακόλουθες πλειάδες συγχρόνως:

S#	P#	SINCE
S1	P1	d04

S#	P#	SINCE
S1	P1	d05

Απαίτηση R9: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx ικανό να προμηθεύσει κάποιο μέρος Py την ημέρα d, τότε πρέπει επίσης να παρουσιάζει τον προμηθευτή Sx να είναι στο πλαίσιο της σύμβασης την ημέρα d.
Ο ξένος βασικός περιορισμός από την σχέση SP_SINCE στη σχέση S_SINCE φροντίζει το μέρος αυτής της απαίτησης ("οποιοσδήποτε προμηθευτής αυτήν την περίοδο ικανός να προμηθεύσει κάποιο μέρος πρέπει να είναι αυτήν την περίοδο στο πλαίσιο της σύμβασης"), αλλά χρειαζόμαστε επίσης:

```
CONSTRAINT CR9 IS_EMPTY
                ((S_SINCE JOIN SP_SINCE) WHERE SINCE < S#_SINCE);
```

("κανένας προμηθευτής δεν μπορεί να προμηθεύσει οποιοδήποτε μέρος προτού να είναι ο προμηθευτής στο πλαίσιο της σύμβασης"). Χωρίς αυτόν τον περιορισμό, οι σχέσεις S_SINCE και η SP_SINCE μπορεί αντίστοιχα να περιέχουν τις ακόλουθες πλειάδες συγχρόνως:

S#	S#_SINCE	.....
S1	d04	.....

S#	S#_SINCE	.....
S1	d02	.....

Συγκρίνετε και αντιπαραβάλατε τον περιορισμό XST1 στο κεφάλαιο 2 (παράγραφος 2.2).

Για να συμπληρώσετε αυτό το τμήμα, το σχήμα 6.3 παρουσιάζει αναθεωρημένη και ολοκληρωμένη την έκδοση του αρχικού σχεδίου βάσεων δεδομένων από το σχήμα 6.1.

Παράδειγμα 6.3

```
S _SINCE {S#, S#,_SINCE_STATUS. STATUS_SINCE}
        KEY {S#}

CONSTRAINT CR6 IS_ EMPTY
   (S_SINCE WHERE STATUS_SINCE < S#_SINCE)
        SP_SINCE {S#, P#, SINCE}
        KEY {S# P#}
        FOREIGN KEY {S#} REFERENCES S _SINCE

CONSTRAINT CR9 IS_ EMPTY
     ((S_SINCE JOIN SP_SINCE) WHERE SINCE < S#_SINCE)
```

6.4 Μόνο ιστορικές Σχέσεις

Τώρα γυρίζουμε σε μια χρονική βάση δεδομένων που περιέχει ιστορικές σχέσεις μόνο. Η βάση δεδομένων (δείτε το παράδειγμα 6.4) αποτελείται ουσιαστικά από τις σχέσεις "during" από το παράδειγμα 5.1. Παρουσιάζουμε τώρα τους PACKED ON *και* WHEN/THEN περιορισμούς που ισχύουν γι' αυτές τις σχέσεις. Η βάση δεδομένων είναι πλήρως χρονική, αλλά δεν χωρίζει τις τρέχουσες και ιστορικές πληροφορίες. Μερικές τιμές δειγμάτων παρουσιάζονται στο σχήμα 6.5, μια αρκετά τροποποιημένη έκδοση του σχήματος 5.3.

Όπως ξέρουμε από το κεφάλαιο 4, ένα πλεονέκτημα αυτού του σχεδίου είναι ότι μπορεί ρητά να αντιπροσωπεύσει τις πληροφορίες για το μέλλον καθώς επίσης και το παρελθόν. Εντούτοις, υπάρχει ένα μειονέκτημα. Για *να* είμαστε συγκεκριμένοι, θα πρέπει πιθανώς να χρησιμοποιήσουμε μια τεχνητή τιμή "end-of-time" ως τιμή end (τέλους) για οποιοδήποτε διάστημα που αναφέρεται είτε σε μια τρέχουσα είτε σε κάποια μελλοντική παρούσα κατάσταση (εκτός αν συμβαίνει να *ξέρουμε τον πραγματικό* χρόνο τέλους(end), φυσικά). Ο προμηθευτής S7 είναι η περίπτωση στο σχήμα 6.5 (έχουμε υποθέσει χάριν του παραδείγματος ότι το d99 είναι "η τελευταία ημέρα").

Παράδειγμα 6.4
Ιστορικές σχέσεις μόνο

```
S_DURING {S#, DURING}
PACKED ON DURING
KEY {S#, DURING}

S_STATUS_DURING {S#, STATUS, DURING}
   PACKED ON DURING
   WHEN UNPACKED ON DURING THEN KEY {S#, DURING}
   KEY {S#, DURING}
```

SP_DURING {S#, P#, DURING}
PACKED ON DURING
KEY {S#, P#, DURING}

Παράδειγμα 6.5
Ιστορικές σχέσεις μόνο

S_DURING

S#	DURING
S2	[d02:d04]
S6	[d03:d05]
S7	[d03:d99]

SP_DURING

S#	P#	DURING
S2	P1	[d02:d04]
S2	P2	[d03:d03]
S2	P5	[d03:d04]
S6	P3	[d03:d05]
S6	P4	[d04:d04]
S6	P5	[d04:d05]
S7	P1	[d03:d04]
S7	P1	[d06:d07]
S7	P1	[d09:d99]

S_STATUS_DURING

S#	STATUS	DURING
S2	5	[d02:d02]
S2	10	[d03:d04]
S6	5	[d03:d04]
S6	7	[d05:d05]
S7	15	[d03:d08]
S7	20	[d09:d99]

Παρακάτω εξετάζουμε ποιες επίσημες εκδόσεις των απαιτήσεων R1 - R9 μπορεί να μοιάσουν με αυτήν την βάση δεδομένων.

Απαίτηση R1: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να είναι *στο* πλαίσιο της σύμβασης την ημέρα d τότε πρέπει να περιέχει ακριβώς μία πλειάδα που να δείχνει αυτό το γεγονός.
Ο περιορισμός PACKED ON στη σχέση S_DURING φροντίζει αυτήν την απαίτηση. Χωρίς αυτό, *η* σχέση μπορεί να περιέχει, και τις δύο ακόλουθες πλειάδες συγχρόνως:

S#	DURING
S2	[d02:d03]

S#	DURING
S2	[d03:d04]

Η βάση δεδομένων μας περιέχει μόνο ιστορικές σχέσεις. Συνεπεία αυτού του γεγονότος, η τιμή σχέσης S_DURING είναι πάντα ίση με το αποτέλεσμα της έκφρασης

USING DURING ◄ S_STATUS_DURING {S#, DURING} ►

Κατά συνέπεια, μπορεί να θεωρηθεί ότι εάν η σχέση S_DURING παρουσιάζει τον προμηθευτή Sx να είναι στο πλαίσιο της σύμβασης την ημέρα d, τότε η σχέση

S_STATUS_DURING κάνει το ίδιο επίσης, παραβιάζοντας την απαίτηση Rl. Για να είμαστε συγκεκριμένοι, η πρόταση "ο προμηθευτής Sx είχε κάποια θέση την ημέρα d" δεν υπονοεί λογικά την πρόταση "ο προμηθευτής Sx ήταν στο πλαίσιο της σύμβασης την ημέρα d," και η σχέση S_STATUS_DURING δεν δείχνει ότι οι προμηθευτές ήταν στο πλαίσιο της σύμβασης.

Απαίτηση R2: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να είναι στο πλαίσιο της σύμβασης τις ημέρες d και d+1, κατόπιν πρέπει να περιέχει ακριβώς μια πλειάδα που δείχνει αυτό το γεγονός.

Ο PACKED ON περιορισμός στη σχέση S_DURING φροντίζει αυτήν την απαίτηση. Χωρίς αυτή την σχέση μπορεί να περιέχει και τις δύο ακόλουθες πλειάδες συγχρόνως:

S#	DURING
S2	[d02:d02]

S#	DURING
S2	[d03:d04]

Απαίτηση R3: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να είναι στο πλαίσιο της σύμβασης την ημέρα d, τότε πρέπει να παρουσιάζει τον προμηθευτή Sx να έχει κάποια θέση την ημέρα d.
Οι προμηθευτές παρουσιάζονται να είναι στο πλαίσιο της σύμβασης στην σχέση S_DURING και να έχουν κάποια θέση στην σχέση S_STATUS_DURING. Η απαίτηση R3 υπονοεί ότι κάθε πλειάδα του ζεύγους προμηθευτής Sx με την ημέρα d στην αποσυμπιεσμένη(unpacked) μορφή του S_DURING θα πρέπει επίσης να εμφανίζεται στην αποσυμπιεσμένη μορφή της προβολής της S_STATUS_DURING πάνω στο S# και DURING. Ως εκ τούτου:

```
CONSTRAINT HR3 USING DURING
  ◄S_DURING ⊆ S_STATUS_DURING {S#, DURING} ►;
```

Χωρίς αυτόν τον περιορισμό, η σχέση S_DURING μπορεί να περιέχει, τις ακόλουθες πλειάδες για τον προμηθευτή S7

S#	DURING
S7	[d03:d99]

ενώ συγχρόνως η σχέση S_STATUS_DURING περιέχει, ακριβώς την ακόλουθη πλειάδα για τον προμηθευτή S7:

S#	STATUS	DURING
S7	20	[d09:d99]

εάν η σχέση S_STATUS_DURING κρατήθηκε πραγματικά αποσυμπιεσμένη πάνω στο DURING , τότε {S#, DURING} θα ήταν ένα κλειδί υποψηφίων γι' αυτή την σχέση. Επιπλέον, εάν η σχέση S_DURING κρατήθηκε επίσης αποσυμπιεσμένο πάνω στο

DURING, τότε {S#, DURING} θα ήταν ένα ξένο κλειδί στην S_DURING. Κατά συνέπεια, εάν το {S#, DURING} θεωρείται ως U_key *για την* S_STATUS_DURING, τότε το {S#, DURING} μπορεί να θεωρηθεί ως ξένο U_key *στην* S_DURING! Επομένως προτείνουμε μια άλλη στενογραφία. Για να είμαστε συγκεκριμένοι, προτείνουμε ότι ο καθορισμός οποιοδήποτε δεδομένης σχέσης R2 επιτρέπεται να περιλαμβάνει μια προδιαγραφή της μορφής

USING (ACL) FOREIGN KEY {K} REFERENCES R1

Όπως η συζήτησή μας έχει δείξει λίγο πολύ, ότι η σημασιολογία είναι ότι εάν το R1 *και* R2 κρατιόνταν αποσυμπιεσμένα στις ιδιότητες που διευκρινίστηκαν *στο* ACL, τότε το K εντός του R2 θα ήταν ένα ξένο κλειδί που ταιριάζει με το κλειδί υποψηφίων *του* R1, *που* υπονοείται από τον αντίστοιχο καθορισμό U_key *για το* R1.
Για να επιστρέψουμε στην απαίτηση R3 συγκεκριμένα, μπορούμε τώρα να διατυπώσουμε αυτήν την απαίτηση έτσι:

USING DURING FOREIGN KEY {S#, DURING}
REFERENCES S_STATUS_DURING

Απαίτηση R4: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να έχει κάποια θέση την ημέρα d, τότε πρέπει να περιέχει ακριβώς μία πλειάδα που να δείχνει αυτό το γεγονός.
Οι PACKED ON *και* WHEN/THEN περιορισμοί στη σχέση S_STATUS_DURING φροντίζουν αυτήν την απαίτηση. Χωρίς τον περιορισμό PACKED ON, η σχέση μπορεί να περιέχει, και τις δύο ακόλουθες πλειάδες συγχρόνως:

S#	STATUS	DURING
S7	15	[d03:d06]

S#	STATUS	DURING
S7	15	[d05:d08]

Και χωρίς τον περιορισμό WHEN/THEN της σχέσης μπορεί να περιέχει και τις δύο ακόλουθες πλειάδες συγχρόνως:

S#	STATUS	DURING
S7	15	[d03:d06]

S#	STATUS	DURING
S7	20	[d05:d08]

Απαίτηση R5: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να έχει την ίδια θέση τις ημέρες d *και* d + 1, τότε πρέπει να περιέχει ακριβώς μία πλειάδα που να δείχνει αυτό το γεγονός.
Ο PACKED ON περιορισμός στην σχέση S_STATUS_DURING φροντίζει αυτήν την απαίτηση. Χωρίς αυτή την σχέση μπορεί να περιέχει, και τις δύο ακόλουθες πλειάδες συγχρόνως:

S#	STATUS	DURING
S7	15	[d03:d06]

S#	STATUS	DURING
S7	15	[d07:d08]

Απαίτηση R6: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να έχει κάποια θέση την ημέρα d, τότε πρέπει επίσης να παρουσιάζει τον προμηθευτή Sx να είναι στο πλαίσιο της σύμβασης την ημέρα d.
Η απαίτηση R6 είναι το αντίστροφο της απαίτησης R3. Ως εκ τούτου

USING DURING FOREIGN KEY {S#, DURING}
REFERENCES S_DURING

Χωρίς αυτόν τον περιορισμό, η σχέση S_STATUS_DURING μπορεί να περιέχει, τις ακόλουθες πλειάδες για τον προμηθευτή S7

S#	STATUS	DURING
S7	20	[d0:d99]

ενώ συγχρόνως η σχέση S_DURING περιέχει, μόνο την ακόλουθη πλειάδα και καμία άλλη για τον προμηθευτή S7:

S#	DURING
S7	[d10:d99]

Παρατηρείστε, ότι καθεμιά απ' τις σχέσεις S_DURING *και* S_STATUS_DURING έχει ένα ξένο U_key που αναφέρεται στο άλλο.

Απαίτηση R7: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να είναι ικανός να προμηθεύσει κάποιο συγκεκριμένο μέρος Py την ημέρα d, τότε πρέπει να περιέχει ακριβώς μία πλειάδα που να δείχνει αυτό το γεγονός.
Ο περιορισμός PACKED ON στην σχέση SP_DURING φροντίζει αυτήν την απαίτηση. Χωρίς αυτή την σχέση μπορεί να περιέχει, και τις δύο ακόλουθες πλειάδες συγχρόνως:

S#	P#	DURING
S2	P1	[d02:d03]

S#	P#	DURING
S2	P1	[d03:d04]

Απαίτηση R8: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx ικανό να προμηθεύσει το ίδιο μέρος Py τις ημέρες d *και* d+1 τότε πρέπει να περιέχει ακριβώς μία πλειάδα που να δείχνει αυτό το γεγονός.
Ο περιορισμός PACKED ON στην σχέση SP_DURING φροντίζει αυτήν την απαίτηση. Χωρίς αυτό η σχέση μπορεί να περιέχει και τις δύο ακόλουθες πλειάδες συγχρόνως:

S#	P#	DURING
S2	P1	[d02:d02]

S#	P#	DURING
S2	P1	[d03:d04]

Απαίτηση R9: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx ικανό να προμηθεύσει κάποιο μέρος Py την ημέρα d, τότε πρέπει επίσης να παρουσιάζει τον προμηθευτή Sx να είναι στο πλαίσιο της σύμβασης την ημέρα d.

Ο περιορισμός

USING DURING FOREIGN KEY {S#, DURING}
REFERENCES S_DURING

φροντίζει αυτήν την απαίτηση. Χωρίς αυτό η σχέση μπορεί να περιέχει και τις δύο ακόλουθες πλειάδες συγχρόνως για τον προμηθευτή S7:

S#	P#	DURING
S7	P1	[d09:d99]

ενώ συγχρόνως η σχέση S_DURING περιέχει μόνο την ακόλουθη πλειάδα και καμία άλλη για τον προμηθευτή S7:

S#	DURING
S7	[d10:d99]

Για να συμπληρώσουμε αυτό το τμήμα, το σχήμα 6.6 παρουσιάζει την αναθεωρημένη και ολοκληρωμένη έκδοση του αρχικού σχεδίου των βάσεων δεδομένων του σχήματος 6.4. Παρατηρήστε ότι τώρα χρησιμοποιούμε στενογραφίες U_key.

Παράδειγμα 6.6
Ιστορικές σχέσεις
Ολοκληρωμένος σχεδιασμός

```
S_DURING {S#, DURING}
    USING DURING KEY {S#, DURING}
    USING DURING FOREIGN KEY {S#, DURING}
        REFERENCES S_STATUS_DURING

S_STATUS_DURING {S#, STATUS, DURING}
        USING DURING KEY {S#, DURING}
        USING DURING FOREIGN KEY {S#, DURING}
            REFERENCES S_DURING

SP_DURING {S#, P#, DURING}
        USING DURING KEY {S#, P#, DURING}
        USING DURING FOREIGN KEY {S#, DURING}
            REFERENCES S_DURING
```

6.5 Τρέχουσες και ιστορικές σχέσεις

Σε αυτό το τμήμα εξετάζουμε την πλήρως χρονική βάση δεδομένων του σχήματος 5.1. Παρουσιάζουμε τη βάση δεδομένων πάλι στο σχήμα 6.7, εκτός από το ότι παρουσιάζουμε τώρα όλους τους εφαρμόσιμους PACKED ON *και* WHEN/THEN

περιορισμούς. Αυτή η βάση δεδομένων κρατά τις τρέχουσες πληροφορίες στις σχέσεις "since" και τις ιστορικές πληροφορίες στις σχέσεις "during".

Όπως εξηγήσαμε στην παράγραφο 4.6, οι σχέσεις "during" δεν περιέχουν οποιαδήποτε πληροφορία σχετικά με το παρόν ή το μέλλον, δηλαδή κάθε τιμή END (DURING) και κάθε τιμή BEGIN(DURING) είναι μικρότερη από την σημερινή ημερομηνία (που, χάριν του σχήματος 6.8, υποθέτουμε ότι άλλη μια φορά είναι η 10^7 ημέρα). Εντούτοις, οι σχέσεις "since" περιλαμβάνουν τις υπονοούμενες πληροφορίες για το μέλλον, και περιλαμβάνουν τις ρητές πληροφορίες επίσης.

Σχετικά με την σχέση S_SINCE: Παρατηρούμε ότι η σχέση S_SINCE υπόκειται στον περιορισμό (που τον καλούμε περιορισμό X) ότι εάν κάποια πλειάδα S_SINCE περιλαμβάνει μια S#_SINCE τιμή d που αντιστοιχεί σε κάποια ημερομηνία στο μέλλον, τότε η τιμή STATUS_SINCE σε αυτή την πλειάδα έπρεπε να είναι καλύτερα d. Εάν ήταν μεγαλύτερο από το d, τότε η απαίτηση R3 θα παραβιαζόταν. (Δεν μπορεί να είναι μικρότερη από το d, χάρη στον περιορισμό BR6_A). Εάν θα ήμαστε σε θέση να δηλώσουμε τον περιορισμό X τυπικά είναι

παράδειγμα 6.7
παρούσες και ιστορικές
σχέσεις μαζί

```
S SINCE {S#, S#_SINCE, STATUS, STATUS _SINCE}
                KEY {S#}

 SP_SINCE {S#, P#, SINCE}
       KEY {S#, P#}
 FOREIGN KEY {S#} REFERENCES S_SINCE

S_DURING {S#, DURING}
 PACKED ON DURING
KEY {S#, DURING}

S_STATUS_DURING {S#, STATUS, DURING }
PACKED ON DURING
WHEN UNPACKED ON DURING THEN KEY { S#, DURING}
 KEY {S#, DURING}

SP_DURING {S#, P#, DURING}
PACKED ON DURING
 KEY {S#, P#, DURING}
```

Παράδειγμα 6.8 παρούσες και ιστορικές σχέσεις μαζί ενδεικτικές τιμές

SP_SINCE

S#	P#	SINCE
S1	P1	d04
S1	P2	d05
S1	P3	d09
S1	P4	d05
S1	P5	d04
S1	P6	d06
S2	P1	d08
S2	P2	d09
S3	P2	d08
S4	P5	d05

S_DURING

S#	DURING
S2	[d02:d04]
S6	[d03:d05]

SP_DURING

S#	P#	DURING
S2	P1	[d02:d04]
S2	P2	[d03:d03]
S3	P5	[d05:d07]
S4	P2	[d06:d09]
S4	P4	[d04:d08]
S6	P3	[d03:d03]
S6	P3	[d05:d05]

S_STATUS_DURING

S#	STATUS	DURING
S1	15	[d04:d05]
S2	5	[d02:d02]
S2	10	[d03:d04]
S4	10	[d04:d04]
S4	25	[d05:d07]
S6	5	[d03:d04]
S6	7	[d05:d05]

σημειώστε ότι θα έπρεπε πιθανώς να περιλάβει μια δοκιμή της μορφής "IF d >TODAY()", όπου TODAY είναι ένας μηδενικός ενσωματωμένος τελεστής που επιστρέφει την σημερινή ημερομηνία και αυτό δεν είναι καθόλου σαφές ότι έχουμε την άδεια, ή θα θέλαμε να έχουμε την άδεια, *για* να περιλάβουμε μια τέτοια δοκιμή μέσα σε έναν επίσημο περιορισμό. Γιατί όχι; Επειδή ένας τέτοιος περιορισμός ή ο έλεγχος ακεραιότητας που υπονοήθηκε από έναν τέτοιο περιορισμό, μάλλον μπόρεσε να πετύχει *για μια* ημέρα και να αποτύχει για την επόμενη, ακόμα κι αν η βάση δεδομένων δεν ήταν ενημερωμένη στο μεσοδιάστημα!

Προχωράμε τώρα να εξετάσουμε ποιες επίσημες εκδόσεις των απαιτήσεων R1 - R9 μπορεί να μοιάσει με αυτήν την βάση δεδομένων.

Απαίτηση Rl:Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να είναι *στο* πλαίσιο της σύμβασης την ημέρα d τότε πρέπει να περιέχει ακριβώς μία πλειάδα που να δείχνει αυτό το γεγονός.
Θυμηθείτε την απαίτηση Rl που έχει να κάνει με την αποφυγή του πλεονασμού. Συγκεκριμένα ο πλεονασμός μέσα ή στην σχέση S_SINCE *και* S_DURING, λαμβάνοντας υπόψη τη βάση δεδομένων του σχήματος 6.7. Τώρα ο βασικός περιορισμός κλειδιού στη S_SINCE εγγυάται ότι η σχέση S_SINCE από μόνη της δεν μπορεί να παραβιάσει την απαίτηση (παράγραφος 6.3), και ο περιορισμός PACKED ON *στην* S_DURING εγγυάται ότι η σχέση S_DURING από μόνο του δεν μπορεί να την

παραβιάσει. Αυτό που χρειαζόμαστε επομένως, είναι ένας πρόσθετος περιορισμός για να εξασφαλίσει ότι και οι δύο σχέσεις δεν παρουσιάζουν τον προμηθευτή Sx να είναι στο πλαίσιο της σύμβασης την ίδια ημέρα:

```
CONSTRAINT BR1 IS _EMPTY
    ((S _SINCE JOIN S _DURING)
        WHERE S#_SINCE ≤END (DURING));
```

Χωρίς αυτόν τον περιορισμό, οι σχέσεις S_SINCE *και* S_DURING αντίστοιχα μπορεί να περιέχουν τις ακόλουθες πλειάδες συγχρόνως:

S#	S#_SINCE	.....
S1	d04	.....

S#	DURING
S1	[d06:d08]

και οι δύο αυτές πλειάδες δείχνουν ότι ο προμηθευτής S1 ήταν στο πλαίσιο της σύμβασης την $7^{η}$ ημέρα.
Υποθέστε ότι εισάγουμε μια τροποποιημένη έκδοση του S_SINCE' της σχέσης S_SINCE που περιλαμβάνει μια ιδιότητα διαστήματος DURING αντί των ιδιοτήτων σημείου S#_SINCE. Υποθέστε ακόμα ότι η τρέχουσα τιμή S_SINCE' είναι ίδια με την S_SINCE, εκτός από το ότι, όποτε η S_SINCE έχει την S#_SINCE τιμή d, *η* S_SINCE' έχει την DURING τιμή [d:d99] (όπου υποθέτουμε χάριν της συζήτησης ότι d99 είναι "η τελευταία ημέρα"). Κατόπιν ο περιορισμός BRl υπονοεί τα εξής:

- εάν πάρουμε την προβολή της σχέσης S_SINCE' πάνω στην S# *και* DURING και αποσυμπιέσουμε αυτήν την προβολή πάνω στην DURING, *και*
- εάν επίσης αποσυμπιέσουμε την σχέση S_DURING πάνω στην DURING, τότε
- η τομή αυτών των δύο αποσυμπιεσμένων αποτελεσμάτων είναι άδεια

Απαίτηση R2: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να είναι *στο* πλαίσιο της σύμβασης τις ημέρες d *και* d+1, κατόπιν πρέπει να περιέχει ακριβώς μια πλειάδα που να δείχνει αυτό το γεγονός.
Η απαίτηση R2 έχει να κάνει με την αποφυγή της περίφρασης στις σχέσεις S_SINCE *και* S_DURING. Ο βασικός περιορισμός της S_SINCE εγγυάται ότι η σχέση S_SINCE από μόνη της δεν μπορεί να παραβιάσει την απαίτηση, και ο PACKED ON περιορισμός πάνω στην S_DURING εγγυάται ότι η σχέση S_DURING από μόνη της δεν μπορεί να την παραβιάσει, επίσης. Αυτό που χρειαζόμαστε είναι ένας επιπρόσθετος περιορισμός για να διαβεβαιώσουμε ότι εάν η σχέση S_SINCE δείχνει τον προμηθευτή Sx να είναι στο πλαίσιο της σύμβασης από την ημέρα d, τότε η σχέση S_DURING δεν δείχνει τον προμηθευτή Sx να είναι στο πλαίσιο της σύμβασης την αμέσως προηγούμενη ημέρα d.

```
CONSTRAINT BR2 IS_EMPTY
    ((S_ SINCE JOIN S_ DURING)
WHERE PRIOR_DATE (S#_SINCE) = END (DURING));
```

Χωρίς αυτόν τον περιορισμό, οι σχέσεις S_SINCE *και* S_DURING μπορεί να περιέχουν τις ακόλουθες πλειάδες συγχρόνως:

S#	S#_SINCE	....
S1	d04	....

S#	DURING
S1	[d01:d03]

οι περιορισμοί BR1 *και* BR2 μπορούν να συνδυαστούν σε μία ενιαία διατύπωση, ως εξής:

```
CONSTRAINT BR1 _2  IS _EMPTY
((S_ SINCE JOIN S_ DURING)
   WHERE PRIOR_DATE (S#_SINCE) ≤END (DURING));
```

Εντούτοις, υπάρχει ένα μικρό πρόβλημα εδώ. Όπως εξηγείται στο 3° κεφάλαιο, η έκφραση PRIOR_DATE (S#_SINCE) δεν ορίζεται εάν η S#_SINCE συμβαίνει να είναι "η πρώτη ημέρα" και ο περιορισμός BR1_2 όπως διατυπώνεται μπορεί να προκαλέσει λάθη χρόνου εκτέλεσης. Δημιουργούμε έναν άλλο τελεστή, IS_PRIOR_T που παίρνει δύο ορίσματα του ίδιου τύπου σημείου, p1 *και* p2 (σε αυτήν την σειρά), και επιστρέφει αληθή(true) εάν το p1 είναι ο άμεσος προκάτοχος του p2 αλλιώς ψεύδή(false). Μπορούμε να επαναδιατυπώσουμε τον περιορισμό BR1_2 ως εξής:

```
CONSTRAINT BR1 _2 IS _EMPTY
      ((S _SINCE JOIN S_ DURING)
         WHERE END (DURING) ≥S#_ SINCE
          OR IS_PRIOR_DATE (END (DURING), S#_SINCE));
```

Απαίτηση R3: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να είναι στο πλαίσιο της σύμβασης την ημέρα d, τότε πρέπει να παρουσιάζει τον προμηθευτή Sx *να* έχει κάποια θέση την ημέρα d.
Οι σχέσεις που παρουσιάζουν τον δεδομένο προμηθευτή να είναι στο πλαίσιο της σύμβασης μια δεδομένη ημέρα είναι τα S_SINCE *και* S_DURING, *οι* σχέσεις που παρουσιάζουν τον δεδομένο προμηθευτή να έχει μια δεδομένη θέση μια δεδομένη ημέρα είναι τα S_SINCE *και* S_STATUS_DURING.
Εξετάζουμε την πλειάδα S_SINCE, *με* S# _ SINCE *και* S_STATUS_SINCE τις τιμές d *και* d' αντίστοιχα. Εάν d =d', τότε αυτή η πλειάδα ικανοποιεί την απαίτηση R3. Ωστόσο, εάν d <d', τότε η απαίτηση R3 υπονοεί ότι οι πληροφορίες θέσης για κάθε σημείο στο διάστημα [d:d'-1] για τον εν λόγω προμηθευτή πρέπει να εμφανιστούν στην σχέση S_STATUS_DURING.
Επίσης, η απαίτηση R3 υπονοεί ότι κάθε τιμή {S#, DURING} που εμφανίζεται στην αποσυμπιεσμένη μορφή της σχέσης S_DURING πρέπει να εμφανιστεί και στην αποσυμπιεσμένη μορφή της σχέσης S_STATU5_DURING επίσης. Ως εκ τούτου:

```
CONSTRAINT BR3
  WITH (S _SINCE WHERE S#_ SINCE < STATUS _SINCE) AS T1,
     (EXTEND T1
       ADD INTERVAL_DATE ([S #_SINCE:
                    PRIOR_ DATE (STATUS_ SINCE)])
```

```
    AS DURING) {S#, DURING} AS T2,
(T2 UNION S_DURING) AS T3,
 S_STATUS_DURING {S#, DURING} AS T4:
USING DURING ◄ T3 ⊆ T4 ►;
```

Χωρίς αυτόν τον περιορισμό, θα ήταν δυνατό η σχέση S_SINCE να περιλαμβάνει, την ακόλουθη πλειάδα:

S#	S#_SINCE	STATUS	STATUS_SINCE
S1	dO4	20	d06

ενώ συγχρόνως η αποσυμπιεσμένη μορφή της σχέσης S_STATUS_DURING *δεν* περιέχει πλειάδες της μορφής:

S#	STATUS	DURING
S1	.....	[d04:d04]
S1	.....	[d05:d05]

Η τιμή status στην πλειάδα για τον προμηθευτή S1 και το διάστημα [d05:d05] εδώ δεν θα μπορούσε να είναι 20, χάρη στον περιορισμό BR5.

ΣΗΜΕΙΩΣΗ: *Εάν θέλαμε, θα μπορούσαμε να διευκρινίσουμε τον ακόλουθο ξένο περιορισμό U_key ως τμήμα του καθορισμένου σχέσης S_DURING:*

```
USING DURING FOREIGN KEY (S#, DURING)
REFERENCES S_STATUS_DURING
```

Εάν μπορούσαμε να απλοποιήσουμε τον περιορισμό BR3 ελαφρώς με τη διαγραφή της γραμμής 7 και την αντικατάσταση της αναφοράς στη γραμμή 9 *το* T3 από μια αναφορά *στο* T2 άντ' αυτού.
Παρατηρήστε τώρα ότι ο περιορισμός BR3 όπως δηλώνεται περιλαμβάνει μια επίκληση PRIOR_DATE. Σε αντίθεση με την κατάσταση με τον BR2 περιορισμό, όπου η επίκληση είναι εγγυημένη για να μην αποτύχει, δηλαδή το επιχείρημα STATUS_SINCE είναι εγγυημένο για να μην αξιολογήσει "την πρώτη ημέρα".
Σημειώστε τελικά ότι είναι μια λογική συνέπεια της απαίτησης R3 ότι ο κάθε αριθμός προμηθευτών που εμφανίζεται στην σχέση S_DURING εμφανίζεται επίσης και *στην* σχέση S_STATUS_DURING.

Απαίτηση R4: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να έχει κάποια θέση την ημέρα d, τότε πρέπει να περιέχει ακριβώς μία πλειάδα που να δείχνει αυτό το γεγονός.
Ο βασικός περιορισμός της σχέσης S_SINCE *και των* PACKED ON *και* WHEN/THEN περιορισμών της σχέσης S_STATUS_DURING είναι επαρκές για να εγγυηθεί ότι καμιά από αυτές τις σχέσεις δεν μπορεί να παραβιάσει αυτήν την απαίτηση από μόνη της. Αλλά πρέπει να προσθέσουμε:

```
CONSTRAINT BR4 IS_EMPTY
  ((S SINCE JOIN S_STATUS_DURING (S#, DURING))
     WHERE STATUS_SINCE ≤END (DURING));
```

Χωρίς αυτόν τον περιορισμό, οι σχέσεις S_SINCE *και* S_STATUS_DURING μπορεί να περιέχουν τις ακόλουθες πλειάδες συγχρόνως:

S#	S#_SINCE	STATUS	STATUS_SINCE
S1	d04	20	d05

S#	STATUS	STATUS_SINCE
S1	20	[d04:d05]

Εναλλακτικά (αυτό που είναι χειρότερο) είναι ότι μπορούν να περιέχουν τις ακόλουθες πλειάδες συγχρόνως:

S#	S#_SINCE	STATUS	STATUS _SINCE
S1	d04	20	d05

S#	STATUS	DURING
S1	10	[d04:d05]

Τώρα υποθέστε ότι εισάγουμε μια τροποποιημένη έκδοση S_SINCE' της σχέσης S_SINCE που περιλαμβάνει μια ιδιότητα διαστήματος DURING αντί των ιδιοτήτων σημείου STATUS_SINCE. Υποθέστε ακόμα ότι η τρέχουσα τιμή S_SINCE' είναι ίδια με *την* S_SINCE, εκτός από το ότι, όποτε η S_SINCE έχει *τη* STATUS_SINCE τιμή d, *η* S_SINCE' έχει την τιμή DURING [d:d99] (όπου υποθέτουμε άλλη μια φορά ότι d99 είναι "η τελευταία ημέρα"). Κατόπιν ο περιορισμός BR4 υπονοεί τα εξής:

- Εάν πάρουμε την προβολή της σχέσης S_SINCE' πάνω στην S#, STATUS *και* DURING *και* αποσυμπιέσουμε αυτήν την προβολή πάνω στην DURING *και*
- Εάν επίσης αποσυμπιέσουμε της σχέσης S_STATUS_DURING πάνω στην DURING, τότε
- Η τομή των δύο αυτών αποσυμπιεσμένων αποτελεσμάτων είναι άδεια.

Απαίτηση R5: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να έχει την ίδια θέση τις ημέρες d *και* d + 1, τότε πρέπει να περιέχει ακριβώς μία πλειάδα που να δείχνει αυτό το γεγονός.
Ο βασικός περιορισμός της σχέσης S_SINCE και του περιορισμού PACKED ON πάνω *στο* S_STATUS_DURING είναι κατάλληλα. Επιπλέον χρειαζόμαστε

```
CONSTRAINT BR5 IS _EMPTY
  ((S_ SINCE JOIN S_ STATUS _DURING)
    WHERE PRIOR_DATE (STATUS_SINCE) = END (DURING));
```

Ή προτιμότερα

```
CONSTRAINT BR5 IS_ EMPTY
   ((S_SINCE JOIN S _STATUS _DURING)
     WHERE IS_PRIOR_DATE (END (DURING), STATUS_SINCE));
```

χωρίς αυτόν τον περιορισμό οι σχέσεις S_SINCE *και* S_STATUS_DURING μπορεί να περιέχουν τις ακόλουθες πλειάδες συγχρόνως:

S#	S# _SINCE	STATUS	STATUS _SINCE
S1	d04	20	d05

S#	STATUS	DURING
S1	20	[d04:d04]

Απαίτηση R6: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να έχει κάποια θέση την ημέρα d, τότε πρέπει επίσης να παρουσιάζει τον προμηθευτή Sx να είναι στο πλαίσιο της σύμβασης την ημέρα d.

Αυτήν η απαίτηση υπονοεί ότι εάν οι τιμές S# _SINCE *και* STATUS _SINCE σε κάποια S_SINCE πλειάδα είναι d *και* d', τότε πρέπει να έχουμε d' ≥d. Αντιθέτως, εάν η S_SINCE πλειάδα για τον προμηθευτή Sx έχει d'<d τότε η απαίτηση R6 υπονοεί ότι ο προμηθευτής Sx πρέπει να είναι στο πλαίσιο της σύμβασης την ημέρα d-1 και έτσι η πλειάδα επιβεβαιώνοντας αυτό το γεγονός πρέπει να εμφανιστεί στην σχέση S_DURING. Αλλά μια τέτοια δήλωση θα παραβίαζε την απαίτηση R2. Ως εκ τούτου έχουμε:

```
CONSTRAINT BR6 _A IS _EMPTY
   (S_SINCE WHERE STATUS_SINCE < S#_SINCE);
```

Αυτός ο περιορισμός είναι ίδιος με τον περιορισμό CR6 από την παράγραφο 6.3, εκτός από το όνομά του.

ΣΗΜΕΙΩΣΗ: Ονομάζουμε τον περιορισμό "BR6_A" επειδή, όπως θα δούμε σε μια στιγμή, χρειαζόμαστε έναν πρόσθετο περιορισμό ("BR6_B") προκειμένου να ικανοποιηθεί πλήρως η απαίτηση R6. Πρέπει να εξηγήσουμε επίσης ότι επιλέγουμε να αντιπροσωπεύσουμε την απαίτηση R6 από δύο χωριστούς περιορισμούς καθαρά για παιδαγωγικούς λόγους.

Όπως σημειώνεται στην παράγραφο 6.4, η απαίτηση R6 είναι αποτελεσματικά το αντίστροφο της απαίτησης R3. Κατά συνέπεια, μια κατάλληλη διατύπωση του περιορισμού που απαιτείται για να φροντίσει το υπόλοιπο της απαίτησης R6 μπορεί να ληφθεί από τον περιορισμό BR3 με την αντικατάσταση του τελεστή "U_⊆" *στην* τελευταία γραμμή αυτού του περιορισμού από έναν "U_⊆" τελεστή. Εντούτοις, έχει προφανώς νόημα να συνδυάσουμε τους δύο περιορισμούς με τη χρησιμοποίηση του "U_ =" τελεστή άντ' αυτού:

```
CONSTRAINT BR3_6_B
  WITH (S_ SINCE WHERE S# _SINCE < STATUS _SINCE) AS T1
(EXTEND T1
        ADD INTERVAL _DATE ((S# _SINCE:
                PRIOR _DATE (STATUS _SINCE)])
        AS DURING) {S#, DURING} AS T2,
       (T2 UNION S_ DURING) AS T3,
        S_STATUS_DURING {S#, DURING} AS T4:
        USING DURING ◄T3 = T4 ►;
```

Χωρίς αυτόν τον περιορισμό, θα ήταν δυνατό για την σχέση S_STATUS_DURING *να* περιέχει, την ακόλουθη πλειάδα:

S#	.....	DURING
S1	.....	[d04:d04]

όσο την ίδια στιγμή ούτε η σχέση S_SINCE ούτε η σχέση S_DURING δεν περιέχει την πλειάδα που δείχνει ότι ο S1 προμηθευτής είναι στο πλαίσιο της σύμβασης από την $4^{η}$ ημέρα.
Έπ' ευκαιρία δεν θα ήταν σωστό να διευκρινιστεί ο ακόλουθος ξένος περιορισμός U_key ως τμήμα της καθορισμένης σχέσης S_STATUS_DURING

```
USING DURING FOREIGN KEY {S#, DURING}
        REFERENCES S_DURING
```

Σημειώστε τελικά ότι είναι μια λογική συνέπεια της απαίτησης R6 ότι κάθε αριθμός προμηθευτών που εμφανίζεται στην σχέση S_STATUS_DURING εμφανίζεται επίσης στην σχέση S_DURING ή στην σχέση S_SINCE ή και στα δύο.

Απαίτηση R7: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx να είναι ικανός να προμηθεύσει κάποιο συγκεκριμένο μέρος Py την ημέρα d, τότε πρέπει να περιέχει ακριβώς μία πλειάδα που να δείχνει αυτό το γεγονός.
Ο βασικός περιορισμός της σχέσης SP_SINCE *και* ο περιορισμός PACKED ON *της* σχέσης SP_DURING είναι επαρκής για να εγγυηθούν ότι καμιά απ' αυτές τις σχέσεις δεν μπορεί να παραβιάσει αυτήν την απαίτηση. Αλλά πρέπει να προσθέσουμε:

```
CONSTRAINT BR7 IS_EMPTY
   ((SP _SINCE JOIN SP_DURING)
        WHERE SINCE ≤END (DURING));
```

Χωρίς αυτόν τον περιορισμό, οι σχέσεις SP _SINCE *και* SP _DURING αντίστοιχα μπορεί να περιέχουν, τις ακόλουθες πλειάδες συγχρόνως:

S#	P#	SINCE
S1	P1	d04

S#	P#	DURING
S1	P1	[d06:d08]

Τώρα υποθέστε ότι εισάγουμε μια τροποποιημένη έκδοση SP_SINCE' της σχέσης SP_SINCE που περιλαμβάνει μια ιδιότητα διαστήματος DURING αντί των ιδιοτήτων σημείου SINCE. Υποθέστε περαιτέρω ότι η τρέχουσα τιμή της SP _SINCE είναι ίδια με αυτήν της SP_SINCE , εκτός από το ότι, όταν η SP_SINCE έχει την SINCE τιμή d, *η* SP _SINCE' έχει την DURING τιμή [d:d99] τότε ο περιορισμός BR7 υπονοεί τα ακόλουθα:

- Εάν αποσυμπιέσουμε την σχέση SP_SINCE' πάνω στο DURING, *και*
- Εάν αποσυμπιέσουμε επίσης την σχέση SP_DURING , τότε
- Η τομή των δύο αυτών αποσυμπιεσμένων αποτελεσμάτων είναι άδεια

Απαίτηση R8: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx ικανό να προμηθεύσει το ίδιο μέρος Py τις ημέρες d *και* d+1 τότε πρέπει να περιέχει ακριβώς μία πλειάδα που να δείχνει αυτό το γεγονός.
Ο βασικός περιορισμός κλειδιού στον SP_SINCE *και τον* PACKED ON περιορισμό πάνω στο SP_DURING είναι σχετικός. Επιπλέον, χρειαζόμαστε:

```
CONSTRAINT BR8 IS_EMPTY
  ((SP_SINCE JOIN SP_DURING)
    WHERE PRIOR_DATE (SINCE) = END (DURING));
```

Ή προτιμότερα:

```
CONSTRAINT BR8 IS_EMPTY
  ((SP _SINCE JOIN SP_ DURING)
    WHERE IS_PRIOR_DATE (END (DURING), SINCE));
```

Χωρίς αυτόν τον περιορισμό, οι σχέσεις SP_SINCE *και* SP_DURING μπορεί να περιέχουν τις ακόλουθες πλειάδες συγχρόνως:

S#	P#	SINCE
S1	P1	d04

S#	P#	DURING
S1	P1	[d01:d03]

Οι περιορισμοί BR7 *και* BR8 μπορούν να συνδυαστούν όπως φαίνεται παρακάτω:

```
CONSTRAINT BR7 _8 IS _EMPTY
  ((SP_ SINCE JOIN SP _DURING)
      WHERE END (DURING) ≥SINCE
        OR IS_PRIOR_DATE (END ( DURING ), SINCE));
```

Απαίτηση R9: Εάν η βάση δεδομένων παρουσιάζει τον προμηθευτή Sx ικανό να προμηθεύσει κάποιο μέρος Py την ημέρα d, τότε πρέπει επίσης να παρουσιάζει τον προμηθευτή Sx να είναι στο πλαίσιο της σύμβασης την ημέρα d.

Ο βασικός ξένος περιορισμός από την σχέση SP_SINCE στην σχέση S_SINCE φροντίζει γι' αυτήν την απαίτηση(κάθε προμηθευτής που είναι ικανός να προμηθεύσει κάποιο μέρος πρέπει να είναι στο πλαίσιο της σύμβασης), χρειαζόμαστε επίσης:

```
CONSTRAINT BR9_A IS_EMPTY
   ((S_SINCE JOIN SP_SINCE)
   WHERE S#_SINCE > SINCE};
```

(κανένας προμηθευτής κάτω από τη σύμβαση, δεν μπορεί να παράγει κανένα μέρος πριν αρχίσει η σύμβαση). Επιπλέον, χρειαζόμαστε:

```
CONSTRAINT BR9_ B
  WITH (EXTEND S_ SINCE
        ADD INTERVAL _DATE ([S#_ SINCE: LAST_ DATE ( ) ] }
     AS DURING) {S#, DURING} AS T1,
      (T1 UNION S_DURING) AS T2,
      SP_DURING {S#, DURING} AS T3:
  USING DURING ◄T3 ⊆ T2►;
```

Χωρίς αυτόν τον περιορισμό θα ήταν πιθανό η σχέση SP_DURING να περιείχε την ακόλουθη πλειάδα:

S#	P#	DURING
S1	P1	[d04:d04]

ενώ ταυτόχρονα ούτε η σχέση S_SINCE, αλλά ούτε και η σχέση S_DURING περιέχει μία πλειάδα που να δείχνει ότι ο προμηθευτής Sx ήταν στο πλαίσιο της σύμβασης την $4^{η}$ ημέρα.

Παρατηρείστε ότι ο περιορισμός BR9_B περιλαμβάνει μια επίκληση επιλογέα διαστήματος στην οποία ο χρόνος τέλους(end time) διευκρινίζεται ως η τελευταία ημέρα και μετά συνεχίζει να ζητά για μια σχέση που να περιέχει τέτοια διαστήματα.

Σημειώστε, ότι είναι μια λογική συνέπεια της απαίτησης R9 ότι κάθε αριθμός προμηθευτή που εμφανίζεται στην σχέση SP_DURING εμφανίζεται επίσης και στην σχέση S_DURING ή στην σχέση S_SINCE ή και στις δύο.

ΚΕΦΑΛΑΙΟ 7: ΠΡΑΞΕΙΣ ΣΤΟ ΓΕΝΙΚΕΥΜΕΝΟ ΜΟΝΤΕΛΟ

7.1 Εισαγωγή

Σ' αυτό το κεφάλαιο θα επικεντρωθούμε στη διατύπωση ερωτημάτων πάνω σε μια χρονική βάση δεδομένων. Τα παραδείγματά μας θα βασίζονται στο παράδειγμα των προμηθευτών-αποστολών του κεφαλαίου 6. Η ακόλουθη λίστα δείχνει τα ερωτήματα που θα χρησιμοποιήσουμε αργότερα.

- Query Q1: Δώσε την κατάσταση του προμηθευτή S1 *την* dn ημέρα
- Query Q2:Δώσε τα ζευγάρια των αριθμών των προμηθευτών, ώστε οι ενδεικνυόμενοι προμηθευτές όρισαν την παρούσα κατάστασή τους την ίδια μέρα
- Query Q3: Δώσε τους αριθμούς των προμηθευτών, για τους προμηθευτές που μπορούσαν να προμηθεύσουν παράλληλα τα μέρη P1 και P2
- Query Q4: Δώσε τους αριθμούς των προμηθευτών, για τους προμηθευτές που δεν μπορούσαν να προμηθεύσουν παράλληλα τα μέρη P1 και P2
- Query Q5: Δώσε τους αριθμούς των προμηθευτών, για τους προμηθευτές που μπορούσαν να προμηθεύσουν κάποιο μέρος που έχει αλλάξει την κατάσταση τους, από τότε που αυτοί μπορούσαν να προμηθεύσουν κάποιο μέρος
- Query Q6: Δώσε τα διαστήματα κατά τα οποία τουλάχιστον ένας προμηθευτής είχε συμβόλαιο
- Query Q7: Υποθέστε ότι το αποτέλεσμα απ' το Query Q6 *κρατείται* σε μια μεταβλητή BUSY. Χρησιμοποιήστε την μεταβλητή BUSY για να πάρετε διαστήματα κατά τα οποία κανένας προμηθευτής δεν είχε συμβόλαιο
- Query Q8: Δώσε τους αριθμούς των προμηθευτών, για τους προμηθευτές *που* προσεχώς έχουν συμβόλαιο ενώ είχαν επίσης και προηγουμένως
- Query Q9: Δώσε τις τριάδες S#-PARTS-DURING ώστε ένας συγκεκριμένος προμηθευτής *να* μπορούσε να προμηθεύσει μια συγκεκριμένη σειρά μερών, κατά την διάρκεια ενός συγκεκριμένου διαστήματος
- Query Q10: Υποθέστε ότι το αποτέλεσμα απ' το Query Q9 *κρατείται* σε μια μεταβλητή S_PARTS_DURING. Χρησιμοποιήστε τη μεταβλητή S_PARTS_DURING για να αποσπάσετε τριάδες S# -P# -DURING ώστε ο συγκεκριμένος προμηθευτής μπορούσε να προμηθεύσει το συγκεκριμένο μέρος, κατά την διάρκεια ενός συγκεκριμένου διαστήματος
- Query Q11: Έστω μια σχέση INFLATION, με ιδιότητες ID, DURING *και* PERCENTAGE, ενώ οι {ID} *και* {DURING} είναι μαζί κλειδί. Δώσε ζευγάρια DURING- PERCENTAGE ώστε το ποσοστό πληθωρισμού για το συγκεκριμένο διάστημα, ήταν το ενδεικνυόμενο ποσοστό
- Query Q12: υποθέστε ότι το αποτέλεσμα απ' το Query Q11 κρατείται σε μια μεταβλητή INFLATION (που αντικαθιστά την προηγούμενη σχέση με το ίδιο όνομα). Δώσε διαστήματα όπου το ποσοστό είναι μικρότερο από 5%.

7.2 Παρούσες Μεταβλητές Σχέσης

Σ' αυτή την ενότητα περιορίζουμε την προσοχή μας σε ερωτήματα της βάσης δεδομένων που περιλαμβάνουν μόνο παρούσες μεταβλητές. Το παρακάτω παράδειγμα δείχνει τον ορισμό της βάσης δεδομένων. Σας θυμίζουμε ότι αυτή η βάση δεν μπορεί να αναπαραστήσει ιστορικές πληροφορίες. Μπορεί να αναπαραστήσει όμως μελλοντικές πληροφορίες.

```
S_SINCE {S#, S#_SINCE, STATUS, STATUS_SINCE}
        KEY {S#}

CONSTRAINT CR6 IS_EMPTY
   (S_SINCE WHERE STATUS_SINCE < S#_SINCE)

SP_SINCE {S#, P#, SINCE}
        KEY {S#, P#}
        FOREIGN KEY {S#} REFERENCES S_SINCE

CONSTRAINT CR9 IS EMPTY
   ((S_SINCE JOIN SP_SINCE) WHERE SINCE < S#_SINCE)
```

Ορισμένα από τα ερωτήματα που θα δούμε σ' αυτή την ενότητα δεν έχουν και τόσο χρονική φύση. Γι' αυτό θα παραλείψουμε τα ερωτήματα Q11 *και* Q12 σ' αυτή την ενότητα αφού δεν ταιριάζουν στο ημιχρονικό πνεύμα.

Query Q1: όπως έχουμε αναφέρει οι σχέσεις S_SINCE *και* SP_SINCE επιτρέπεται να περιλαμβάνουν πληροφορίες για το μέλλον. Πιο συγκεκριμένα η σχέση S_SINCE μπορεί να δείχνει ότι κάποιος προμηθευτής θα οριστεί σε μια κατάσταση σε μια μελλοντική ημερομηνία.

Υπενθυμίζουμε ότι αν μια πλειάδα S_SINCE λέει ότι ο προμηθευτής Sx είχε ή θα έχει μια κατάσταση st απ' την ημέρα d και μετά, το ερμηνεύουμε σαν την πλειάδα που ο προμηθευτής Sx είχε, έχει ή θα έχει την κατάσταση st απ' την ημέρα d ως την τελευταία μέρα. Επομένως το Query Q1 μπορεί να απαντηθεί απ' την παραπάνω βάση εάν και μόνο εάν η μέρα dn δεν προηγείται απ' την μέρα STATUS_SINCE για τον προμηθευτή S1. Αν ικανοποιείται αυτή η απαίτηση, μια κατάλληλη φόρμα του ερωτήματος είναι η παρακάτω:

```
(S_SINCE WHERE S# = S# ('S1')) {STATUS}
```

Αλλά μια πιο ικανοποιητική φόρμα είναι η ακόλουθη:

```
(S_SINCE WHERE S# = S# ('S1')
        AND dn ≥STATUS_SINCE) {STATUS}
```

Στην τελευταία φόρμα επιστρέφεται κενό αποτέλεσμα, που σημαίνει (1)απ' την υπόθεσή μας η σχέση S_SINCE περιέχει μια πλειάδα για τον προμηθευτή S1, αλλά (2) η μέρα STATUS_SINCE σ' αυτή την πλειάδα είναι μεγαλύτερη απ' την ημέρα dn

Query Q2: αρχικά να πούμε ότι το ερώτημα αυτό θα έπρεπε να ρωτά για τους προμηθευτές που ορίστηκε ή θα οριστεί η παρούσα κατάστασή τους την ίδια μέρα. Παρατηρήστε ότι ένας προμηθευτής έχει μια παρούσα κατάσταση εάν και μόνο εάν υπάρχει μια πλειάδα γι' αυτό το προμηθευτή στη σχέση S_SINCE. Και αν υπάρχει αυτή η πλειάδα, τότε η τιμή STATUS_SINCE σ' αυτή τη πλειάδα δίνει την ημέρα που αυτή η κατάσταση ορίστηκε σ' αυτό το προμηθευτή. Έτσι:

```
WITH S_ SINCE {S#, STATUS_SINCE} AS T1,
     (T1 RENAME S# AS X#) AS T2,
     (T1 RENAME S# AS Υ#) AS T3,
     (T2 JOIN T3) AS T4,
     (T4 WHERE X# < Υ#) AS T5
T5 {X#, Υ}
```

Η 5η γραμμή δίνει ένα περιορισμό στη σχέση T4 για τις πλειάδες όπου ο αριθμός του προμηθευτή X# είναι μικρότερος απ' τον αριθμό του προμηθευτή Y#. Η ιδέα είναι να αποβληθούν τα ζευγάρια των αριθμών προμηθευτών του τύπου (x, x) και να εξασφαλίσει ότι τα ζευγάρια (x, y) *και* (y, x) δεν εμφανίζονται ταυτόχρονα.

Query Q3: η φόρμα του ερωτήματος είναι:

```
WITH (SP_SINCE WHERE P# = P# ('P1')) {S#} AS T1,
     (SP_SINCE WHERE P# = P# ('P2')) {S#} AS T2:
T1 JOIN T2
```

Query Q4: εδώ πρέπει να δούμε τις σχέσεις S_SINCE *και* SP_SINCE, επειδή ο προμηθευτής που έχει τώρα συμβόλαιο αλλά δεν μπορεί να προμηθεύσει κανένα μέρος, είναι αυτός που δεν μπορεί τώρα να προμηθεύσει τα μέρη P1 και P2 μαζί, και αυτός ο προμηθευτής θα αναπαριστάται στη σχέση S_SINCE κι όχι στη σχέση SP_SINCE. Έτσι:

```
WITH (SP_SINCE WHERE P# = P# ('P1')) {S#} AS T1,
     (SP_SINCE WHERE P# = P# ('P2')) {S#} AS T2,
     (T1 JOIN T2) AS T3:
S_SINCE {S#} MINUS T3 {S#}
```

Query Q5: οι προμηθευτές που μπορούν να προμηθεύσουν τώρα κάποιο μέρος αναπαρίστανται στη σχέση SP_SINCE, και η μέρα που προσφάτως μπορούσε να προμηθεύσει κάποιο μέρος, μπορεί επίσης να αναπαρασταθεί από αυτή τη σχέση. Έτσι:

```
WITH (SUMMARIZE SP_SINCE
        PER SP_SINCE {S#}
        ADD MAX (SINCE) AS MOST_RECENT_SP _DATE) AS T1,
     (S_SINCE JOIN T1) AS T2,
     (T2 WHERE STATUS_SINCE > MOST_RECENT_SP_DATE) AS T3:
T3 {S#}
```

Query Q6: η καλύτερη προσπάθεια για να απαντήσουμε αυτό το ερώτημα είναι να πούμε ότι εάν η νωρίτερη μέρα S#_SINCE στη σχέση SP_SINCE είναι d, τότε τουλάχιστον ένας προμηθευτής έχει συμβόλαιο απ' την ημέρα d ως την τελευταία μέρα του συμβολαίου του.

```
RELATION {TUPLE {DURING INTERVAL_DATE
                 ([MIN (S_SINCE {S#_SINCE}) : LAST_DATE ( ) ])}}
```

Η έκφραση αυτή είναι *του* τύπου RELATION {TUPLE {DURING INTERVAL_DATE ([b: e])}}. Επιστρέφει μια σχέση με μια ιδιότητα, την DURING, και μια πλειάδα. Η έκφραση μέσα στο εξωτερικό σύνολο των αγκυλών είναι μια πλειάδα επίκλησης επιλογέων. Επιστρέφει την μια πλειάδα στη σχέση με *μια* άλλη πλειάδα, και αυτή η πλειάδα έχει ένα μόνο συστατικό, που ονομάζεται DURING. Η τιμή DURING σ' αυτή την πλειάδα ορίζεται εννοώντας επίκληση επιλογέων διαστημάτων του τύπου INTERVAL_DATE([b: e]). Η τιμή b αντιπροσωπεύει το MIN και η τιμή e αντιπροσωπεύει την LAST_DATE.
Υπάρχει ένα πρόβλημα με αυτή τη μορφή. Αυτή η μορφή θα δοκιμάζεται με την προϋπόθεση ότι μπορούμε να είμαστε σίγουροι ότι η σχέση S_SINCE δεν είναι άδεια. Αν είναι άδεια, τότε η επίκληση MIN θα επιστρέφει την "last day", και η γενική έκφραση θα αποφέρει μια σχέση με μια ιδιότητα (DURING) και μια πλειάδα, που θα περιλαμβάνει μια μεταβλητή σχέση της μορφής

```
INTERVAL_DATE ((LAST_DATE ( ) : LAST DATE ( )))
```

Το αποτέλεσμα όμως θα είναι λάθος. Αν δεν μπορούμε να υποθέσουμε ότι η σχέση S_SINCE δεν είναι άδεια, μια πιο σύνθετη μορφή είναι αναγκαία:

```
WITH (SUMMARIZE S_SINCE PER S_SINCE { }
      ADD MIN (S#_SINCE)
      AS EARLIEST) AS T1,
  (EXTEND T1
     ADD INTERVAL_DATE ((EARLIEST: LAST DATE ( )))
     AS DURING) AS T2:
T2 {DURING)
```

Τώρα αν η σχέση S_SINCE είναι άδεια, τα T1 και T2 είναι επίσης άδεια.

Query Q7: υποθέτουμε ότι το αποτέλεσμα του Query Q6 αποθηκεύεται σε μια σχέση BUSY. Αν υποθέσουμε ότι η σχέση BUSY δεν είναι άδεια, τότε θα περιέχει ακριβώς μια σχέση, και η ακόλουθη μορφή θα επαρκεί.

```
RELATION {TUPLE {DURING INTERVAL_DATE
        ([FIRST DATE ( ):
           PRE (DURING FROM (TUPLE FROM BUSY))))))}
```

Η έκφραση αυτή είναι ξανά *του* τύπου RELATION {TUPLE {DURING INTERVAL_DATE ([b: e])}}. Έτσι επιστρέφει μια σχέση με μια ιδιότητα, την DURING, και μια πλειάδα. Η τιμή DURING μέσα σε εκείνη την σχέση ορίζεται για να εξηγήσει την επίκληση επιλογής διαστήματος του τύπου INTERVAL_DATE([b: e]). Η τιμή b αντιπροσωπεύει την τιμή FIRST DATE, και η τιμή e είναι η μέρα που προηγείται απ' το αρχικό σημείο της τιμής DURING, *την* BUSY. Η τιμή e λαμβάνεται σαν έκφραση του τύπου:

```
PRE (DURING FROM (TUPLE FROM BUSY))
```

Υπάρχουν όμως δυο προβλήματα μ' αυτή την μορφή. Αυτή η μορφή επαρκεί με την προϋπόθεση ότι είμαστε σίγουροι ότι η σχέση BUSY δεν είναι άδεια και το μοναδικό διάστημα που περιέχει δεν έχει "τον αρχικό χρόνο" σαν αρχικό του σημείο. Παρακάτω βλέπουμε μια πιο πολύπλοκη μορφή που θα δουλεύει σωστά σε όλες τις περιπτώσεις:

```
USING DURING ◄RELATION {TUPLE {DURING INTERVAL_DATE
                        ([FIRST_DATE ( ): LAST_DATE ( )])}}
             MINUS BUSY►
```

Query Q8: δίνοντας μόνο την ημιχρονική βάση δεδομένων του παραδείγματος, αυτό το ερώτημα δεν μπορεί να απαντηθεί. Στην πραγματικότητα δεν μπορεί να δοθεί η μορφή του ερωτήματος.

Query Q9: εδώ πρέπει να σχεδιάσουμε μια σχέση που μοιάζει με την μεταβλητή σχέση S_PART_DURING.

```
WITH (EXTEND SP_SINCE ADD
      (INTERVAL_P# ([P#: P#]) AS PARTS,
       INTERVAL_DATE ([SINCE: LAST_DATE ( )])
                      AS DURING))
    AS T:
USING (PARTS, DURING) ◄T (S#, PARTS, DURING) ►
```

Query Q10: υποθέστε ότι το αποτέλεσμα απ' το Query Q9 κρατείται σε μια σχέση S_PARTS_DURING. Χρησιμοποιείστε τη σχέση S_PARTS_DURING για να πάρετε τριάδες S#-P#-DURING ώστε ο συγκεκριμένος προμηθευτής να μπορούσε να προμηθεύσει το συγκεκριμένο μέρος κατά την διάρκεια ενός συγκεκριμένου διαστήματος.

```
WITH (UNPACK S_PARTS_DURING ON PARTS) AS T1,
     (EXTEND T1 ADD POINT FROM PARTS AS P#) AS T2:
USING DURING ◄T2 (ALL BUT PARTS) ►
```

7.3 Ιστορικές Μεταβλητές Σχέσεις

Το παρακάτω παράδειγμα δείχνει τον ορισμό της βάσης δεδομένων. Η βάση αυτή είναι πλήρως χρονική, αλλά δεν διακρίνει παρούσες και ιστορικές πληροφορίες. Μπορεί να αναπαριστά πληροφορίες για το μέλλον όσο και για το παρελθόν, αλλά τυπικά πρέπει να χρησιμοποιούμε τεχνητές τιμές "end-of-time" για να σημειώσουμε το τέλος κάθε διαστήματος, του οποίου το αληθινό τέλος είναι την παρούσα στιγμή άγνωστο.

```
S_DURING {S#, DURING}
  USING DURING KEY {S#, DURING}
  USING DURING FOREIGN KEY {S#, DURING}
          REFERENCES S_STATUS_DURING

S_STATUS_DURING {S#, STATUS, DURING}
  USING DURING KEY {S#, DURING}
  USING DURING FOREIGN KEY {S#, DURING}
          REFERENCES S_DURING

SP_DURING {S#, P#, DURING}
  USING DURING KEY {S#, P#, DURING}
  USING DURING FOREIGN KEY {S#, DURING}
          REFERENCES S_DURING
```

Query Q1: αυτό το ερώτημα διατυπώνεται ως εξής:

```
(S_STATUS_DURING WHERE S#= S# ('S1')
      AND dn ∈ DURING) {STATUS}
```

Αυτή η μορφή δουλεύει ακόμα κι αν το dn είναι ημερομηνία στο μέλλον.

Query Q2: επειδή η βάση δεδομένων περιέχει ιστορικές σχέσεις μόνο, είναι απαραίτητο να αναθεωρήσουμε το ερώτημα. Παρακάτω βλέπουμε μια κατάλληλη μορφή του:

```
WITH (EXTEND S_STATUS_DURING
          ADD BEGIN (DURING) AS STB) {S#, STB) AS T1,
      (T1 RENAME S# AS X#) AS T2,
      (T1 RENAME S# AS Y#) AS T3,
      (T2 JOIN T3) AS T4,
      (T4 WHERE X# < Y#) AS T5:
T5 {X#, Y#)
```

Query Q3: ας δούμε ξανά πιο αναλυτικά το ερώτημα. Αυτή εδώ είναι μια κατάλληλη μορφή:

```
WITH (SP_DURING WHERE P# = P# ('P1')) {S#, DURING} AS T1,
      (SP_DURING WHERE P# = P# ('P2')) {S#, DURING} AS T2,
      (USING DURING ◄T1 JOIN T2►) AS T3:
T3 {S#}
```

Query Q4: άλλη μια φορά πρέπει να ορίσουμε το ερώτημα. Αυτή είναι η φόρμα του:

```
WITH (SP_DURING WHERE P# = P# ('P1')) {S#, DURING} AS T1,
      (SP_DURING WHERE P# = P# ('P2')) {S#, DURING} AS T2,
      (USING DURING ◄T1 JOIN T2►) AS T3:
S_DURING {S#} MINUS T3 {S#}
```

Query Q5: η αναθεωρημένη έκδοση παρακάτω είναι πιο πολύπλοκη. Σημειώστε την απαίτηση ότι ορισμένα συγκεκριμένα γεγονότα απαντώνται ενώ το συμβόλαιο που ερωτάται ήταν σε ισχύ. Αυτή η απαίτηση υπολογίζεται στις γραμμές 3 και 6.

```
WITH (S_STATUS_DURING RENAME DURING AS X) {S#, X} AS T1,
      (T1 JOIN S_DURING) AS T2,
      (T2 WHERE X ⊆ DURING) AS T3,
      (SP_DURING RENAME DURING AS Y) {S#, Y} AS T4,
      (T4 JOIN T3) AS T5,
      (T5 WHERE Y ⊆ DURING) AS T6,
      (SUMMARIZE T6 PER T6 {S#, DURING} ADD
           (MAX (BEGIN (X)) AS BXMAX,
            MAX (BEGIN (Y)) AS BYMAX)) AS T7,
      (T7 WHERE BXMAX > BYMAX) AS T8:
T8 {S#}
```

Query Q6: τα ζητούμενα διαστήματα μπορούν να δοθούν με την ακόλουθη φράση:

USING DURING ◄S_DURING {DURING} ►

Query Q7: αυτή η μορφή είναι ίδια ακριβώς ίδια με του προηγούμενου ερωτήματος.

USING DURING ◄RELATION (TUPLE {DURING INTERVAL_DATE
([FIRST_DATE ( ): LAST_DATE ( )])}}
MINUS BUSY►

Query Q8: όπως και σε προηγούμενο κεφάλαιο θα χρησιμοποιήσουμε τον τελεστή TODAY που επιστρέφει την σημερινή ημέρα.

WITH (S_DURING WHERE TODAY ( ) ∈ DURING) {S#} AS T1,
S_DURING {S#} WHERE TODAY ( ) > END (DURING) AS T2:
T1 JOIN T2

Query Q9: η απάντηση δίνεται απ' την παρακάτω μορφή:

WITH (EXTEND SP_DURING
ADD INTERVAL_P# ([P#: P#]) AS PARTS) AS T:
USING (PARTS, DURING) ◄T {S#, PARTS, DURING} ►

Query Q10: αυτή η έκφραση είναι ισοδύναμη με την προηγούμενη ενότητα

WITH (UNPACK S_PARTS_DURING ON PARTS) AS T1,
(EXTEND T1 ADD POINT FROM PARTS AS P#) AS T2:
USING DURING ◄T2 {ALL BUT PARTS} ►

Query Q11: το σημείο που αξίζει να αναφερθούμε εδώ είναι ότι το αποτέλεσμα δεν "συμπιέζεται" *στην* DURING

INFLATION {DURING, PERCENTAGE}

Query Q12: το ζητούμενο αποτέλεσμα δίνεται απ' την παρακάτω μορφή:

PACK (INFLATION WHERE PERCENTAGE < 5) ON DURING

7.4 Παρούσες Και Ιστορικές Σχέσεις Μαζί

Σ' αυτή την ενότητα θα δούμε ερωτήματα πάνω στη βάση που περιλαμβάνουν παρούσες και ιστορικές σχέσεις. Το παρακάτω παράδειγμα δείχνει τον ορισμό της βάσης δεδομένων σε περίληψη. Αυτή η βάση κρατάει τις παρούσες πληροφορίες στις σχέσεις "since" και τις ιστορικές στις "during".

Σημειώστε ότι τα ερωτήματα είναι τα ίδια με την προηγούμενη ενότητα. Παραλείπουμε τα ερωτήματα 10, 11, 12 επειδή η τυπική μορφή τους είναι ακριβώς η ίδια με την προηγούμενη ενότητα.

```
S_SINCE {S#, S#_SINCE, STATUS, STATUS_SINCE}
        KEY {S#}

SP_SINCE {S#, P#, SINCE}
         KEY {S#, P#}
         FOREIGN KEY {S#} REFERENCES S_SINCE

S_DURING {S#, DURING}
         USING DURING KEY {S#, DURING}

S_STATUS DURING {S#, STATUS, DURING}
                USING DURING KEY {S#, DURING}

SP_DURING {S#, P#, DURING}
          USING DURING KEY {S#, P#, DURING}
```

Query Q1: η δυσκολία είναι ότι δεν ξέρουμε αν η απάντηση σ' αυτό το ερώτημα είναι στην σχέση S_SINCE ή στην σχέση S_STATUS DURING. Έτσι:

```
WITH (S_SINCE WHERE S# = S# ('S1')) AS T1,
     (EXTEND T1
      ADD INTERVAL_DATE ([STATUS SINCE: LAST DATE ( )])
      AS DURING) AS T2,
     T2 (STATUS, DURING} AS T3,
     (S_STATUS_DURING WHERE S# = S# ('S1')) AS T4,
     T4 (STATUS, DURING} AS T5,
     (T4 UNION T5) AS T6:
  (T6 WHERE dn ∈ DURING) {STATUS}
```

Μια εναλλακτική μορφή, που περιλαμβάνει ένα τεστ ώστε να δούμε ποια σχέση περιλαμβάνει την επιθυμητή πληροφορία, είναι η εξής:

```
WITH (TUPLE FROM (S_SINCE WHERE S# = S# ('S1'}}) AS t,
      (STATUS FROM t) AS s,
      (STATUS_SINCE FROM t) AS d:
  IF dn ≥d THEN RELATION (TUPLE (STATUS s}}
         ELSE (S STATUS DURING WHERE S# = S# ('S1')
                        AND dn ∈ DURING) {STATUS}
  END IF
```

Query Q2: το περίπλοκο εδώ είναι ότι η σχέση S_SINCE ίσως να δείχνει κάποιο προμηθευτή Sx που να έχει οριστεί σε κάποια παρούσα κατάσταση την ημέρα d και η σχέση S_STATUS DURING ίσως να δείχνει κάποιο προμηθευτή Sy που να έχει οριστεί σε κάποια ιστορική κατάσταση την ίδια ημέρα d. Έτσι:

```
WITH (EXTEND S_STATUS_DURING
       ADD BEGIN (DURING) AS STATUS_SINCE)
            {S#, STATUS_SINCE} AS T1,
      (T1 UNION S_SINCE {S#, STATUS_SINCE}) AS T2,
      (T2 RENAME S# AS X#) AS T3,
      (T2 RENAME S# AS ϒ#) AS T4,
      (T3 JOIN T4) AS T5,
      (T5 WHERE X# < ϒ#) AS T6:
T6 {X#, ϒ#}
```

Query Q3:

```
          WITH (EXTEND SP_SINCE
                 ADD INTERVAL_DATE ([SINCE: LAST_DATE ( )])
                 AS DURING) {S#, P#, DURING} AS T1,
                (SP_DURING UNION T1) AS T2,
                (T2 WHERE P# = P# ('P1')) {S#, DURING} AS T3,
                (T2 WHERE P# = P# ('P2')) {S#, DURING} AS T4,
                (USING DURING ◄T3 JOIN T4►) AS T5:
          T5 { S#}
```

Query Q4:

```
          WITH (EXTEND SP_SINCE
                 ADD INTERVAL DATE {[SINCE: LAST DATE ( )])
                 AS DURING) T S#, P#, DURING} AS T1,
                (SP_DURING UNION T1) AS T2
                (T2 WHERE P# = P# ('P1')) {S#, DURING} AS T3,
                (T2 WHERE P# = P# ('P2')) {S#, DURING} AS T4,
                (USING DURING ◄T3 JOIN T4 ►) AS T5:
S_DURING {S#} MINUS T5 {S#}
```

Query Q5:

```
WITH (EXTEND S_SINCE
        ADD INTERVAL_DATE ([S#_SINCE: LAST DATE ( )])
        AS DURING) {S#, DURING} AS T1,
      (T1 UNION S_DURING) AS T2,
      (EXTEND S_SINCE
    ADD INTERVAL_DATE ([STATUS SINCE: LAST DATE ( )])
    AS DURING) {S#, STATUS, DURING} AS T3,
   (T3 UNION S_STATUS_DURING) AS T4,
   (EXTEND SP_SINCE
    ADD INTERVAL_DATE ([SINCE: LAST_DATE ( )])
    AS DURING) AS T5,
   (T5 UNION SP_DURING) AS T6,
   (T4 RENAME DURING AS X) {S#, X} AS T7,
   (T6 RENAME DURING AS Y) {S#, Y} AS T8,
   (S_DURING JOIN T7) AS T9,
   (T9 WHERE ⊆ DURING) AS T10,
   (T10 JOIN T8) AS T11,
   (T11 WHERE Y ⊆ DURING) AS T12,
   (SUMMARIZE T12 PER T12 {S#, DURING} ADD
        (MAX (BEGIN (X)) AS BXMAX,
         MAX (BEGIN (Y)} AS BYMAX)) AS T13,
        (T13 WHERE BXMAX > BYMAX) AS T14:
T14 {S#}
```

Query Q6:

```
WITH (EXTEND S_SINCE
        ADD INTERVAL [S#_SINCE: LAST DATE ( )]
        AS DURING) AS T1,
      (T1 {S#, DURING} UNION S_DURING) AS T2:
USING DURING ◄T2 {DURING}►
```

Query Q7:

```
USING DURING ◄RELATION {TUPLE {DURING INTERVAL_DATE
        ([FIRST_DATE ( ): LAST_DATE ( )])}}
            MINUS BUSY►
```

Query Q8:

```
(S_SINCE JOIN S_DURING) {S#}
```

Query Q9:

```
WITH (EXTEND SP_SINCE ADD
        (INTERVAL_P# ([P#: P#]) AS PARTS,
         INTERVAL_DATE ([SINCE: LAST DATE ( )))
                                  AS DURING)) AS T1,
     T1 {S#, PARTS, DURING} AS T2,
     (EXTEND SP_DURING
      ADD INTERVAL_P# ([P#: P#)) AS PARTS) AS T3,
     T3 {S#, PARTS, DURING} AS T4:
USING (PARTS, DURING) ◄T3 UNION T4►
```

7.5 Η Βοήθεια Των Ιδεατών Σχέσεων

Μερικά απ' τα ερωτήματα της προηγούμενης ενότητας ήταν αρκετά πολύπλοκα. Σ' αυτή την ενότητα θα προσπαθήσουμε να κάνουμε ευκολότερη τη ζωή του χρήστη, ορίζοντας μια κατάλληλη συλλογή από ιδεατές σχέσεις (virtual relvars).

Οι σχέσεις αυτές ανατρέπουν την οριζόντια και κάθετη ανάλυση που περιγράψαμε στο κεφάλαιο 4. Το παράδειγμά μας είναι πολύ απλό για να διαφωτίσει αυτή μας την ιδέα, και γι' αυτό θα το επεκτείνουμε λίγο. Ας θυμηθούμε λίγο την πληροφορία για την πόλη του προμηθευτή (supplier city), επαναφέροντας τις ιδιότητες CITY *και* CITY_SINCE στην σχέση S_SINCE. Επαναφέρουμε επίσης τη σχέση S_CITY_DURING *με τις* ιδιότητες S#, CITY *και* DURING.

```
S_SINCE {S#, S#_SINCE,
         STATUS, STATUS_SINCE,
         CITY, CITY_SINCE}
       KEY {S#}

S_CITY_DURING {S#, CITY, DURING}
              KEY {S#, DURING}
```

Τώρα μπορούμε να εξηγήσουμε τις ιδεατές σχέσεις που έχουμε στο μυαλό μας. Πρώτα ορίζουμε τέσσερις ιδεατές σχέσεις S_DURING', S_STATUS_DURING', S_CITY_DURING' *και* SP_DURING' που συνδυάζουν παρούσες και ιστορικές πληροφορίες, εξουδετερώνοντας την οριζόντια ανάλυση:

```
VAR S_DURING' VIRTUAL
     S_DURING UNION
    (EXTEND S_SINCE
    ADD INTERVAL_DATE ([S#_SINCE: LAST_DATE ( )])
    AS DURING) {S#, DURING};
```

```
VAR S_STATUS_DURING' VIRTUAL
      S_STATUS_DURING UNION
     (EXTEND S_SINCE
      ADD INTERVAL_DATE ([STATUS_SINCE: LAST DATE ( )])
      AS DURING) {S#, STATUS, DURING};

VAR S_CITY_DURING' VIRTUAL
      S_CITY_DURING UNION
     (EXTEND S_SINCE
      ADD INTERVAL_DATE ([CITY_SINCE: LAST DATE ( )])
      AS DURING) {S#, STATUS, DURING};

VAR SP_DURING' VIRTUAL
      SP_DURING UNION
     (EXTEND SP_SINCE
     (ADD INTERVAL_DATE ([SINCE: LAST DATE ( )])
      AS DURING) {S#, P#, DURING};
```

Μετά ορίζουμε μια ιδεατή σχέση S'' που συνδυάζει την κατάσταση του προμηθευτή με την πόλη, εξουδετερώνοντας την κάθετη ανάλυση.

```
VAR S" VIRTUAL
  USING DURING ◄S_STATUS_DURING' JOIN S_CITY_DURING' ►;
```

Δεν είναι ανάγκη να συμπεριλάβουμε την σχέση S_DURING' στην συνένωση, επειδή κάθε τιμή {S#, DURING} που εμφανίζεται στην S_DURING', εμφανίζεται επίσης στις σχέσεις S_STATUS_DURING' *και* S_CITY_DURING', και το αντίθετο. Για να είμαστε πλήρης θα ορίσουμε ακόμα μια ιδεατή σχέση SP'' για τις αποστολές, όπως και *η* S'' *για* τους προμηθευτές:

```
VAR S'' VIRTUAL
      SP_DURING';
```

Το παρακάτω παράδειγμα δείχνει την δομή των ιδεατών αυτών σχέσεων:

```
S_DURING' {S#, DURING}
     USING DURING KEY {S#, DURING}

S_STATUS_DURING' {S#, STATUS, DURING}
     USING DURING KEY {S#, DURING}

S_CITY_DURING' {S#, CITY, DURING}
     USING DURING KEY {S#, DURING}
```

```
SP_DURING' {S#, P#, DURING}
      USING DURING KEY {S#, P#, DURING}

S'' {S#, STATUS, CITY, DURING}
      USING DURING KEY {S#, DURING}

SP'' {S#, P#, DURING}
      USING DURING KEY {S#, P#, DURING}
```

Ας δούμε τώρα τα ερωτήματα Q1-Q10. Τα ερωτήματα Q11 και Q12 παραμένουν αμετάβλητα.

Query Q1: αυτό το ερώτημα είναι το πιο εύκολο.

```
(S'' WHERE S# = S# ('S1') AND dn ∈ DURING) {STATUS}
```

Query Q2: αυτή η έκφραση μοιάζει με το αντίστοιχο ερώτημα στην ενότητα 7.4, με την διαφορά ότι: (1) η αναφορά στην σχέση S_STATUS_DURING έχει αντικατασταθεί με την σχέση S_STATUS_DURING' και (2) δεν υπάρχει αναφορά στην σχέση S_SINCE.

```
WITH (EXTEND S_STATUS_DURING'
        ADD BEGIN (DURING) AS STB) {S#, STB} AS T1,
      (T1 RENAME S# AS X#) AS T2,
      (T1 RENAME S# AS Y#} AS T3,
      (T2 JOIN T3) AS T4,
      (T4 WHERE X# < Υ#) AS T5:
T5 {X#, Υ#}
```

Query Q3: αυτή τη φορά η έκφραση είναι πιο εύκολη απ' την αντίστοιχη της προηγούμενης ενότητας.

```
WITH (SP'' WHERE P# = P# ('P1')) {S#, DURING} AS T1,
    (SP'' WHERE P# = P# ('P2')) {S#, DURING} AS T2,
    (USING DURING ◄T1 JOIN T2►) AS T3:
T3 S#}
```

Query Q4: και αυτή η έκφραση είναι πιο απλή απ' την αντίστοιχη της προηγούμενης ενότητας.

```
WITH (SP'' WHERE P# = P# ('P#')) {S#, DURING} AS T1,
      (SP" WHERE P# = P# ('P2')) {S#, DURING} AS T2,
      (USING DURING ◄T1 JOIN T2►) AS T3:
S'' {S#} MINUS T3 {S#}
```

Query Q5:

```
WITH (S" RENAME DURING AS X) {S#, X} AS T1,
     (SP" RENAME DURING AS Y) {S#, Y} AS T2,
     (S'' JOIN T2) AS T3,
     (T3 WHERE X ⊆ DURING) AS T4,
     (T4 JOIN T2) AS T5,
     (T5 WHERE Y ⊆ DURING) AS T6,
     (SUMMARIZE T6 PER T6 {S#, DURING} ADD
          (MAX (BEGIN (X)) AS BXMAX,
           MAX (BEGIN (Y)) AS BYMAX)) AS T7,
     (T7 WHERE BXMAX > BYMAX) AS T8:
T8 {S#}
```

Query Q6:

```
USING DURING ◄S_DURING {DURING} ►
```

Θα μπορούσαμε να είχαμε ορίσει την σχέση S'' στην θέση της σχέσης S_DURING'

Query Q7: σ' αυτό το ερώτημα η βοήθεια των ιδεατών σχέσεων δεν είναι δυνατή

Query Q8: θα προτιμήσουμε γι' αυτό το ερώτημα τη μορφή που δώσαμε στην ενότητα 13.4, αφού είναι δύσκολη η έκφρασή του με ιδεατές σχέσεις.

Query Q9:

```
WITH (EXTEND SP_DURING'
     ADD INTERVAL_P# ([P#: P#]) AS PARTS) AS T:
USING (PARTS, DURING) ◄T {SH, PARTS, DURING} ►
```

Query Q10: και σ' αυτό το ερώτημα οι ιδεατές σχέσεις που δημιουργήσαμε δε μας βοηθούν

Σ' αυτό το σημείο θα δώσουμε την προσοχή μας στη διατύπωση ερωτημάτων για την ενημέρωση της βάσης δεδομένων μας. Τα παραδείγματά μας θα βασιστούν στη βάση δεδομένων των προμηθευτών και αποστολών που περιγράφτηκε στο κεφάλαιο 4, ενώ επίσης υποθέτουμε ότι ισχύουν οι περιορισμοί ακεραιότητας που είδαμε στα κεφάλαια 5 *και* 6.

Σας θυμίζουμε ότι χρησιμοποιούμε τον όρο update(ενημέρωση) για να αναφερθούμε στους τελεστές INSERT(εισαγωγή), DELETE(διαγραφή) και UPDATE(ενημέρωση). Όταν θέλουμε να αναφερθούμε στον τελεστή UPDATE, τον ορίζουμε πάντα με κεφαλαία γράμματα.

7.6 Παρούσες Σχέσεις Μόνο

Σ' αυτή την ενότητα θα περιορίσουμε την προσοχή μας στην απλή μορφή ενημέρωσης της βάσης δεδομένων που περιλαμβάνει τις παρούσες σχέσεις S_SINCE και SP_SINCE. Το παράδειγμα 7.1 δείχνει κάποιες ενδεικτικές τιμές. Τα παραδείγματά μας θα βασιστούν σ' αυτές τις συγκεκριμένες τιμές. Σας θυμίζουμε ότι αυτή η βάση δεν μπορεί να αναπαραστήσει ιστορικές πληροφορίες. Μπορεί όμως να αναπαραστήσει μελλοντικές πληροφορίες.

Παράδειγμα 7.1 παρούσες σχέσεις

S_SINCE

S#	S#_SINCE	STATUS	STATUS_SINCE
S1	d04	20	d06
S2	d07	10	d07
S3	d03	30	d03
S4	d14	20	d14
S5	d02	30	d02

SP_SINCE

S#	P#	SINCE
S1	P1	d04
S1	P2	d05
S1	P3	d09
S1	P4	d05
S1	P5	d04
S1	P6	d06
S2	P1	d08
S2	P2	d09
S3	P2	d08
S4	P5	d14

- S_SINCE: *ο προμηθευτής* Sx ήταν στο πλαίσιο της σύμβασης απ' την ημέρα dc και βρισκόταν στην κατάσταση st απ' την ημέρα ds
- SP_SINCE: *ο προμηθευτής* Sx μπορούσε να προμηθεύσει το μέρος Py *απ' την* ημέρα d

UPDATE U1: προσθέστε μια πρόταση για να δείξετε ότι ο προμηθευτής S9 ξεκίνησε με συμβόλαιο, με την κατάσταση 15.

```
INSERT S_SINCE
    RELATION {TUPLE {S#  S# ('S9'),
                     S#_SINCE TODAY ( ),
                     STATUS   15,
                     STATUS_SINCE TODAY ( )}};
```

Η σύνταξη για τον τελεστή INSERT είναι: INSERT R rel-exp;
Το R είναι το όνομα μιας σχέσης και το rel-exp είναι μια έκφραση που αποφέρει μια σχέση, την r, του ίδιου τύπου με την R, και στόχος είναι να εισάγουμε το σύνολο των πλειάδων της σχέσης R στη σχέση r. Το σύνολο των πλειάδων της σχέσης r είναι η πηγή και η σχέση R είναι ο στόχος. Στο παράδειγμά μας ο στόχος είναι η σχέση S_SINCE και η πηγή είναι το σύνολο που περιέχει μια πλειάδα.

UPDATE U2: αφαιρέστε την πρόταση που δείχνει ότι ο προμηθευτής S5 έχει συμβόλαιο

```
DELETE S_SINCE WHERE S# = S# ('S5');
```

Αυτή η μορφή είναι μια συντομογραφία της σχεσιακής εντολής:

```
S_SINCE:= S_SINCE WHERE NOT (S# = S# ('S5'));
```

Μπορεί να θεωρηθεί σαν την πρόταση "ο προμηθευτής S5 είχε συμβόλαιο απ' την ημέρα dc και είχε μια κατάσταση st απ' την ημέρα ds", όπου dc, st και ds είναι οι τιμές S#_SINCE, STATUS και STATUS_SINCE για τον προμηθευτή S5. Υποθέστε ότι η πρόταση μιλούσε για τον προμηθευτή S1 αντί για τον S5. Τότε ο τελεστής DELETE θα είχε αποτύχει λόγω παραβίασης του ξένου κλειδιού, επειδή η σχέση SP_SINCE περιέχει κάποιες πλειάδες για τον προμηθευτή S1.

UPDATE U3: αντικαταστήστε την πρόταση δείχνοντας ότι ο προμηθευτής S1 τοποθετήθηκε στο πλαίσιο του συμβολαίου την $4^{η}$ μέρα, απ' αυτήν που παρουσιάζει ότι ο ίδιος προμηθευτής τοποθετήθηκε στο πλαίσιο του συμβολαίου την $3^{η}$ μέρα.

```
UPDATE S_SINCE WHERE S# = S# ('S1')
        {S#_SINCE:= d03};
```

Αυτό το UPDATE αποτελεί την συντομογραφία που μπορεί να μοιάζει ως εξής:

```
S_SINCE:=
    WITH (S_SINCE WHERE S# = S# ('S1')) AS T1,
         (EXTEND T1 ADD d03 AS NEW S#_SINCE) AS T2,
         T2 {ALL BUT S#_SINCE} AS T3,
         (T3 RENAME NEW S#_SINCE AS S#_SINCE) AS T4:
  (S_SINCE MINUS T1) UNION T4;
```

Προσέξτε ότι αυτή η πρόταση δουλεύει σωστά ακόμα και στην ειδική περίπτωση όπου η σχέση S_SINCE δεν περιέχει καμία πλειάδα για τον προμηθευτή S1.
Κλείνοντας, σημειώνουμε ότι η συμπεριφορά της σχέσης SP_SINCE σε σχέση με τους τελεστές INSERT, DELETE *και* UPDATE, δεν διαφέρει με αυτή της σχέσης S_SINCE.

7.7 Ιστορικές Σχέσεις (Ι)

Σ' αυτήν την ενότητα, αλλά και στην επόμενη, θα δούμε ενημερώσεις σε μια έκδοση της βάσης δεδομένων που περιλαμβάνει μόνο ιστορικές πληροφορίες. Το θέμα θα χωριστεί σε δυο ενότητες, επειδή χρειαζόμαστε να συζητήσουμε δυο ξεχωριστές σκέψεις κάτω απ' την ίδια επικεφαλίδα.
Η βάση δεδομένων περιέχει τρεις ιστορικές σχέσεις, *τις* S_DURING, S_STATUS_DURING *και* SP_DURING. Το παράδειγμα 7.2 δείχνει κάποιες ενδεικτικές τιμές γι' αυτές τις σχέσεις. Η βάση είναι πλήρως χρονική, αλλά δεν διαχωρίζει παρούσες και χρονικές πληροφορίες. Μπορεί να αναπαραστήσει πληροφορίες για το μέλλον, αλλά και για το παρελθόν. Τυπικά όμως πρέπει να χρησιμοποιήσουμε τεχνητές τελικές τιμές(όπως την d99 στο παράδειγμα 7.2) για να δείξουμε το τέλος ενός διαστήματος του οποίου το αληθινό τελικό είναι προς το παρόν άγνωστο.

Παράδειγμα 7.2

S_DURING

S#	DURING
S2	[d02:d04]
S6	[d03:d05]
S7	[d03:d99]

SP_DURING

S#	P#	DURING
S2	P1	[d02:d04]
S2	P2	[d03:d03]
S2	P5	[d03:d04]
S6	P3	[d03:d05]
S6	P4	[d04:d04]
S6	P5	[d04:d05]
S7	P1	[d03:d04]
S7	P1	[d06:d01]
S7	P1	[d09:d99]

S_STATUS_DURING

S#	STATUS	DURING
S2	5	[d02:d02]
S2	10	[d03:d04]
S6	5	[d03:d04]
S6	7	[d05:d05]
S7	15	[d03:d08]
S7	20	[d09:d99]

Ας δούμε τώρα τα ορίσματα των σχέσεων:

- S_DURING: ο προμηθευτής Sx ήταν στο πλαίσιο της σύμβασης σε όλο το διάστημα i
- S_STATUS_DURING: ο προμηθευτής Sx είχε την κατάσταση st σε όλο το διάστημα i
- SP_DURING: ο προμηθευτής Sx μπορούσε να προμηθεύσει το μέρος Py σε όλο το διάστημα i

UPDATE U4: προσθέστε την πρόταση "ο προμηθευτής S2 μπορούσε να προμηθεύσει το μέρος P4 την $2^{η}$ μέρα".
Η ακόλουθη έκφραση INSERT ικανοποιεί την ανάγκη μας:

```
INSERT SP_DURING
  RELATION {TUPLE
          {S# S# ('S2'),
           P# P# ('P4'),
       DURING INTERVAL_DATE ([d02: d02])}};
```

Υποθέστε τώρα ότι η πρόταση είχε το μέρος P5 αντί το P4. Τότε δεν μπορούμε να εισάγουμε την αντίστοιχη πλειάδα στην σχέση SP_DURING, γιατί αν το κάναμε η σχέση θα περιείχε τις ακόλουθες δυο πλειάδες:

S#	P#	DURING
S2	P5	[d02:d02]

S#	P#	DURING
S2	P5	[d03:d04]

Προκειμένου να εξετάσουμε το πρόβλημα, εισάγουμε μια γενική μορφή της INSERT, στην οποία αναφερόμαστε ως U_ INSERT. Αυτός εδώ είναι ο ορισμός. Η πρόταση:

```
USING (ACL) INSERT R r;
```

Ορίζεται σαν συντομογραφία για την σχεσιακή πρόταση:

```
R: =USING (ACL) ◄R UNION r►;
```

Ισοδύναμα:

```
R: =PACK
      ((UNPACK R ON (ACL))
            UNION
      (UNPACK r ON (AC)))
   ON (ACL);
```

Η U_INSERT μπορεί να θεωρηθεί, ασαφώς βέβαια, σαν μια κανονική INSERT εντολή της μη συμπιεσμένης έκδοσης της πηγαίας σχέσης, μέσα σε μια μη συμπιεσμένη έκδοση της σχέσης που μας ενδιαφέρει, ακολουθούμενη από μια κατάλληλη επανασυμπίεση της σχέσης.
Γυρίζοντας τώρα *στο* UPDATE U4, *το* ακόλουθο U_INSERT ικανοποιεί την ανάγκη μας, είτε το μέρος είναι το P4, είτε το P5.

```
USING DURING INSERT SP_DURING
    RELATION {TUPLE
            {S# S# ('S2'),
             P# P# ('Py'),
             DURING INTERVAL_DATE ([d02: d02])}};
```

Στην περίπτωση του κανονικού INSERT, ο τελεστής INSERT θα αποτύχει, εάν παραβιάσει κάποιο περιορισμό κλειδιού. Με τον ίδιο τρόπο, ο τελεστής U_INSERT *θα* αποτύχει εάν παραβιάσει κάποιο περιορισμό κλειδιού U_KEY.

UPDATE U5: αφαιρέστε την πρόταση "ο προμηθευτής S6 μπορούσε να προμηθεύσει *το* μέρος P3 απ' την $3^{η}$ *ως την* $5^{η}$ μέρα".
Η βάση δεδομένων περιέχει ιστορικές εγγραφές, όπου αν μια δεδομένη πρόταση p *που* περιλαμβανόταν στην βάση δεδομένων, δεν είναι αυτή η πρόταση ιστορική εγγραφή που θα έπρεπε να κρατείται στην βάση δεδομένων για *πάντα*.
Ας δούμε τώρα πως μπορούμε να πετύχουμε τον επιθυμητό στόχο. Στην πραγματικότητα, ο τελεστής DELETE θα μας βοηθήσει:

```
DELETE SP_DURING
    WHERE S# = S# ('S6')
    AND    P# = P# ('P3')
    AND DURING = INTERVAL_DATE ([d03: d05]);
```

Υποθέστε όμως ότι η πρόταση περιείχε μόνο την $4^{η}$ μέρα αντί το διάστημα 3-5. Τότε *η* αντίστοιχη πρόταση DELETE

```
DELETE SP_DURING
    WHERE S# = S# ('S6')
    AND    P# = P# ('P3')
    AND DURING = INTERVAL_DATE ([d04: d04]);
```

δεν θα έχει κανένα αποτέλεσμα, επειδή δεν υπάρχει καμία πλειάδα για τον προμηθευτή S6 και το μέρος P3 με το DURING = [d04:d04] στην σχέση SP_DURING. Σ*την* πραγματικότητα μια τέτοια πλειάδα εμφανίζεται μόνο όταν αποσυμπιέζουμε αυτή την σχέση στο DURING. Η πρόταση

```
USING (ACL) DELETE R WHERE P;
```

ορίζεται σαν συντομογραφία της σχεσιακής πρότασης

R: = USING (ACL) ◄R WHERE NOT (p) ►;

Ισοδύναμα:

```
R: =ACK
      ((UNPACK R ON (ACL)) WHERE NOT (P))
   ON (ACL);
```

Έτσι, επιστρέφοντας *στο* UPDATE U5, το ακόλουθο U_DELETE μας ικανοποιεί είτε το διάστημα [di: dj] *είναι το* [d03: d05] είτε το [d04: d04].

```
USING DURING DELETE SP_DURING
   WHERE S# = S# ('S6')
   AND    P# = P# ('P3')
   AND    POINT FROM DURING ∈
                 INTERVAL_DATE ([di: dj]);
```

UPDATE U6: αντικαταστήστε την πρόταση "*ο προμηθευτής* S2 μπορούσε να προμηθεύσει το μέρος P5 απ' την $3^{η}$ μέρα ως την $4^{η}$", με την πρόταση "ο προμηθευτής S2 μπορούσε να προμηθεύσει το μέρος P5 απ' την $2^{η}$ μέρα ως την $4^{η}$".

```
UPDATE SP_DURING
     WHERE S# = S# ('S2')
     AND    P# = P# ('P5')
     AND DURING = INTERVAL DATE ([d03: d04])
   (DURING: = INTERVAL_DATE ([d02: d04])};
```

Σημειώνουμε ότι μας επιτρέπεται η ενημέρωση των ιδιοτήτων(attribute update), *και στην* τελευταία μας γραμμή θα μπορούσε να εκφραστεί ως εξής:

BEGIN (DURING): = d02

Αυτή η απλούστερη μορφή σημαίνει ότι το αρχικό σημείο BEGIN του διαστήματος DURING μπορεί να τεθεί ως d02, ενώ το τελικό σημείο END μένει αμετάβλητο. Με τον ίδιο τρόπο, η έκφραση

END (DURING): = d07

θα εννοούσε ότι το τελικό σημείο του διαστήματος μπορεί να τεθεί ως d07 ενώ το αρχικό σημείο BEGIN μένει αμετάβλητο.
Πίσω στο παράδειγμά μας υποθέστε ότι η απαίτηση ήταν να αντικαταστήσουμε την πρόταση "ο προμηθευτής S2 μπορούσε να προμηθεύσει το μέρος P5 την $3^{η}$ μέρα", *με την* πρόταση "ο προμηθευτής S2 μπορούσε να προμηθεύσει το μέρος P5 την $2^{η}$ μέρα". Τότε

η αντίστοιχη πρόταση UPDATE,

```
UPDATE SP_DURING
      WHERE S# = S# ('S2')
      AND    P# = P# ('PS')
      AND DURING = INTERVAL_DATE ([d03: d03])
     (DURING: = INTERVAL_DATE ([d02: d02])} ;
```

δεν θα είχε καμιά επιρροή, επειδή δεν υπάρχει πλειάδα για τον προμηθευτή S2 *και το* μέρος P5 με το DURING = [d03:d03] στην σχέση SP_DURING.

Η πρόταση

```
USING (ACL) UPDATE R WHERE P {attribute updates};
```

ορίζεται σαν συντομογραφία της σχεσιακής πρότασης

```
R: = PACK
      (((UNPACK R ON (ACL)) WHERE NOT (P))
          UNION
        f ((UNPACK R ON (ACL)) WHERE P))
       ON (ACL);
```

Όπου f είναι μια συνάρτηση που επιστρέφει την σχέση που είναι το αποτέλεσμα της εφαρμογής της συγκεκριμένης ενημέρωσης ιδιοτήτων (attribute updates). Αν το ACL είναι άδειο, το U_UPDATE γίνεται μια κανονική σχέση UPDATE. Επιστρέφοντας στο UPDATE U6, το ακόλουθο U_UPDATE ικανοποιεί την ανάγκη μας:

```
USING DURING
UPDATE SP_DURING
          WHERE S# = S# ('S2')
          AND P# = P# ('P5)
          AND DURING = INTERVAL DATE ([d03: d03])
         {DURING: = INTERVAL_DATE ([d01: d01])};
```

7.8 Ιστορικές Σχέσεις (II)

Ας γυρίσουμε τώρα στο ερώτημα της ενημέρωσης των ιστορικών σχέσεων S_DURING *και* S_STATUS_DURING. Πρώτα σας υπενθυμίζουμε το σχέδιο του κατάλληλου τμήματος της βάσης δεδομένων:

```
S_DURING {S#, DURING}
    USING DURING KEY {S#, DURING}
    USING DURING FOREIGN KEY {S#, DURING}
        REFERENCES S_STATUS_DURING
```

```
S_STATUS_DURING {S#, STATUS, DURING}
   USING DURING KEY {S#, DURING}
   USING DURING FOREIGN KEY {S#, DURING}
        REFERENCES S_DURING
```

Σημειώνουμε ότι καθεμιά απ' αυτές τις σχέσεις περιλαμβάνει ένα ξένο κλειδί U_KEY που αναφέρει το άλλο: στη σχέση S_STATUS_DURING απεικονίζει το γεγονός ότι κάθε προμηθευτής που έχει μια κατάσταση σε κάποια χρονική στιγμή, και στη σχέση S_DURING απεικονίζει το γεγονός ότι κάθε προμηθευτής με συμβόλαιο σε κάποια χρονική στιγμή, πρέπει να βρίσκεται σε κάποια κατάσταση σ' αυτή τη στιγμή.

UPDATE U7: προσθέστε την/τις απαραίτητες προτάσεις για να δείξετε ότι ο προμηθευτής S9 μόλις τοποθετήθηκε με συμβόλαιο (από σήμερα), με την κατάσταση 15. Ο παρακάτω κώδικας *μας* ικανοποιεί:

```
INSERT S_DURING RELATION {TUPLE
    {S#    S# ('S9'),
     DURING INTERVAL_DATE ([TODAY ( ): d99])}},

INSERT S_STATUS_DURING RELATION {TUPLE
    {S#    S# ('S9'),
     STATUS 15,
     DURING INTERVAL_DATE ([TODAY ( ): d99])}};
```

ΕΞΗΓΗΣΗ: είναι φανερό ότι χρειαζόμαστε δυο INSERT εδώ: οι σχέσεις που στοχεύουμε είναι η S_DURING *και η* S_STATUS_DURING, και οι αντίστοιχες εκφράσεις είναι απλές επικλήσεις επιλογέων σχέσης. Προσέξτε ότι τα δυο INSERTs χωρίζονται με κόμμα, κι όχι με άνω τελεία, και συγκεντρώνονται σε μια μόνο πρόταση. Η πρόταση αυτή αποτελεί συντομογραφία μιας πολύπλοκης εντολής που μοιάζει με την παρακάτω:

```
S_DURING
   : = S_DURING UNION RELATION {TUPLE
        {S#    S# ('S9'),
         DURING INTERVAL_DATE ([TODAY ( ): d99])}},

S_STATUS_DURING
   : = S_STATUS_DURING UNION RELATION {TUPLE
        {S#    S# ('S9'),
         STATUS 15,
         DURING INTERVAL_DATE ([TODAY ( ): d99])}};
```

Η σημασιολογία είναι ως ακολούθως:

1) Πρώτον, οι εκφράσεις πηγής αξιολογούνται
2) Δεύτερον, οι μεμονωμένες εντολές εκτελούνται στη σειρά όπως γράφονται

UPDATE U8: αφαιρέστε την/τις προτάσεις που δείχνουν ότι ο προμηθευτής S7 είχε συμβόλαιο.
Πάλι θα χρειαστεί μια πολλαπλή έκφραση ενημέρωσης:

```
DELETE SP_DURING WHERE S# = S# ('S7'),
DELETE S_STATUS_DURING WHERE S# = S# ('S7'),
DELETE S_DURING WHERE S# = S# ('S7');
```

Εδώ είναι η γενική επέκταση από την άποψη της σχεσιακής ανάθεσης:

```
SP_DURING:= SP_DURING
                WHERE NOT (S# = S# ('S7')),

S_STATUS_DURING:= S_STATUS_DURING
                        WHERE NOT (S# = S# ('S7')),

S_DURING:= S_DURING
              WHERE NOT (S# = S# (' S7'));
```

UPDATE U9: το συμβόλαιο του προμηθευτή S7 μόλις έχει τερματιστεί. Ενημερώστε την βάση κατάλληλα.
Η ενημέρωση *στο* UPDATE U2(στην ημιχρονική βάση) περιείχε τον τελεστή DELETE. Η ενημέρωση UPDATE U9 σε αντίθεση, (σε μια πλήρως χρονική βάση) περιλαμβάνει τον τελεστή UPDATE:

```
UPDATE S_DURING WHERE S# = S# ('S7')
                    AND TODAY ( ) ∈ DURING
            {END (DURING):= TODAY ( )),

UPDATE S_STATUS_DURING WHERE S# = S# ('S7')
                      AND TODAY ( ) ∈ DURING
          {END (DURING):= TODAY ( )},

UPDATE SP_DURING WHERE S# = S# ('S7')
    AND TODAY ( ) ∈ DURING
  {END (DURING):= TODAY ( )};
```

7.9 Ιστορικές Και Παρούσες Σχέσεις Μαζί

Τώρα θα δούμε την προτιμότερη για μας έκδοση της βάσης δεδομένων, που περιλαμβάνει παρούσες και ιστορικές σχέσεις μαζί. Το παράδειγμα 7.3 δείχνει κάποιες ενδεικτικές τιμές, στις οποίες θα βασίσουμε τα παραδείγματά μας. Η βάση είναι πλήρως χρονική.

Σας θυμίζουμε ότι οι σχέσεις "during" δεν περιλαμβάνουν καμιά πληροφορία όσον αφορά το παρόν ή το μέλλον.

Παράδειγμα 7.3

S_SINCE

S#	S#_SINCE	STATUS	STATUS_SINCE
S1	d04	20	d06
S2	d07	10	d07
S3	d03	30	d03
S4	d04	20	d08
S5	d02	30	d02

S_DURING

S#	DURING
S2	[d02:d04]
S6	[d03:d05]

SP_DURING

S#	P#	DURING
S2	P1	[d02:d04]
S2	P2	[d03:d03]
S3	P5	[d05:d07]
S4	P2	[d06:d09]
S4	P4	[d04:d08]
S6	P3	[d03:d03]
S6	P3	[d05:d05]

SP_SINCE

S#	P#	SINCE
S1	P1	d04
S1	P2	d05
S1	P3	d09
S1	P4	d05
S1	P5	d04
S1	P6	d06
S2	P1	d08
S2	P2	d09
S3	P2	d08
S4	P5	

S_STATUS_DURING

S#	STATUS	DURING
S1	15	[d04:d05]
S2	5	[d02:d02]
S2	10	[d03:d04]
S4	10	[d04:d04]
S4	25	[d05:d07]
S6	5	[d03:d04]
S6	7	[d05:d05]

Εδώ βλέπουμε τις κατάλληλες δηλώσεις:

- S_SINCE: ο προμηθευτής Sx είχε συμβόλαιο απ' την ημέρα dc και είχε την κατάσταση st απ' την ημέρα ds.
- SP_SINCE: ο προμηθευτής Sx μπορούσε να προμηθεύσει το μέρος Py *απ' την* ημέρα d
- S_DURING: ο προμηθευτής Sx είχε συμβόλαιο κατά το διάστημα i
- S_STATUS_DURING: *ο* προμηθευτής Sx είχε την κατάσταση st κατά το διάστημα i
- SP_DURING: ο προμηθευτής Sx μπορούσε να προμηθεύσει το μέρος Py κατά το διάστημα i

UPDATE U10: προσθέστε την πρόταση "ο προμηθευτής S4 μπορούσε να προμηθεύσει το μέρος P4 ως την $10^{η}$ μέρα".
Η ακόλουθη σύνταξη της INSERT μας δίνει την απάντηση που θέλουμε:

```
INSERT SP_SINCE
        RELATION {TUPLE {S#     S# ('S4'),
                         P#     P# ('P4'),
                         SINCE d10}};
```

Υποθέστε όμως ότι η πρόταση μιλούσε *για την* $9^{η}$ μέρα αντί την $10^{η}$. Με τις τιμές του παραδείγματος 7.3, το αντίστοιχο INSERT θα είχε αποτύχει ως παραβίαση του περιορισμού. Αυτός ο περιορισμός προορίζεται να επιβάλει την απαίτηση ότι αν η βάση δείχνει τον προμηθευτή Sx με δυνατότητα να προμηθεύσει το μέρος Py τις μέρες d *και* d+1, τότε πρέπει να περιέχει ακριβώς μια πλειάδα που δείχνει αυτό το γεγονός. Αν επιτρεπόταν το INSERT, οι σχέσεις SP_SINCE *και* SP_DURING θα περιείχε τις ακόλουθες πλειάδες και θα είχαμε παραβίαση του περιορισμού.

S#	P#	SINCE
S4	P4	d09

S#	P#	DURING
S4	P4	[d04:d08]

Για να προσθέσουμε την ανανεωμένη πρόταση, θα πρέπει να κάνουμε τα ακόλουθα:

- Διαγραφή της πλειάδας για τον προμηθευτή S4 και το μέρος P4 με την DURING = [d04:d08] απ' την σχέση SP_DURING
- Εισαγωγή μιας πλειάδας για τον προμηθευτή S4 και το μέρος P4 με το SINCE=d04 στην σχέση SP_SINCE

Αυτές οι δυο ενημερώσεις πρέπει να εκτελεστούν σαν μέρος της ίδιας πρότασης. Ο παρακάτω κώδικας λύνει το πρόβλημα:

```
DELETE SP_DURING WHERE S# = S# ('S4')
        AND P# = P# ('P4')
        AND END (DURING) = d08,

INSERT SP_SINCE
    RELATION (TUPLE (S#      S# ('S4'),
                     P#      P# ('P4'),
                     SINCE d04)};
```

UPDATE U11: ο προμηθευτής S1 δεν μπορεί να προμηθεύσει πια το μέρος P1. Ενημερώστε την βάση αντίστοιχα.
Σημειώστε απ' το παράδειγμα 7.3 ότι ο προμηθευτής S1 μπορούσε να προμηθεύσει το μέρος P1 ως την $4^{η}$ μέρα. Ο παρακάτω κώδικας δίνει την απάντηση:

```
INSERT SP_DURING RELATION
    {TUPLE{S# S# ('S1'), P# P# ('P1'),
           DURING INTERVAL_DATE ([d04: TODAY ( ))])},
  DELETE SP_SINCE WHERE S1 = S# ('S1') AND P# = P# ('P1');
```

UPDATE U12: αντικαταστήστε την πρόταση "ο προμηθευτής S2 μπορούσε να προμηθεύσει το μέρος P1 ως την $8^{η}$ μέρα", με την πρόταση "ο προμηθευτής S2 μπορούσε να προμηθεύσει το μέρος P1 ως την $7^{η}$ μέρα".
Η ακόλουθη έκφραση UPDATE μας δείχνει τον τρόπο:

```
  UPDATE SP_SINCE WHERE S# = S# ('S2') AND P# = P# ('P1')
      {SINCE: = d07};
```

Υποθέστε όμως ότι η νέα τιμή SINCE ήταν η $5^{η}$ αντί την $7^{η}$. Δίνοντας τις τιμές του παραδείγματος 7.3, το αντίστοιχο UPDATE θα είχε αποτύχει σαν παραβίαση του περιορισμού. Αν επιτρεπόταν το UPDATE, *οι* σχέσεις SP_SINCE *και* SP_DURING *θα* περιείχαν αντίστοιχα τις δυο ακόλουθες πλειάδες. Πάλι όμως θα υπήρχε παραβίαση του περιορισμού.

S#	P#	SINCE
S2	P1	d05

S#	P#	DURING
S2	P1	[d02:d04]

Αυτός εδώ είναι ο σωστός κώδικας:

```
DELETE SP_DURING WHERE S# = S# ('S2') AND P# = P# ('P1')
        AND END (DURING) = d04,

UPDATE SP_SINCE WHERE S# = S# ('S2') AND P# = P# ('P1')
      {SINCE: = d02};
```

Γενικότερα, ο κώδικας που αντικαθιστά την πρόταση "ο προμηθευτής Sx μπορούσε να προμηθεύσει το μέρος Py ως την μέρα d", με την *πρόταση* "ο προμηθευτής Sx μπορούσε να προμηθεύσει το μέρος Py ως την μέρα d":

```
WITH {SP_DURING WHERE S# = Sx AND P# = Py AND
                      {IS_PRIOR_DATE {END (DURING), d) OR
                        END (DURING) ≥d)) AS R1;
IF IS_EMPTY (R1)
      THEN UPDATE SP_SINCE WHERE S# = Sx AND P# = Py
            {SINCE: = d'};
      ELSE
          SP_DURING:= SP_DURING MINUS R1,
          UPDATE SP_SINCE WHERE S# = Sx AND P# = Py
            {SINCE: = SINCE FROM (TUPLE FROM (
            (SUMMARIZE R1 PER R1 {S#, P#)
              ADD MIN (BEGIN (DURING)) AS SINCE))));
END IF;
```

Κλείνουμε αυτή την ενότητα με τις ακόλουθες παρατηρήσεις. Πρώτα, έχουμε εξετάσει τις προσπάθειες για να αλλάξουμε τα συστατικά του SINCE σε μια πλειάδα SP_SINCE, αλλά όχι τις προσπάθειες για να αλλάξουμε τα συστατικά των S# ή P#. Δεύτερον, έχουμε εξετάσει τις ενημερώσεις που επηρεάζουν τις σχέσεις S_SINCE, S_DURING *και* S_STATUS_DURING.

7.10 Η Βοήθεια Των Ιδεατών Σχέσεων

Είναι φανερό απ' τις προηγούμενες ενότητες ότι η ενημέρωση μιας χρονικής βάσης δεδομένων είναι ένα πολύπλοκο γεγονός. Αν η βάση περιέχει παρούσες και ιστορικές σχέσεις, τότε:

- ✓ Είναι συχνά αδύνατο να μιλήσουμε για ένα απ' τους τρεις τελεστές ενημέρωσης(INSERT, DELETE *και* UPDATE) μεμονωμένα
- ✓ Είναι συχνά αδύνατο να μιλήσουμε για ενημέρωση μόνο παρούσων σχέσεων, ή μόνο ιστορικών σχέσεων

Ωστόσο μπορούμε να δώσουμε συντακτικές συντομογραφίες για να κάνουμε την ζωή του χρήστη ευκολότερη. Ας δούμε για παράδειγμα τους ακόλουθους ορισμούς σχέσεων απ' το κεφάλαιο 6:

```
VAR S_SINCE RELATION
   {S# S#, ..., STATUS INTEGER,
     STATUS_SINCE DATE SINCE FOR {STATUS}
          -HISTORY IN (S STATUS DURING)}
   KEY {S#};
```

```
VAR S_STATUS_DURING RELATION
    {S# S#, STATUS INTEGER, DURING INTERVAL_DATE}
    USING DURING KEY {S#, DURING};
```

Τώρα σας προτείνουμε μια επιπλέον συντακτική επέκταση:

```
  VAR S_SINCE RELATION
    {S# S#, ..., STATUS INTEGER,
     STATUS_SINCE DATE SINCE_FOR {STATUS}
                    HISTORY_IN (S_STATUS_DURING)
                    COMBINED_IN (S_STATUS_DURING')}
       KEY {S#};
```

Σκοπός αυτού του νέου ορισμού είναι να αναγκάσει το σύστημα να παρέχει, αυτόματα, έναν εικονικό ορισμό σχέσης, της μορφής

```
VAR S_STATUS_DURING' VIRTUAL
       S_STATUS_DURING UNION
      (EXTEND S_SINCE
       ADD INTERVAL_DATE ([STATUS_SINCE: LAST_DATE ( )])
       AS DURING) {S#, STATUS, DURING};
```

UPDATE U13: Η κατάσταση του προμηθευτή S1 μόλις άλλαξε κι έγινε 15. Ενημερώστε *την* βάση κατάλληλα.

```
    USING DURING UPDATE S_STATUS_DURING'
                    WHERE S# = S# ('S1')
                    AND POINT FROM DURING ∈
                          INTERVAL_DATE ([TODAY ( ): d99])
                    {STATUS: = 15};
```

Υποθέστε ότι η ιδεατή σχέση S_STATUS_DURING' περιείχε την ακόλουθη παρούσα πλειάδα για τον προμηθευτή S1:

S#	STATUS	DURING
S1	20	[d04:d99]

Υποθέστε επίσης ότι η σημερινή μέρα είναι η $10^{η}$. Μετά την ενημέρωση η σχέση θα περιέχει τις ακόλουθες δυο πλειάδες:

S#	STATUS	DURING
S1	15	[d10:d99]
S1	20	[d04:d09]

ΚΕΦΑΛΑΙΟ 8 :ΤΙΘΕΜΕΝΟΙ ΚΑΙ ΚΑΤΑΓΕΓΡΑΜΜΕΝΟΙ ΧΡΟΝΟΙ

8.1 Εισαγωγή

Σε αυτό το κεφάλαιο, ρίχνουμε μια πιο στενή ματιά στις έννοιες του έγκυρου χρόνου και του χρόνου συναλλαγής, τους οποίους αντιμετωπίσαμε αρχικά στο κεφάλαιο 2. Μέσω μιας γρήγορης αναθεώρησης, εξετάζουμε το ακόλουθο απλό παράδειγμα. Υποθέστε ότι η σχέση S_DURING περιέχει αυτήν την περίοδο μόνο μια πλειάδα για τον προμηθευτή S2, κατά συνέπεια:

S#	DURING
S2	[d02:d04]

Ο έγκυρος χρόνος για την πρόταση "ο προμηθευτής S2 ήταν στο πλαίσιο της σύμβασης" είναι το διάστημα από την $2^{η}$ ημέρα έως την $4^{η}$ ημέρα.
Τώρα υποθέστε ότι η προηγούμενη πλειάδα υπήρξε στη βάση δεδομένων από την χρονική στιγμή tl έως την χρονική στιγμή t2. Ο χρόνος συναλλαγής για την πρόταση "ο προμηθευτής S2 ήταν στο πλαίσιο της σύμβασης από την $2^{η}$ ημέρα ως την $4^{η}$ ημέρα" είναι το διάστημα από tl *στο* t2.

Παρατηρήστε, επομένως, ότι:

1. Οι έγκυροι χρόνοι αναφέρονται σε κάτι που πιστεύουμε αυτήν την περίοδο ότι είναι αληθές, ενώ οι χρόνοι συναλλαγής αναφέρονται στο πότε μια βάση δεδομένων είπε ότι κάτι ήταν αληθινό.
2. Οι έγκυροι χρόνοι και οι χρόνοι συναλλαγής, είναι και οι δύο σύνολα σχέσεων. Στο παράδειγμά μας, ο έγκυρος χρόνος για την πρόταση "ο προμηθευτής S2 ήταν στο πλαίσιο της σύμβασης" είναι το διάστημα {[d02:d04]}. Εάν η σχέση S_DURING δείχνει ότι ο προμηθευτής S2 ήταν επίσης στο πλαίσιο της σύμβασης από την $6^{η}$ *ημέρα* μέχρι την $9^{η}$, τότε ο έγκυρος χρόνος για την πρόταση "ο προμηθευτής S2 ήταν στο πλαίσιο της σύμβασης" θα είναι το διάστημα {[d02:d04] *και* [d06:d09]}. Ωστόσο, όταν το σύνολο διαστημάτων που είναι κάποιος ιδιαίτερος έγκυρος χρόνος ή χρόνος συναλλαγής περιέχει μόνο ένα διάστημα, λέμε συχνά, ανεπίσημα, ότι ο εν λόγω έγκυρος χρόνος ή ο χρόνος συναλλαγής είναι ακριβώς αυτό καθ' εαυτό το ενιαίο διάστημα.
3. Οι έγκυροι χρόνοι είναι αναθεωρήσιμοι, αλλά οι χρόνοι συναλλαγής δεν είναι
4. Οι έγκυροι χρόνοι μπορούν να αναφερθούν στο παρελθόν στο παρόν, ή στο μέλλον, αλλά οι χρόνοι συναλλαγής μπορούν να αναφερθούν μόνο στο παρελθόν και το παρόν. Για παρόμοιους λόγους, οι χρόνοι συναλλαγής δεν μπορούν επίσης να αναφερθούν σε έναν χρόνο στο παρελθόν νωρίτερα από το χρόνο *που* προστέθηκε η σχέση στη βάση δεδομένων.
5. Όλοι οι "χρόνοι" που έχουμε εξετάσει σε αυτό το βιβλίο πριν από το παρόν κεφάλαιο είναι "έγκυροι χρόνοι".

Μια τελευταία παρατήρηση: Ο χρόνος συναλλαγής ισχύει για όλες τις πιθανές σχέσεις, χρονικές ή όχι. Σε αντίθεση, η έννοια έγκυρος χρόνος ισχύει μόνο για χρονικές σχέσεις, επειδή οι χρονικές σχέσεις είναι οι μόνες που περιλαμβάνουν τις ιδιότητες του έγκυρου χρόνου.

Θυμηθείτε πάλι την S_DURING πλειάδα

S#	DURING
S2	[d02:d04}

υποθέτουμε όπως πρίν ότι:

- ✓ Αυτή η πλειάδα δηλώνει την πρόταση, που την ονομάζουμε p "ο προμηθευτής S2 ήταν στο πλαίσιο της σύμβασης από την $2^{η}$ έως την $4^{η}$ ημέρα "
- ✓ Ο χρόνος συναλλαγής για την p είναι το διάστημα από το t1 έως το t2

Παρατηρήστε τώρα πολύ προσεκτικά, ότι η *πρόταση* p δεν υπονοεί *την* πρόταση "ο προμηθευτής S2 ήταν στο πλαίσιο της σύμβασης"! Μάλλον, υπονοεί *την* πρόταση "ο προμηθευτής S2 ήταν στο πλαίσιο της σύμβασης σε κάποιο χρόνο". Κατά συνέπεια, μπορούμε εύλογα να πούμε ότι αυτή η τελευταία πρόταση, όπως η πρόταση p από την οποία προέρχεται, έχει το χρόνο συναλλαγής στο διάστημα από το t1 *στο* t2. Εντούτοις, δεν μπορούμε να πούμε το ίδιο για την πρόταση "ο προμηθευτής S2 ήταν στο πλαίσιο της σύμβασης:

Στην πραγματικότητα, η πρόταση "ο προμηθευτής S2 ήταν στο πλαίσιο της σύμβασης" δεν παρουσιάζεται στην βάση μας. Ενώ ένα επιχείρημα μπορεί να προβληθεί από την κοινή αίσθηση ότι η πρόταση "υπάρχει ένας χρόνος t έτσι ώστε ο προμηθευτής S2 ήταν στο πλαίσιο της σύμβασης στο χρόνο t" υπονοεί την πρόταση " ο προμηθευτής S2 ήταν στο πλαίσιο της σύμβασης", ένα τέτοιο επιχείρημα δεν ισχύει. Εδώ είναι ένα διαφορετικό παράδειγμα που κάνει το θέμα πιο σαφές: Η πρόταση "όλοι οι πολιτικοί είναι διεφθαρμένοι αυτό το έτος" δεν υπονοεί την πρόταση "όλοι οι πολιτικοί είναι διεφθαρμένοι". (Ακόμα κι αν η τελευταία πρόταση είναι αληθινή, εντούτοις *δεν* μπορούμε να συμπεράνουμε την αλήθεια της πρότασης "όλοι οι πολιτικοί είναι διεφθαρμένοι αυτό το έτος").

Πίσω στο αρχικό παράδειγμα. Παρατηρήστε τώρα ότι η δήλωση "ο χρόνος συναλλαγής για την πρόταση p είναι το διάστημα από *το* t1 έως t2" είναι από μόνη της μια πρόταση και μπορεί επομένως να αντιπροσωπευθεί ως πλειάδα:

S#	DURING	X_DURING
S2	[d02:d04]	[t1: t2]

(χρησιμοποιούμε το X_DURING για να δηλώσουμε το χρόνο συναλλαγής)

Αυτή η πλειάδα, περιλαμβάνει δύο ευδιάκριτες χρονικά σφραγισμένες προτάσεις, η μία που αντιπροσωπεύει έναν έγκυρο χρόνο και η άλλη έναν χρόνο συναλλαγής.
Σημειώστε προσεκτικά, ότι η έγκυρη χρονική σφραγισμένη πρόταση σε αυτήν την πλειάδα ισχύει για μια πρόταση και η χρονική σφραγισμένη πρόταση συναλλαγής ισχύει για μια διαφορετική πρόταση. Όπως προτείνει το παράδειγμα, μια τέτοια χρονική πλειάδα μπορεί να θεωρηθεί, λίγο αόριστα, ως πλειάδα που παρουσιάζει τον χρόνο συναλλαγής tt του έγκυρου χρόνου vt κάποιας πρότασης.

Υποθέστε ότι η βάση δεδομένων περιλαμβάνει την ακόλουθη πλειάδα:

Figure	#_OF_SIDES
Triangle	3

Η ερμηνεία είναι "τα τρίγωνα έχουν τρεις πλευρές". Αυτήν η πρόταση, που την ονομάζουμε q δεν έχει καθόλου έγκυρο χρόνο. Ωστόσο, μπορούμε να σκεφτούμε την πρόταση q να έχει έναν έγκυρο χρόνο. Για να είμαστε ακριβής, είναι η περίπτωση που πιστεύουμε τώρα ότι η q είναι, ήταν πάντα, και θα είναι για πάντα αληθινή. Κατά συνέπεια, ο έγκυρος χρόνος για το q είναι το πάντα (δηλ., ο "συνολικός χρόνος" ή ισοδύναμα το διάστημα από την αρχή του χρόνου έως το τέλος του χρόνου). Και επειδή ο έγκυρος χρόνος είναι πράγματι ο "συνολικός χρόνος" δεν υπάρχει κανένα σημείο να δηλώσουμε αυτό τον έγκυρο χρόνο στη βάση δεδομένων. Κατά συνέπεια, μπορούμε εύλογα να πούμε ότι, γενικά, μια πρόταση που αντιπροσωπεύεται στη βάση δεδομένων χωρίς ρητή έγκυρη χρονικά σφραγισμένη πρόταση έχει σιωπηρά έγκυρη χρονικά σφραγισμένη πρόταση "το πάντα".

Παρατηρήστε ότι κάθε πρόταση που καταγράφεται στη βάση δεδομένων έχει έναν χρόνο συναλλαγής, ανεξάρτητα από το εάν περιλαμβάνει ρητή έγκυρη χρονικά σφραγισμένη πρόταση. Παραδείγματος χάριν, εάν η πλειάδα που αντιστοιχεί στην πρόταση q "*τα* τρίγωνα έχουν τρεις πλευρές" καταγράφεται στη βάση δεδομένων από το χρονικό t1 *στο* χρονικό t2, τότε ο χρόνος συναλλαγής για την πρόταση "q είναι πάντα αληθινός" και είναι το διάστημα από το t1 *στο* t2.

εξετάστε πάλι την ακόλουθη χρονική πλειάδα (την ονομάζουμε bt):

S#	DURING	X _DURING
S2	[d02:d04]	[tl: t2]

θυμηθείτε ότι η πρόταση είναι "ο χρόνος συναλλαγής για την πρόταση p είναι το διάστημα από το t1 έως το t2", όπου η πρόταση p είναι στην συνέχεια η πρόταση "ο προμηθευτής S2 ήταν στο πλαίσιο της σύμβασης από την $2^{η}$ ημέρα έως την $4^{η}$ ημέρα"

Υποθέστε τώρα ότι η πλειάδα από μόνη της εμφανίζεται στη βάση δεδομένων. Τότε η αντίστοιχη πρόταση έχει έναν έγκυρο χρόνο και έναν χρόνο συναλλαγής. Ο έγκυρος χρόνος είναι, σιωπηρά, "το πάντα".

Τι γίνεται με τον χρόνο συναλλαγής; Η πλειάδα BT είναι αυτή που παρεμβάλλεται στη βάση δεδομένων σε κάποιο χρόνο t>=t2 και εάν η πλειάδα BT, όπως δηλώνεται, αυτήν την περίοδο εμφανίζεται ακόμη στη βάση δεδομένων, και έχει κάνει χρόνο t. Τότε ο χρόνος συναλλαγής για την πρόταση που αντιστοιχεί *στη* BT είναι σαφώς το διάστημα από το χρόνο t μέχρι το τέλος του χρόνου.

- ✓ *Ο έγκυρος χρόνος για μια πρόταση p είναι το σύνολο χρόνων t έτσι ώστε,* σύμφωνα με αυτό που δηλώνει η βάση δεδομένων αυτήν την περίοδο (που πρόκειται να πει, σύμφωνα με τις τρέχουσες πεποιθήσεις μας), *το* p είναι, ήταν, ή θα είναι αληθή στο χρόνο t.

- ✓ Ο χρόνος συναλλαγής για μια πρόταση q είναι το σύνολο χρόνων t έτσι ώστε, σύμφωνα με αυτό που δηλώνει η βάση δεδομένων ή δήλωσε στο χρόνο t, *η* q είναι, ήταν, ή θα είναι αληθινής.

Εν συντομία:

1. **Οι έγκυροι** χρόνοι είναι οι χρόνοι (παρελθόν ή/και παρόν ή/και μέλλον) όταν, σύμφωνα με αυτό που πιστεύουμε, κάτι είναι, ήταν, ή θα είναι αληθινό.

2. Οι χρόνοι **συναλλαγής** είναι οι χρόνοι (παρελθόν ή/και παρόν) όταν είπε η βάση δεδομένων πιστέψαμε ότι κάτι είναι, ήταν, ή θα είναι αληθινό.

Επιπλέον, υποθέτουμε ότι (α) ο έγκυρος χρόνος για οποιαδήποτε δεδομένη πρόταση *p* αντιπροσωπεύεται ως σύνολο διαστημάτων, και ότι το σύνολο κρατιέται σε συντομευμένη μορφή (2) ο χρόνος συναλλαγής για οποιαδήποτε δεδομένη πρόταση *q* αντιπροσωπεύεται από ένα σύνολο διαστημάτων, και αυτό το σύνολο κρατιέται επίσης σε συντομευμένη μορφή.

8.2 Η Βάση Δεδομένων Και Ο Λογάριθμος

Εξαιτίας των ορισμών και των εξηγήσεων του προηγούμενου τμήματος, τώρα παρατηρούμε ότι υπάρχει μια σημαντική λειτουργική διαφορά μεταξύ του έγκυρου χρόνου και του χρόνου συναλλαγής, διαφορά που μπορεί να χαρακτηριστεί ανεπίσημα ως εξής:

Οι έγκυροι χρόνοι κρατιούνται στη βάση δεδομένων, ενώ οι χρόνοι συναλλαγής κρατιούνται στο λογάριθμο.

Σημειώστε ότι μια βάση δεδομένων είναι πραγματικά μια μεταβλητή, η λειτουργία "της ενημέρωσης της βάσης δεδομένων" αναγκάζει την τρέχουσα τιμή αυτής της μεταβλητής να αντικατασταθεί από μια άλλη τιμή. Οι εν λόγω τιμές είναι τιμές βάσεων δεδομένων, και η μεταβλητή είναι μια μεταβλητή βάσεων δεδομένων. Με άλλα λόγια, η κρίσιμη διάκριση μεταξύ των τιμών και των μεταβλητών που συζητήσαμε στο κεφάλαιο 1, ισχύει για τις βάσεις δεδομένων επίσης.

Για να είμαστε συγκεκριμένοι, μια αναπροσαρμογή των βάσεων δεδομένων μπορεί να θεωρηθεί, όχι τόσο πολύ ως αντικατάσταση της τρέχουσας τιμής βάσεων δεδομένων από μια άλλη, αλλά ως προσδιορισμό μιας "νέας" τιμής βάσεων δεδομένων από μια "παλαιά", κρατώντας συγχρόνως την παλαιά τιμή γύρω από το σύστημα. Η "παλαιά" τιμή βάσεων δεδομένων είναι, φυσικά, αυτή που ήταν τρέχουσα αμέσως πριν από την αναπροσαρμογή.

Η γενική βάση δεδομένων μπορεί να θεωρηθεί ως ακολουθία τιμών βάσεων δεδομένων, όπου κάθε τέτοια τιμή είναι χρόνος που σφραγίζεται με το χρόνο της αναπροσαρμογής που το παρήγαγε, και η πλήρη ακολουθία διατάσσεται χρονολογικά. Η πιο πρόσφατη τιμή βάσεων δεδομένων στην ακολουθία είναι φυσικά, η τρέχουσα. Είναι η βάση δεδομένων όπως εμφανίζεται "αμέσως" όπως ήταν. Και το μόνο είδος λειτουργίας αναπροσαρμογών που μπορούμε να απευθύνουμε στη γενική βάση δεδομένων, μιλώντας εννοιολογικά, είναι αυτό που παίρνει την τρέχουσα τιμή βάσεων δεδομένων και προσδιορίζει από αυτή μια νέα τιμή, η οποία γίνεται τρέχουσα στη συνέχεια.

Σημειώστε ότι η γενική ακολουθία τιμών βάσεων δεδομένων μπορεί ωφέλιμα να θεωρηθεί ως λογάριθμος και είναι αποτελεσματικά μια αφαίρεση του [log] αποκατάστασης όπως εφαρμόζεται στα πραγματικά συστήματα βάσεων δεδομένων, και παρέχει ένα πλήρες ιστορικό αρχείο κάθε αναπροσαρμογής που έχει γίνει στη βάση δεδομένων. Και η πιο πρόσφατη είσοδος [log] , η πιο πρόσφατη τιμή βάσης δεδομένων στην χρονολογική ακολουθία, παρέχει ένα αρχείο των τρεχουσών πεποιθήσεών μας.

Επιστρέφουμε στο παράδειγμα της παραγράφου 2.3 στο 2^{o} κεφάλαιο, στο οποίο η πλειάδα που παρουσιάζει τον προμηθευτή S1 να είναι στο πλαίσιο της σύμβασης κατά τη διάρκεια κάποιου διαστήματος παρεμβλήθηκε στη βάση δεδομένων σε χρόνο t1 *και* αντικαταστάθηκε σε χρόνο t2, και αυτή η πλειάδα αντικατάστασης διαγράφηκε έπειτα στο χρόνο t3. Έστω ότι το t2' *και* t3' είναι χρονικά σφραγισμένες προτάσεις των log εισόδων που είναι οι τιμές της βάσης δεδομένων που προηγούνται αμέσως αυτών για τους χρόνους t2 *και* t3, αντίστοιχα.

Οι log είσοδοι για τους χρόνους t1 *και* t2' θα περιλαμβάνουν την αρχική πλειάδα, αυτές για τους χρόνους t2 *και* t3' *θα* περιλαμβάνουν την πλειάδα αντικατάστασης, και αυτές για το χρόνο t3 και εμπρός θα περιλαμβάνουν μία μη ανταποκρινόμενη πλειάδα. (Υποθέτουμε χάριν του παραδείγματος ότι καμία πλειάδα δεν παρουσιάζει τον προμηθευτή s1 να είναι στο πλαίσιο της σύμβασης στη βάση δεδομένων είτε πριν από χρόνο t1 είτε μετά χρόνο t3). Οι χρόνοι συναλλαγής για τις εφαρμόσιμες προτάσεις μπορούν σαφώς να ληφθούν από τις χρονικά σφραγισμένες προτάσεις που συνδέονται με τις σχετικές log καταχωρήσεις)

Τι γίνεται όσον αφορά τον έγκυρο χρόνο; Θα πρέπει να γίνει σαφές ότι αυτοί δεν ανταποκρίνονται στις log εισόδους. Πράγματι όπως είδαμε, εάν το ρ είναι μία πρόταση στην οποία η έννοια του έγκυρου χρόνου ισχύει, εμείς δεν κρατάμε το ρ (υπό αυτήν τη μορφή) στη βάση δεδομένων, αλλά κρατάμε την αντιστοιχία η την επέκταση του ρ στη βάση δεδομένων. Ως εκ τούτου, οι έγκυροι χρόνοι, που είναι αυτοί οι χρόνοι που

θεωρούμε αυτήν την περίοδο έγκυροι, εμφανίζονται στην τρέχουσα τιμή της βάσης δεδομένων, η οποία αντιστοιχεί πάντα, πάλι εξ ορισμού, στην πιο πρόσφατη log είσοδο.

Προχωράμε τώρα να εξετάσουμε διάφορα περαιτέρω σημεία που προκύπτουν σχετικά με την ιδέα ότι "το log είναι η πραγματική βάση δεδομένων."

- Όπως σημειώνεται, μια πρόταση p μπορεί να έχει έναν χρόνο συναλλαγής που αποτελείται από αρκετά ιδιαίτερα διαστήματα, από t1 *στο* t2, μετά από t3 *στο* t4, μετά από t5 *στο* t6, *κ.τ.λ.* Δηλαδή οι χρόνοι t1, t3, t5,κ.τ.λ. αντιστοιχούν στις αναπροσαρμογές που ανάγκασαν το p να εμφανιστεί στη βάση δεδομένων (είτε σιωπηρά είτε ρητά), ενώ οι χρόνοι t2 , t4, t6 κ.τ.λ., αντιστοιχούν στις αναπροσαρμογές που ανάγκασαν το p να εξαφανιστεί πάλι.
- Επιπλέον, μια πρόταση q μπορεί να έχει έναν έγκυρο χρόνο που αποτελείται από διάφορα ιδιαίτερα διαστήματα. Εδώ, τα σχετικά διαστήματα είναι απλά τιμές που εμφανίζονται αυτήν την περίοδο στη βάση δεδομένων. Παραδείγματος χάριν, υποθέστε ότι η σχέση S_DURING περιέχει ακριβώς τις δύο ακόλουθες πλειάδες για τον προμηθευτή S2:

S#	DURING
S2	[d02:d04]

S#	DURING
S2	[d06:d09]

Τότε ο έγκυρος χρόνος για την πρόταση "ο προμηθευτής S2 ήταν στο πλαίσιο της σύμβασης" είναι το σύνολο διαστημάτων [d02:d04], [d06:d09] .

- Σημειώστε το γεγονός ότι οι ορισμοί είναι ασύμμετροι, εάν το p είναι μια πρόταση για την οποία η έγκυρη χρονική έννοια ισχύει και το q είναι μια πρόταση για την οποία η χρονική έννοια συναλλαγής ισχύει, τότε το p και το q δεν είναι η ίδια πρόταση. Βεβαίως φαίνεται να είναι δύσκολο να βρεθεί μια μη-τετριμμένη πρόταση που έχει και έναν μη-τετριμμένο σχετικό έγκυρο χρόνο και έναν μη-τετριμμένο σχετικό χρόνο συναλλαγής.
- Μπορεί να εκπλαγείτε όταν μάθετε ότι οι έγκυροι χρόνοι είναι λιγότερο ενδιαφέροντες από τους χρόνους συναλλαγής. Γιατί έτσι; Έχουμε δει ότι οι έγκυροι χρόνοι αντιπροσωπεύονται με τη βοήθεια των ιδιοτήτων των σχέσεων βάσεων δεδομένων. Επιπλέον, μπορούμε να έχουμε σχέσεις με έγκυρες χρονικές ιδιότητες, και οι έγκυροι χρόνοι μπορούν έτσι να ενημερωθούν καθώς επίσης και να ρωτηθούν, δηλαδή τους έγκυρους χρόνους μπορούμε να τους αλλάξουμε *για* να απεικονίσουμε τις μεταβαλλόμενες πεποιθήσεις. Σε γενικές γραμμές, οι έγκυρες χρονικά ιδιότητες δεν είναι σημαντικά διαφορετικές από τις ιδιότητες βάσεων δεδομένων οποιουδήποτε άλλου είδους, τουλάχιστον όσον αφορά στη δυνατότητά τους να συμμετέχουν σε διαδικασίες ερώτησης ή αναπροσαρμογών.
- Τι γίνεται λοιπόν με τους χρόνους συναλλαγής; Εάν οι χρόνοι συναλλαγής είναι πραγματικά βασισμένοι σε log χρονικά σφραγισμένες προτάσεις, τότε πρέπει να γίνει σαφές ότι διατηρούνται από το σύστημα, όχι από τους χρήστες. Πράγματι, επισημάναμε νωρίτερα ότι οι χρόνοι συναλλαγής είναι μη-αναθεωρήσιμοι, ότι δεν μπορούν να ενημερωθούν από τους χρήστες. Εντούτοις, οι χρήστες πρέπει να είναι σε θέση να τους ρωτήσει. Παραδείγματος χάριν, μπορεί να θελήσουμε να

ρωτήσουμε, "πότε η βάση δεδομένων είπε ότι ο προμηθευτής S2 ήταν στο πλαίσιο της σύμβασης την $3^{η}$ ημέρα";

- Τουλάχιστον δύο προβλήματα προκύπτουν αμέσως από αυτήν την απαίτηση. Καταρχήν, το σύστημα δεν διατηρεί πραγματικά το log στη μορφή που έχουμε περιγράψει και κατά συνέπεια, είναι ιδιαίτερα απίθανο το σύστημα να μας επιτρέψει να διατυπώσουμε τις ερωτήσεις ενάντια στην τιμή βάσεων δεδομένων σε κάποιο αυθαίρετο προηγούμενο χρόνο t. Δεύτερον, ακόμα κι αν θα μπορούσαμε να διατυπώσουμε μια τέτοια ερώτηση, η αντίστοιχη χρονικά σφραγισμένη πρόταση t δεν αντιπροσωπεύεται από μόνο της ως τμήμα αυτής της τιμής βάσεων δεδομένων αλλά μάλλον, ένα είδος ετικέτας που συνδέεται με αυτήν την τιμή με συνέπεια να μην μπορούμε ακόμη να διατυπώσουμε ένα ερώτημα, όχι ένα ερώτημα σχέσης, που αναφέρεται άμεσα σ'αυτήν την χρονικά σφραγισμένη πρόταση.
- Προκύπτει από τα προηγούμενα ότι το φανταστικό log για το οποίο έχουμε μιλήσει είναι η καλύτερη σκέψη, όχι ως αφαίρεση του συνηθισμένου log αποκατάστασης, αλλά μάλλον ως διαδρομή του ελέγχου. Τελικά, ο σκοπός του είναι ουσιαστικά να χρησιμεύσει ως μια ιστορία των γεγονότων αναπροσαρμογών και να επιτρέψει στις ερωτήσεις να υποβληθούν αργότερα για αυτό που έγινε στη βάση δεδομένων όταν, τέτοιες ερωτήσεις είναι θέματα φύσης ελέγχου και σχετίζονται ελάχιστα με την αποκατάσταση.

- Έχουμε μιλήσει σαν οι αναπροσαρμογές να εφαρμόζονται στη βάση δεδομένων τη στιγμή στην οποία εκτελούνται. Στην πράξη φυσικά, οι αναπροσαρμογές εφαρμόζονται στη βάση δεδομένων μόνο εάν και όταν φθάνει η σχετική συναλλαγή σε μια επιτυχή λήξη. Ακολουθεί ότι η στιγμιαία χρονική συναλλαγή,(δηλ., το χρονικό σημείο) σε μια ιδιαίτερη αναπροσαρμογή δεν είναι ακριβώς ο χρόνος της λειτουργίας αναπροσαρμογών αυτό καθ' εαυτό αλλά είναι, ο χρόνος της αντιστοιχίας COMMIT λειτουργίας στα τέλη της συναλλαγής (που υπονοεί μεταξύ άλλων ότι διάφορες ευδιάκριτες αναπροσαρμογές μπορούν να συνδεθούν με την ίδια χρονική στιγμή συναλλαγής). Αυτό το γεγονός μπορεί να προκαλέσει ορισμένα προβλήματα στην εφαρμογή.

8.3 Ορολογία

Καταρχήν, σχετικά με τον έγκυρο χρόνο, πρέπει να πούμε ότι δεν χρειάζεται να αναφερθούμε ρητά στην έννοια, ακριβώς επειδή θεωρούμε τους έγκυρους χρόνους σαν κανονικά στοιχεία.

Θα επιθυμούσαμε να βρούμε έναν όρο που συλλαμβάνει, καλύτερα, αυτά τα γεγονότα από ότι ο "έγκυρος χρόνος", έτσι ώστε να έχουμε έναν καλύτερο όρο για να χρησιμοποιήσουμε όποτε πρέπει να αναφερθούμε σ' αυτή την έννοια. Όσον αφορά το χρόνο συναλλαγής, εδώ σίγουρα πρέπει να αναφερθούμε στην έννοια βάση ονόματος από χρόνο σε χρόνο και έτσι θα επιθυμούσαμε σίγουρα να βρούμε έναν καλύτερο όρο.

Κατά συνέπεια:

- Ο τιθέμενος χρόνος για μια πρόταση ρ είναι το σύνολο των χρόνων t έτσι ώστε, σύμφωνα με αυτό που η βάση δεδομένων δηλώνει αυτήν την περίοδο, το p είναι, ήταν, ή θα είναι αληθινό στο χρόνο t.
- Ο καταγεγραμμένος χρόνος για μια πρόταση q είναι το σύνολο των χρόνων t έτσι ώστε, σύμφωνα με αυτό που η βάση δεδομένων δηλώνει *στο* χρόνο t, *η* πρόταση q είναι, ήταν, ή θα είναι αληθινή.

Επίσης, να επισημάνουμε ότι:

1. Στην περίπτωση του καταγεγραμμένου χρόνου, το μεταβλητό t κυμαίνεται κατά τη διάρκεια του συνόλου όλων των χρόνων, από το χρόνο που η βάση δεδομένων δημιουργήθηκε μέχρι τώρα.
2. Στην περίπτωση του τιθέμενου χρόνου, το μεταβλητό t κυμαίνεται κατά τη διάρκεια του συνόλου όλων των χρόνων που εμφανίζονται ρητά ή σιωπηρά ως χρονικά σφραγισμένη πρόταση στην τρέχουσα τιμή της βάσης δεδομένων.

8.4 Σχέσεις LOGGED-TIME

Επισημάναμε στην παράγραφο 8.2 ότι ακόμα κι αν οι χρήστες δεν μπορούν να ενημερώσουν τους καταγεγραμμένους χρόνους, πρέπει να είναι σε θέση να τους ρωτήσουν. Επίσης επισημαίνουμε ότι αυτή η απαίτηση υπονοεί ότι:

- Οι τιθέμενοι χρόνοι πρέπει να είναι διαθέσιμοι σε τυποποιημένη σχετική φόρμα

- Επιπλέον, πρέπει να είναι διαθέσιμοι ως τμήμα της τρέχουσας τιμής βάσεων δεδομένων,

Ουσιαστικά, αυτό που προτείνουμε είναι ότι εάν *η* R είναι μία σχέση (και είναι πραγματική, μη εικονική), τότε ο καθορισμός *του ρ* θα πρέπει να περιλάβει ένα αίτημα για την αυτόματη παροχή και τη συντήρηση μιας βοηθητικής σχέσης, που δίνει την ιστορία της τιθέμενης χρονικής στιγμής γι'αυτήν τη σχέση R. Παραδείγματος χάριν:

```
VAR S_ DURING RELATION
 {S# S#, DURING INTERVAL_DATE)
 LOGGED_TIMES_IN (S_DURING_LOG);
```

Η επίδραση αυτής της προδιαγραφής είναι να αναγκάσει το σύστημα να παρέχει σε μία σχέση αποκαλούμενη S_DURING_LOG, τις ιδιότητες S#, DURING *και* X_DURING, και πλειάδες που αντιπροσωπεύουν μαζί τους τιθέμενους χρόνους για όλες τις πλειάδες που έχουν εμφανιστεί, ρητά ή σιωπηρά, στη σχέση S_DURING. Παραδείγματος χάριν, υποθέστε ότι σήμερα είναι $75^{η}$ ημέρα. Τότε το παράδειγμα 8.1 παρουσιάζει μια πιθανή

τρέχουσα τιμή για το S_DURING, μαζί με μια πιθανή αντίστοιχη τιμή για την S_DURING_LOG.

FIGURE 8.1
Σχέσεις S_DURING
Και S_DURING_LOG
Ενδεικτικές τιμές

S_DURING

S#	DURING
S2	[d02:d04]
S6	[d03:d05]

S_DURING_LOG

S#	DURING	X_DURING
S2	[d02:d04]	[d04:d07]
S2	[d02:d04]	[dl0:d20]
S2	[d02:d04]	[d50:d75]
S6	[d02:d05]	[d15:d25]
S6	[d03:d05]	[d26:d15]
S1	[d0l:d0l]	[d20:d30]
S1	[d05:d06]	[d40:d50]

ΕΞΗΓΗΣΗ:

Καταρχήν, η σχέση S_DURING αυτήν την περίοδο λέει ότι ο προμηθευτής S2 ήταν στο πλαίσιο της σύμβασης από την $2^{η}$ μέχρι την $4^{η}$ ημέρα. Η πλειάδα 1 της σχέσης S_DURING_LOG μας λέει ότι η σχέση S_DURING είπε το ίδιο πράγμα νωρίτερα, από *την* $4^{η}$ μέχρι την $7^{η}$ ημέρα. Η πλειάδα 2 της σχέσης S_DURING_LOG μας λέει ότι η σχέση S_DURING είπε επίσης το ίδιο πράγμα νωρίτερα, από την $10^{η}$ μέχρι την $20^{η}$ ημέρα.

Η πλειάδα 3 της σχέσης S_DURING_LOG μας λέει ότι η σχέση S_DURING έχει πει το ίδιο πράγμα πάλι, ότι δηλαδή ο προμηθευτής S2 ήταν στο πλαίσιο της σύμβασης από την $2^{η}$ ημέρα ως *την* $4^{η}$, *κι* από την $50^{η}$ ημέρα μέχρι σήμερα (δηλ., $75^{η}$ ημέρα). Τώρα, στο κεφάλαιο 4, παράγραφος 4.5, υποστηρίξαμε ότι η αντιπροσώπευση "της σημερινής ημέρας" στη βάση δεδομένων από μια πραγματική τιμή όπως d75 ήταν μια κακή ιδέα. Ειδικότερα, επισημάναμε ότι όλα αυτά τα d75's θα έπρεπε να αντικατασταθούν από d76's στο κτύπημα των μεσάνυχτων την $75^{η}$ ημέρας. Εντούτοις, αυτά τα επιχειρήματα δεν ισχύουν για την παρούσα κατάσταση, επειδή:

- Η σχέση S_DURING_LOG δεν χρειάζεται να υπάρξει πραγματικά όλες τις στιγμές, είναι ικανοποιητικό για το σύστημα να υλοποιηθεί όταν παραπέμπεται σε κάποιο ερώτημα. Με άλλα λόγια, είναι ένας χρήσιμος τρόπος να σκεφτούμε ότι η S_DURING_LOG είναι εικονική σχέση, αλλά όχι όπως άλλες εικονικές σχέσεις, που καθορίζονται από την άποψη άλλων σχέσεων στη βάση δεδομένων, αλλά που καθορίζονται από την άποψη του συστήματος log. Επιπλέον, ο ορισμός αυτής της εικονικής σχέσης από την άποψη του log παρέχεται από το σύστημα, και όχι από κάποιον χρήστη.

- Η σχέση S_DURING_LOG είναι αναθεωρήσιμη από το σύστημα αλλά όχι από τους συνηθισμένους χρήστες. Κατά συνέπεια, η διαδικασία της αντικατάστασης όλων των d75's από d76's *στο* κτύπημα των μεσάνυχτων την $75^{η}$ ημέρα πραγματοποιείται από το σύστημα, και όχι από κάποιο χρήστη.

- Εν πάση περιπτώσει, η πρόταση "από την 50η ημέρα ως την 75η ημέρα, η βάση δεδομένων είπε ότι ο προμηθευτής S2 ήταν στο πλαίσιο της σύμβασης από την 2η ως την 4η ημέρα" είναι αληθινή! Βεβαίως δεν πρέπει να χρησιμοποιήσουμε το τεχνητό τέχνασμα σχετικά με τους καταγεγραμμένους χρόνους που μερικές φορές χρησιμοποιούμε σχετικά με τους τιθέμενους χρόνους, επειδή π.χ. η πρόταση "από την 50η ημέρα μέχρι το τέλος του χρόνου, η βάση δεδομένων είπε ότι ο προμηθευτής S2 ήταν στο πλαίσιο της σύμβασης από την 2η ως την 4η ημέρα" είναι ψευδής.

- Η σχέση S_DURING αυτήν την περίοδο λέει ότι ο προμηθευτής S6 ήταν στο πλαίσιο της σύμβασης από την 3η ως την 5η ημέρα. Σε αντίθεση, η πλειάδα 4 της σχέσης S_DURING_LOG μας λέει ότι η σχέση S_DURING (από την 15η ως την 25η ημέρα) ότι ο προμηθευτής S6 ήταν στο πλαίσιο της σύμβασης από την 2η ως την 5η ημέρα, η πλειάδα 5 της σχέσης S_DURING_LOG μας λέει ότι από την 26η ημέρα, η σχέση S _DURING έχει πει ότι ο προμηθευτής S6 ήταν στο πλαίσιο της σύμβασης από την 3η ως την 5η ημέρα

- Τέλος, η σχέση S_DURING δεν έχει αυτήν την περίοδο τίποτα να πεί για τον προμηθευτή S1, η πλειάδα 6 της σχέσης S_DURING_LOG μας λέει ότι η σχέση S_DURING από την 20η μέχρι την 30η ημέρα ότι ο προμηθευτής S1 ήταν στο πλαίσιο της σύμβασης, την 1η ημέρα μόνο. Επιπλέον, η πλειάδα 1 μας λέει ότι η σχέση S_DURING από την 40η μέχρι την 50η ημέρα ότι ο προμηθευτής S1 ήταν στο πλαίσιο της σύμβασης από την 5η ημέρα μέχρι την 6η ημέρα. (συμπεραίνουμε λοιπόν ότι όλες οι πληροφορίες σχετικά με τον προμηθευτή S1 διαγράφηκαν από τη σχέση S_DURING την 30η και πάλι την 50η ημέρα, πιθανώς επειδή ανακαλύφθηκαν να είναι ανακριβείς.)

Έστω ότι η R να είναι μία κανονική σχέση, και η R' είναι η σχετική τιθέμενη χρονική σχέση. Επίσης, δίνουμε στο R τις ιδιότητες διαστήματος A1, A2..., An ενδεχομένως με άλλες ιδιότητες. Τότε:

- Ο τίτλος του R'είναι ο ίδιος με αυτόν του R, εκτός από το ότι περιέχει μια πρόσθετη ιδιότητα διαστήματος αποκαλούμενη X_DURING.

- Για κάθε πλειάδα t που έχει εμφανιστεί στην πλήρως αποσυμπιεσμένη μορφή της σχέσης R, η σχέση R' περιέχει m ευδιάκριτες πλειάδες για μερικές m >0. Κάθε τέτοια πλειάδα αποτελείται από t επεκτάσεις με μία τιμή X_DURING, έτσι ώστε ο χρόνος συναλλαγής για το t (ή μάλλον, για την πρόταση που συνδέεται με t) είναι ακριβώς το σύνολο που περιέχει ακριβώς αυτές τις τιμές m X_DURING. Η σχέση R' "κρατιέται συμπιεσμένη πάνω στο (A1, A2...,An , X_DURING). Επ'

ευκαιρία παρατηρήστε ότι η συμπίεση *του* R' πρέπει να γίνει στις ιδιότητες X_DURING στο τέλος, όπως υποδεικνύεται.

- Τέλος, εάν *η* R ικανοποιεί τον περιορισμό WHEN UNPACKED ON (A1, A2, ., ., An) THEN KEY {K}, τότε η R' *ικανοποιεί* τον περιορισμό WHEN UNPACKED ON (Al, A2,..., An, X_DURING) THEN KEY {K, X_DURING}.

Σαν βάση για ένα δεύτερο παράδειγμα, γυρίζουμε στη συνηθισμένη τρέχουσα σχέση S_SINCE *μας*:

```
VAR S_ SINCE RELATION
        {S#             S#,
         S#_SINCE       DATE,
         STATUS         INTEGER,
         STATUS_SINCE DATE}
LOGGED_TIMES_IN (S_SINCE_LOG);
```

Το σχήμα 8.2 παρουσιάζει μερικές τιμές δειγμάτων. Σας αφήνουμε μια λεπτομερή εξέταση αυτού του παραδείγματος, κι εφιστάμε την προσοχή μας στην πλειάδα S_SINCE_LOG για τον προμηθευτή S2 στην οποία οι τιμές S#_SINCE *και* S_STATUS_SINCE είναι και οι δύο d07, ενώ η τιμή X_DURING είναι [d06:d75]. Τι υπονοεί αυτός ο συνδυασμός τιμών; Απάντηση: *την* $6^{η}$ ημέρα μία πλειάδα παρεμβλήθηκε στη σχέση S_SINCE για να πει ότι ο προμηθευτής S2 θα τοποθετούταν στο πλαίσιο της σύμβασης την $7^{η}$ ημέρα, μια μέρα στο μέλλον, κατά την διάρκεια της INSERT.

FIGURE 8.2
σχέσεις
S_SINCE KAI
S_SINCE_LOG-
Ενδεικτικές τιμές

S_SINCE

S#	S#_SINCE	STATUS	STATUS_SINCE
S1	d04	20	d06
S2	d01	10	d01
S3	d03	30	d03
S4	d04	20	d08
S5	d02	30	d02

S_SINCE_LOG

S#	S#_SINCE	STATUS	STATUS SINCE	X DURING
S1	d04	15	d04	[d04:d05]
SI	d04	20	d06	[d06:d75]
S2	d02	5	d02	[d01:d02]
S2	d02	10	d03	[d03:d04]
S2	D01	10	d07	[d06:d75]
S3	d03	30	d03	[d03:d75]
S4	d04	10	d04	[d04:d04]
S4	d04	25	d05	[d05:d07]
S4	d04	20	d08	[d08:d75]
S5	d02	30	d02	[d02:d75]
S6	d03	5	d03	[d03:d04]
S6	d03	7	d05	[d05:d05]

ΚΕΦΑΛΑΙΟ 9: ΤΥΠΟΙ ΣΗΜΕΙΟΥ ΚΑΙ ΔΙΑΣΤΗΜΑΤΟΣ

9.1 Εισαγωγή

Στο κεφάλαιο 3 είπαμε ότι οποιοσδήποτε δεδομένος τύπος διαστήματος περιλαμβάνει ένα θεμελιώδη τύπο σημείου, και ένας τύπος σημείου είναι ένας τύπος που έχει μια λειτουργία διαδόχων.
Ένας δεδομένος τύπος T είναι χρήσιμος ως τύπος σημείου εάν όλα τα ακόλουθα ορίζονται για το T:

Μια συνολική διαταγή, σύμφωνα με την οποία ο τελεστής παρεμβολής ">" (μεγαλύτερος από) καθορίζεται για κάθε ζευγάρι των τιμών v1 *και* v2 του τύπου T, εάν *τα* v1 *και* v2 είναι ευδιάκριτα, τότε μια από τις εκφράσεις "v1 > v2" *και* "v2 > v1" επιστρέφει αληθή(true) και οι άλλες επιστρέφουν ψευδή(false).

Μηδενικοί "πρώτοι" και "τελευταίοι" τελεστές, που επιστρέφουν τη μικρότερη και τη μεγαλύτερη τιμή του T, αντίστοιχα, σύμφωνα με την προαναφερθείσα διαταγή.
Μοναδιαίοι "επόμενοι " και "προγενέστεροι" τελεστές, που επιστρέφουν το διάδοχο και τον προκάτοχο, αντίστοιχα, οποιασδήποτε δεδομένης τιμής του τύπου T σύμφωνα με την προαναφερθείσα διαταγή. Σημείωση: Ο "επόμενος" τελεστής είναι η λειτουργία διαδόχων.

9.2 Κληρονομιά τύπων

Αρχίζουμε με ένα απλό παράδειγμα. Θυμηθείτε τον τύπο δεδομένων στις "γρηγοριανές ημερομηνίες" άλλη μια φορά. Υποθέστε για την προσδιορισιμότητα ότι αυτός ο τύπος δεν ενσωματώνεται. Θα αναφερθούμε σε αυτόν τον τύπο όχι ως DATE, αλλά ως DDATE, προκειμένου να υπογραμμιστεί το γεγονός ότι δείχνει τις ημερομηνίες στο γρηγοριανό ημερολόγιο που είναι ακριβείς στην ημέρα. Φυσικά, DDATE είναι ένας τύπος που οι τιμές του διατάσσονται χρονολογικά, και η λειτουργία διαδόχων είναι βασικά "η επόμενη ημέρα". Εδώ είναι ένας κατάλληλος ορισμός:

```
TYPE DDATE
        POSSREP DPRI {DI   INTEGER
                        CONSTRAINT DI ≥1 AND DI ≤N};
```

ΕΞΗΓΗΣΗ:

. **Ο τύπος** DDATE έχει μια πιθανή αντιπροσώπευση *που ονομάζεται* DPRI που λέει ότι οποιαδήποτε δεδομένη τιμή DDATE, μπορεί να αντιπροσωπευθεί από έναν θετικό ακέραιο αριθμό αποκαλούμενο DI.
. **Ο περιορισμός** της προδιαγραφής POSSREP καθορίζει έναν τύπο περιορισμού *για τον* τύπο DDATE, που περιορίζει τις τιμές του DI για να παραμείνουν στο εύρος από το 1 έως κάποιο μεγάλο ακέραιο αριθμό N.

Είδαμε στο κεφάλαιο 1 ότι οποιοσδήποτε δεδομένος τύπος απαιτεί ένα συνδεδεμένο «ίσο» τελεστή ("=") και, για κάθε πιθανή αντιπροσώπευση, ένας σχετικός τελεστής επιλογής και ένα σύνολο THE_τελεστών. Εδώ είναι οι ορισμοί αυτών των τελεστών για τον τύπο DDATE.

```
OPERATOR DPRI (DI INTEGER) RETURNS DDATE;
      /* κώδικας που επιστρέφει την αντιπροσωπευτική τιμή DDATE */
      /* από τον δεδομένο ακέραιο DI  * /
END OPERATOR;

OPERATOR THE_DI (DD DDATE) RETURNS INTEGER;
       /* κώδικας που επιστέφει τον ακέραιο που αντιπροσωπεύει*/
       /* την δεδομένη DDATE τιμή DD          */
END OPERATOR;

OPERATOR "=" (DDl DDATE, DD2 DDATE) RETURNS BOOLEAN;
RETURN (THE_DI (DDl) = THE_DI (D02));
END OPERATOR;
```

Ο τελεστής επιλογής DPRI επιτρέπει στο χρήστη να διευκρινίσει μια τιμή DDATE *με* την παροχή του κατάλληλου ακέραιου αριθμού. Παραδείγματος χάριν, η επίκληση DPRI

```
DPRI (5263)
```

θα επιστρέψει τιμή DDATE που συμβαίνει να είναι το 5, 263rd. Εντούτοις, αυτό που λείπει είναι ένας φιλικός προς το χρήστη τρόπος να διευκρινίζονται τέτοιες DDATE τιμές. Γι' αυτό επεκτείνουμε τον τύπο καθορισμού για να προσθέσουμε ένα δεύτερο πιθανό ορισμό, ως εξής:

```
TYPE DDATE
      POSSREP DPRI {DI   INTEGER
                    CONSTRAINT DI ≥ 1 AND DI ≤ N}
      POSSREP DPRC {DC CHAR
                    CONSTRAINT ...};
```

Ο περιορισμός προδιαγραφής για την πιθανή αντιπροσώπευση DPRC, οι λεπτομέρειες *της οποίας* παραλείπονται εδώ, αποτελεί έναν άλλο τύπο περιορισμού για τον τύπο DDATE. Αποτελείται από μια πιθανώς περίπλοκη έκφραση που περιορίζει τις τιμές του τύπου DDATE για να είναι τέτοιες, που να μπορούν ενδεχομένως να αντιπροσωπευθούν από τιμές της μορφής 'yyyy/mm/dd', όπου *το* yyyy είναι ένας θετικός ακέραιος αριθμός στη σειρά από 0001 μέχρι 9999, mm είναι ένας θετικός ακέραιος αριθμός στη σειρά από 01 μέχρι 12, dd είναι ένας θετικός ακέραιος αριθμός στη σειρά από 01 μέχρι 31, και η γενική σειρά "yyyy/mm/dd" τηρεί τους συνηθισμένους γρηγοριανούς ημερολογιακούς κανόνες! Φυσικά, κάθε τιμή του τύπου DDATE έχει και μια αντιπροσώπευση DPRI *και μια* DPRC αντιπροσώπευση, ένα γεγονός που υπονοεί, μεταξύ άλλου, ότι κάθε τιμή της

πιθανής αντιπροσώπευσης DPRI πρέπει να είναι αντιπροσωπεύσιμη ως τιμή DPRC *και* αντίστροφα.

Χρειαζόμαστε επίσης έναν επιλογέα και ένα THE_τελεστή αντίστοιχα στην πιθανή αντιπροσώπευση του DPRC:

```
OPERATOR DPRC (DC CHAR) RETURNS DDATE;
        /* κώδικας που επιστρέφει την αντιπροσωπευτική τιμή DDATE */
        /* από την δεδομένη yyyy/mm/dd string DC        */
    END OPERATOR;

    OPERATOR THE_DC (DD DDATE) RETURNS CHAR;
        /* κώδικας που επιστρέφει την  eely/mm/dd string that */
        /* αντιπροσωπεύει την δεδομένη DDATE τιμή DD        */
    END OPERATOR;
```

Τώρα ο χρήστης μπορεί να γράψει, DPRC ("2001/01/18") για να διευκρινίσει την γρηγοριανή ημερομηνία Ιανουάριος 18^{η} 2001, *και το* THE_DC (d) για να λάβει την αντιπροσώπευση ενός χαρακτήρα σειράς μιας δεδομένης DDATE τιμής d.

Σημείωση: Στην πράξη, θα θέλαμε σίγουρα να καθορίσουμε πολλούς πρόσθετους τελεστές σχετικά με τον τύπο DDATE: έναν για να επιστρέψει το έτος που αντιστοιχεί σε μια δεδομένη τιμή DDATE, άλλον για να επιστρέψει το μήνα, άλλον για να επιστρέψει το έτος και το μήνα υπό μορφή σειράς "yyyy/mm", άλλον για να επιστρέψει την ημέρα της εβδομάδας, διάφορους αριθμητικούς τελεστές (π.χ. για να επιστρέψει τις ημέρες ημερομηνίας n πριν από ή μετά από μια καθορισμένη ημερομηνία), και πολλοί άλλοι.

Μπορούμε να χρησιμοποιήσουμε τον τύπο DDATE για να επεξηγήσουμε την έννοια της κληρονομιάς τύπων με την παρατήρηση ότι μερικές φορές δεν ενδιαφερόμαστε για τις ημερομηνίες που είναι ακριβής στην ημέρα. Ισοδύναμα, μπορούμε να πούμε ότι, για ορισμένους λόγους, ενδιαφερόμαστε μόνο για αυτές τις τιμές του τύπου DDATE *που αντιστοιχούν* στην πρώτη του μήνα, όπως όταν μετράμε τις δεκάδες, ενδιαφερόμαστε μόνο για εκείνους τους αριθμούς που αντιστοιχούν στο πρώτο κάθε διαδοχικής ακολουθίας δέκα ακέραιων αριθμών (0 ..10 ..20, κ.λπ.).

Εάν ενδιαφερόμαστε μόνο για ένα υποσύνολο των τιμών που αποτελούν κάποιο τύπο, τότε εξ ορισμού εξετάζουμε έναν υποτύπο, που τον ονομάζουμε Τ'. Στην πραγματικότητα, ο τύπος Τ' είναι ένας υποτύπος του Τ εάν και μόνο εάν κάθε τιμή του τύπου Τ' είναι μία τιμή του Τ. Επίσης εάν υπάρχει τουλάχιστον μία τιμή του Τ που δεν είναι τιμή του Τ' τότε το Τ' είναι ένας κατάλληλος υποτύπος του Τ. Εάν και μόνο εάν το Τ' είναι ένας υποτύπος του Τ, τότε το Τ είναι ένας υπερτύπος του Τ'

Έτσι, για να συνεχίσουμε με το τρέχων παράδειγμά μας, καθορίζουμε έναν τύπο MDATE που είναι ένας κατάλληλος υποτύπος του τύπου DDATE.

```
TYPE MDATE
        IS DDATE
                CONSTRAINT fIRST_Of_MONTH (DDATE)
        POSSREP MPRI {MI = THE_DI (DDATE))
        POSSREP MPRC {MC = SUBSTR (THE_DC (DDATE), I, 7)) ;
```

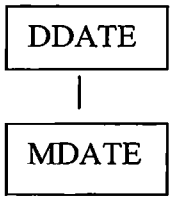
Ο τύπος MDATE ορίζεται να είναι ένας υποτύπος του τύπου DDATE. *Στην* πραγματικότητα, μια τιμή DDATE είναι μια τιμή MDATE εάν και μόνο εάν η τιμή DDATE αντιστοιχεί στην πρώτη του μήνα (έχουμε υποθέσει την ύπαρξη ενός τελεστή αποκαλούμενο FIRST_OF_MONTH που παίρνει μια δεδομένη τιμή DDATE *και* επιστρέφει αληθή εάν αυτή η τιμή αντιστοιχεί στην πρώτη του μήνα, ειδάλλως ψευδή). Όπως τον τύπο DDATE, ο τύπος MDATE έχει δύο πιθανές αντιπροσωπεύσεις, MPRI *και* MPRC, καθεμιά από τις οποίες προέρχεται από την αντίστοιχη πιθανή αντιπροσώπευση για τον τύπο DDATE. Το MPRC μας επιτρέπει να σκεφτούμε τις τιμές MDATE *σαν να* είναι χαρακτήρες σειράς της μορφής "yyyylmm". ΣΗΜΕΙΩΣΗ: Στην πράξη, θα θέλαμε *να καθορίσουμε* ακόμα μια πιθανή αντιπροσώπευση για τον τύπο MDATE, τον "αριθμό του μήνα:" πάλι προερχόμενος με κάποιο τρόπο από κάποια πιθανή αντιπροσώπευση για τον τύπο DDATE.

Εδώ τώρα σας παρουσιάζουμε τους απαραίτητους επιλογείς και τους THE_τελεστές για *τον* τύπο MDATE:

```
OPERATOR MPRI (MI INTEGER) RETURNS MDATE;
                /* κώδικας που επιστρέφει την αντιπροσωπευτική τιμή MDATE  */
                /* από το δεδομένο ακέραιο MI (φυσικά το , MI πρέπει να */
                /* αντιστοιχεί στην πρώτη του μήνα)*/
END OPERATOR;

OPERATOR THE_MI (MD MDATE) RETURNS INTEGER;
          /* κώδικας που επιστρέφει τον ακέραιο που αντιπροσωπεύει*/
          /* την δεδομένη MDATE τιμή MD       */
END OPERATOR;

OPERATOR MPRC (MC CHAR) RETURNS MDATE;
          /* κώδικας που επιστρέφει την αντιπροσωπευτική τιμή MDATE */
          /* από την δεδομένη yyyy/mm string MC  */
END OPERATOR;

OPERATOR THE_MC (MD MDATE) RETURNS CHAR;
        /* κώδικας που επιστρέφει την yyyy/mm string η οποία  */
        /* αντιπροσωπεύει την δεδομένη MDATE τιμή MD   */
END OPERATOR;
```

Έχουμε καθορίσει τώρα τον τύπο MDATE ως κατάλληλο υποτύπο του τύπου DDATE (βλ. την απλή ιεραρχία τύπων που παρουσιάζεται στο σχήμα 9.1). Ποιες είναι οι επιπτώσεις αυτού του γεγονότος; Πιθανώς η σημαντικότερη είναι η ιδιοκτησία, γνωστή ως τιμή υποκαταστασιμότητας.

σχήμα 9.1



Όπου το σύστημα αναμένει μια τιμή του τύπου DDATE, μπορούμε πάντα να αντικαταστήσουμε μια τιμή του τύπου MDATE άντ' αυτού.

Ειδικότερα, εάν Op είναι ένας τελεστής που απευθύνεται στις τιμές DDATE, μπορούμε πάντα να εφαρμόσουμε το Op *σε μια* MDATE τιμή, παραδείγματος χάριν, τους τελεστές "=", THE_DI *και* THE_DC που καθορίζονται παραπάνω για τις τιμές του τύπου DDATE, όλοι ισχύουν και για τις τιμές του τύπου MDATE επίσης. Φυσικά, το αντίστροφο δεν είναι αληθινοί τελεστές που καθορίζονται γιατί οι τιμές του τύπου MDATE δεν ισχύουν για τις τιμές του τύπου DDATE, επειδή οι τιμές DDATE δεν είναι τιμές MDATE.

Ακολουθεί ο ακόλουθος κώδικας:

```
VAR I INTEGER;
VAR MDX MDATE;
VAR DDX DDATE;

I   := THE_DI ( MDX );
IF MDX=DDX THEN ...END IF;
/* κ.τ.λ. κ.τ.λ. κ.τ.λ. */
```

Υποθέτουμε για το υπόλοιπο αυτού του τμήματος ότι η ιεραρχία των τύπων περιέχει τους τύπους DDATE *και* MDATE μόνο. Προσέξτε τις δύο θεμελιώδεις διαδικασίες, της σύγκρισης ανάθεσης και ισότητας. Κατ' αρχάς, η ανάθεση, εξετάζει το ακόλουθο παράδειγμα:

```
VAR DDX DDATE;
        DDX:= MPRC ('2000/09') ;
```

Στην ανάθεση, η μεταβλητή στόχων στην αριστερή πλευρά δηλώνεται να είναι τύπου DDATE, αλλά η έκφραση πηγής στη δεξιά πλευρά δείχνει μια τιμή του τύπου MDATE. Και μετά από την ανάθεση, η μεταβλητή περιέχει μια τιμή DDATE που ικανοποιεί τον τύπο περιορισμού για τον τύπο MDATE (δηλ., είναι στην πραγματικότητα μια τιμή

MDATE). Μπορούμε επομένως να πούμε ότι, ενώ ο δηλωμένος τύπος του μεταβλητού DDX είναι DDATE, ο τρέχων και ο πιο συγκεκριμένος τύπος, που είναι ο τύπος της τρέχουσας τιμής αυτής της μεταβλητής, είναι MDATE.

Τώρα, είναι σημαντικό να καταλάβουμε ότι η διάκριση που συζητάμε εδώ, μεταξύ δηλωμένων και τρεχόντων συγκεκριμένων τύπων, που ισχύουν όχι μόνο για τις μεταβλητές αυτές καθ' εαυτές αλλά στην πραγματικότητα στις εκφράσεις της αυθαίρετης γενικότητας. Με άλλα λόγια, κάθε έκφραση έχει και έναν δηλωμένο τύπο και έναν τρέχοντα πιο συγκεκριμένο τύπο. Για να είναι πιο συγκεκριμένος, ο δηλωμένος τύπος και ο τρέχων τύπος μιας έκφρασης X είναι, ο δηλωμένος και τρέχων τύπος του πιο ακραίου τελεστή που συμμετέχει στο X.

Σημείωση: Ο δηλωμένος τύπος ενός τελεστή είναι ο τύπος που ονομάζεται RETURNS προδιαγραφή στον ορισμό του εν λόγω τελεστή. Ο τρέχων πιο συγκεκριμένος τύπος είναι ο πιο συγκεκριμένος τύπος της τιμής που επιστρέφεται από τη συγκεκριμένη εν λόγω επίκληση.

Για να συνεχίσουμε με το παράδειγμα, υποθέτουμε ότι η ακόλουθη ανάθεση εκτελείται τώρα:

DDX: = DPRC ('1999/10/17');

Ο τρέχων πιο συγκεκριμένος τύπος του μεταβλητού DDX είναι τώρα το DDATE

Ένα ακόμη παράδειγμα:

DDX: = DPRC ('2000/11/01');

Σε αυτήν την περίπτωση, η επίκληση DPRC στη σωστή πλευρά επιστρέφει πραγματικά μια τιμή του τύπου όχι μόνο DDATE αλλά MDATE (επειδή η εν λόγω τιμή ικανοποιεί τον περιορισμό "πρώτη του μήνα" για τον τύπο MDATE). Αναφερόμαστε με αυτή την έννοια ως «ειδίκευση από τον περιορισμό», λόγω του ότι το αποτέλεσμα της επίκλησης DPRC είναι ειδικευμένο στον τύπο MDATE (ακριβώς επειδή ικανοποιεί τον περιορισμό τύπων για τον τύπο MDATE). Και μετά από την ανάθεση, ο τρέχων πιο συγκεκριμένος τύπος του μεταβλητού DDX είναι MDATE.

Έστω ότι το V είναι μια μεταβλητή του δηλωμένου τύπου T και έστω ότι το T' είναι ένας κατάλληλος υποτύπος του T. Τότε:

- Εάν ο τρέχων τύπος του V είναι T και μια τιμή του v του τύπου T' ορίζεται σε V, ο τρέχων τύπος του V γίνεται T". Αυτή η επίδραση είναι μια λογική συνέπεια του φαινομένου που καλούμε ειδίκευση από τον περιορισμό.

- Αντιθέτως, εάν ο τρέχων τύπος του V είναι T' και μια τιμή του V του τύπου T ορίζεται σε V, ο τρέχων τύπος του V γίνεται T. Εμείς αναφερόμαστε με αυτή την έννοια ως γενίκευση από τον περιορισμό, λόγω του ότι ο τρέχων πιο

συγκεκριμένος τύπος του V είναι περαιτέρω γενικευμένος. Καθιερώνεται ως T επειδή η τιμή V ικανοποιεί τον περιορισμό τύπων για το T και όχι για *οποιαδήποτε* κατάλληλο υποτύπο του T.

Όπως μπορείτε να δείτε από τα προηγούμενα παραδείγματα, ένας κανόνας σχετικά με την ανάθεση είναι ότι ο τύπος της τιμής πηγής μπορεί να είναι οποιουδήποτε υποτύπου του δηλωμένου τύπου της μεταβλητής στόχων. Τι γίνεται με τις συγκρίσεις ισότητας; Η σύγκριση v1 = v2 θα απαιτούσε τις τιμές v1 *και* v2 να είναι του ίδιου τύπου. Με την τιμή υποκαταστασιμότητας, μπορούμε να αντικαταστήσουμε μια τιμή οποιουδήποτε υποτύπου του T για το v1 *και* μία τιμή οποιασδήποτε υποκατηγορίας του T για v2. Για *τη* σύγκριση v1 = v2, για να είναι νόμιμη, οι πιό συγκεκριμένοι τύποι της σύγκρισης έχουν έναν κοινό τύπο, *τον* υπερτύπο. Φυσικά, η σύγκριση θα δίνει αληθή εάν και μόνο εάν v1 *και* v2 είναι στην πραγματικότητα η ίδια τιμή.

9.3 Τύποι σημείου

Πρέπει να είναι σαφές ότι ο τύπος DDATE από το προηγούμενο τμήμα καλύπτει τις απαιτήσεις ενός τύπου σημείου. Μπορούμε βεβαίως να καθορίσουμε μια συνολική διαταγή βασισμένη στον τελεστή ">" για τις τιμές του τύπου, και μπορούμε βεβαίως να καθορίσουμε τον απαραίτητο "πρώτο:" first "τελευταίο:" last "επόμενο:" next *και* "προγενέστερο:" prior τελεστή. Στην πραγματικότητα, εφηύραμε μια νέα λέξη κλειδί ORDINAL που μπορεί να περιληφθεί σε έναν καθορισμό τύπων, όπως εδώ :

```
TYPE DDATE ORDINAL
   POSSREP DPRI {DI      INTEGER
                 CONSTRAINT DI ≥ 1 AND DI ≤N}
   POSSREP DPRC {DC    CHAR
                 CONSTRAINT...};
```

Ερμηνεύουμε την προδιαγραφή ORDINAL όπως υπονοείται ότι οι ">" "first" "last:" "next:" *και* "prior" τελεστές πρέπει να καθοριστούν για τον τύπο. Εδώ είναι μερικοί κατάλληλοι ορισμοί για αυτούς τους τελεστές:

```
OPERATOR ">" (DD1 DDATE, DD2 DDATE) RETURNS BOOLEAN;
      RETURN (THE_DI (DD1) > THE_DI (DD2));
END OPERATOR;

OPERATOR FIRST_DDATE ( ) RETURNS DDATE;
      RETURN DPRI (1);
END OPERATOR;

OPERATOR LAST DDATE ( ) RETURNS DDATE;
      RETURN DPRI (N);
END OPERATOR;
```

```
OPERATOR NEXT_DDATE (DD DDATE) RETURNS DDATE;
        IF DD = LAST_DDATE ( )
            THEN signal error;
            ELSE RETURN DPRI (THE_DI (DD) + 1);
        END IF;
END OPERATOR;

OPERATOR PRIOR DDATE (DD DDATE) RETURNS DDATE;
        IF DD = FIRST_DDATE ( )
            THEN signal error;
            ELSE RETURN DPRI (THE_DI (DD) -1 ) ;
        END IF;
END OPERATOR;
```

ΣΗΜΕΙΩΣΗ: Όσον αφορά τον τελεστή "next" εδώ (δηλ., NEXT_DDATE), ο ακόλουθος απλούστερος καθορισμός στην πραγματικότητα θα ήταν ικανοποιητικότερος:

```
OPERATOR NEXT _DDATE  (DD DDATE) RETURNS DDATE;
        RETURN DPRI (THE_DI (DD) + 1);
END OPERATOR;
```

Με άλλα λόγια, δεν είναι λογικά απαραίτητο να περιληφθεί μια ρητή δοκιμή για να δούμε εάν το επιχείρημα που αντιστοιχεί στην παράμετρο dd συμβαίνει να είναι "η τελευταία ημερομηνία". Εάν είναι, η επίκληση επιλογέων DPRI θα αποτύχει σε μια παραβίαση περιορισμού τύπων. Μια ανάλογη παρατήρηση ισχύει για την PRIOR_DDATE.
Πίσω στο κύριο νήμα της συζήτησής μας. Ο τύπος DDATE σαφώς, καλύπτει τις απαιτήσεις για έναν τύπο σημείου. Σημειώστε πολύ προσεκτικά, ωστόσο, ότι οι "first", "last" και "next και prior" τελεστές ονομάζονται FIRST_T: LAST_T: NEXT_T: και PRIOR_T: αντίστοιχα, όπου το T είναι το όνομα του εν λόγω τύπου (DDATE στο παράδειγμα).

Αρχίζουμε με την παρατήρηση ότι, αφού ο τύπος MDATE είναι ένας κατάλληλος υποτύπος του τύπου DDATE και ο DDATE είναι ένας τακτικός τύπος, μπορούμε να θεωρήσουμε και τον MDATE ως τακτικό τύπο. Έτσι πρέπει να καθορίσουμε τους ">" "first", "last" και "next και prior" τελεστές για τον τύπο MDATE. Ο τελεστής ">", πρέπει να είναι κληρονομημένος από τον τύπο DDATE. Το ίδιο και τελεστές NEXT_DDATE, αυτοί οι τελεστές δεν είναι "next" (κ.λπ....) τελεστές που χρειαζόμαστε για τον τύπο MDATE! Παραδείγματος χάριν, λαμβάνοντας υπόψη τη MDATE τιμή 1η Αυγούστου 2001, ο τελεστής NEXT_DDATE θα επιστρέψει την ημερομηνία της επόμενης ημέρας (Αύγουστος 2η, 2001), και όχι την ημερομηνία του επόμενου μήνα (Σεπτέμβριος 1η 2001). Κατά συνέπεια, χρειαζόμαστε μερικούς νέους τελεστές για τον τύπο MDATE:

```
OPERATOR FIRST MDATE ( ) RETURNS MDATE;
RETURN MPRI- (op ( 1 ) ) ;
        /* op ( 1 ) υπολογίζει τον ακέραιο που αντιστοιχεί*/
```

```
        /* στην πρώτη ημέρα του πρώτου μήνα*/
END OPERATOR;

OPERATOR LAST MDATE ( ) RETURNS MDATE;
RETURN MPRI (op ( N ) ) ;
        /* op ( N ) υπολογίζει τον ακέραιο που αντιστοιχεί */
      /* στην πρώτη ημέρα, του τελευταίου μήνα*/
END OPERATOR;

OPERATOR NEXT MDATE (MD MDATE) RETURNS MDATE;
IF MD = LAST_MDATE ( )
THEN signal error;
ELSE RETURN MDATE (THE MI (MD) + incr) ;
/* incr is 28, 29, 30, or 31, as applicable */
END IF;
END OPERATOR;

OPERATOR PRIOR MDATE (MD MDATE) RETURNS MDATE;
IF MD = FIRST_MDATE ( )
I THEN signal error;
        , ELSE RETURN MDATE (THE MI (MD) -decr);
        /* decr is 28, 29, 30, or 31, as applicable */
        END IF;
END OPERATOR;
```

Μπορούμε τώρα να δούμε γιατί οι "first", "last" *και* "next *και* prior" τελεστές για έναν δεδομένο τύπο T ονομάζονται FIRST_T, LAST_T, NEXT_T *και* PRIOR_T αντίστοιχα. Παραδείγματος χάριν, δίνοντας *στην* MDATE την τιμή 1η Αυγούστου *του* 2001, *ο* τελεστής NEXT_DDATE θα επιστρέψει την 2η Αυγούστου του 2001 (όπως ξέρουμε ήδη), ο τελεστής NEXT_MDATE, σε αντίθεση, θα επιστρέψει την 1η Σεπτεμβρίου *του* 2001. Με άλλα λόγια, NEXT_DDATE *και* NEXT_MDATE είναι διαφορετικοί τελεστές.

9.4 Τύποι διαστημάτων

Στο κεφάλαιο 3 καθορίσαμε έναν τύπο διαστήματος να είναι ένας δημιουργικός τύπος της μορφής INTERVAL_T όπου το T είναι ένας τύπος σημείου. Και οι τύποι διαστήματος συνδέουν ορισμένους γενικούς τελεστές με τους επιλογείς διαστήματος, BEGIN *και* END, PRE *και* POST, POINT, FROM, $\in$ *και* $\ni$, COUNT, Allen's τελεστές, UNION, INTERSECT, and MINUS. Σε αυτό το τμήμα, εξετάζουμε την επίδραση των ιδεών που συζητούνται σε αυτό το κεφάλαιο μέχρι τώρα, σχετικά με όλους αυτούς τους τελεστές.

Αρχίζουμε με τη σημείωση ότι εάν το T' είναι ένας υποτύπος του T τότε το T' περιέχει δύο τουλάχιστον τιμές σημείων INTERVAL_T' που δεν είναι υποτύποι του INTERVAL_T επειδή τα διαστήματα του τύπου INTERVAL_T' δεν είναι διαστήματα

του INTERVAL_T'. Πράγματι ακόμη και αν τα διαστήματα ι και ι' των τύπων INTERVAL_T' και INTERVAL_T έχουν τα ίδια σημεία αρχής και τέλους b και e δεν είναι και πάλι το ίδιο διάστημα, συγκεκριμένα έχουν διαφορετικά σύνολα περιλαμβανόμενων σημείων. Παραδείγματος χάριν, πάρτε το T και T' ως, INTEGER και EVEN_INTEGER αντίστοιχα και εξετάστε το ι= [2:6] και το ι'= [2:6] τότε το ι περιέχει μόνο EVEN INTEGER 2, 4 και 6.

Ακόμη και στην ειδική περίπτωση όπου το b = e, όπου σ' αυτή την περίπτωση τα διαστήματα ι και ι' προφανώς έχουν το ίδιο σύνολο περιλαμβανόμενων σημείων, δεν είναι ακόμα το ίδιο διάστημα, ακριβώς επειδή είναι διαφορετικών τύπων. Παραδείγματος χάριν, οι επικλήσεις των τελεστών POST (ι) και POST(ι') θα δώσουν διαφορετικά αποτελέσματα, ακόμη και σε αυτήν την ειδική περίπτωση (εκτός αν το e συμβαίνει να είναι η τελευταία τιμή των τύπων T και T', φυσικά, σ'αυτή την περίπτωση και οι δύο POST επικλήσεις είναι απροσδιόριστες).

Όπως μπορείτε να δείτε, ο τύπος ενός δεδομένου διαστήματος δεν μπορεί να προκύπτει από τον τύπο της αρχής και του τέλους σημείων. Ακολουθεί, όπως ήδη ξέρουμε από το προηγούμενο κεφάλαιο, ότι οι επιλογείς διαστήματος πρέπει επίσης να περιλάβουν τον "τελεστήστή_T'", ακριβώς όπως των "first" and "next" τελεστών. Εδώ είναι δύο παραδείγματα:

INTERVAL INTEGER ([2:6])
INTERVAL_EVEN_ INTEGER ([2:6])

Μια σημαντική παρατήρηση είναι η ακόλουθη. Μόλις είδαμε ότι, εάν το T΄ είναι ένας κατάλληλος υποτύπος του T, τότε το INTERVAL_T' δεν είναι ένας υποτύπος του INTERVAL_T. Στην πραγματικότητα, εάν η IT είναι ο τύπος διαστήματος INTERVAL_T τότε δεν υπάρχει κανένας τύπος διαστήματος της μορφής INTERVAL_T' που το ονομάζουμε IT΄ που είναι ένας κατάλληλος υποτύπος της IT.

Υποθέστε, ότι ένας τέτοιος τύπος IT υπάρχει στην πραγματικότητα. Έστω ότι το r = [b':e'] είναι ένα αυθαίρετο διάστημα του IT τύπου. Τότε το i πρέπει να είναι ένα διάστημα του τύπου IT επίσης. Αλλά ακόμα κι αν το b' και το e' συμβαίνουν να είναι οι τιμές του τύπου t r δεν μπορούν να είναι ένα διάστημα του τύπου IT, επειδή τα περιλαμβανόμενα σημεία του είναι καθορισμένα από τη λειτουργία διαδόχων για το t", το οποίο είναι εξ ορισμού ευδιάκριτο από τη λειτουργία διαδόχων για το t, αντίφαση!

Οι παρατηρήσεις στην προηγούμενη παράγραφο είναι ευρέως αληθινές, αλλά μερικές δευτερεύουσες εξαιρέσεις πρέπει να αναφερθούν:

1. Εάν το T' είναι κενό τότε το IT' είναι ένας υποτύπος όλων των πιθανών τύπων διαστήματος. Εντούτοις, το IT' δεν περιέχει κανένα διάστημα σε αυτήν την περίπτωση.

2. Το πιο σημαντικό είναι να καθοριστεί ένας IT τύπος "που να είναι ένας κατάλληλος υποτύπος του IT' ". Παραδείγματος χάριν, το IT' μπορεί να καθοριστεί για να αποτελείται από ακριβώς εκείνα τα διαστήματα του IT' τύπου που συμβαίνουν να είναι

μοναδιαία διαστήματα. Αλλά εάν το T περιέχει τουλάχιστον δύο σημεία, τότε ένας τέτοιος τύπος IT' "δεν μπορεί να καθοριστεί απλά με την επίκληση του τύπου INTERVAL" και έτσι δεν θα ήταν, όπως δηλώνεται, ένας τύπος της μορφής INTERVAL_T'.

Έπειτα, παρατηρούμε ότι τα διαστήματα είναι τιμές, επομένως όπως όλες οι τιμές, φέρνουν τον τύπο τους μαζί τους. Δηλαδή εάν το ι είναι ένα διάστημα, τότε το ι μπορεί να θεωρηθεί ότι φέρνει μαζί του ένα είδος σημαίας που αναγγέλλει ότι "είμαι ένα διάστημα του τύπου INTERVAL_INTEGER" ή "είμαι ένα διάστημα του τύπου INTERVAL_EVEN_INTEGER" ή "είμαι ένα διάστημα του τύπου INTERVAL_DDATE" (κ.λπ., κ.λπ.). Από αυτές τις παρατηρήσεις και από τις προηγούμενες παραγράφους, συμπεραίνουμε ότι οποιοδήποτε δεδομένο διάστημα φέρνει μαζί του έναν ακριβώς τύπο, και έτσι μπορούμε να μιλήσουμε σαφώς για τον τύπο οποιουδήποτε δεδομένου διαστήματος.

Τώρα εξετάζουμε τον τελεστή POST, για τον οποίο, λαμβάνοντας υπόψη το διάστημα ι = [b:e], επιστρέφει τον άμεσο διάδοχο e + 1 του τελικού σημείου το e του I. Η εφαρμόσιμη λειτουργία διαδόχων είναι γνωστή, και έτσι δεν υπάρχει καμία ανάγκη για να περιληφθεί ένας χαρακτηριστής _T. Κατά συνέπεια, τα εξής είναι μια έγκυρη POST επίκληση:

POST (INTERVAL_INTEGER ([2:6]))

Το αποτέλεσμα, φυσικά, είναι 7. Σε αντίθεση, η POST επίκληση

POST (INTERVAL_EVEN_INTEGER ([2:6]))

επιστρέφει 8, και όχι 7.

Πρέπει να είναι σαφές ότι οι ανάλογες παρατηρήσεις ισχύουν εξίσου για όλους τους PRE, BEGIN, END τελεστές διαστήματος, επειδή η εφαρμόσιμη λειτουργία διαδόχων, όταν απαιτείται, είναι πάντα γνωστή. Με άλλα λόγια, οι μόνοι τελεστές διαστήματος που απαιτούν τον χαρακτηριστή _T είναι οι επιλογείς διαστήματος(οι επιλογείς διαστήματος διαφέρουν από τους άλλους τελεστές που αναφέρθηκαν, διότι οι τελεστέοι είναι σημεία και όχι διαστήματα).

9.5 Συνεχείς Τύποι Σημείου

Υπάρχει ένα τελευταίο ζήτημα που πρέπει να συζητήσουμε σε αυτό το κεφάλαιο, και αυτό είναι η δυνατότητα των συνεχών τύπων σημείων. Όλα τα σημεία τύπων που έχουμε εξετάσει μέχρι τώρα ήταν ιδιαίτεροι ή «τμηματικοί» τύποι, και είχαμε υποθέσει ότι τα διαστήματα καθορίζονται πάντα πέρα από αυτούς τους τύπους και αποτελούνται από μια πεπερασμένη ακολουθία ιδιαίτερων σημείων. Επομένως η ερώτηση είναι: τι θα συμβεί εάν απορρίψουμε αυτήν την υπόθεση; Με άλλα λόγια θα ήταν δυνατό να χτιστεί μία προσέγγιση στα διαστήματα (και ιδιαίτερα στα χρονικά) που να είναι βασισμένη στους συνεχείς τύπους σημείου;

Η προφανέστερη και θεμελιώδης παρατήρηση είναι ότι οι πραγματικοί αριθμοί δεν *έχουν* καμία λειτουργία διαδόχων (εάν το n είναι ένας πραγματικός αριθμός, δεν υπάρχει κανένας next πραγματικός αριθμός n'). Συνέπεια αυτού του γεγονότος, είναι ότι κάποιοι από τους τελεστές που καθορίστηκαν νωρίτερα σε αυτό το βιβλίο για τα σημεία διαστήματος, δεν είναι πλέον διαθέσιμοι.

- **Όσον αφορά τα σημεία:** οι τελεστές σύγκρισης ("=" ">" κ.λ.π...) ακόμα δουλεύουν, αλλά οι τελεστές NEXT_T, PRIOR_T, IS_NEXT_T *και* IS_PRIOR_T δεν έχουν κανένα νόημα. Όσον αφορά τους FIRST_T *και* LAST_T τελεστές δουλεύουν ακόμα, στους χρονικούς όρους επιστρέφουν 'την αρχή του χρόνου' και 'το τέλος του χρόνου' αντίστοιχα, αλλά υπόκεινται σε μια μικρή ανωμαλία όπως θα δούμε παρακάτω.

- **Όσον αφορά τα διαστήματα:** *η* COUNT δεν ισχύει πλέον, *οι* Allen's *τελεστές* ισχύουν, αλλά η MEETS απαιτεί προσεκτικό καθορισμό και δεν μπορούμε ενδεχομένως να πούμε ότι τα διαστήματα [b: n] *και* [n': e] συναντιούνται(meet), επειδή (πάλι) δεν μπορούμε να πούμε ότι το n'είναι ο διάδοχος του n. Εντούτοις, εάν υιοθετήσουμε την closed-open μορφή τότε μπορούμε να πούμε ότι τα διαστήματα [b: n) *και* [n': e) συναντιούνται (meet). Στην πραγματικότητα, ούτε *η* closed-closed ούτε η open-open μορφή για το γράψιμο των διαστημάτων δεν έχει πολύ νόημα στις περιπτώσεις συνοχής, και οι τελεστές επιλογέων διαστήματος θα πρέπει να περιοριστούν σε μια από τις άλλες δύο μορφές

Τώρα μπορούμε να εξηγήσουμε την «ανωμαλία» που παρουσιάζεται κατά τη σύνδεση *του* FIRST_T *και* LAST_T τελεστή. Εάν χρησιμοποιήσουμε την open-closed μορφή για τους επιλογείς διαστημάτων, τότε τα διαστήματα που περιλαμβάνουν "την αρχή του χρόνου" δεν μπορούν να εκφραστούν. Εναλλακτικά εάν χρησιμοποιήσουμε την closed – open μορφή, τότε τα διαστήματα που περιλαμβάνουν "το τέλος του χρόνου" δεν μπορούν να εκφραστούν. Στην πραγματικότητα, ούτε η open-closed, ούτε η closed – open μορφή δεν είναι η αληθινή αντιπροσώπευση του τύπου διαστήματος, και η ίδια παρατήρηση ισχύει και για την open-open μορφή.

Από αυτό το σημείο, θα υποθέτουμε την closed – open μορφή. Σε αυτές τις περιπτώσεις, *οι* τελεστές PRE *και* END δεν είναι πλέον διαθέσιμοι. Σημειώστε, ωστόσο, ότι το "τελικό σημείο" e του διαστήματος ι = [b: ε) δίνεται από την POST (ι). Παρόλ' αυτά "το σημείο τέλους" δεν συμπεριλαμβάνεται πραγματικά στο διάστημα ι και δεν είναι έτσι αληθινά ένα τελικό σημείο υπό αυτήν τη μορφή, υπό την έννοια στην οποία αυτός ο όρος καθορίστηκε στο κεφάλαιο 3.

Τέλος, οι EXPAND *και* UNPACK τελεστές δεν είναι πλέον διαθέσιμοι. Η απώλεια του UNPACK ειδικότερα είναι ένα θέμα για κάποια ανησυχία. Ο "u_" *τελεστής* μας καθορίζεται από τον UNPACK. Κατά συνέπεια, θα πρέπει να ξαναεπισκεφτούμε όλους αυτούς τους τελεστές για να δούμε εάν θα μπορούσαμε να τους επαναπροσδιορίσουμε με τέτοιο τρόπο ώστε να αποφύγουμε να στηριχθούμε στον UNPACK. Θα πρέπει επίσης να ελέγξουμε ότι αυτοί οι αναθεωρημένοι τελεστές ήταν όλοι εκτελέσιμοι.

ΕΥΡΕΤΗΡΙΟ ΕΝΝΟΙΩΝ

ΒΙΒΛΙΟΓΡΑΦΙΑ

- TEMPORAL DATA AND THE RELATIONAL MODEL
  C. J. DATE, HUGH DARWEN, NIKOS LORENTZOS
- James F. Allen: "Maintaining Knowledge about Temporal Intervals"
- Gad Ariav: "A Temporally Oriented Data Model"
- J. Ben-Zvi: "The Time Relational Model"
- Michael H. Bohlen: " Temporal Database System implementations"
- James Clifford: "A Model for Historical Databases"
- James Clifford and Albert Croker: "The historical Relational Data Model (HRDM) Revisited"
- C. J. Date: "The Primacy of Primary Keys: An investigation"
- C. J. Date: "The Final Normal Form!"
- C. J. Date: An introduction to Database Systems
- C. J. Date: The database Relational Model: A retrospective review and Analysis
- C. J. Date and Hugh Darwen: Foundation for Future Database Systems: The Third Manifesto
- Ronald Fagin: "Normal Forms and Relational Database Operators"
- Nikos A. Lorentzos: "DBMS Support For Time and Totally Ordered Compound Data Types"
- Nikos A. Lorentzos and R. G. Johnson: "An Extension of the Relational Model to Support Generic Intervals"
- E. McKenzie and R. Snodgrass: "Supporting Valid Time: An Historical Algebra"