ΣΧΕΔΙΑΣΤΙΚΑ ΠΡΟΤΥΠΑ ΓΙΑ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

Η
ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

Υποβάλλεται στην

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

από την

Ευγενία Σταθοπούλου

ως μέρος των Υποχρεώσεων

για τη λήψη

του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ ΣΤΟ ΛΟΓΙΣΜΙΚΟ

Ιούλιος 2007

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω, πρωτίστως, τον επιβλέποντα καθηγητή μου κ.κ. Παναγιώτη Βασιλειάδη, ο οποίος με βοήθησε, σε όλη την διάρκεια της διατριβής μου, τόσο σε γνωστικό, όσο και σε ψυχολογικό επίπεδο. Τον ευχαριστώ πολύ, κυρίως, για την υποστήριξη και την υπομονή του. Θα ήθελα να ευχαριστήσω, επίσης, τους φίλους και συνάδελφους του εργαστηρίου μου (*Ιωάννη Κρομμύδα*, *Τάσο Καραγιάννη*, *Ευτυχία Μπαϊκούση*, *Φωτεινή Πεχλιβάνη*), οι οποίοι αποτέλεσαν σημαντικό παράγοντα για την εκπόνηση της διατριβής αυτής. Τους ευχαριστώ πολύ για το υπέροχο κλίμα εργασίας και τη βοήθεια, που μου προσέφεραν, και τους εύχομαι καλή σταδιοδρομία. Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου και τους φίλους μου, χωρίς τη βοήθεια και την πίστη των οποίων δεν θα μπορούσα να είχα επιτεύξει αυτόν τον στόχο.

ΠΕΡΙΕΧΟΜΕΝΑ

Σελ

.

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ

Σχήμα Σελ.

ΠΕΡΙΛΗΨΗ

Ευγενία Σταθοπούλου του Μαρίνη και της Παναγιώτας. MSc, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Ιούλιος, 2007. Σχεδιαστικά Πρότυπα για Βάσεις Δεδομένων. Επιβλέποντας: Παναγιώτης Βασιλειάδης.

Η διαδικασία σχεδίασης πληροφοριακών συστημάτων που οργανώνονται γύρω από μια βάση δεδομένων ξεκινά με την απεικόνιση, με γενικό και τυπικό τρόπο, των οντοτήτων και των συσχετίσεων που εμφανίζονται μεταξύ τους, κατά την αναπαράσταση ενός μέρους του πραγματικού κόσμου. Η αναπαράσταση αυτή, μετατρέπεται σε σχεδιαστικές λύσεις για τη δομή της βάσης δεδομένων στα αρχικά στάδια της σχεδίασής της. Τα σχεδιαστικά πρότυπα, άμεσα συνδεδεμένα με την διαδικασία της σχεδίασης, αποτελούν γενικούς ορισμούς λύσεων σε προβλήματα, τα οποία εμφανίζονται συχνά κατά τη σχεδίαση ενός λογισμικού. Δυστυχώς, τα σχεδιαστικά πρότυπα, που έχουν παρουσιαστεί μέχρι σήμερα, αφορούν, κυρίως, στον αντικειμενοστρεφή προγραμματισμό. Η έλλειψη σχεδιαστικών προτύπων για τις Βάσεις Δεδομένων, αποτελεί, για μας, ένα σημαντικό κίνητρο για την δημιουργία αυτής της εργασίας. Σε αυτή τη μελέτη ασχολούμαστε με σημαντικά, συχνά συναντώμενα προβλήματα, που εμφανίζονται κατά τη σχεδίαση μιας βάσης. Οι εναλλακτικές δυνατότητες επίλυσης των προβλημάτων αυτών παρουσιάζονται συστηματικά με τη μορφή σχεδιαστικών προτύπων. Κάθε σχεδιαστικό πρότυπο αντιστοιχίζεται σε ένα συγκεκριμένο πρόβλημα, το οποίο ορίζεται πριν την απόδοση οποιασδήποτε λύσης. Το όνομα κάθε προτύπου, περιγράφει με συνοπτικό και σαφές τρόπο το πρόβλημα, στο οποίο αντιστοιχίζεται. Για κάθε πρόβλημα, δίδεται το σύνολο των σχεδιαστικών λύσεων, μέσα από το οποίο μπορεί να επιλέξει ένας σχεδιαστής. Εκτός από τη σχηματική αναπαράσταση κάθε σχεδιαστικής λύσης, αρκετές φορές, δίδεται και ο αντίστοιχος κώδικας υλοποίησης σε SQL ή PL/SQL.

Η θεώρησή μας στηρίζεται σε δύο θεμελιώδεις «οπτικές γωνίες», τη συντηρησιμότητα και την αποτίμηση της εσωτερικής ποιότητας κάθε προτεινόμενης λύσης. Σε σχέση με την πρώτη οπτική γωνία, για κάθε λύση, μελετάται η συμπεριφορά της βάσης, που σχεδιάζεται με βάση τη λύση αυτή, σε επίπεδο στιγμιότυπου (εγγραφών) και σε επίπεδο σχήματος (πεδίων) όλων των σχετικών πινάκων. Σε σχέση με την δεύτερη οπτική γωνία, μια γενική αποτίμηση του μεγέθους, της απόδοσης και του κόστους συντήρησης της εκάστοτε λύσης, δίδεται μέσα από την μελέτη μετρικών αποτίμησης της εσωτερικής ποιότητας των προτεινόμενων σχεδιαστικών λύσεων, και συγκεκριμένα, του μεγέθους, μήκους, όγκου, συνδεσιμότητας και συνεκτικότητας της κάθε σχεδιαστικής λύσης. Με βάση όλα τα παραπάνω, ο σχεδιαστής είναι σε θέση να επιλέξει την καλύτερη, γι' αυτόν, σχεδίαση, γνωρίζοντας εκ των προτέρων τα πλεονεκτήματα και τα μειονεκτήματά της.

EXTENDED ABSTRACT IN ENGLISH

Stathopoulou, Eugenia. MSc. Computer Science Department, University of Ioannina, Greece. July, 2007. Design Patterns for Databases. Thesis Supervisor: Panagiotis Vassiliadis.

The last four decades, in computer science field and especially in software engineering, the idea of conceptual modeling has flourished. The purpose of conceptual modeling is to capture and present, using a general and formal way, the entities and their relationship of a part of the real world. Design patterns, based on this idea, define in general way common problems that occur, very often, during the design process. So far the related literature has focused to object-oriented patterns. Thus, the lack of patterns for the case of a database design was our prime motivation. In this study, we present nine design patterns, which represent commonly encountered problems of database design. Each pattern focuses on one specific problem. Given alternative solutions for a problem, a designer can choose and adopt the most suitable, solution. The description of a pattern-based solution to a problem includes a formal description both the schema and the instance level. Wherever necessary, we also give the code in SQL or PL/SQL. In our effort to equip the designer with the appropriate "tools" for the selection of the most suitable solutions for his needs, we maintain two fundamental viewpoints over the introduced patterns: maintenance and design quality. With respect to the former, we study the impact of evolution events both at instance and schema level for each of the proposed solutions. For the latter, we use established metrics to measure the size, the effectiveness and the effort of maintenance of the database, for each alternative. In this way, the designer will be able to know, right from the beginning, the advantages and disadvantages of his development. Our belief

is that, through the recording and the learning of the design patterns, a designer is in position to get a design “right” faster, minimizing –or more hopefully avoiding- the redesign.

ΚΕΦΑΛΑΙΟ 1. ΕΙΣΑΓΩΓΗ

1.1. Στόχοι

Ένα *σχεδιαστικό πρότυπο* (*design pattern*) αποτελεί τυπική περιγραφή του τρόπου επίλυσης ενός προβλήματος, το οποίο εμφανίζεται συχνά κατά τον σχεδιασμό ενός λογισμικού. Σκοπό ενός σχεδιαστικού προτύπου αποτελεί η γενίκευση ενός προβλήματος και των λύσεων του, με τέτοιο τρόπο ώστε να μπορεί να εφαρμοστεί, με μικρές τροποποιήσεις, και σε παρόμοια προβλήματα. Ένα από τα πιο συχνά προβλήματα που εμφανίζονται κατά τον σχεδιασμό μιας Βάσης Δεδομένων, αποτελεί το παράδειγμα του τεχνητού κλειδιού. Στις περισσότερες περιπτώσεις, ο σχεδιαστής βρίσκεται αντιμέτωπος με ένα μεγάλο σχεδιαστικό δίλημμα, αν θα πρέπει, ή όχι, να χρησιμοποιήσει τεχνητό κλειδί σε έναν πίνακα κατά τον σχεδιασμό μιας βάσης. Μελετώντας τις λειτουργικές απαιτήσεις του συστήματος, σχεδιάζεται μια νέα λύση, προσανατολισμένη στο πρόβλημα. Το μειονέκτημα, της εκ νέου σχεδίασης μιας Βάσης Δεδομένων, αφορά στην εκ των υστέρων «αποτίμηση» της σχεδίασης. Δηλαδή, ο σχεδιαστής, εφόσον έχει επιλέξει κάποια από τις δύο πιθανές σχεδιαστικές λύσεις, θα πρέπει να ελέγξει κατά πόσο η βάση που έχει δημιουργήσει, *i)* αποτελείται από συνεπή δεδομένα, *ii)* έχει όσο το δυνατόν μικρότερο κόστος διατήρησης και *iii)* λειτουργεί αποδοτικά κατά την εφαρμογή ερωτήσεων (queries). Τα παραπάνω προβλήματα, σχεδόν εκμηδενίζονται, με την κατασκευή και την εκμάθηση σχεδιαστικών προτύπων. Κάθε σχεδιαστικό πρότυπο μοντελοποιεί ένα συγκεκριμένο πρόβλημα, με γενικό τρόπο, και παρέχει όλες τις δυνατές σχεδιαστικές λύσεις του. Η γνώση, εκ των προτέρων, της συμπεριφοράς της βάσης που σχεδιάζεται, αποτελεί

σημαντικό πλεονέκτημα της εκμάθησης των σχεδιαστικών προτύπων. Επιπλέον, η γενική μορφή της παρουσίασης ενός σχεδιαστικού προτύπου επιτρέπει την εφαρμογή του σχετικού προτύπου σε οποιοδήποτε στάδιο της ανάπτυξης ενός λογισμικού συστήματος, π.χ., κατασκευή διεπαφής μέσω αντικειμενοστρεφούς προγραμματισμού, σχεδίαση μιας βάσης κ.τ.λ..

Η ιδέα της μοντελοποίησης ενός μέρους του κόσμου, το οποίο ουσιαστικά αποτελείται από ένα πρόβλημα και το σύνολο των πιθανών λύσεων του, με έναν γενικό και κάπως αφαιρετικό τρόπο απεικόνισης, ξεκίνησε στις αρχές στα μέσα της δεκαετίας ’60. Το σημαντικότερο, βήμα, προσανατολιζόμενο στις Βάσεις Δεδομένων, πραγματοποιείται στα μέσα του ‘70 δεκαετίας με τον ορισμό των *σημασιολογικών μοντέλων δεδομένων* (*semantic data models*), όπως το *μοντέλο οντοτήτων-συσχετίσεων* (*entity-relationship model*) του Chen και το *σχεσιακό μοντέλο* (*relational model*) του Codd, τα οποία συνδύαζαν τεχνικές αναπαράστασης γνώσης με την τεχνολογία των Βάσεων Δεδομένων, οδηγώντας σε συστήματα που υπόσχονταν τόσο δύναμη μοντελοποίησης όσο και απόδοση [18]. Μία δεκαετία αργότερα, εμφανίζεται ο όρος *εννοιολογική μοντελοποίηση* (*conceptual modeling*), ο οποίος εστιάζει κυρίως στον αντικειμενοστρεφή προγραμματισμό, αντιπροσωπεύει τη διαδικασία της τυπικής περιγραφής του φυσικού κόσμου, με σκοπό την κατανόηση και την επικοινωνία μεταξύ των κατασκευαστών/σχεδιαστών. Η εννοιολογική μοντελοποίηση, αποδεικνύεται ότι, υπερτερεί τόσο της φυσικής γλώσσας, εξαιτίας της φορμαλιστικής περιγραφής της, όσο και της μαθηματικής γλώσσας, λόγω του ότι οι δομές και οι δραστηριότητες τις οποίες υποστηρίζει βασίζονται στο φυσικό κόσμο και είναι πιο κατανοητές. Βασιζόμενες στην ιδέα της εννοιολογικής μοντελοποίησης, το 2005 προτείνονται στο [3], οι εννιά, πιο σημαντικές, γενικές συσχετίσεις, οι οποίες εμφανίζονται συχνά κατά τη σχεδίαση ενός λογισμικού. Οι γενικές συσχετίσεις αποτελούν υψηλού επιπέδου πρότυπα, μέσω των οποίων παρουσιάζονται οι ομοιότητες, που εμφανίζονται στις συσχετίσεις μεταξύ των οντοτήτων (αντικειμένων) του πραγματικού κόσμου, απομακρύνοντας την πλεονάζουσα πληροφορία που τις διαφοροποιεί. Η εργασία των Gamma et al. [8], αποτελεί την πιο αντιπροσωπευτική μελέτη των σχεδιαστικών προτύπων, όπου ορίζονται και περιγράφονται συγκεκριμένα πρότυπα, που μπορούν να εφαρμοστούν κατά τον αντικειμενοστρεφή προγραμματισμό λογισμικού. Σε αυτή την εργασία, τα πρότυπα παρουσιάζονται ως

ένας τρόπος αναπαράστασης μιας κοινά αποδεκτής δομής σχεδίασης συγκεκριμένων προβλημάτων, η οποία είναι δυνατόν να επαναχρησιμοποιηθεί σε παρόμοια προβλήματα μοντελοποίησης. Για κάθε πρότυπο, ορίζεται το γενικό πρόβλημα που επιλύεται, παρουσιάζεται η λύση του προβλήματος με την περιγραφή των στοιχείων (σχέσεις και συσχετίσεις) που μετέχουν στον σχεδιασμό, και τέλος δίδονται τα πλεονεκτήματα και τα μειονεκτήματα της χρήσης του συγκεκριμένου προτύπου. Η μελέτη, όμως, αυτή, καθώς και όλες οι άλλες που στηρίχτηκαν σε αυτή, είναι προσανατολισμένες στον αντικειμενοστρεφή προγραμματισμό.

Βασιζόμενοι στον τρόπο περιγραφής των σχεδιαστικών προτύπων του [8], και δεδομένου ότι δεν έχουν παρουσιαστεί πρότυπα, συγκεκριμένα, για την περίπτωση των Βάσεων Δεδομένων, σκοπός της συγκεκριμένης εργασίας είναι η κατασκευή σχεδιαστικών προτύπων για τις Βάσεις Δεδομένων. Για μας πρότυπο αποτελεί ο γενικός ορισμός ενός, συχνά επανεμφανιζόμενου, προβλήματος σχεδίασης μιας βάσης και η απόδοση όλων των σχεδιαστικών λύσεών του. Καθορίζοντας τα στοιχεία (σχέσεις και συσχετίσεις), που εμφανίζονται κατά την μοντελοποίηση, παρουσιάζουμε όλους τους δυνατούς τρόπους σχεδίασης μιας βάσης, για το συγκεκριμένο πρόβλημα. Η δομή της κάθε σχεδιαστικής δίδεται σχηματικά, για το γενικό αλλά και για ένα πιο συγκεκριμένο πρόβλημα, ώστε να είναι πλήρως κατανοητή από τον σχεδιαστή. Σε επόμενο στάδιο μελετάται, για κάθε προτεινόμενη λύση, η συμπεριφορά της βάσης κατά την επιλογή/εισαγωγή/διαγραφή/ανανέωσης πλειάδων και πεδίων από τους διάφορους πίνακες. Τέλος, παραθέτοντας τα πλεονεκτήματα και τα μειονεκτήματα κάθε λύσης, ο σχεδιαστής είναι σε θέση να επιλέξει τον κατάλληλο σχεδιασμό, δεδομένου των λειτουργικών απαιτήσεων της βάσης του. Δεδομένου, ότι, η εργασία απευθύνεται σε σχεδιαστές όλων των επιπέδων, αλλά κυρίως σε μη έμπειρους, δίδονται, στις περισσότερες περιπτώσεις, τρόποι περιγραφής των λύσεων στο πρότυπο SQL-1999 και στην Oracle10g. Επιπλέον, όταν η εύρεση μιας πληροφορίας ή η κατασκευή μιας διαδικασίας, είναι πολύπλοκη, παρατίθεται ο σχετικός κώδικας σε SQL-99 ή PL/SQL.

Μετά τη μελέτη της συμπεριφοράς της βάσης, απομένει η αξιολόγηση του σχεδιασμού της ως προς, *i)* τη συνέπεια των δεδομένων της, *ii)* την αποδοτική εφαρμογή ερωτήσεων σε αυτή, και *iii)* το κόστος διατήρησής της. Σε αυτή την

εργασία, υιοθετούνται οι μετρικές *μέγεθος*, *μήκος*, *συνδεσιμότητα*, *συνεκτικότητα*, που παρουσιάζονται στο [1], καθώς και η μετρική *μέγεθος δεδομένων*. Οι μετρικές, ο οποίες ορίζονται στο [1], αν και είναι προσανατολισμένες στην περίπτωση αντικειμενοστρεφούς προγραμματισμού, μεταφέρονται, με λίγες τροποποιήσεις, και στην περίπτωση μιας Βάσης Δεδομένων. Πιο συγκεκριμένα, στην περίπτωσή μας, σύστημα αποτελεί το σχεδιαστικό πρότυπο και οι πιθανές επερωτήσεις που μπορεί να τεθούν προς αυτό. Μετρώντας το μέγεθος του συστήματος, υπολογίζεται το κόστος της κατασκευής και της διατήρησης της συγκεκριμένης σχεδιαστικής λύσης. Υπολογίζοντας το μήκος του συστήματος, είμαστε σε θέση να προβλέψουμε σε τι ποσοστό επηρεάζεται το σχήμα της βάσης κατά την εισαγωγή/διαγραφή ενός γνωρίσματος μιας σχέσης. Η συνδεσιμότητα, όπως και η συνεκτικότητα, δεν αφορούν σε όλο το σύστημα, αλλά στα επιμέρους υποσυστήματά του. Ως υποσυστήματα, θεωρούμε τις σχέσεις και τις επερωτήσεις. Η ύπαρξη μεγάλης συνδεσιμότητας μεταξύ των σχέσεων του προτύπου, υποδηλώνει μεγάλη εξάρτηση μεταξύ των γνωρισμάτων των σχέσεων (δηλαδή η εισαγωγή/διαγραφή μιας πλειάδας ενός πεδίου ενός πίνακα απαιτεί την ενημέρωση επιπλέον πεδίων και άλλων πινάκων), συμπεραίνοντας ότι η διατήρηση, της συγκεκριμένης βάσης, έχει μεγάλο κόστος. Μια σχέση ή μια ερώτηση παρουσιάζει μεγάλη συνεκτικότητα όταν εκτελεί μια και μονό εργασία. Με πιο απλά λόγια, όταν η βάση είναι σε κανονική μορφή, και αντίστοιχα η απάντηση της ερώτησης δεν προϋποθέτει συνενώσεις πινάκων, θεωρούμε ότι έχουμε μέγιστη συνεκτικότητα. Τέλος, το μέγεθος των δεδομένων αφορά στα στοιχεία που αποθηκεύονται σε κάθε πίνακα της βάσης, δίδοντας μια γενική εκτίμηση του συνολικού αποθηκευτικού χώρου που θα πρέπει να δεσμευτεί για τη συγκεκριμένη βάση.

Σκοπός, λοιπόν, αυτής της εργασίας είναι η παρουσίαση γενικών προβλημάτων, που εμφανίζονται συχνά κατά τον σχεδιασμό μιας Βάσης Δεδομένων, και η αποτίμηση όλων δυνατών λύσεων κάθε προβλήματος. Πιστεύουμε ότι μέσω της μελέτης των προτύπων, ο σχεδιαστής «απελευθερώνεται» από το «βάρος» μιας εξ' ολοκλήρου νέας σχεδίασης, η οποία έχει μη προβλεπόμενα αποτελέσματα. Η εκ των προτέρων γνώση της απόδοσης, του κόστους κατασκευής και του κόστους διατήρησης μιας σχεδίασης, αποτελεί τον πιο σημαντικό παράγοντα για να επιλέξει κανένας την υιοθέτηση ενός σχεδιαστικού προτύπου, έναντι μιας νέας σχεδίασης. Με βάση όλα τα

παραπάνω, πιστεύουμε ότι η εκμάθηση των προτύπων, μπορεί να συντελέσει σε μια αποδοτική και πιο γρήγορη σχεδίαση ενός λογισμικού συστήματος.

1.2. Δομή της Διατριβής

Η εργασία έχει την ακόλουθη οργάνωση. Στο Κεφάλαιο 2, παρουσιάζονται μελέτες των τελευταίων 10 ετών σε ότι αφορά στην εννοιολογική μοντελοποίηση και σε διάφορους φορμαλιστικούς ορισμούς που προτάθηκαν για την απεικόνιση του πραγματικού κόσμου. Επιπλέον, παρατίθενται πρόσφατες πειραματικές μελέτες, αποδεικνύοντας το κατά πόσο και σε ποιες περιπτώσεις, η εκμάθηση των προτύπων αποδίδει κατά το σχεδιασμό. Στο Κεφάλαιο 3, παρουσιάζεται το κίνητρο της δημιουργίας των προτύπων για την περίπτωση των Βάσεων Δεδομένων καθώς και η δομή των σχεδιαστικών προτύπων που θα μελετηθούν στο Κεφάλαιο 4. Στο Κεφάλαιο 4 παρουσιάζονται, αναλυτικά, εννιά σχεδιαστικά πρότυπα, τα οποία επιλύουν εννιά σημαντικά προβλήματα των Βάσεων Δεδομένων. Για κάθε πρότυπο, δίδονται όλες οι εναλλακτικές σχεδιαστικές λύσεις, μέσω των οποίων μπορεί να κατασκευαστεί το πρότυπο, μελετώντας τη δομή και τη συμπεριφορά της κάθε μίας. Στο Κεφάλαιο 5, παρατίθενται οι μετρικές ποιότητας των προτύπων, αποτιμώντας, για κάθε σχεδιαστική λύση, τον απαιτούμενο αποθηκευτικό χώρο, την απόδοση και τη δυσκολία της διατήρησης της βάσης, που κατασκευάζεται με βάση τη συγκεκριμένη σχεδίαση. Τέλος, στο Κεφάλαιο 6, παρουσιάζονται τα συμπεράσματα που προκύπτουν από την εργασία μας, ενώ συμπεριλαμβάνονται οι κατευθύνσεις μελλοντικών επεκτάσεων της.

ΚΕΦΑΛΑΙΟ 2. ΒΙΒΛΙΟΓΡΑΦΙΑ

Σε αυτό το κεφάλαιο παρουσιάζονται αναλυτικά οι βασικές αρχές *σχεδιαστικών προτύπων* (*design patterns*), η αναγκαιότητα δημιουργίας τους για την *εννοιολογική μοντελοποίηση* (*conceptual modeling*) και αναπαράσταση πραγμάτων του φυσικού κόσμου, καθώς επίσης και κάποιες πρόσφατες μελέτες που έχουν πραγματοποιηθεί με θέμα τη σχεδίαση προτύπων γενικά. Οι μελέτες ταξινομούνται, στην επόμενη παράγραφο, ανάλογα με το θέμα που πραγματεύονται στις κατηγορίες: *Βασικές αρχές* (*Fundamentals*), *Φορμαλισμοί* (*Formalisms*), *Πειράματα* (*Experiments*) και *Διάφορα* (*Miscellaneous*).

2.1. Βασικές αρχές (Fundamentals)

Στο [17] παρουσιάζονται οι βασικές αρχές της εννοιολογικής μοντελοποίησης (Conceptual Modeling) καθώς επίσης προτείνεται μια γλώσσα τυπικής περιγραφής της γνώσης για την εννοιολογική μοντελοποίηση του φυσικού κόσμου.

Τα τελευταία χρόνια έχουν δημιουργηθεί τρεις περιοχές μελέτης, η *Εννοιολογική Μοντελοποίηση* (*Conceptual Modeling*), η *Αναπαράσταση Γνώσης* (*Knowledge Representation*) και η *Σημασιολογική Μοντελοποίηση Δεδομένων* (*Semantic Data Modeling*), οι οποίες έχουν ως κοινό θέμα την απεικόνιση γνώσης για ένα συγκεκριμένο πρόβλημα. Το Knowledge Representation εστιάζει στη δημιουργία

λογικών προτύπων τα οποία στη συνέχεια επεξεργάζονται από κάποιο «expert system». Το Conceptual Modeling απεικονίζει μέρος του κόσμου, με σκοπό την καλύτερη κατανόηση του από τον άνθρωπο. Το Semantic Data Modeling, παρόμοια με το Conceptual Modeling, απεικονίζει ένα μέρος του κόσμου με σχήματα τα οποία όμως έχουν τελικό αποδέκτη σε κάποιο μηχάνημα.

Το Conceptual Modeling χρησιμοποιείται για την οργάνωση και την υποστήριξη ενός *πληροφοριακού συστήματος* (*information system*). Με τον όρο πληροφοριακό σύστημα εννοούμε ένα σύστημα το οποίο διατηρεί πληροφορία σχετική με την απεικόνιση ενός μέρους του πραγματικού κόσμου. Υπάρχουν τέσσερα είδη πληροφορίας, τα οποία απεικονίζονται σε ένα πληροφοριακό σύστημα και χαρακτηρίζονται ως περιβάλλοντα (worlds):

1. *Περιβάλλον υποκειμένου* (*Subject world*). Αποτελεί το μέρος του πραγματικού κόσμου για το οποίο διατηρείται η πληροφορία.
2. *Περιβάλλον συστήματος* (*System world*). Αποτελεί το περιβάλλον καταχώρησης της πληροφορίας.
3. *Περιβάλλον χρήσης* (*Usage world*). Αποτελεί το περιβάλλον των χρηστών, δηλαδή αυτών που χειρίζονται την πληροφορία.
4. *Περιβάλλον ανάπτυξης* (*Development world*). Αποτελεί το περιβάλλον στο οποίο εφαρμόζεται η διαδικασία ανάπτυξης του πληροφοριακού συστήματος.

Στις σημαντικότερες γλώσσες μοντελοποίησης αναφέρεται, εκτός των άλλων, η *RML* (*Requirements Modeling Language*) η οποία συνδυάζει την αντικειμενοστρέφεια, παρέχοντας κατασκευαστικές δυνατότητες, με μια υπογλώσσα, η οποία χρησιμοποιείται για τον καθορισμό περιορισμών και παραγωγικών κανόνων. Παρατηρείται, όμως, ότι: *i)* η όψη του περιβάλλοντος απεικονίζεται έτσι ώστε οι ορισμοί των οντοτήτων και των δραστηριοτήτων τους να μπορούν να κατασκευάσουν από μια γλώσσα, και *ii)* οι οντότητες και οι δραστηριότητες δίδουν ένα μικρό και από μέρος του κόσμού, αδυνατώντας να απεικονίσουν το πραγματικό περιβάλλον χρήσης.

Θέλοντας να εξαλειφθούν τα παραπάνω προβλήματα εισάγεται η γλώσσα Telos, στην οποία ορίζονται μεταγνωρίσματα. Τα μεταγνωρίσματα χρησιμοποιούνται για την ταξινόμηση των γνωρισμάτων των κλάσεων, δηλαδή δημιουργούνται κατηγορίες

γνωρισμάτων. Για τον ορισμό των περιορισμών χρησιμοποιείται μια γλώσσα πρωτοβάθμιας λογικής, η οποία βασίζεται στη φορμαλιστική απεικόνιση των γνωρισμάτων.

Σκοπός της μελέτης [18] είναι η ανάπτυξη θεωριών, εργαλείων και τεχνικών για τη διαχείριση και την καλύτερη χρήση της πληροφορίας. Η μοντελοποίηση της πληροφορίας προσανατολίζεται στην κατανόηση και τη μοντελοποίηση τεσσάρων περιβαλλόντων, του περιβάλλοντος υποκειμένου, του περιβάλλοντος συστήματος, του περιβάλλοντος χρήσης και του περιβάλλοντος ανάπτυξης.

Μια *βάση πληροφορίας* (*information base*) είναι μια δομή συμβόλων, η οποία βασίζεται στη μοντελοποίηση της πληροφορίας και περιγράφει μια συγκεκριμένη εφαρμογή. Μπορεί να θεωρηθεί ως μια «αποθήκη» πληροφορίας, η οποία εισάγεται με τη μορφή προτάσεων και είναι δομημένη σύμφωνα με το περιεχόμενο (ενδιαφέροντος) της. Το πληροφοριακό μοντέλο θεωρείται ότι είναι ανεξάρτητο της γλώσσας με την οποία επικοινωνούν οι χρήστες, π.χ., SQL queries, αλλά και του *συμβολισμού* (*notation*), π.χ., χρήση data flow diagrams. Χαρακτηριστικά μιας information base αποτελούν: *i) οι όροι* (*terms*), με τα οποία περιγράφονται συγκεκριμένα μεμονωμένα στοιχεία (individuals), π.χ., `Μαρία, Γιώργος, 7`, *ii)* οι *γενικές έννοιες* (*generic concepts*), με βάση τις οποίες κατηγοριοποιούνται τα terms (π.χ., `Μαθητής, Εργαζόμενος`), και *iii)* οι *συσχετίσεις* (*associations*), οι οποίες δηλώνουν τις συσχετίσεις του πραγματικού κόσμου.

Για την δημιουργία και την αξιολόγηση ενός conceptual model θα πρέπει να ληφθούν υπόψη οι εξής παράγοντες:

- *Οντολογίες* (*Ontologies*): θεωρήσεις για την φύση της εφαρμογής που θα μοντελοποιηθεί (*τι* θα μοντελοποιηθεί).
- *Μηχανισμοί γενίκευσης (Abstract Mechanisms)*: τρόποι οργάνωσης της πληροφορίας οδηγώντας σε πιο εύχρηστες πληροφοριακές βάσεις, οι οποίες μπορούν να επεκταθούν αλλά και να εξερευνηθούν πιο αποτελεσματικά.
- *Εργαλεία (Tools)*: για την κατασκευή, ανάλυση και διαχείριση της πληροφοριακής βάσης ώστε να παρέχει ασφάλεια, συνέπεια και ευχρηστία.

Μια οντολογία χαρακτηρίζει κάποια πεδία ενδιαφέροντος μια κλάσης της εφαρμογής, π.χ., μια οντολογία κατασκευαστών μπορεί να αποτελείται από διεργασίες, φυσικούς και ανθρώπινους πόρους κ.α.. Η οντολογία είναι ένα πλαίσιο (framework), το οποίο παρέχει κοινή ορολογία για την απεικόνιση μιας πληροφορίας. Για να κατανοήσουμε την έννοια της οντολογίας, ας φανταστούμε δυο μοντέλα τα οποία αφορούν στην ίδια κλάση αλλά χρησιμοποιούν διαφορετικούς όρους απεικόνισης. Ορίζοντας μια οντολογία, δημιουργούμε ένα κοινό τρόπο «απόδοσης» της κλάσης παρέχοντας συνοχή ανάμεσα στα μοντέλα. Η οντολογία μπορεί να είναι:

1. *Στατική (Static)*: η οποία απεικονίζει τη στατική πληροφορίας μιας εφαρμογής περιγράφοντας ποια πράγματα υπάρχουν (οντότητες), ποια είναι τα γνωρίσματά τους και ποιες είναι οι μεταξύ τους συσχετίσεις. Η στατική οντολογία δεν μπορεί να εφαρμοστεί σε συστήματα πραγματικού χρόνου, όπου η ύπαρξη, τα γνωρίσματα και οι συσχετίσεις των οντοτήτων μεταβάλλονται με το χρόνο. Επιπλέον είναι αδύνατο να διαχωριστεί η φυσική ύπαρξη, π.χ., η Μαρία είναι φυσικό πρόσωπο, από την αφαιρετική ύπαρξη, π.χ., ο αριθμός 7 έχει αφαιρετική ύπαρξη και όχι φυσική.
2. *Δυναμική (Dynamic)*: η οποία απεικονίζει την δυναμική άποψη μιας εφαρμογής, χρησιμοποιώντας τους όρους *καταστάσεις* (*states*), *μεταβάσεις κατάστασης* (*state transitions*) και *διεργασίες* (*processes*). Μια διεργασία αποτελεί μια συλλογή από μερικώς διατεταγμένες ενέργειες, μέσα από τις οποίες επιτυγχάνεται κάποιος στόχος. Οι διεργασίες εκτελούνται από *πράκτορες* (*agents)*, ανθρώπους ή γενικά. Με την δυναμική οντολογία απεικονίζεται η αλλαγή της κατάστασης μια οντότητας όταν σε αυτή εφαρμοστεί μια διεργασία από κάποιον πράκτορα.
3. *Κινήτρου (Intentional)*: η οποία απεικονίζει τον κόσμο των agents και τα πράγματα που τους αφορούν, όπως τι θέλουν, τι πιστεύουν, τι εγκρίνουν, κ.ο.κ.. Ανάλογα με τις απόψεις και τους σκοπούς του, ο κάθε agent είναι σε θέση να πραγματοποιήσει κάποιες ενέργειες, δηλαδή κάθε agent έχει συγκεκριμένη συμπεριφορά. Οι agents παίρνουν αποφάσεις ανάλογα με τις ερωτήσεις που τους θέτονται, τις επιλογές που έχουν και τα κριτήρια που θεωρούν σημαντικά. Τέτοιες αποφάσεις μπορούν να εφαρμοστούν για θέματα χρηστικότητα, ασφάλειας και εμπιστοσύνης κατά την μοντελοποίηση ενός συστήματος.

4. *Κοινωνική (Social)*: η οποία απεικονίζει την κοινωνική πλευρά της εφαρμογής, χρησιμοποιώντας όρους όπως *συμμετέχοντες* (*actor*), *ρόλος* (*role*), *αρχή* (*authority*), *δέσμευση* (*commitment*) κ.α.. Οι actors παρουσιάζονται να έχουν εξαρτήσεις μεταξύ τους, π.χ., όταν ένας actor έχει δεσμευτεί να εκπληρώσει έναν σκοπό, μέσω μιας διαδικασίας ως προς έναν άλλο actor. Με αυτόν τον τρόπο μπορούμε να απαντήσουμε σε ερωτήσεις του τύπου «*γιατί συμβαίνει αυτό*».

Οι μηχανισμοί γενίκευσης οργανώνουν την πληροφοριακή βάση και καθοδηγούν τους χρήστες για τη σωστή χρήση της, κάνοντας ευκολότερη την ανανέωση (update) και αναζήτηση (search) σε αυτή. Σε αυτή την ενότητα αναφέρονται εφτά (7) αφαιρετικοί μηχανισμοί:

1. *Κατηγοριοποίηση* (*Classification*). Ένα άτομο (οντότητα, συσχέτιση, γνώρισμα, δραστηριότητα ή οτιδήποτε άλλο) μιας πληροφοριακής βάσης μοντελοποιείται σε ένα ή περισσότερα γενικά άτομα (κλάσεις). Το άτομο θεωρείται ότι αποτελεί στιγμιότυπο της γενικής κλάσης, έχοντας τις ιδιότητες της κλάσης. Η κατηγοριοποίηση μπορεί να είναι αναδρομική, δηλαδή μια κλάση να αποτελεί στιγμιότυπο μιας άλλης κλάσης. Σε τέτοιες περιπτώσεις η κατηγοριοποίηση μπορεί να πραγματοποιείται με περιορισμούς ή χωρίς περιορισμούς, Όταν το classification έχει περιορισμούς, σημαίνει ότι κάθε άτομο της πληροφοριακής βάσης ανήκει σε κάποιο επίπεδο της ιεραρχίας και είναι σε θέση να αποτελεί στιγμιότυπο κλάσεων μόνο του συγκεκριμένου επιπέδου ιεραρχίας, π.χ., το άτομο `Jenny` αποτελεί στιγμιότυπο της κλάσης `STUDENT` και όχι της μετακλάσης `PERSON`. Αντίθετα στην περίπτωση όπου δεν υπάρχουν περιορισμοί, μία κλάση και ένα στιγμιότυπό της μπορεί να αποτελούν στιγμιότυπα μιας άλλης κλάσης (ή μετακλάσης) υψηλότερου επιπέδου. Πολλά σχήματα επιτρέπουν την πολλαπλή κατηγοριοποίηση ενός ατόμου, π.χ., το άτομο `Jenny` μπορεί να αποτελεί στιγμιότυπο των κλάσεων `STUDENT`, `PERSON`, `EMPLOYEE` την ίδια στιγμή. Τέλος, ορισμένα σχήματα αντιμετωπίζουν την κατηγοριοποίηση ως μια σταθερή ιδιότητα ενός ατόμου, ενώ αλλά επιτρέπουν την αλλαγή κατηγορίας ως συνέπεια του χρόνου, π.χ., το άτομο `Jenny` στην κλάση `PERSON` απεικονίζει μια σταθερή ιδιότητα, ενώ στην κλάση `STUDENT` μια ιδιότητα που θα αλλάξει.

2. *Γενίκευση (Generalization).* Τα άτομα και οι κλάσεις μιας πληροφοριακής βάσης οργανώνονται σε ιεραρχίες ανάλογα με την γενικότητα ή την εξειδίκευση που παρουσιάζουν. Σημαντική ιδιότητα αυτού του μηχανισμού αποτελεί η κληρονομικότητα των γνωρισμάτων και των ιδιοτήτων μιας κλάσης σε όλες τις κλάσεις που αποτελούν εξειδικεύσεις της. Π.χ., εξειδίκευση της `PERSON` αποτελεί η κλάση `STUDENT`. Τα γνωρίσματα ηλικία (`Age`) και διεύθυνση (`Address`) της `PERSON` κληρονομούνται και στην κλάση `STUDENT`. Η διαφορά μεταξύ κατηγοριοποίησης και γενίκευσης δίδεται με την βοήθεια των προτάσεων: «`a shark is a fish`» (generalization) και «`Clyde is a fish`» (classification).
3. *Συνάθροιση* (*Aggregation*). Τα αντικείμενα αντιμετωπίζονται ως συναθροίσεις των *συστατικών* (*components*) ή των *μερών* (*parts*) από τα οποία αποτελούνται. Στη συνάθροιση είναι δυνατό να υπάρχει μεταβατικότητα, δηλαδή τα αντικείμενα να αποτελούν συστατικά συνάθροισης άλλων αντικειμένων. Ανάμεσα στα αντικείμενα και στα συστατικά τους, είναι δυνατόν να υπάρχουν εξαρτήσεις ως προς τον χρόνο ζωής και των δύο. Δηλαδή η διαγραφή ενός αντικειμένου να οδηγεί στην διαγραφή όλων των συστατικών του, και αντίστροφα. Τέλος, ένα συστατικό μπορεί να ανήκει *αποκλειστικά* (*exclusive*) σε μια συνάθροιση, αποκλείοντας την εμφάνιση του ως συστατικό ή μέρος άλλης συνάθροισης, ή να είναι *διαμοιραζόμενο* (*shared*), οπότε μπορεί να αποτελεί συστατικό πολλών διαφορετικών αντικειμένων.
4. *Contextualization*. Ο μηχανισμός εφαρμόζει κατάτμηση και ομαδοποίηση των περιγραφών που εισάγονται σε μια πληροφοριακή βάση, λαμβάνοντας υπόψη μια οπτική πλευρά του προβλήματος. Παράδειγμα contextualization για τις Βάσεις Δεδομένων αποτελούν οι *όψεις* (*views*), οι οποίες παρουσιάζουν μια μερική αλλά συνεπή πλευρά των περιεχομένων μιας βάσης σε διαφορετικούς χρήστες.
5. *Materialization*. Απεικονίζεται η συσχέτιση μεταξύ μιας *αφηρημένης* (*abstract*) κλάσης και μιας *συγκεκριμένης* (*concrete*) κλάσης. Π.χ., materialization εμφανίζεται μεταξύ των κλάσεων CarModel και Car, όπου η κλάση CarModel αποτελεί abstract κλάση καταχωρώντας πληροφορία σχετικά με κάθε αυτοκίνητο, όπως όνομα αυτοκινήτου, αριθμός πορτών κ.α.,

ενώ η Car αποτελεί concrete κλάση, όπου καταχωρείται πιο συγκεκριμένη πληροφορία, όπως ο ιδιοκτήτης του αυτοκινήτου, ο αριθμός κυκλοφορίας του κ.α.. Φορμαλιστικά, το materialization αποτελεί συνδυασμό των συσχετίσεων κατηγοριοποίηση και γενίκευση. Το materialization μελετάται πιο λεπτομερώς σε επόμενες παραγράφους.

6. *Κανονικοποίηση* (*Normalization*). Ο μηχανισμός της κανονικοποίησης μοντελοποιεί αρχικά τις πιο γενικές (τυπικές) οντότητες, καταστάσεις και γεγονότα, και στη συνέχεια μελετά πιο συγκεκριμένες (εξαιρετικές) καταστάσεις και πως αυτές μπορεί να αντιμετωπιστούν. Με αυτόν τον τρόπο απλοποιείται η μοντελοποίηση. Ο μηχανισμός αυτός επιτυγχάνει εάν υπάρχει κάποιος συστηματικός τρόπος εύρεσης των πιο ασυνήθιστων (εξεζητημένων) καταστάσεων, και κυρίως εάν υπάρχει τρόπος να καθοριστούν οι καταστάσεις αυτές ως σημειώσεις (footnotes/annotations) στο γενικό μοντέλο της απεικόνισης.
7. *Παραμετροποίηση* (*Parameterization*). Ο μηχανισμός αυτός εισάγει παραμέτρους στο μοντέλο απαιτήσεων κάνοντάς το χρηστικό και επαναχρησιμοποιήσιμο. Δηλαδή, με την εισαγωγή παραμέτρων σε ένα μοντέλο είναι δυνατό να χρησιμοποιηθεί σε διάφορες περιπτώσεις απαιτήσεων. Π.χ., κατασκευάζεται ένα μοντέλο σύμφωνα με κάποιες απαιτήσεις και δέχεται δυο παραμέτρους, `απόθεμα` και `καταναλωτής`, εκτελεί δυο πράξεις, `ζήτηση` και `κράτηση`, και έχει τον εξής περιορισμό, «μια `κράτηση` θα συμβεί για ένα διαθέσιμο `απόθεμα` εάν υπάρχει κάποιος `καταναλωτής που` το `ζητήσει`». Μπορούμε να χρησιμοποιήσουμε το παραπάνω μοντέλο για την περίπτωση μιας δανειστικής βιβλιοθήκης, θέτοντας στις παραμέτρους τις τιμές: `απόθεμα = βιβλίο` και `καταναλωτής = χρήστης βιβλιοθήκης`.

Για την ανάλυση, σχεδίαση και διαχείριση της απεικόνισης του μοντέλου στο επίπεδο υπολογιστή, χρησιμοποιούνται ορισμένα εργαλεία. Εργαλεία ανάλυσης θεωρούνται η *πιστοποίηση* (*verification*) και ο έλεγχος *εγκυρότητας* (*validation*). Με την εφαρμογή της πιστοποίησης ελέγχεται, με τη βοήθεια κάποιων κανόνων, η σύνταξη και η σημασιολογία των περιεχομένων της πληροφοριακής βάσης. Ο έλεγχος της εγκυρότητας, επικεντρώνεται στο κατά πόσο η βάση είναι συνεπής προς την

εφαρμογή. Εργαλεία σχεδίασης αποτελούν ορισμένοι κανόνες, οι οποίοι κατευθύνουν τη σχεδίαση μιας εφαρμογής και αποτιμούν την ορθότητα του αποτελέσματος της. Παράδειγμα τέτοιων κανόνων αποτελούν οι *κανονικές μορφές* (*normal forms*) των σχεσιακών βάσεων, με την βοήθεια των οποίων εξαλείφονται οι εμφανίσεις συγκεκριμένων ανωμαλιών που μπορεί να προκύψουν κατά τη εισαγωγή/διαγραφή/ανανέωση μιας εγγραφής. Τέλος τα εργαλεία για την διαχείριση ξεκινούν με μια «καλή» υλοποίηση, η οποία προσφέρει δυνατότητες, όπως κατασκευή, πρόσβαση και ανανέωση μιας πληροφοριακής βάσης. Επιπλέον η υλοποίηση θα πρέπει να είναι και «αποδοτική», δηλαδή να είναι *επεκτάσιμη*, με την έννοια ότι οι πρωταρχικές λειτουργίες της δεν επηρεάζονται από το μέγεθος της πληροφοριακής βάσης.

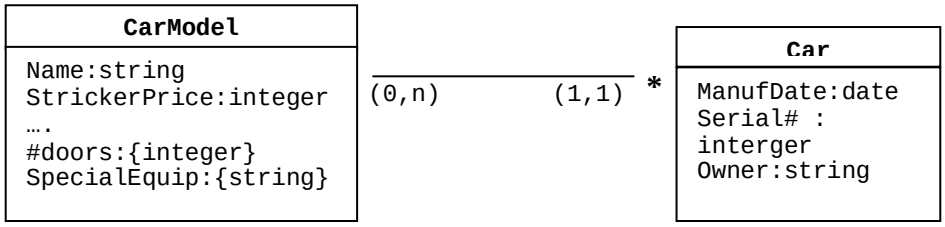
Στο [3] μελετώνται εφτά (7) βασικές *γενικές συσχετίσεις* (*generic relationships*) με βάση χαρακτηριστικά της κοινής σημασιολογία τους. Στη συνέχεια δίνονται ορισμένοι σημαντικοί κανόνες σχεδίασης μιας νέας γενική συσχέτισης. Τέλος, μελετάται η εφαρμογή μετακλάσεων στις συσχετίσεις. Παραδείγματα των γενικών συσχετίσεων παρουσιάζονται και στο [11].

Οι γενικές συσχετίσεις αποτελούν αφαιρετικούς μηχανισμούς μοντελοποίησης ορισμένων πτυχών φυσικών και κοινωνικών συστημάτων. Οι εφτά πιο σημαντικές γενικές συσχετίσεις, πλην της κατηγοριοποίησης και της γενίκευσης, είναι:

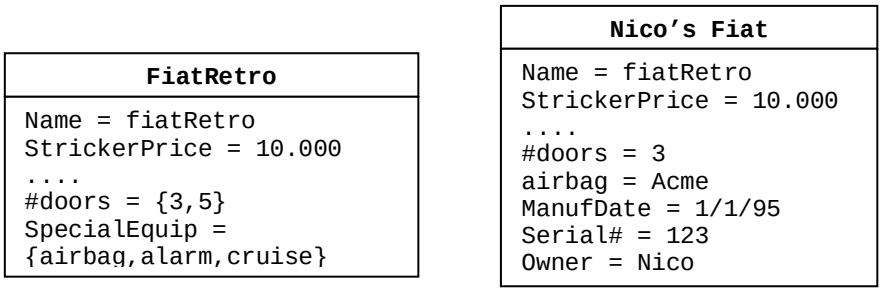
1. *Materialization*. Το materialization είναι μια δυαδική συσχέτιση ανάμεσα σε μια κλάση κατηγοριών και μια κλάση πιο συγκεκριμένων αντικειμένων. Δηλαδή, συσχετίζει μια *αφηρημένη* κλάση (*Abstract*) με μια πιο *συγκεκριμένη* (*Concrete*). Η σχηματική αναπαράσταση του materialization έχει τη μορφή:

   `Abstract —∗ Concrete.`

   Παράδειγμα materialization αποτελεί η περίπτωση καταγραφής του μοντέλου ενός αυτοκινήτου (abstract) και του αυτοκινήτου ενός ιδιοκτήτη (Concrete), με τα γνωρίσματα όπως αυτά φαίνονται στο σχήμα 2.1. Στιγμιότυπα των δύο κλάσεων παρουσιάζονται στο σχήμα 2.2



Σχήμα 2.1 Αναπαράσταση παραδείγματος materialization [σχ.2, [3]].



Σχήμα 2.2 Αναπαράσταση στιγμιότυπου του παραδείγματος materialization [σχ3, [3]].

Η *πληθικότητα* της abstract κλάσης είναι συνήθως `(0,n)`, ενώ της concrete `(1,1)`. Δηλαδή μια abstract κλάση μπορεί να έχει μηδέν (0) ή περισσότερες concrete κλάσεις ενώ μια concrete σχετίζεται ακριβώς με μια abstract κλάση. Υπάρχουν τρεις τύποι *προώθησης γνωρισμάτων* από την abstract στην concrete κλάση:

i. Τύπος 1: το στιγμιότυπο της concrete κλάσης κληρονομεί τις τιμές των πεδίων του στιγμιότυπου της abstract, π.χ., το πεδίο `name= fiatRetro`.
ii. Τύπος 2: το στιγμιότυπο της abstract έχει πλειότιμα γνωρίσματα. Μια τιμή εκ των οποίων προωθείται στο αντίστοιχο γνώρισμα της concrete κλάσης. Π.χ., στο στιγμιότυπο της abstract το πλειότιμο πεδίο έχει τιμές `#doors = {3,5}`, ενώ στο στιγμιότυπο της concrete έχει μια συγκεκριμένη τιμή του συνόλου, `#doors = 3`.
iii. Τύπος 3: Η abstract κλάση περιλαμβάνει κατηγορίες γνωρισμάτων (μεταγνωρίσματα), τα οποία για κάθε στιγμιότυπο της αντιστοιχίζονται σε διαφορετικό σύνολο στοιχείων. Το κάθε στιγμιότυπο της concrete

κλάσης περιλαμβάνει τόσα ξεχωριστά πεδία, όσο είναι το πλήθος των στοιχείων που εμφανίζονται στο αντίστοιχο στιγμιότυπο της abstract κλάσης. Π.χ., το πεδίο `SpecilaEquip` της abstract κλάσης, για το στιγμιότυπο `FiatRetro` αντιστοιχίζεται στο σύνολο `SpecialEquip = {airbag,alarm,cruise}`. Στο στιγμιότυπο της concrete, κάθε ένα στοιχείο του συνόλου `SpecialEquip` αποτελεί ξεχωριστό πεδίο και έχει συγκεκριμένη τιμή, π.χ., `airbag = Acme`.

Οι materializations μπορούν να συνδυαστούν και να δημιουργήσουν ιεραρχίες στις οποίες:

- μια concrete κλάση να αποτελεί abstract κλάση μιας άλλης materialization, π.χ., ένα `κώμικ` μπορεί να θεωρηθεί συγκεκριμένου τύπου `περιοδικό`, ενώ τα `περιοδικά` μπορεί να θεωρηθούν συγκεκριμένου τύπου `βιβλία`. Τα περιοδικά στην πρώτη περίπτωση θεωρούνται abstract κλάση, ενώ στη δεύτερη concrete.
- μια abstract κλάση μπορεί να συμμετέχει σε πολλές διαφορετικές materializations, π.χ., η abstract κλάση `Movie` μπορεί να γίνει materialize στις κλάσεις `DVD` και `VideoTape`.
- μια concrete κλάση μπορεί να αποτελεί materialization πολλών διαφορετικών abstract κλάσεων, π.χ., μια ταινία (`Movie`) μπορεί να προέρχεται είτε από ένα βιβλίο (`Novel`) είτε από ένα θεατρικό έργο (`Play`).

Οι εξαρτήσεις, ως προς το χρόνο ζωής, που παρουσιάζονται μεταξύ των κλάσεων βασίζονται στην πληθικότητα της κάθε κλάσης. Έτσι η διαγραφή ενός στιγμιότυπου της abstract κλάσης συνεπάγεται την διαγραφή του αντίστοιχου στιγμιότυπου της concrete κλάσης. Η διαγραφή ενός στιγμιότυπου της concrete κλάσης μπορεί να οδηγεί σε διαγραφή του αντίστοιχου στιγμιότυπου της abstract κλάσης μόνο όταν η abstract συμμετέχει στη συσχέτιση με βαθμό πληθικότητας τουλάχιστον 1.

Σημασιολογικά το materialization αποτελεί συνδυασμό των συσχετίσεων της γενίκευσης και της κατηγοριοποίησης. Η κατασκευή μια two-facet απεικόνισης καθώς και ο αναλυτικός τρόπος προώθησης των γνωρισμάτων μελετώνται αναλυτικά στο [2].

2. *Ownership*. Η ownership είναι μια δυαδική συσχέτιση μεταξύ μια κλάσης που απεικονίζει έναν *ιδιοκτήτη* (*owner*) και μιας κλάσης που απεικονίζει μια *ιδιοκτησία* (*property*) του. Π.χ., όταν ένα άτομο της κλάσης `PERSON` είναι ιδιοκτήτης ενός τραπεζικού λογαριασμού (`BankAcount`), η κλάση `PERSON` αντιστοιχίζεται στη `owner` κλάση, ενώ η κλάση `BankAccount` στη `property` κλάση της συσχέτισης ownership. Η σχηματική απεικόνιση της ownership συσχέτισης είναι η εξής:

   `Owner ←−−− Property.`

   Η πληθικότητα της owner κλάσης είναι `(0,n)` ενώ της property `(1,n)`, εφόσον ένας ιδιοκτήτης μπορεί να έχει καμία ή πολλές ιδιοκτησίες και μια ιδιοκτησία μπορεί να ανήκει τουλάχιστον σε έναν ιδιοκτήτη.

   Προώθηση τιμών μεταξύ των δύο κλάσεων παρατηρείται, όταν κάποια χαρακτηριστικά της ιδιοκτησία μπορούν να θεωρηθούν χαρακτηριστικά και του ιδιοκτήτη και αντίστροφα. Π.χ., η διεύθυνση που αναγράφεται στον τραπεζικό λογαριασμό ενός ατόμου μπορεί να θεωρηθεί και γενικά η διεύθυνση κατοικία του ιδιοκτήτη.

   Οι συσχετίσεις ownership μπορούν να συνδυαστούν δημιουργώντας ιεραρχίες, όπου:

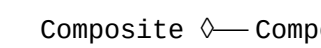
   i. μια κλάση property μιας ownership μπορεί να αποτελεί owner κλάση μιας άλλης ownership. Π.χ., ένα εργοστάσιο ανήκει σε μία εταιρία, η οποία ανήκει σε έναν όμιλο εταιριών.
   ii. μια owner κλάση μπορεί να έχει πολλές διαφορετικές properties.
   iii. μια property να έχει συμμετέχει σε πολλές διαφορετικές ownership συσχετίσεις, π.χ., ένα μαγαζί μπορεί να ανήκει σε μια εταιρία ή σε ένα άτομο.

iv. ισχύει η αναδρομικότητα, π.χ., μια εταιρία μπορεί να ανήκει σε πολλές εταιρίες.

Μια ownership μπορεί να είναι *αποκλειστική* (*exclusive*) ή *κοινή* (*joint*) ανάλογα με το αν μια ιδιοκτησία ανήκει μόνο σε έναν ή σε περισσότερους του ενός ιδιοκτήτες, αντίστοιχα. Στην περίπτωση που η ownership είναι *joint*, τότε είτε είναι *χωρίς δεσμεύσεις* (*free joint*), είτε έχει *καθορισμένα δικαιώματα* για κάθε owner (*percentage* ή *equal joint*).

Η διαγραφή μιας ιδιοκτησία μπορεί να οδηγήσει σε διαγραφή του αντίστοιχου ιδιοκτήτη. Π.χ., στην περίπτωση καταγραφής του ιδιοκτήτη ενός αυτοκινήτου (owner) και του αυτοκινήτου (property), η διαγραφή του αυτοκινήτου οδηγεί στη διαγραφή και του ιδιοκτήτη διότι δεν υπάρχει λόγος ύπαρξης της συγκεκριμένης πληροφορίας. Ανάλογα, εάν διαγραφεί ο μοναδικός ιδιοκτήτης μιας ιδιοκτησίας, τότε διαγράφεται και η ιδιοκτησία.

3. *Aggregation*. Το aggregation αποτελεί δυαδική συσχέτιση μεταξύ κλάσεων, όπου η μία αντιπροσωπεύει ένα *αντικείμενο* (*whole/composite*) και η άλλη τα *μέρη* ή τα *συστατικά* (*part/component*) από τα οποία αποτελείται. Η σχηματική αναπαράσταση του aggregation είναι η εξής:



Η πληθικότητα της composite κλάσης είναι συνήθως (1,n), ενώ της component ποικίλει.

Η προώθηση τιμών πραγματοποιείται τόσο από την πλευρά του composite όσο και από την πλευρά του component. π.χ., το σώμα του αυτοκινήτου αποτελεί μέρος του αυτοκινήτου, τότε το χρώμα του αυτοκινήτου αποτελεί και χρώμα του σώματος του αυτοκινήτου (προώθηση από το composite στο component), αλλά και ο αριθμός των πορτών που έχει το σώμα του αυτοκινήτου είναι και ο αριθμός πορτών που έχει όλο το αυτοκίνητο (προώθηση από το component στο composite).

Οι συσχετίσεις aggregation μπορούν να συνδυαστούν δημιουργώντας ιεραρχίες, όπου:

i. η component κλάση ενός aggregation μπορεί να αποτελεί composite κλάση ενός άλλου aggregation. Π.χ., ο τοίχος ενός δωματίου αποτελεί μέρος ενός δωματίου, το οποίο αποτελεί μέρος ενός σπιτιού.
ii. ένα component μπορεί να αποτελείται από πολλά composites.
iii. ένα composite μπορεί να συμμετέχει σε πολλά διαφορετικά aggregation, δηλαδή να αποτελεί μέρος πολλών components.

Γενικά η μεταβατικότητα δεν ισχύει στη συσχέτιση aggregation, π.χ., η μπάντα αποτελείται από μουσικούς, ενώ μέρος ενός μουσικού αποτελεί το χέρι του. Δεν μπορούμε να ισχυριστούμε ότι μια μπάντα αποτελείται από χέρια.

Ανάλογα με την πληθικότητα της κλάσης component, ένα aggregation μπορεί να είναι *αποκλειστικό* (*exclusive*), οπότε ένα component ανήκει μόνο σε ένα composite, ή *διαμοιραζόμενο* (*shared*), δηλαδή δεν υπάρχουν περιορισμοί.

Ανάμεσα στις δύο κλάσεις μπορεί να υπάρξουν εξαρτήσεις σε σχέση με τον χρόνο ζωής τους. Η *εξάρτηση μέρος-από το-όλο* (*part-to-whole dependency*) σημαίνει ότι η ύπαρξή του μέρους (part) εξαρτάται από την ύπαρξη του όλου (whole) και σε περίπτωση διαγραφής του whole διαγράφονται και όλα τα μέρη του. Ενώ όταν έχουμε *εξάρτηση όλο-από το-μέρος* (*whole-to-part dependency*) σημαίνει ότι η ύπαρξη του όλου εξαρτάται από την ύπαρξη του μέρους, και σε περίπτωση διαγραφής ενός μέρους διαγράφεται και το whole.

4. *Role Relationship.* Η role συσχετίζει μια *κλάση αντικειμένου* (*Object Class*) και μια *κλάση ρόλου* (*Role Class*), τον οποίον έχει το συγκεκριμένο αντικείμενο. Η Object Class ορίζει μόνιμες ιδιότητες του αντικειμένου, ενώ η Role Class ορίζει προσωρινές ιδιότητες που μπορεί να έχει ένα αντικείμενο σε μια ορισμένη περίοδο. Π.χ., συσχέτιση ρόλου εμφανίζεται ανάμεσα στις κλάσεις `PERSON` και `STUDENT`, όπου ένα άτομο έχει το ρόλο του μαθητή για κάποιο χρονικό διάστημα. Η διαφορά μεταξύ των συσχετίσεων ρόλου και

γενίκευσης, είναι ότι η συσχέτιση role απεικονίζει μεταβλητές ιδιότητες ενός αντικειμένου, ενώ η γενίκευση απεικονίζει μόνιμες, π.χ., όταν ένα άτομο είναι μαθητής, αυτό απεικονίζεται μέσω της συσχέτισης ρόλου, αλλά το γεγονός ότι το άτομο αυτό είναι γυναίκα θα πρέπει να απεικονιστεί μέσω γενίκευσης. Η σχηματική αναπαράσταση της συσχέτισης role είναι η εξής:

Role Class →○ Object Class.

Η πληθικότητα της Role Class είναι (1,1), δηλαδή κάθε ρόλος αντιστοιχίζεται σε ένα αντικείμενο, ενώ της Object Class είναι (0,n), δηλώνοντας ότι ένα αντικείμενο μπορεί να έχει κανέναν ή πολλών ρόλους.

Δεν υπάρχει κληρονομικότητα μεθόδων και γνωρισμάτων μεταξύ των κλάσεων.

Οι συσχετίσεις role μπορούν να συνδυαστούν σε ιεραρχίες, όπου:

i. η Role Class μιας συσχέτισης μπορεί να συμμετέχει σε άλλη συσχέτιση ως Object Class.
ii. μια Role Class μπορεί να σχετίζεται με πολλές κλάσεις αντικειμένου. Π.χ., το ρόλο του συμβούλου μπορεί να παίξει τόσο ένας καθηγητής όσο και ένας μαθητής.
iii. Η κλάση αντικειμένου μπορεί να σχετίζεται με διαφορετικές κλάσεις ρόλων. Π.χ., ένα αντικείμενο της κλάσης PERSON μπορεί να έχει τον ρόλο του ΦΟΙΤΗΤΗ αλλά και του ΕΡΓΑΖΟΜΕΝΟΥ, ταυτόχρονα.

Η role συσχέτιση δεν μπορεί να είναι αναδρομική, αν και μπορεί να είναι μεταβατική. Π.χ., ένα άτομο είναι διευθυντής, αλλά ο διευθυντής έχει και το ρόλο του εργαζομένου, τότε εφαρμόζοντας την μεταβατικότητα, το άτομο έχει και το ρόλο του εργαζομένου.

Η διαγραφή ενός αντικειμένου συνεπάγεται την διαγραφή και του ρόλου του. Η διαγραφή ενός ρόλου μπορεί να οδηγήσει στην διαγραφή ενός αντικειμένου

το οποίο συμμετέχει στη συσχέτιση με βαθμό πληθικότητας 1, δηλαδή έχει τουλάχιστον έναν ρόλο.

5. *Grouping*. Το grouping αποτελεί δυαδική συσχέτιση κατά την οποία μια συλλογή ενός συνόλου μελών θεωρείται ως ένα, ανώτερου επιπέδου, αντικείμενο. Ουσιαστικά το grouping ομαδοποιεί ένα σύνολο μελών σε ένα άλλο σύνολο, και μπορεί να θεωρηθεί ένα είδος aggregation. Η σχηματική αναπαράσταση του grouping είναι η εξής:

   `Member Class` ↠ `Set Class.`

   Η Set κλάση περιλαμβάνει ένα *set-determining* γνώρισμα, δηλαδή ένα γνώρισμα που χαρακτηρίζει και ομαδοποιεί το σύνολο των μελών. Επιπλέον περιλαμβάνει έναν αριθμό *set-describing* γνωρισμάτων, τα οποία χαρακτηρίζουν και διαφοροποιούν κάθε μέλος.

   Ισχύουν τα χαρακτηριστικά του aggregation όσον αφορά την πληθικότητα των κλάσεων, τη δημιουργία ιεραρχιών και την προώθηση τιμών ανάμεσα στις κλάσεις. Διαφοροποιείται από το aggregation στο ότι δεν υπάρχει εξάρτηση σε σχέση με το χρόνο ζωής ανάμεσα στις κλάσεις. Π.χ., όταν κάποιος διαγραφτεί από έναν όμιλο, δεν συνεπάγεται και τη διαγραφή του από τη Member κλάση. Παρόλα αυτά μπορεί να δημιουργηθούν εξαρτήσεις λόγω περιορισμών ακεραιότητας. Π.χ., εάν η Member κλάση `EMPLOYEE` συμμετέχει στο grouping με την Set κλάση `DEPARTMENT`, με βαθμό πληθικότητας `(1,1)`, τότε η διαγραφή του `DEPARTMENT` οδηγεί στη διαγραφή του `EMPLOYEE`.

   Η συμμετοχή της member κλάσης στο grouping μπορεί να είναι είτε μερική, οπότε ισχύει *μερική κάλυψη* (*partial covering*), είτε ολική με *ολική κάλυψη* (*complete covering*) του συνόλου των μελών. Ας θεωρήσουμε το παράδειγμα, όπου οι παίχτες τέννις ομαδοποιούνται σε έναν όμιλο. Εάν ισχύει ολική κάλυψη τότε κάθε παίχτης ανήκει στον όμιλο, αντίθετα όταν υπάρχει τουλαχιστον ένας παιχτης που δεν ανήκει στον όμιλο τότε ισχύει μερική κάλυψη του συνόλου των παιχτών.

6. *Viewpoint*. Το viewpoint αποτελεί δυαδική συσχέτιση μεταξύ μιας κλάσης και μιας όψης που αποκτά η κλάση, υπό το πρίσμα μια συγκεκριμένης οπτικής γωνίας. Στις Βάσεις Δεδομένων ο μηχανισμός της *όψης* (*view*) χρησιμοποιείται για την αναπαράσταση ορισμένων δεδομένων σε διαφορετικές ομάδες χρηστών. Οι όψεις χωρίζονται σε δυο κατηγορίες: *i) Object-generating*, όπου δημιουργείται ένα νέο αντικείμενο και *ii) Object-preserving*, όπου εξάγονται ήδη υπάρχοντα αντικείμενα από μια κλάση ή μια όψη. Η σχηματική αναπαράσταση μιας viewpoint είναι η εξής:

`Class ⟝⊙ View`.

Συνήθως, μια κλάση συμμετέχει με βαθμό πληθικότητας (`0,n`) όταν έχουμε object-generating views, ενώ με (`0,1`) βαθμό όταν έχουμε object-preserving views. Οι όψεις συμμετέχουν πάντα στη συσχέτιση με βαθμό (`1,1`).

Μια όψη κληρονομεί την δομή και τις μεθόδους από τις κλάσεις ή τις όψεις που καθορίζει, αν και μπορεί να είναι πολύ διαφορετική. Δηλαδή, μπορεί να έχει νέα γνωρίσματα/μεθόδους, να αποκρύψει κάποια άλλα ή και να έχει άλλο όνομα σε κάποιο γνώρισμα/μέθοδο. Αυτό που δεν μπορεί να αλλάξει είναι το πεδίο ορισμού ενός γνωρίσματος μιας κλάσης.

Οι όψεις μπορούν να συνδυαστούν σχηματίζοντας ιεραρχίες, όπου:

i. μια view ενός viewpoint μπορεί αποτελεί κλάση ενός άλλου viewpoint.
ii. Μια κλάση να ανήκει σε πολλά viewpoints, π.χ., η κλάση `EMPLOYEE` μπορεί να έχει δύο όψεις `MARRIED_EMP` και `SINGLE_EMP`.
iii. Μια όψη μπορεί να ανήκει σε πολλές κλάσεις, π.χ., η όψη `MARRIED` μπορεί να αφορά την κλάση `EMPLOYEE` αλλά και την κλάση `PROFESSOR`.

Υπάρχει εξάρτηση μεταξύ της κλάσης και της παραγόμενης όψης. Η εξάρτηση εμφανίζεται σε επίπεδο στιγμιότυπου, αλλά και στο χρόνο ζωής της κλάσης. Σε επίπεδο στιγμιότυπου ισχύει ότι, δύο διακριτά στιγμιότυπα μιας κλάσης (ή όψης) δεν μπορούν να μοιράζονται το ίδιο (να αντιστοιχίζονται στο ίδιο) στιγμιότυπο μιας παραγόμενης όψης. Π.χ., εάν θεωρήσουμε μια όψη,

η οποία δημιουργείται πάνω σε έναν πίνακα μιας βάσης, η όψη αυτή απεικονίζει τα περιεχόμενα του πίνακα κάθε χρονική στιγμή. Όταν το περιεχόμενο του πίνακα αλλάζει, τότε αλλάζει και το αποτέλεσμα της όψης. Σε ότι αφορά στην εξάρτηση της όψης σε επίπεδο χρόνου ζωής, η διαγραφή/αλλαγή ενός αντικειμένου μιας παράγουσας κλάσης ή όψης οδηγεί στην διαγραφή/αλλαγή του παραγόμενου αντικειμένου της όψης.

7. *Generation*. Το generation αποτελεί δυαδική συσχέτιση μεταξύ μιας *κλάσης εισόδου* (*Input Class*), η οποία απεικονίζει τα αντικείμενα εισόδου, και μιας *κλάσης εξόδου* (*Output Class*), η οποία απεικονίζει τα αντικείμενα που παράγονται ως αποτέλεσμα μιας διεργασίας. Η σχηματική αναπαράσταση του generation έχει τη μορφή:

   `Input Class` ⇒ `Output Class`.

   Κατά το generation, τα αντικείμενα εισόδου μπορούν είτε να διατηρηθούν, οπότε και έχουμε *παραγωγή* (*production*) ενός αντικειμένου εξόδου, είτε να καταναλωθούν, όπου έχουμε *μετατροπή* (*transformation*) των αντικειμένων εισόδου στο αντικείμενο της εξόδου.

   Η πληθικότητα της κλάσης είναι Input (`1,n`), που σημαίνει θα πρέπει να υπάρχει τουλάχιστον ένα αντικείμενο εισόδου, ενώ της Output κλάσης είναι (`0,n`), το οποίο δηλώνει ότι μπορεί να μην παραχθεί κανένα αντικείμενο (λόγω αποτυχίας) ή να παραχθούν πολλά.

   Οι τιμές ορισμένων γνωρισμάτων της Output κλάσης εξαρτώνται από τις τιμές των αντίστοιχων γνωρισμάτων της Input κλάσης. Π.χ., ας θεωρήσουμε την περίπτωση μιας `παραγγελίας` (Input Class), η οποία οδηγεί στην `παραγωγή` (generation) ενός `αντικειμένου` (Output Class). Τότε η `ημερομηνία` της παραγωγής του αντικειμένου εξαρτάται από την ημερομηνία της παραγγελίας.

   Οι generations μπορούν να συνδυαστούν σχηματίζοντας ιεραρχίες, όπου:

   i. οι Output κλάσεις ενός generation να αποτελούν Input κλάσεις άλλου generation.

ii. ισχύει η αναδρομική ιδιότητα, όπου ένα αντικείμενο εξόδου μπορεί να παράγεται από πολλά αντικείμενα εξόδου, και αντίστοιχα ένα αντικείμενο εισόδου μπορεί να παράγει πολλά αντικείμενα εξόδου.

Δεν υπάρχει εξάρτηση των αντικειμένων εξόδου με τα αντικείμενα εισόδου, διότι τα αντικειμένων εξόδου είναι *νέα αυτόνομα* προϊόντα.

Μπορούμε να έχουμε *αναστρέψιμη παραγωγή* (*reversible generation*), όταν δεν έχει γίνει καμία αλλαγή που να εμποδίζει την αναστροφή της generation, ενώ όταν δεν μπορούμε να ξαναπάρουμε τα προϊόντα εισόδου από τα προϊόντα εξίσου έχουμε *μη αναστρέψιμη παραγωγή* (*irreversible generation*).

Για την κατασκευή μιας νέας γενικής συσχέτισης θα πρέπει να εφαρμοστούν τα παρακάτω βήματα:

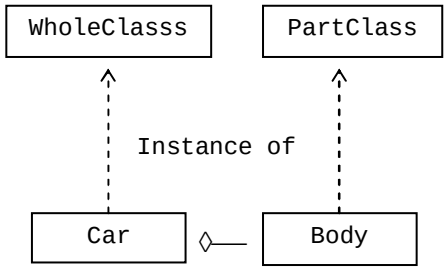
1. <u>Καθορισμός της συσχέτισης:</u> Τα βασικά δομικά στοιχεία των αφαιρετικών μοντέλων είναι οι *οντότητες* (ή τύποι, κλάσεις), οι οποίες αναπαριστούν τα σημαντικά πράγματα στο πεδίο της εφαρμογής, και οι *συσχετίσεις*, οι οποίες αναπαριστούν τις σχέσεις που αναπτύσσονται μεταξύ αυτών των πραγμάτων. Ο καθορισμός των οντοτήτων από τον πραγματικό κόσμο είναι πολύ εύκολος, εν αντιθέσει με τον καθορισμό των συσχετίσεων που υπάρχουν μεταξύ τους. Οι γενικές συσχετίσεις που προαναφέρθηκαν είναι ανεξάρτητες των εφαρμογών δίνοντας τους μεγάλη ευελιξία. Πρακτικά, για την επιλογή της συσχέτισης, που μοντελοποιεί καλύτερα μια σχέση, ο σχεδιαστής θα πρέπει να επιλέξει: *i) μια νέα ad hoc συσχέτισης*, δηλαδή την κατασκευή μιας νέας συσχέτισης η οποία αντιστοιχίζεται στο συγκεκριμένο πρόβλημα, *ii)* μια συσχέτιση που προκύπτει από μια γενική συσχέτιση, ή *iii)* μια συγκεκριμένη συσχέτιση που προκύπτει από μια νέα γενική συσχέτιση που έχει προσαρμοστεί στο πεδίο της εφαρμογής. Στις περισσότερες περιπτώσεις θα πρέπει να προτιμάται η *ii)* επιλογή.
2. <u>Ορισμός:</u> Μια νέα συσχέτιση που έχει καθοριστεί, θα πρέπει να οριστεί καλά, να είναι πλήρως κατανοητή, να ανταποκρίνεται σε ένα σημαντικό αριθμό από στιγμιότυπα στο πεδίο της εφαρμογής, η σημασιολογία της να είναι φορμαλιστικά ορισμένη και να συσχετίζεται με τον κατάλληλο *γραφικό*

συμβολισμό (*graphical notation*). Η διαισθητική σημασιολογία έχει να κάνει με τον σκοπό της συσχέτισης ως προς την εφαρμογή. Με την τυπική σημασιολογία ορίζονται τα χαρακτηριστικά της συσχέτισης (όπως πληθικότητα, σύνθεση,…). Ένας γραφικός συμβολισμός μιας συσχέτισης περιλαμβάνει έναν συμβολισμό για τις συμμετέχοντες κλάσεις και για την ίδια τη συσχέτιση.

3. Σύγκριση: Θα πρέπει να πραγματοποιηθεί σύγκριση με παρόμοιες υπάρχουσες συσχετίσεις, κατά την οποία να τονίζονται τόσο οι διαφορές όσο και οι ομοιότητες τους.

Οι *μετακλάσεις* (*metaclasses*) παίζουν τον ίδιο ρόλο με τις κλάσεις σε ένα μοντέλο κλάσης/στιγμιότυπου, δηλαδή περιγράφουν τη δομή και τη συμπεριφορά των κλάσεων, μέσω γνωρισμάτων και μεθόδων. Ορίζοντας μετακλάσεις, οι κλάσεις των γενικών συσχετίσεων αποτελούν στιγμιότυπα των μετακλάσεων, κληρονομώντας χαρακτηριστικά τους. Η σημασιολογία των συσχετίσεων ορίζεται μέσω της δομής των μετακλάσεων. Οι μετακλάσεις περιλαμβάνουν χαρακτηριστικά και παραμέτρους, στα οποία όταν δοθούν συγκεκριμένες τιμές (ανάλογα με τη συσχέτιση που αναφερόμαστε) δηλώνεται η αντίστοιχη σημασιολογία της συσχέτισης. Για την υλοποίηση των γενικών συσχετίσεων υπάρχουν τρεις προσεγγίσεις κατασκευής μετακλάσεων:

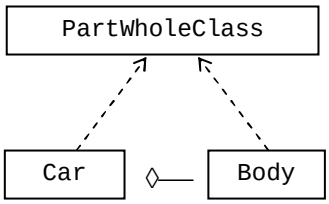
1. *Προσέγγιση δύο μετακλάσεων* (*Two-metaclass approach*): ορίζεται μία μετακλάση για κάθε μία κλάση (*ρόλο*) της συσχέτισης. Π.χ., για τη γενική συσχέτιση aggregation μεταξύ των κλάσεων `Car` και `Body`, η σχηματική αναπαράσταση αυτής της προσέγγισης δίνεται από το σχήμα 2.3.



Η κλάση `Car` κληρονομεί από τη μετακλάση της, `WholeClass`, γνωρίσματα και μεθόδους που χαρακτηρίζουν τη σημασιολογία ανάμεσα σε μια κλάση και το στιγμιότυπό της.

Σχήμα 2.3 Αναπαράσταση της προσέγγισης δύο μετακλάσεων [σχ. 16, [3]].

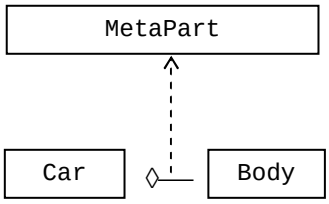
2. *Προσέγγιση μίας μετακλάσης* (*Single metaclass approach*): ορίζεται μια μετακλάση για τους δύο ρόλους της συσχέτισης. Η σχηματική αναπαράσταση της προσέγγισης για το παραπάνω παράδειγμα του aggregation δίνεται από το σχήμα 2.4.



Οι κλάσεις `Car` και `Body` κληρονομούν από τη μετακλάση της, `PartWholeClass`, γνωρίσματα και μεθόδους.

Σχήμα 2.4 Αναπαράσταση της προσέγγισης μιας μετακλάσης [σχ. 16, [3]].

3. *Προσέγγιση συσχέτισης - μετακλάσης* (*Relationship – Metaclass approach*): ορίζεται μια μετακλάση για τη συσχέτιση. Η σχηματική αναπαράσταση της προσέγγισης για το παραπάνω παράδειγμα παρουσιάζεται στο σχήμα 2.5.



Η συσχέτιση `aggregation` κληρονομεί χαρακτηριστικά και μεθόδους από την μετακλάση `MetaPart`.

Σχήμα 2.5 Αναπαράσταση της προσέγγισης μιας μετακλάσης [σχ. 16, [3]].

Στο [10] παρουσιάζεται ένα framework για την μοντελοποίηση των απαιτήσεων ενός συστήματος εφαρμογής, *RMF - Requirements Models Framework*. Σκοπός του RMF είναι η σύνθεση κάποιων αρχών, που είναι σημαντικές στη μοντελοποίηση των απαιτήσεων. Οι αρχές αυτές παρέχονται από τις απαντήσεις των ερωτήσεων: *i)* τι πληροφορία θα πρέπει να απεικονίζεται στο μοντέλο, και *ii)* πως θα πρέπει να είναι δομημένο το μοντέλο. Απαντώντας στις παραπάνω ερωτήσεις, επιλέχθηκε ότι το μοντέλο αποτελείται: *i)* από μια συλλογή *βασικών οντοτήτων* (*conceptual entities*) δημιουργώντας *κατηγορίες οντοτήτων* (*entity categories*), και *ii)* μια συλλογή *συσχετίσεων* (*properties*) που αναπτύσσονται μεταξύ των κατηγοριών, καθορίζοντας

και τον *τύπο των συσχετίσεων* (*property category*). Στο RMF χρησιμοποιούνται τρεις *αφαιρετικοί μηχανισμοί* (*abstract mechanisms*) για την απόδοση της γενικής πληροφορίας, απομακρύνοντας τις λεπτομέρειες. Οι μηχανισμοί αυτοί είναι:

1. Συνάθροιση (Aggregation): όπου μια οντότητα θεωρείται ως μια συλλογή από συστατικά (components) ή μέρη (parts).
2. Κατηγοριοποίηση (Classification): όπου μια νέα οντότητα, κλάση, μοιράζεται τα κοινά χαρακτηριστικά της κατηγορίας στην οποία ανήκει. Ο μηχανισμός αυτός εστιάζει στα κοινά χαρακτηριστικά των στιγμιότυπων μιας κλάσης, αποκρύπτοντας τις διαφορές τους.
3. Γενίκευση (Generalization): όπου απεικονίζονται γενικευμένα τα κοινά χαρακτηριστικά κάποιων κλάσεων. Με αυτό το μηχανισμό εστιάζεται το ενδιαφέρον μόνο στις ιδιότητες, που μια συλλογή κλάσεων έχει κοινές, χωρίς να εμφανίζει τις ατομικές τους διαφορές.

Στο RMF οι οντότητες κατηγοριοποιούνται σε *σύμβολα* (*tokens*), *κλάσεις* (*classes*) και *μετακλάσεις* (*metaclasses*). Τα tokens αποτελούν στιγμιότυπα των κλάσεων, ενώ οι κλάσεις αποτελούν στιγμιότυπα των μετακλάσεων. Οι οντότητες σχετίζονται με άλλες οντότητες έχοντας κάποιες ιδιότητες (properties). Οι ιδιότητες αποτελούνται από τρία μέρη πληροφορίας, το *υποκείμενο* (*subject*), ένα *χαρακτηριστικό γνώρισμα* (*attribute*) του υποκειμένου, και μια *τιμή* (*value*) του γνωρίσματος. Μια ιδιότητα μπορεί να είναι *πραγματική* (*factual*) ή *προσδιοριστική* (*definitional*). Η πραγματική ιδιότητα δηλώνει συγκεκριμένες τιμές σε ένα token ή μια κλάση, π.χ., εάν θέλουμε να καταγράψουμε ότι όλα τα άτομα της κλάσης `PERSON` έχουν ηλικία `21` τότε η δήλωση έχει τύπο: `<PERSON, age, 21>`, αυτό σημαίνει ότι και ο `John` ως token της κλάσης `PERSON` έχει την ίδια τιμή σε αυτό το γνώρισμα, δηλαδή `<john, age 21>`. Η προσδιοριστική ιδιότητα καθορίζει τη γενική πληροφορία που πηγάζει από τα στιγμιότυπα, π.χ., η πληροφορία ότι κάθε στιγμιότυπο της κλάσης `PERSON` έχει ένα γνώρισμα ηλικία (`age`), η οποία έχει κάποια τιμή (`AGE_VALUE`), δηλώνεται ως εξής: `<PERSON, age, AGE_VALUE>`. Η προσδιοριστική ιδιότητα μιας κλάσης αποτελεί την γενίκευση της πραγματικής ιδιότητας ενός αντικειμένου της.

Μέχρι στιγμής εφαρμόστηκε ο μηχανισμός της κατηγοριοποίησης για την απεικόνιση της συσχέτισης μεταξύ των tokens και των κλάσεων, και μεταξύ κλάσεων και

μετακλάσεων. Επιπλέον χρησιμοποιήθηκε ο μηχανισμός της συνάθροισης για την ομαδοποίηση όλων των ιδιοτήτων μιας οντότητας. Για την υποστήριξη του μηχανισμού γενίκευσης (generalization) εισάγεται μια νέα συσχέτιση, η *υποκλάση*, η οποία μπορεί να εφαρμοστεί ανάμεσα σε δύο κλάσεις ή μετακλάσεις. Τα χαρακτηριστικά της γενίκευσης είναι τα εξής: *i)* το στιγμιότυπο της υποκλάσης αποτελεί πάντα στιγμιότυπο της και της γενικής κλάσης, και *ii)* όλες οι προσδιοριστικές ιδιότητες της γενικής κλάσης κληρονομούνται στην υποκλάση. Η γενική ιδέα της γενίκευσης είναι ότι η δόμηση του μοντέλου ξεκινάει μοντελοποιώντας τις πιο γενικές κλάσεις και στη συνέχεια τις πιο εξειδικευμένες (υποκλάσεις).

Γενικά, το RMF απεικονίζει το περιβάλλον σε σχέση με το χρόνο, στον οποίον λαμβάνει χώρα η μοντελοποίηση. Δηλαδή ποιες οντότητες υπάρχουν και ποιες συσχετίσεις παρουσιάζονται μεταξύ τους την δεδομένη χρονική στιγμή. Στην εργασία δεν παρουσιάζεται ο τρόπος ορισμού και καταχώρησης των σχετικών πεδίων, οντότητες και συσχετίσεις, της μελέτης.

Στο [25], παρουσιάζεται ένα τυπικό μοντέλο ενός πληροφοριακού συστήματος, λαμβάνοντας υπόψη τρεις βασικούς σκοπούς: *i)* τον ορισμό ενός συνόλου βασικών στοιχείων, το οποίο μπορεί να χρησιμοποιηθεί για τον καθορισμό της δομής και της συμπεριφοράς ενός πληροφοριακού συστήματος, *ii)* την καλύτερη κατανόηση των στατικών και δυναμικών ιδιοτήτων ενός πληροφοριακού συστήματος, και *iii)* τη δυνατότητα δημιουργίας προβλέψεων για το πληροφοριακό σύστημα, οι οποίες βασίζονται στις ιδιότητες του. Η προσέγγιση του μοντέλου βασίζεται στην θεωρία της οντολογίας, η οποία είναι μια τροποποίηση και επέκταση της θεωρίας του Bunge.

Για την ανάλυση των στατικών και δυναμικών ιδιοτήτων των πληροφοριακών συστημάτων, παρουσιάζονται τα βασικά οντολογικά χαρακτηριστικά, τα οποία περιγράφονται με τυπικό μαθηματικό τρόπο. Το βασικότερο χαρακτηριστικό του μοντέλου που παρουσιάζεται είναι ένα *πράγμα* (*thing*). Σε κάθε thing αντιστοιχεί ένας *χώρος δυνατών καταστάσεων* (*possible state space*), ο οποίος αντιπροσωπεύει όλες τις πιθανές τιμές που μπορούν να λάβουν οι μεταβλητές των γνωρισμάτων του thing. Οι

τιμές των γνωρισμάτων μπορεί να περιορίζονται από κάποιους κανόνες – φυσικούς ή/και τεχνικούς–, οι οποίοι καθορίζουν αν μια κατάσταση είναι *νόμιμη* (*lawful*), ορίζοντας με αυτό τον τρόπο το *χώρο νόμιμων καταστάσεων* (*lawful state space*) του thing. Όταν μία τουλάχιστον από τις ιδιότητες (γνωρίσματα) ενός thing αλλάξει τιμή συντελείται ένα *γεγονός* (*event*). Και σε αυτή την περίπτωση όλα γεγονότα δεν είναι νόμιμα. Οπότε ορίζεται η *νόμιμη μετάβαση* (*lawful transformation*), ως το σύνολο όλων το μεταβάσεων από έναν χώρο νόμιμων καταστάσεων στον εαυτό του, και ένα *νόμιμο γεγονός* (*lawful event*), ως ένα διατεταγμένο ζεύγος καταστάσεων, οι οποίες ανήκουν στο χώρο νόμιμων καταστάσεων και η μετάβαση από την μια κατάσταση στην άλλη πραγματοποιείται μέσω μιας νόμιμης μετάβασης. Οι αλλαγές της κατάστασης ενός thing δημιουργεί ένα *ιστορικό* (*history*) του thing. Με την βοήθεια του ιστορικού καθορίζεται πότε δύο things είναι ανεξάρτητα μεταξύ τους (έχουν ανεξάρτητα ιστορικά) και πότε *εξαρτώμενα* (*coupled*) το ένα από το άλλο. Με την βοήθεια των ορισμών του thing και του coupling, ορίζεται το *σύστημα* (*system*). Ένα σύστημα είναι ένα σύνολο από things όπου κάθε thing συνδέεται με τουλάχιστον ένα άλλο thing του συνόλου, και επιπλέον είναι δυνατή η διαμέριση του συνόλου ώστε οι δύο διαμερίσεις να έχουν ανεξάρτητα ιστορικά. Για τον ορισμό του *υποσυστήματος* (*subsystem*) είναι απαραίτητη η εισαγωγή των ορισμών: i) *σύνθεση* (*composition*), ii) *περιβάλλον* (*environment*) και iii) *δομή* (*structure*). Σύνθεση είναι το σύνολο των things που υπάρχουν στο σύστημα την δεδομένη χρονική στιγμή, περιβάλλον είναι το σύνολο των στοιχείων που δεν ανήκουν στο σύστημα αλλά επιδρούν σε αυτό, ενώ δομή είναι το σύνολο των δεσμών που υπάρχουν μεταξύ της σύνθεσης και του περιβάλλοντος του συστήματος. Το υποσύστημα ορίζεται, τώρα, ως το υποσύνολο των συνθέσεων και των δομών ενός συστήματος, ενώ το περιβάλλον του υποσυνόλου αποτελείται από στοιχεία που ανήκουν στη σύνθεση ή/και στο περιβάλλον του αρχικού συνόλου.

Ένα *συστατικό εισόδου* (*input component*) είναι ένα thing του συστήματος στο οποίο επιδρά ένα thing του περιβάλλοντος, προκαλώντας ένα *εξωτερικό γεγονός* (*external event*), μεταβάλλοντας την κατάσταση του. Τα *εσωτερικά γεγονότα* (*internals events*) είναι οι μεταβολές των άλλων συστατικών του συστήματος, τα οποία προκαλούνται από το εξωτερικό γεγονός. Αντίστοιχα, *συστατικό εξόδου* (*output component*) είναι

κάποιο thing, το οποίο επιδρά με το περιβάλλον προκαλώντας αλλαγή της καταστάσεως του.

Τέλος, ορίζεται η έννοια της *αποσύνθεσης* (*decomposition)* αλλά και της «*καλής» αποσύνθεσης* (“*good” decomposition*) ενός συστήματος. Η αποσύνθεση είναι ένα σύνολο από υποσυστήματα τα οποία έχουν τα εξής χαρακτηριστικά: *i)* κάθε στοιχείο της σύνθεσης του συστήματος περιέχεται σε τουλάχιστον ένα υποσύστημα, *ii)* η διαφορά της ένωσης των περιβαλλόντων των υποσυστημάτων και της σύνθεσης του συστήματος δίνει το περιβάλλον του συστήματος και *iii)* κάθε στοιχείο της δομής του συστήματος περιέχεται σε τουλάχιστον ένα από τα υποσυστήματα της αποσύνθεσης. Με την βοήθεια της αποσύνθεσης δημιουργείται μια *δομή διαστρωμάτωσης* (*level structure*) η οποία παρουσιάζει το σύστημα από την πιο αφαιρετική του μορφή έως την πιο λεπτομερειακή. Εάν το σύστημα αλλάξει κατάσταση, γνωρίζουμε ότι τουλάχιστον ένα από τα υποσυστήματα της αποσύνθεσής του άλλαξε κατάσταση προωθώντας την αλλαγή αυτή προς τα πάνω στρώματα της δομής. Αντίστοιχα αν κάποια αλλαγή παρατηρηθεί στο ανώτερο επίπεδο της δομής (στο επίπεδο του συστήματος), αυτή η αλλαγή «επιβάλλεται» και στα άλλα, πιο λεπτομερή, επίπεδα της δομής. Κάθε κατάσταση του συστήματος αντιστοιχίζεται τουλάχιστον σε μία κατάσταση του κάθε υποσυστήματος της αποσύνθεσής του. Επειδή μια κατάσταση του συστήματος μπορεί να αντιστοιχίζεται σε δύο ή και περισσότερες καταστάσεις του κάθε υποσυστήματος, αυτό σημαίνει ότι δεν συντελούν όλες οι αλλαγές των καταστάσεων των υποσυστήματος σε αλλαγή της κατάστασης του συστήματος. Η αποσύνθεση πραγματοποιείται για δύο βασικούς σκοπούς: 1) αποτελεί τη βάση για την κατανόηση των πολύπλοκων συστημάτων και 2) είναι βασικό χαρακτηριστικό της σχεδίασης ενός συστήματος. Μια «καλή» αποσύνθεση δημιουργείται όταν τα γεγονότα που προκαλούνται σε ένα υποσύστημα προκαλούν τις αναμενόμενες (προβλεπόμενες) αλλαγές στις καταστάσεις των άλλων υποσυστημάτων. Π.χ., τα εξωτερικά γεγονότα, τα οποία προκαλούνται από το περιβάλλον δεν είναι όλα προβλέψιμα, εν αντιθέσει με τα εσωτερικά γεγονότα που σε κάθε περίπτωση θα πρέπει να είναι όλα αναμενόμενα και καλώς ορισμένα.

Με την παρουσίαση του παραπάνω μοντέλου, πραγματοποιείται ένα βήμα προς την κατανόηση κατασκευών της επιστήμης των υπολογιστών και της επιστήμης της

πληροφορίας, αναλύοντας τη σημασία των βασικών χαρακτηριστικών (όπως δεδομένα, προγράμματα, συστήματα πραγματικού χρόνου, κ.α.) Η ανάλυση αυτή παρουσιάζεται να είναι ανεξάρτητη της εφαρμογής. Επιπλέον μέσω της διαδικασίας της αποσύνθεσης, είναι δυνατός ο έλεγχος της αποδοτικότητας και αποτελεσματικότητας διαφόρων συστημάτων.

2.2. Φορμαλισμοί (Formalisms)

Στο [4], παρουσιάζονται τρόποι για την αναπαράσταση της δομής και της συμπεριφοράς κάθε προτύπου, που εμφανίζεται στις εφαρμογές και στις συνθέσεις προτύπων σχεδίασης. Με τις προτάσεις αυτής της εργασίας είναι δυνατή η διατήρηση της πληροφορίας, που σχετίζεται με τα πρότυπα, σε κλάσεις και διαγράμματα. Με αυτό τον τρόπο τα πρότυπα είναι αναγνωρίσιμα στις εφαρμογές και στις συνθέσεις τους με άλλα.

Για τη σχεδίαση προτύπων έχει καθιερωθεί η σχηματική αναπαράσταση με χρήση της UML. Έχει παρατηρηθεί όμως ότι με την χρήση αυτής της αναπαράστασης, η πληροφορία σχετικά με τα πρότυπα χάνεται, κυρίως όταν πραγματοποιείται σύνθεση προτύπων. Η αλλαγή των ονομάτων των κλάσεων, των λειτουργιών και των γνωρισμάτων ενός προτύπου, καθιστά δύσκολη την δουλεία του σχεδιαστή ο οποίος πρέπει να λάβει κάποιες αποφάσεις σχετικά με την απόδοση του προτύπου, εφόσον ο ρόλος του προτύπου ουσιαστικά έχει χαθεί.

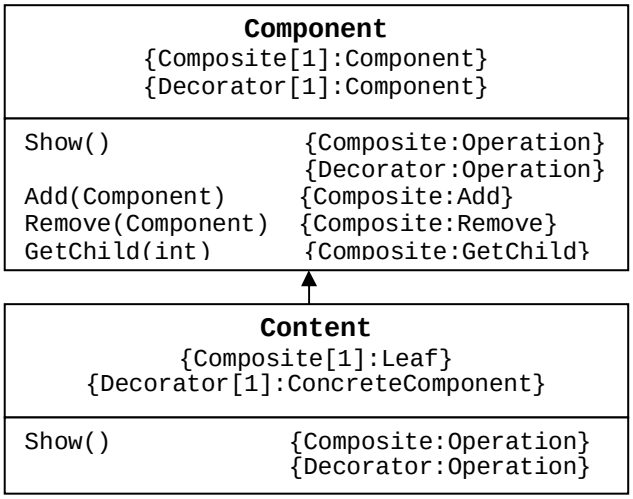
Έχουν παρουσιαστεί αρκετές προτάσεις σχηματικής αναπαράστασης, οι οποίες προσπαθούν να διαχωρίσουν κάθε πρότυπο σε μία εφαρμογή.

- *Venn diagram-style pattern annotation*. Η αναπαράσταση αυτή σκιάζει κάθε πρότυπο με μία απόχρωση του γκρι. Το πρόβλημα που παρουσιάζει αυτή η πρόταση έχει δυο σκέλη. Πρώτον, καθώς αυξάνεται ο αριθμός των προτύπων στα οποία συμμετέχει μια κλάση, είναι δύσκολη η σκίαση των διαφορετικών προτύπων καθώς υπάρχει υπερκάλυψη των περιοχών σκιαγράφησης. Δεύτερον, είναι δύσκολος ο καθορισμός των ρόλων που παίζει κάθε στοιχείο στα διάφορα πρότυπα που εμφανίζονται.

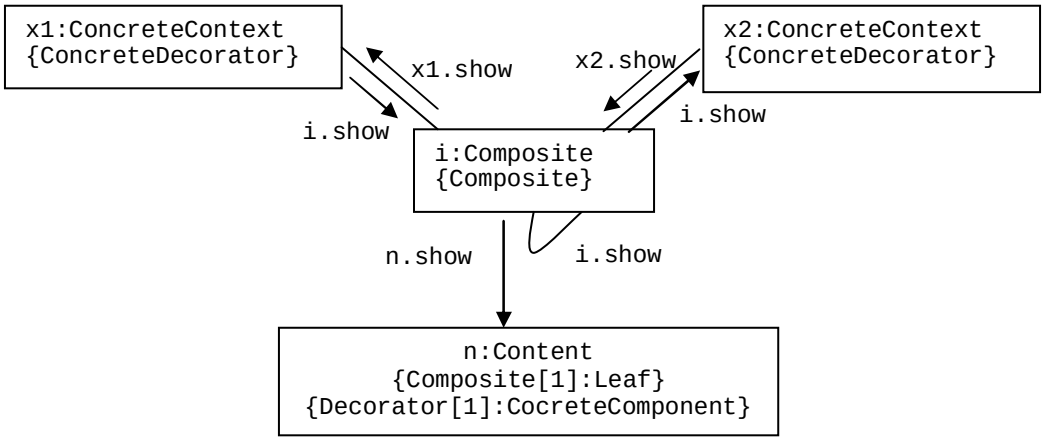
- *Dotted bounding pattern annotation.* Αντί να σκιαγραφείται το κάθε πρότυπο, χρησιμοποιούνται περιγράμματα διακεκομμένων γραμμών που τα οριοθετούν. Με αυτόν τον τρόπο επιλύεται το πρόβλημα της υπερκάλυψης των περιοχών σκιαγράφησης, αλλά εκκρεμεί ακόμα το δεύτερο πρόβλημα με τον καθορισμό των ρόλων.
- *UML collaboration notation.* Σε αυτή την πρόταση χρησιμοποιούνται διακεκομμένες ελλείψεις με τα ονόματα των προτύπων, και διακεκομμένες γραμμές για τη συσχέτιση των κλάσεων με τα αντίστοιχα πρότυπα. Αν και αυτή η πρόταση είναι μια βελτίωση των δύο πρώτων, παρουσιάζει άλλα προβλήματα. Η πληροφορία σχετικά με τις κλάσεις και τους ρόλους τους στα πρότυπα, σε συνδυασμό με τη σχηματική αναπαράσταση της δομής της εφαρμογής έχει ως αποτέλεσμα ένα πολύπλοκο και δύσκολο στην κατανόηση σχεδιάγραμμα. Επιπλέον, αν και ο ρόλος της κάθε κλάσης έχει αναπαρασταθεί στο σχεδιάγραμμα, δεν έχει αναπαρασταθεί ο πιθανός ρόλος των λειτουργιών (ή των γνωρισμάτων).
- *Pattern: role annotations*. Ο Gamma πρότεινε αυτή την αναπαράσταση κατά την οποία σε κάθε κλάση αντιστοιχίζεται μία ετικέτα σκιαγραφημένη, στην οποία αναφέρεται το πρότυπο και τον ρόλο της κλάσης σε αυτό. Και σε αυτή την αναπαράσταση παρουσιάζεται το πρόβλημα με την απόδοση της σκιαγράφησης ειδικά κατά την εκτύπωση, αλλά και το ότι δεν αναπαρίστανται οι ρόλοι των λειτουργιών. Επιπλέον, ο χώρος που καταλαμβάνει η σχηματική αναπαράσταση είναι αρκετά μεγάλος. Τέλος, είναι αδύνατος ο διαχωρισμός των διαφορετικών στιγμιότυπων του ίδιου προτύπου.

Η αναπαράσταση των ρόλων των λειτουργιών και των γνωρισμάτων κατά τη σχεδίαση είναι πολύ σημαντική. Για παράδειγμα, η ακριβής αναπαράσταση των λειτουργιών και των γνωρισμάτων του κλειδιού δεν βοηθάει μόνο στην εφαρμογή, εφόσον το πρότυπο επιβάλλει ορισμένους περιορισμούς μέσω των συσχετίσεων μεταξύ λειτουργιών και γνωρισμάτων, αλλά και στην αναγνώριση του προτύπου.
Για την αναπαράσταση τόσο των ρόλων της κλάσης σε ένα πρότυπο, όσο και των ρόλων των λειτουργιών και των γνωρισμάτων, προτείνεται η επέκταση της UML. Η UML παρέχει τρεις μηχανισμούς επέκτασης: *στερεότυπα* (*stereotypes*), *τιμές σε ετικέτες* (*tagged vales*) και *περιορισμοί* (*constrains*). Τα tagged values είναι της μορφής:

{tag = value}, όπου το tag είναι τύπου συμβολοσειρά ενώ το value μπορεί να είναι συμβολοσειρά, Boolean, ακέραιος. Με τα tagged values επεκτείνονται οι ιδιότητες ενός στοιχείου μοντελοποίησης με κάποια συγκεκριμένη πληροφορία. Χρησιμοποιώντας τον μηχανισμό tagged values, προτείνεται η ιδέα δημιουργίας ενός *προτύπου επεξήγησης με χρήση ετικετών* (*tagged pattern annotation*). Με βάση το προτεινόμενο πρότυπο, για κάθε κλάση δημιουργούνται νέες τιμές σε ετικέτες, με τις οποίες καταχωρείται πληροφορία σχετικά με το πρότυπο, το στιγμιότυπο, το όνομα με το οποίο συμμετέχει η κλάση στο πρότυπο, καθώς επίσης οι λειτουργίες και τα γνωρίσματά της με τα οποία συμμετέχει. Φορμαλιστικά το νέο tag δίδεται μέσω του τύπου: `pattern[instance]:role`. Με αυτόν τον τρόπο έχει αποδοθεί η δομή της κάθε κλάσης, Σχήμα 2.6. Επιπλέον, με παρόμοιο τρόπο, αναπαρίσταται και η συμπεριφορά κάθε προτύπου σχεδίασης, Σχήμα 2.7. Ο μόνος περιορισμός της πρότασης αυτής είναι ο δύσκολος καθορισμός των προτύπων στο σχέδιο, αυτό όμως μπορεί να επιλυθεί εν μέρει με την χρήση του dotted bounding notation. Οι κλάσεις Component και Content, στο Σχήμα 2.6, έχουν δυο διακριτούς ρόλους στην κατασκευή των προτύπων Decorator και Composite. Και οι δύο ρόλοι της κάθε κλάσης δίδονται από τις τιμές των ετικετών.



Σχήμα 2.6 Αναπαράσταση δομής των κλάσεων `Component` και `Content` με την μέθοδο *tagged pattern annotation* [σχ. 5, [4]].



Σχήμα 2.7 Αναπαράσταση σύνθεσης των κλάσεων `Component` και `Content` με την μέθοδο tagged pattern annotation [σχ. 6, [4]].

Στο σχήμα 2.7 φαίνεται η συμπεριφορά της σύνθεσης των προτύπων Composite και Decorator. Τα `x1`, `x2` είναι αντικείμενα της κλάσης ConcreteDecorator, η οποία παίζει το ρόλο ConcreteDecorator στο πρότυπο Decorator. Το στιγμιότυπο `i` της κλάσης Composite παίζει τόσο το ρόλο της Composite κλάσης στο Composite, όσο και της Decorator στο Decorator. Τέλος, το `n` είναι ένα αντικείμενο της κλάσης Content η οποία συμμετέχει στο πρώτο στιγμιότυπο του προτύπου Composite ως φύλλο, αλλά και στο πρώτο στιγμιότυπο του Decorator ως κλάση ConcreteComponent.

Συμπερασματικά αναφέρεται ότι η μεγάλη σημασία που έχει ο διαχωρισμός των προτύπων κατά την αναπαράσταση της δομής και της συμπεριφοράς τους, έγκειται στον ορισμό των προτύπων σχεδίασης. Με τα πρότυπα, επαναχρησιμοποιείται η προηγούμενη πείρα της σχεδίασης, οι σχεδιαστές μπορούν να επικοινωνούν καλύτερα, είναι εμφανής οι αποφάσεις που έχουν παρθεί κατά τη σχεδίαση, καταγράφονται οι *ισορροπία μεταξύ μειονεκτημάτων και πλεονεκτημάτων* (*tradeoffs*), καθώς επίσης και οι σχεδιαστικές εναλλακτικές λύσεις των διάφορων εφαρμογών. Όλη αυτή η πληροφορία που περιέχουν τα πρότυπα πρέπει να διατηρηθεί, αυτός ακριβώς είναι και ο σκοπός αυτής της εργασίας.

Στο [6] παρουσιάζεται η δημιουργία μιας πρακτικής τεχνικής καθορισμού προτύπων, η οποία υποστηρίζει την χρήση προτύπων κατά την διάρκεια της σχεδίασης. Τα πιο δημοφιλή πρότυπα σχεδίασης αποτελούνται από δομημένες, μη τυπικές περιγραφές

λύσεων προβλημάτων. Οι μη τυπικές περιγραφές αποδείχθηκαν αποτελεσματικό μέσο επικοινωνίας μεταξύ των κατασκευαστών, αλλά με την απουσία της τυπικότητας δεν μπορούν να υποστηρίξουν την επακριβή χρήση των προτύπων. Η τυπική περιγραφή των προτύπων πραγματοποιείται με την βοήθεια μαθηματικών τρόπων αναπαράστασης. Η πρόταση που περιγράφεται εδώ υποστηρίζει την επακριβή περιγραφή των προτύπων λύσεων εκφρασμένη σε UML. Ειδικότερα, οι περιγραφές που δημιουργούνται από την προτεινόμενη τεχνική είναι *μεταμοντέλα* (*metamodels*), τα οποία χαρακτηρίζουν τα UML μοντέλα σχεδίασης των προτύπων λύσεων. Ένα μεταμοντέλο ενός προτύπου κατασκευάζεται από τον καθορισμό του UML μεταμοντέλου.

Ένα πρότυπο λύσης περιγράφεται από *δομικής πλευράς* (*structured view*), η οποία αποδίδεται μέσω ενός *διαγράμματος κλάσης* (*class diagram*), και από *λειτουργικής πλευράς* (*interaction view*), η οποία αποδίδεται μέσω των *διαγραμμάτων ακολουθίας* (*sequence diagrams*). Ένα UML class diagram περιγράφει τους *ταξινομητές* (*classifiers*) δηλαδή τις κλάσεις και τις διεπαφές, και τις *συσχετίσεις* (*relationships*) ανάμεσα στους class ταξινομητές. Μια κλάση είναι ένας ταξινομητής, ο οποίος χαρακτηρίζει μια οικογένεια αντικειμένων με όρους κοινών γνωρισμάτων και λειτουργιών. Οι *σχέσεις* (*associations*) μεταξύ των κλάσεων περιγράφουν τις συνδέσεις μεταξύ των *κλάσεων αντικειμένων* (*class objects*). Ένα διάγραμμα ακολουθίας περιγράφει πως τα στιγμιότυπα αλληλεπιδρούν για την επίτευξη μιας αποστολής. Η ενδοεπικοινωνία εκφράζεται με τους όρους *χρόνοι ζωής* (*lifelines*) και *μηνύματα* (*messages*). Ένα lifeline συμμετέχει σε μια αλληλεπίδραση με την μορφή κλάσεων αντικειμένων, ενώ ένα message είναι καθορισμός μιας κλάσης, ο οποίος «περνάει» μεταξύ δυο αντικειμένων.

Το UML μεταμοντέλο χαρακτηρίζει έγκυρα UML μοντέλα και αποτελείται από ένα class diagram και ένα σύνολο από καλά καθορισμένους *κανόνες*. Στο μεταμοντέλο επίσης συμπεριλαμβάνονται και οι μη τυπικές περιγραφές της σημασιολογίας. Το class diagram του μεταμοντέλου αποτελείται από κλάσεις των οποίων τα στιγμιότυπα τους είναι στοιχεία UML μοντέλου. Οι κανόνες περιγράφονται σε Object Constraint Language (OCL) ή σε φυσική γλώσσα.

Ο καθορισμός ενός προτύπου αποτελείται από το *Πρότυπο Καθορισμού της Δομής* (*Structural Pattern Specification, SPS*), με την χρήση class diagrams, και από ένα *Πρότυπο Καθορισμού των Αλληλοεπιδράσεων* (*Interaction Pattern Specification, IPS*). Ένα UML μοντέλο υπακούει σε έναν καθορισμό προτύπου εάν το class diagram του υπακούει στο SPS και οι αλληλεπιδράσεις του που περιγράφονται από τα sequence diagrams υπακούουν στα ISP.

Ένα SPS ορίζει υποτύπους των κλάσεων του UML μεταμοντέλου περιγράφοντας στοιχεία των class diagrams (όπως τις κλάσεις *Class* και *Association* του UML μεταμοντέλου) και καθορίζει σημασιολογικές ιδιότητες χρησιμοποιώντας περιορισμούς. Ένα SPS αποτελείται από *ρόλους* (*roles*), όπου ένας ρόλος καθορίζει ιδιότητες τις οποίες πρέπει να έχει ένα στοιχείο UML μοντέλου ώστε να αποτελεί μέρος της λύσης. Η κλάση του μεταμοντέλου ονομάζεται *βάση* (*base*) του ρόλου. Ένας ρόλος μπορεί να κατηγοριοποιηθεί είτε ως *ρόλος ταξινομητή* (*classifier role*) είτε ως *ρόλος συσχέτισης* (*relationship role*). Ένα SPS πρέπει να έχει τουλάχιστον έναν απαιτούμενο ρόλο. Ένα class diagram υπακούει πλήρως σε ένα SPS, σεβόμενο τη σύνδεση μεταξύ των στοιχείων του μοντέλου και των ρόλων, εάν: *i)* υπακούει δομικά στο SPS, και *ii)* η σημασιολογικές ιδιότητες που εκφράζονται με περιορισμούς στο class diagram υπακούουν στο *πρότυπο των περιορισμών* (*constraint template*) του SPS.

Ένα Interaction Pattern Specification (IPS) περιγράφει ένα πρότυπο αλληλεπιδράσεων και ορίζεται με την μορφή των ρόλων του SPS. Οι SPS ρόλοι χρησιμοποιούνται στον καθορισμό των συμμετεχόντων στο πρότυπο των αλληλεπιδράσεων. Ένα IPS αποτελείται από έναν *ρόλο αλληλεπίδρασης* (*interaction role*), ο οποίος είναι μια δομή από χρόνους ζωής (lifelines) και μηνύματα (messages). Κάθε lifeline ρόλος συσχετίζεται με έναν classifier role. Ο συσχετισμός ορίζεται ώστε, ένας συμμετέχοντας (*participant*), που παίζει έναν lifeline role, να αποτελεί στιγμιότυπο ενός classifier, ο οποίος υπακούει στο classifier role. Τα messages αναπαριστούν κλήσεις λειτουργιών, οι οποίες υπακούουν στη συμπεριφορά που έχει αποδοθεί στο πρότυπο.

Για την επίτευξη του σκοπού της εργασίας αναπτύχθηκε μια γλώσσα καθορισμού προτύπων, η οποία: 1) χρησιμοποιεί τη σύνταξη της UML όσο το δυνατόν επεκτεταμένη, και 2) καθορίζει πρότυπα ως συγκεκριμενοποιήσεις του UML μεταμοντέλου ώστε να υποστηριχθεί η χρήση των προτύπων στη UML μοντελοποίησης συστημάτων. Για να καθοριστεί το κατά πόσο η τεχνική υποστηρίζεται από τα εργαλεία μοντελοποίησης της UML, αναπτύχθηκε ένα πρωτότυπο εργαλείο κατασκευής καθορισμού προτύπων πάνω από το Rational Rose tool. Το εργαλείο επιτρέπει την δημιουργία SPSs και την χρήση τους για την παραγωγή class diagrams που υπακούουν σε αυτά. Ένα πρόβλημα που παρατηρήθηκε όταν επιχειρήθηκε κατά την υποστήριξη δημιουργίας και ενεργοποίηση των προτύπων περιορισμού.

Η εργασία [14] έχει ως σκοπό την παροχή μια γλώσσας μοντελοποίησης η οποία μπορεί να αποκαλύψει την γενική φύση των προτύπων σχεδίασης, αλλά και να είναι αρκετά ακριβής ώστε να είναι δυνατή η επικοινωνία και η κοινή αποδοχή των σταθερές μεταξύ όλων των ασχολουμένων με το θέμα της σχεδίασης.

Τα *pattern leitmotifs* είναι αφαιρετικά σχεδιαστικά μοντέλα, τα οποία «συλλαμβάνουν» οι πιο σημαντικές σταθερές με τις οποίες παράγονται συγκεκριμένες λύσεις ενός προβλήματος.

Τα σχεδιαστικά πρότυπα ορίζουν ιδιότητες δομής και συμπεριφοράς, οι οποίες θα πρέπει να πληρούνται από τις κλάσεις και τα αντικείμενα. Είναι μια μερική περιγραφή, η οποία δείχνει μια *σταθερή* όψη (*invariants*) των συμμετεχόντων κλάσεων και αντικειμένων. Ένα leitmotif μοντέλο ορίζει: *i)* ένα σύνολο από σημαντικούς *ρόλους* τους οποίους κατέχουν δομικές οντότητες (όπως κλάσεις και αντικείμενα), *ii)* τις *οντότητες συμπεριφοράς* (όπως λειτουργίες και μέθοδοι), και *iii)* τη *συνεργασία* (*collaboration*) που εμφανίζεται μεταξύ τους. Εδώ, ρόλοι θεωρούνται οι οντότητες των προτύπων leitmotifs και *actors* οι συμμετέχοντες στις οντότητες. Κάθε ρόλος σε ένα leitmotif μοντέλο μπορεί να αποδοθεί σε έναν μη προκαθορισμένο αριθμό από actors. Οι actors που έχουν τον ίδιο ρόλο μοιράζονται τα ίδια στοιχεία που παρέχονται από τον ρόλο, γι' αυτό το λόγο η γλώσσα περιγραφής θα πρέπει να υποστηρίζει την περιγραφή γενικευμένων σχέσεων μεταξύ των ρόλων. Κάθε ρόλος

μπορεί να παρατηρηθεί μέσω διαφορετικών διαστάσεων, με την κάθε διάσταση να αναπαριστά μια συγκεκριμένη κατηγοριοποίηση των actors. Οι σχέσεις ανάμεσα στους ρόλους δεν είναι ακολουθούν την ένα-προς-ένα αντιστοίχηση με τις σχεδιαστικές κατασκευές του μοντέλου, δηλαδή μπορεί σε μια μόνο κλάση να δύο αποδίδονται ρόλοι.

Για την μοντελοποίηση των ρόλων των patterns leitmotifs χρησιμοποιείται η ιδέα της μετα-μοντελοποίησης. Θεωρείται ότι η δομή των patterns leitmotifs είναι μια UML συνεργασία μεταξύ στοιχείων του UML μεταμοντέλου. Οπότε, οι ρόλοι μπορεί να θεωρηθεί ότι είναι ένας UML ClassifierRole με μετα-επιπέδου στοιχεία. Επειδή όμως, δεν είναι δυνατή η ανάθεση μετα-επιπέδου στοιχεία ως βάση του UML ClassifierRole, χρησιμοποιείται το στερεότυπο <<meta>>. Με αυτό τον τρόπο ένα pattern leitmotif καθορίζεται από ένα UML *διάγραμμα συνεργασίας* (*collaboration diagram*) με <<meta>> στερεότυπα. Το AccosiationRole καθορίζει τις αφαιρετικές σχέσεις ανάμεσα στους actors. Διαφορετικού τύπου ρόλοι σχέσεων καθορίζονται ως στερεότυπα με χρήση <<implicit>>, το οποίο είναι ένα στερεότυπο *σχέσης* (*association*).

Συμπερασματικά, σε αυτή την εργασία υιοθετήθηκε η τεχνική της μετα-μοντελοποίησης, χρησιμοποιώντας το διάγραμμα συνεργασίας για τον καθορισμό της συνεργασίας μεταξύ των στοιχείων του μοντέλου. Γι’ αυτό το σκοπό ορίσθηκαν επιπλέον, καλά καθορισμένοι, κανόνες στο μεταμοντέλο της UML συνεργασίας**.** Εισήχθη μια AssociationClass με το όνομα RoleParticipation, η οποία παρέχει σημασιολογία για τη συμμετοχή των ρόλων. Επιπλέον ορίστηκε ένα σύνολο από στερεότυπα απεικονίζοντας τις σχέσεις ανάμεσα στους ρόλους.

Στο [16], παρουσιάζεται ένα πλαίσιο για την μοντελοποίηση των σχέσεων που αναπτύσσονται μεταξύ των *πηγών* (*sources*) και των *στόχων* (*targets*) σε διάφορα επίπεδα λεπτομέρειας. Ο σκοπός αυτός πραγματοποιείται με την επέκταση της UML ώστε να θεωρούνται τα γνωρίσματα ως *πρώτης κατηγορίας στοιχεία* (*first-class citizens*).

Μια βασική παρατήρηση είναι ότι, όσον αφορά σε κεντρικά συστήματα βάσεων, το conceptual modeling είναι διαφορετικό από αυτό που αφορά στον καθορισμό των λειτουργικών απαιτήσεων από την πλευρά του ανθρώπου.

Το πρόβλημα στο οποίο στοχεύει να λύσει το προτεινόμενο πλαίσιο, είναι η αδυναμία μοντελοποίησης, σε γενικό επίπεδο, των συσχετίσεων που αναπτύσσεται ανάμεσα στα γνωρίσματα. Γενικά, ως *πρώτης-τάξεως στοιχεία μοντελοποίησης* (*first-class modeling elements*) αναφέρονται τα βασικά στοιχεία μοντελοποίησης με την βοήθεια των οποίων κατασκευάζεται τα μοντέλα. Τεχνικά, τα first-class modeling elements έχουν μια ταυτότητα από μόνα τους και πιθανόν ελέγχονται από ορισμένους περιορισμούς ακεραιότητας (π.χ., οι συσχετίσεις πρέπει να έχουν τουλάχιστον δυο άκρα τα οποία να αναφέρονται στις κλάσεις ανάμεσα στις οποίες υπάρχει η συσχέτιση). Στην προσέγγιση αυτή, οι κλάσεις και τα γνωρίσματα ορίζονται κανονικά στην UML. Στις περιπτώσεις όμως, που είναι απαραίτητο τα γνωρίσματα να θεωρούνται ως first-class modeling elements, οι κλάσεις εισάγονται στο *διάγραμμα γνώρισμα/κλάση* (*attribute/class diagram*), όπου τα γνωρίσματα θεωρούνται αυτόματα ως κλάσεις. Κατά την διαδικασία της εισαγωγής από το *διάγραμμα κλάσης* (*class diagram*) στο *attribute/class diagram*, η κλάση η οποία περιέχει τα γνωρίσματα αναφέρεται ως *container class*, και η κλάση η οποία αναπαριστά ένα γνώρισμα ως *attribute class.* Σε συνδυασμό με τον ορισμό του *attribute/class diagram*, ορίζονται και τα στερεότυπα «Attribute» και «Contain» τα οποία αναπαριστούν τα γνωρίσματα των κλάσεων και τη συσχέτιση ανάμεσα σε μια container class και σε ένα attribute class στο *attribute/class diagram*, αντίστοιχα.

Για τον ορισμό του πλαισίου, εκτός από την θεώρηση των γνωρισμάτων ως *first-class modeling elements*, εισάγεται και η έννοια του *διαγράμματος αντιστοίχησης δεδομένων* (*data mapping diagram*), το οποίο είναι ένα διάγραμμα ροής δεδομένων, με διάφορα επίπεδα λεπτομέρειας. Το data mapping diagram είναι συμπληρωματικό με τα τυπικά διαγράμματα κλάσεων και συσχετίσεων της UML, και παρέχει τρία βασικά στοιχεία: 1) τον *προμηθευτή* (*provider)*, το οποίο παράγει τα δεδομένα τα οποία θα προωθηθούν (μπορεί να είναι ένα σχήμα, ένας πίνακας ή κάποιο γνώρισμα), 2) τον *καταναλωτή* (*consumer*), το οποίο λαμβάνει τα παραγόμενα δεδομένα, και 3) την *αντιστοίχηση* (*mapping*), το οποίο είναι ένα ενδιάμεσο βήμα και έχει σχέση με τον

τρόπο με τον οποίο παρουσιάζονται τα δεδομένα καθώς και με τις λειτουργίες μεταβολή (transformation) και φιλτράρισμα (filtering) που πραγματοποιούνται σε αυτό το βήμα τις επεξεργασίας.

Για τη σχηματική αναπαράσταση του data mapping diagram προτείνονται δύο τρόποι, ανάλογα με τον τρόπο με τον οποίο παρουσιάζεται η σημασιολογία του data mapping:

1. *Compact Variant*, όπου το data mapping diagram είναι πιο απλό, πιο λιτό με λιγότερη πληροφορία, ενώ η σημασιολογία εκφράζεται με την μορφή UML *σημειώσεων* (*UML notes*).
2. *Formal variant*, πιο λεπτομερειακή καταγραφή των στοιχείων του μοντέλου και η και η σημασιολογία εκφράζεται με την μορφή *ορισμών ετικετών* (*tag definitions*).

Με την μοντελοποίηση σε επίπεδο γνωρισμάτων επιτυγχάνονται τα εξής:

i. Παρέχεται μια βασική προσέγγιση για το conceptual modeling.
ii. Παρέχονται συμπληρωματικά στοιχεία σχεδίασης, το καθένα από τα οποία περιγράφουν το περιβάλλον σε διαφορετικά επίπεδα λεπτομέρειας.
iii. Καλύτερη κατανόηση του.
iv. Κατασκευή ενός γράφου, όπου οι σχέσεις παραγωγού και καταναλωτή παρουσιάζονται ως εισερχόμενες και εξερχόμενες ακμές αντίστοιχα. Με την βοήθεια του γράφου μπορούν να «μετρηθούν» οι ιδιότητες του μοντέλου.
v. Τέλος με την *οπτικοποίηση* (*visualization*) και την *μέτρηση* (*measurement*) της κατασκευής μπορεί πιο εύκολα να πραγματοποιηθεί η απαιτούμενη ανάλυση σε περιπτώσεις αλλαγών στο σχέδιο.

Οι Taibi και Ngo, στο [23], παρατήρησαν ότι όσο ο αριθμός των προτύπων αυξάνει και απαιτείται για την λύση προβλημάτων συνδυασμός προτύπων, η *περιγραφική περιγραφή* (*textual description*) είναι διφορούμενη και πολλές φορές μπορεί να οδηγήσει σε λαθεμένη κατανόηση και εφαρμογή των προτύπων. Για την

αντιμετώπιση αυτού του προβλήματος θεωρείται αναγκαία η εισαγωγή φορμαλιστικών μέσων, έτσι ώστε να περιγράφονται επ' ακριβώς τα πρότυπα καθώς και οι συνδυασμοί τους. Η φορμαλιστική περιγραφή έχει ως σκοπό να συμπληρώσει τους δυο υπάρχοντες τρόπους, *γραφική/με κείμενο* (*graphical / textual*) περιγραφής, ώστε να επιτευχθεί καλά καθορισμένη σημασιολογία αλλά και λογική για τα πρότυπα. Με την φορμαλιστική περιγραφή μπορεί πιο εύκολα να αποφασιστεί ποιο ή ποια πρότυπα είναι πιο κατάλληλα για τη σχεδίαση ενός προβλήματος, καθώς επίσης και να τεθούν τυπικοί ορισμοί για συνδυασμούς προτύπων.

Σε αυτή την εργασία προτείνεται μια *Balanced Pattern Specification Language* (*BPSL*), η οποία συνδυάζει την τυπική περιγραφή τόσο της *δομής* (*structural*) όσο και της συμπεριφοράς (*behavioral*) των προτύπων. Η δομή περιγράφεται με την βοήθεια της *First Order Logic* (*FOL*), ενώ η συμπεριφορά με την *Temporal Logic of Action* (*TLA*). Για την περιγραφή της δομής χρησιμοποιείται ένα υποσύνολο της FOL, το οποίο εστιάζει σε *μεταβλητές* και σε *σύμβολα κατηγορήματος* (*predicate symbols*), επειδή οι σχέσεις ανάμεσα στα πρότυπα μπορούν να αναπαρασταθούν ως *κατηγορήματα* (*predicates*). Για την τυπική περιγραφή της συμπεριφοράς χρησιμοποιείται ένα υποσύνολο της TLA, το οποίο εστιάζει σε *σταθερές* (*invariants***)** –περιορισμοί που θέτονται κατά την διάρκεια ζωής του συστήματος-, και σε *ενέργειες* (*actions*) – αλλαγή της κατάστασης των μεταβλητών (γνωρίσματα κλάσεων) και συσχετισμός ή διαχωρισμός μέσω προσωρινών συσχετίσεων. Τα βασικά κατασκευαστικά συστατικά της BPSL είναι οι *πρωταρχικές οντότητες-έννοιες* (*primary entities*), οι *συσχετίσεις* (*relations*) και οι *ενέργειες* (*actions*).

Πιο συγκεκριμένα, το υποσύνολο της FOL αποτελείται από *σύμβολα μεταβλητών*, *συνδέσμους* (*connectives*) (κυρίως $\wedge$), *quantifiers* (κυρίως $\exists$) και *predicate symbols* που επιδρούν στις μεταβλητές. Οι μεταβλητές αναπαριστούν κλάσεις, γνώρισμα, μεθόδους, αντικείμενα και *untyped values* - δηλαδή κάθε τύπου τιμές-. Οι πρωταρχικές μόνιμες σχέσεις μπορούν να προκύψουν από την αναπαράσταση των προτύπων μέσω της `UML`. Το υποσύνολο της TLA, θεωρώντας το πρότυπο ως κάποια γενική μηχανή κατάστασης, αποτελείται από *μεταβλητές κατάστασης* (*state variables*) και από *ενέργειες*, οι οποίες μεταβάλλουν τις καταστάσεις. Μια ενέργεια είναι μια Boolean έκφραση, η οποία είτε είναι αληθής είτε ψευδής σε σχέση με ένα ζεύγος καταστάσεων (αρχική και τελική κατάσταση κατά τη μετάβαση). Αντίθετα με τις

μόνιμες συσχετίσεις (*permanent relations*) οι οποίες είναι γενικές και μπορούν να καθορίσουν όλα τα πρότυπα, οι *προσωρινές* (*temporal relations*) είναι συγκεκριμένες για κάθε πρότυπο και πρέπει να καθορίζονται κάθε φορά.

Ο συνδυασμός προτύπων καθιστά την κατανόηση και την χρήση συνδυασμένων προτύπων δύσκολη, οπότε η ανάγκη για τυπική περιγραφή τους είναι μεγάλη. Η αρχή της *ολοκλήρωσης* (*completeness*) των συνδυασμένων προτύπων βασίζεται σε δυο όρους: *i)* κανένα πρότυπο δεν χάνει τις ιδιότητες του μετά τον συνδυασμό, και *ii)* καμία νέα ιδιότητα δεν μπορεί να θεωρηθεί ως απόρροια της σύνθεσης. Κάθε συνδυασμός που ικανοποιεί και τους δυο όρους θεωρείται *faithful* ή *correct*.

Μέσω των FOL αντικαταστάσεων ορίζονται φορμαλιστικά τόσο η δομή όσο και η συμπεριφορά του συνδυαστικού μοντέλου. Σημαντικό στοιχείο στις αντικαταστάσεις αποτελεί η έννοια της επικρατούσας συμπεριφοράς. Δηλαδή όταν κάποιο πρότυπο έχει σημαντική συμπεριφορά, η οποία και επηρεάζει τη συμπεριφορά του συνδυαστικού προτύπου. Οι αντικαταστάσεις σε επίπεδο συμπεριφοράς πραγματοποιείται μόνο σε ενέργειες και συσχετίσεις προσωρινές.

2.3. Πειράματα (Experiments)

Σκοπός του [20], είναι ο καθορισμός των μεγαλύτερων ισχυρισμών όσον αφορά στις ωφέλιμες συνέπειες των προτύπων, και η παραγωγή των αντίστοιχων εργαστηριακών ερωτήσεων. Η συνεισφορά αυτής της εργασίας είναι η περιγραφή σημαντικών μεθοδολογικών απόψεων πρακτικών πειραμάτων και πως αυτά σχετίζονται με τα αποτελέσματα που παρατηρούνται.

Πρώτον, καταγράφονται οι ισχυρισμοί που γίνονται για τα πιθανά πλεονεκτήματα των προτύπων, και για κάθε ισχυρισμό τίθενται δυο ερωτήματα: a) Εάν είναι ο ισχυρισμός σωστός, και b) κάτω από ποιες συνθήκες ισχύει ο ισχυρισμός. Στη συνέχεια συζητούνται οι μέθοδοι με τις οποίες μπορούν να απαντηθούν αυτά τα ερωτήματα και οι περιορισμοί που κάνουν την έρευνα δύσκολη.

Τα κύρια ισχυριζόμενα πλεονεκτήματα των προτύπων είναι τα εξής:

- *Ισχυρισμός LD*: Οι κατασκευαστές μαθαίνουν γρήγορα καλύτερες σχεδιαστικές τεχνικές με την μελέτη των προτύπων.
- *Ισχυρισμός P*: Η χρήση προτύπων βελτιώνει την παραγωγικότητα του σχεδιαστή.
- *Ισχυρισμός QN*: Οι πρωτάρηδες μπορούν να βελτιώσουν την ποιότητα των σχεδιασμών του με την εφαρμογή προτύπων.
- *Ισχυρισμός QE*: Ακόμα και για τους έμπειρους κατασκευαστές, τα πρότυπα ενισχύουν καλύτερες πρακτικές και εξασφαλίζουν υψηλή ποιότητα σχεδιασμού.
- *Ισχυρισμός CD*: Τα σχεδιαστικά πρότυπα βελτιώνουν την επικοινωνία ανάμεσα στους κατασκευαστές.
- *Ισχυρισμός CM*: Τα πρότυπα βελτιώνουν την επικοινωνία ανάμεσα στους κατασκευαστές και στους συντηρητές**.**

Στην εργασία αυτή προτιμήθηκαν τα ελεγχόμενα εργαστηριακά πειράματα (*controlled experiments*), έναντι των πραγματικών περιβαλλόντων ανάπτυξης λογισμικού (*field studies*). Για τα αρχικά πειράματα χρησιμοποιήθηκαν τόσο φοιτητές όσο και επαγγελματίες. Τα προγράμματα των πειραμάτων είναι προβλήματα διατήρησης, διότι το πιθανό πρόβλημα είναι μεγαλύτερο σε μέγεθος απ’ ότι η σχεδίαση του και η επανεγγραφή του από την αρχή.

Τα αποτελέσματα των πειραμάτων έδειξαν ότι:

1. Ο ισχυρισμός *CM* είναι σωστός, και συμπερασματικά η βελτίωση της επικοινωνίας παρατηρείται είτε στην παραγωγικότητα είτε στην ποιότητα των προγραμμάτων.
2. Όσον αφορά στους ισχυρισμούς *QE*, *CM* και *LD* δεν είναι ξεκάθαρο το κατά πόσο είναι σωστοί ή όχι, εφόσον όλα τα δυνατά αποτελέσματα εμφανίστηκαν. Π.χ. στην περίπτωση του το πρότυπο είχε αρνητικό αποτέλεσμα, περίπτωση του Composite/Abstract Factory και Composite/Visitor , τα πρότυπα και οι εναλλακτικές και λύσεις δεν είχαν μεγάλες αποκλείσεις, ενώ για το Decorator το πρότυπο υπερίσχυσε. Το σημαντικότερο αυτού του μέρους του πειράματος είναι ότι τα περισσότερα αποτελέσματα μπορούσαν να προβλεφθούν, εκτός της περίπτωσης του Visitor όπου αναμενόταν αρνητικό αποτέλεσμα το οποίο

δεν επαληθεύτηκε. Η μελέτη των αποτελεσμάτων απέδειξε ότι: *i)* γενικά, η κοινή λογική στην κατασκευή λογισμικών, όταν αυτή εφαρμόζεται προσεκτικά, επιλέγει τη χρησιμότητα των προτύπων έναντι των εναλλακτικών λύσεων, *ii)* τα πρότυπα είναι χρήσιμα αλλά δεν είναι για όλα τα προβλήματα, *iii)* υπάρχουν περιπτώσεις όπου η κατανόηση των σχεδιασμών είναι παραπλανητική οδηγώντας σε αναμενόμενες δυσκολίες στις λάθος περιπτώσεις.

Συμπερασματικά, ως αποτέλεσμα των πειραμάτων θα μπορούσε να αποτελεί η συμβουλή, η χρήση των προτύπων να γίνεται με βάση την κοινή λογική, αλλά χωρίς αμφιβολία είναι προτιμότερη λόγω της ελαστικότητας που παρέχει.

Το [21], σκοπεύει στην εμπειρική απόδειξη του κατά πόσο η χρήση σχεδιαστικών προτύπων είναι ωφέλιμη. Συγκεκριμένα, κάποιος μπορεί να προτιμήσει να χρησιμοποιήσει ένα σχεδιαστικό πρότυπο ακόμα και όταν το πρόβλημα είναι πιο απλό από αυτό το οποίο λύνει το πρότυπο, π.χ. εάν δεν απαιτούνται όλες οι λειτουργίες που παρέχονται από το πρότυπο. Το πείραμα που διεξάγεται μελετάει το πρόβλημα της συντήρησης (*maintenance*) συγκρίνοντας διάφορα πρότυπα με πιο απλές εναλλακτικές λύσεις. Τα αντικείμενα είναι επαγγελματίες μηχανικοί λογισμικού. Στα περισσότερα από τα εννιά παραδείγματα του πειράματος αποδείχθηκε ότι η χρήση προτύπου έχει θετικά αποτελέσματα, αν και δεν έλειψαν τα παραδείγματα όπου τα αποτελέσματα δεν ήταν τα αναμενόμενα. Συμπερασματικά, μέσω του πειράματος αποδείχθηκε ότι, αν δεν υπάρχει προφανής λόγος για την χρήση πιο απλής λύσης, είναι πιο σοφό να επιλεχθεί η «ελαστικότητα» των σχεδιαστικών προτύπων διότι μπορεί να εμφανιστούν νέες μη αναμενόμενες απαιτήσεις.

Η παρούσα εργασία παρέχει εμπειρική απόδειξη στο ότι η χρήση συγκεκριμένων σχεδιαστικών προτύπων μπορεί, όπως αναμένεται, να βελτιώσει την ιδιότητα της συντήρησης των προγραμμάτων. Επιπλέον, εξάγονται δύο όχι και τόσο εμφανή αποτελέσματα: *i)* παρουσιάζεται ένα παράδειγμα όπου η λογική χρήση ενός προτύπου κάνει το πρόγραμμα πιο δύσκολο για τη συντήρηση, και *ii)* παρατηρήθηκε ότι συγκρινόμενο με μια πιο άμεση λύση, το πρότυπο, αν και παρέχει ελαστικότητα που

δεν απαιτείται από το πρόβλημα, μπορεί να συντηρηθεί πιο εύκολα. Στην περίπτωση όπου το πρόβλημα είναι πιο απλό από αυτό που λύνει το πρότυπο υπάρχουν δύο περιπτώσεις. Από τη μία, η εφαρμογή του προτύπου είναι μια καλή λύση δεδομένου των πλεονεκτημάτων της κοινής ορολογίας, των αποδεδειγμένων λύσεων και της καλύτερης πρακτικής. Από την άλλη όμως, μπορεί να είναι μια κακή ιδέα διότι μπορεί να αποτελεί πιο δύσκολη λύση, κάνοντας πιο δύσκολη και την κατανόηση ή την αλλαγή του προγράμματος από χρήστες μη έμπειρους στα πρότυπα.

Στο πείραμα αυτό συγκρίνονται λύσεις με σχεδιαστικά πρότυπα και οι εναλλακτικές λύσεις χωρίς την χρήση προτύπων από ανθρώπους με διαφορετικά επίπεδα γνώσεις στα πρότυπα. Οι παράμετροι που λαμβάνονται υπόψη είναι *ο χρόνος για την αλλαγή κάθε προγράμματος* και *το κατά πόσο η λύση πλήρη τις απαιτήσεις*. Τα αντικείμενα έχουν χωριστεί σε τέσσερις ομάδες, όπου κάθε μία εργάζεται και στα τέσσερα προγράμματα του πειράματος. Σε πρώτο στάδιο ζητάτε από κάθε ομάδα να εργαστεί σε ένα πρόγραμμα στο οποίο να χρησιμοποιήσει ένα πρότυπο και σε ένα άλλο στο οποίο να χρησιμοποιήσει μια εναλλακτική λύση. Στο ενδιάμεσο βήμα του πειράματος πραγματοποιείται εκπαίδευση των μελών κάθε ομάδας στα πρότυπα. Στη συνέχεια, και σαν τελευταίο βήμα, τους ζητάτε να εργαστούν σε δυο διαφορετικά, από τα αρχικά, προβλήματα, όπου στο ένα να χρησιμοποιήσουν πρότυπα και στο άλλο όχι.

Τα αποτελέσματα που πειράματος αποδεικνύουν ότι η ωφέλιμη χρήση προτύπου εξαρτάται από το εκάστοτε πρόβλημα. Για την περίπτωση του Observer το αποτέλεσμα, όπως αναμενόταν, ήταν αρνητικό λόγω της πολυπλοκότητας του προτύπου. Στην περίπτωση των Composite, Visitor το αποτέλεσμα, αν και αναμένονταν αρνητικό λόγω της δυσκολίας στην κατανόηση του προτύπου Visitor, ήταν ουδέτερο χωρίς σημαντική αύξηση του απαιτούμενου χρόνου. Αναμενόμενο ήταν και το θετικό αποτέλεσμα του Decorator. Τέλος στο παράδειγμα των Composite και Abstract Factory όπως ήταν αναμενόμενο, οι αποκλίσεις στους χρόνους ήταν μικρές.

Από το πείραμα έχουν εξαχθεί τα εξής συμπεράσματα:

1. Είναι συνήθως, όχι πάντα, χρήσιμο η επιλογή ενός προτύπου από μια απλή εναλλακτική λύση.

2. Χρησιμοποιείται η κοινή λογική για την εύρεση των εξαιρετικών περιπτώσεων όπου η εναλλακτική λύση είναι προτιμότερη.
3. Ακόμα και αν η κοινή λογική επιτάσσει ότι η χρήση ενός προτύπου μπορεί να μην αποτελεί καλή λύση, όπως στην περίπτωση του *Visitor*, είναι προτιμότερο να χρησιμοποιηθεί το πρότυπο.
4. Η κατανόηση συγκεκριμένων προτύπων βοηθά στη χρήση τους κατά τη συντήρηση των προγραμμάτων ακόμα και όταν τα προγράμματα δεν είναι ούτε πολύ μεγάλα ούτε πολύπλοκα. Εάν αυτή η παρατήρηση ισχύει γενικά, εισάγει περιορισμούς στην χρησιμότητα των προτύπων όταν ο κατάλογος με τα διαθέσιμα πρότυπα αυξάνει και ο προγραμματιστής δεν τα γνωρίζει όλα.

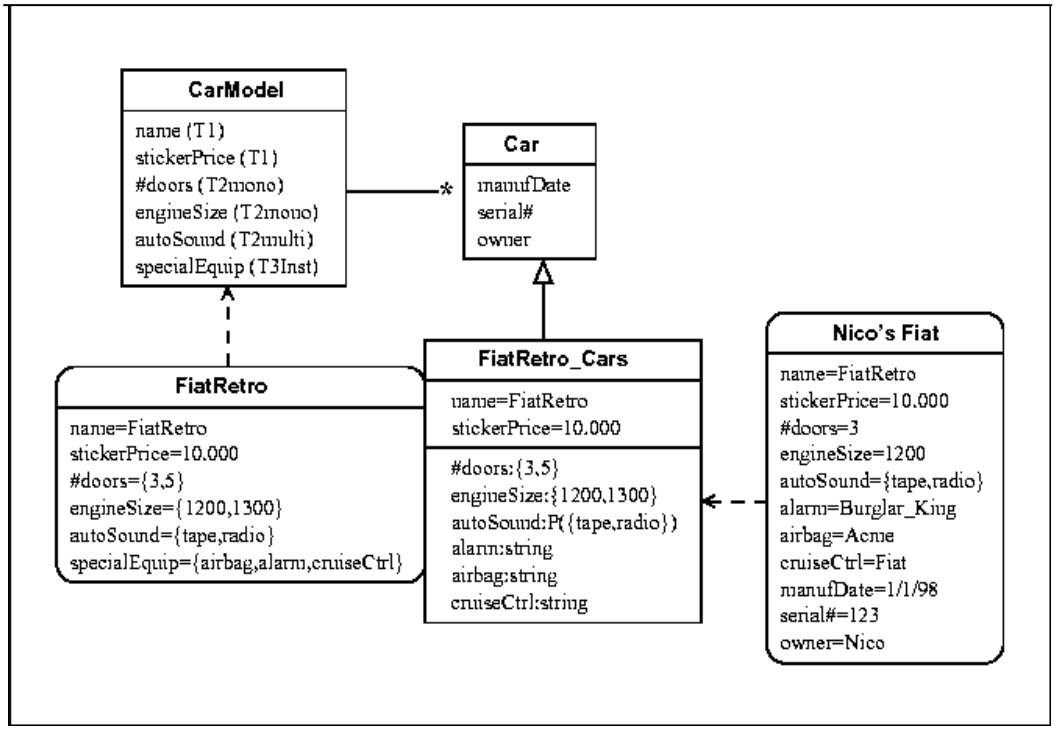
2.4. Διάφορα (Miscellaneous)

Στο [2], ορίζονται τυπικά κάποιες σημασιολογίες του materialization χρησιμοποιώντας την προσέγγιση των μετακλάσεων του μοντέλου δεδομένων TELOS. Επιπλέον προτείνει κάποιες επεκτάσεις του TELOS για τη σύλληψη κάποιων σημασιολογιών του materialization, οι οποίες δεν μπορούν να αναπαρασταθούν από τις υπάρχουσες κατασκευές.

Η σημασιολογία του materialization, `Abstract` —∗ `Concrete`, ορίζεται ως ένας συνδυασμός *IsA* (generalization), *IsOf* (classification) και μιας κλάσης/ μετά-κλάσης, ο οποίος ονομάζεται *δύο όψεων κατασκευή* (*two-faceted construct*). Η κατασκευή αυτή αποτελείται από ένα αντικείμενο, *object facet*, και μια συσχετιζόμενη κλάση, *class facet*. Το object facet είναι ένα στιγμιότυπο της Abstract κλάσης, ενώ το class facet είναι μια υποκλάση της Concrete κλάσης. Η class facet κληρονομεί γνωρίσματα από την Concrete μέσω του μηχανισμού *IsA*, αλλά και γνωρίσματα από την Abstract μέσω του μηχανισμού προώθησης γνωρισμάτων. Τα αντικείμενα της Concrete κλάσης με τιμές γνωρισμάτων κληρονομούμενες από ένα στιγμιότυπο της Abstract, είναι στιγμιότυπα της class facet συσχετιζόμενα με το στιγμιότυπο της Abstract.

Στο Σχήμα 2.8 παρουσιάζεται ένα παράδειγμα materialization με την κατασκευή *two-faceted*, όπου ένα μοντέλο αυτοκινήτου γίνει materialized σε ένα συγκεκριμένο αυτοκίνητο. Η κλάση `CarModel` αποτελεί την Abstract κλάση του materialization,

ενώ η `Car` την Concrete. Η κλάση `FiatRetro` αποτελεί το object facet, και η `FiatRetro_Cars` το class facet της *two*-faceted κατασκευή. Τέλος η κλάση `Nico's Fiat` απεικονίζει το στιγμιότυπο της κλάσης `FiatRetro_Cars`.



Σχήμα 2.8 Αναπαράσταση παραδείγματος materialization ως two-faceted κατασκευή [σχ.3, [2]].

Η προώθηση γνωρισμάτων ορίζεται ως μια μεταφορά πληροφορίας από ένα αφαιρετικό αντικείμενο στη συσχετιζόμενη class facet του. Υπάρχουν τρεις μηχανισμοί προώθησης γνωρισμάτων:

- *Τύπος 1*: μεταφορά μιας τιμής γνωρίσματος από ένα στιγμιότυπο της γενικής κλάσης στο στιγμιότυπα της συγκεκριμένης κλάσης. Η τιμή του γνωρίσματος προωθείται από το object facet στη σχετική class facet ως γνώρισμα κλάσης. Π.χ. τα μονότιμα γνωρίσματα `name` και `stickerPrice` του `CarModel` είναι τύπου 1. Οι τιμές τους στο object facet `FiatRetro` προωθείται και στα γνωρίσματα της σχετικής class facet `FiatRetro_Cars`.
- *Τύπος 2*: αφορά πλειότιμα γνωρίσματα της Abstract κλάσης. Η τιμή του γνωρίσματος στο στιγμιότυπο της Abstract καθορίζει τον τύπο ή το πεδίο του

στιγμιότυπου γνωρισμάτων, μονότιμα ή πλειότιμα, με το ίδιο όνομα στη σχετική class facet. Π.χ. στην περίπτωση μονότιμων γνωρισμάτων το `eng_size` του `Car_Model` έχει δύο δυνατές τιμές `1200` ή `1300`. Η τιμή του γνωρίσματος `eng_size` στο `FiatRetro` καθορίζει και την τιμή του ίδιου γνωρίσματος στη `FiatRetro_Cars`. Περίπτωση πλειότιμων γνωρισμάτων αποτελεί το γνώρισμα `autoSound` του `Car_Model`. Η τιμή του `autoSound ={tape, radio}` ορίζει ότι κάθε `FiatRetro` έχει είτε `radio`, είτε `tape` είτε τίποτα από τα δύο.

- *Τύπος 3*: αφορά πλειότιμα γνωρίσματα της abstract κλάσης, των οποίων η τιμή είναι πάντα ένα σύνολο συμβολοσειρών (strings). Κάθε στιγμιότυπο της Abstract (object facet) αντιστοιχίζει διαφορετικό σύνολο στοιχείων σε αυτά τα γνωρίσματα. Κάθε στοιχείο του συνόλου του γνωρίσματος του object facet γεννά ένα νέο στιγμιότυπο γνώρισμα στη σχετική class facet. Π.χ. γνώρισμα τύπου 3 είναι το `special_equip` του `CarModel`. Η τιμή του `{airbag, alarm, cruise_ctrl}` του `FiatRetro` παράγει τρία μονοδιάστατα στιγμιότυπα γνωρίσματα τύπου συμβολοσειράς, `airbag`, `alarm` και `cruise_ctrl`, στη σχετική class facet `FiatRetro_Cars`.

Το *TELOS* είναι μια γλώσσα αναπαράστασης γνώσης για τα πληροφοριακά συστήματα. Η βάση γνώσης του TELOS είναι μια συλλογή από *προτάσεις* (propositions). Κάθε πρόταση `p` είναι μια τριπλέτα `<from, label, to>`. Οι προτάσεις μπορεί είναι είτε *μεμονωμένες* (*individuals*) είτε *γνωρίσματα* (*attributes)*. Οι individuals αναπαριστούν τα αντικείμενα και τις κλάσεις, ενώ οι attributes αναπαριστούν δυαδικές σχέσεις ανάμεσα στις individuals ή άλλες σχέσεις. Οι προτάσεις μπορούν να είναι στιγμιότυπα άλλων προτάσεων, οι οποίες ονομάζονται classes. Υπάρχουν οι *ω-classes*, οι οποίες αναπαριστούν τις μετακλάσεις, οπότε μπορούν να έχουν στιγμιότυπα πολλών επιπέδων.

Τα μεταγνωρίσματα χρησιμοποιούνται για τον ορισμό κοινών ιδιοτήτων διάφορων κλάσεων. Το TELOS υποστηρίζει την εισαγωγή μια υπογλώσσας για τον καθορισμό των περιορισμών ακεραιότητας και των *κανόνων απόφασης* (*deductive rules***)**, οι οποίοι καθορίζουν τη συμπεριφορά των αντικειμένων στα οποία εφαρμόζονται. Οι

περιορισμοί είναι εισαγωγές ελέγχου της πληροφορίας που παρέχεται στο χρήστη, ενώ οι κανόνες απόφασης είναι εισαγωγές ενδυνάμωσης νέων γεγονότων.

Τέλος, προτείνονται δύο επεκτάσεις του TELOS ώστε να περιλαμβάνει τον ορισμό της πληθικότητας και των κανόνων απόφασης**.**

Στο [5] υποστηρίζεται ότι, οι *σχέσεις* (*associations*), αν και χρησιμοποιούνται ευρέως στις αντικειμενοστρεφείς γλώσσες, δεν έχουν καθορισμένη σημασία στο πεδίο του Conceptual Modeling. Προς το σκοπό αυτό, αρχικά εφαρμόζεται οντολογική ανάλυση της λογισμικής σημασιολογίας των συσχετίσεων και αποδεικνύεται ότι δεν μπορεί να μεταφερθεί στο Conceptual Modeling. Εν συνεχεία, θεωρείται ότι οι συσχετίσεις είναι 'σημασιολογικές συνδέσεις μεταξύ αντικειμένων, και σε αυτή την περίπτωση αποδεικνύεται ότι δεν μπορεί να μεταφερθεί η σημασία στο Conceptual Modeling. Τέλος, προτείνεται μια εναλλακτική λύση της χρήσης συσχετίσεων, η χρήση *κοινών ιδιοτήτων* (*shared properties*), μια κατασκευή η οποία πηγάζει άμεσα από την οντολογία.

Εδώ εξετάζεται η σημασιολογία των συσχετίσεων από πλευράς ανάλυσης του πεδίου της εφαρμογής και όχι από την πλευρά του σχεδιαστή λογισμικού ή του προγραμματιστή, αν και η σημασιολογία των συσχετίσεων και στις άλλες περιπτώσεις είναι επίσης προβληματική. Για τον ορισμό των συσχετίσεων με όρους στοιχείων από το πεδίο της εφαρμογής, θα πρέπει πρώτα να καθοριστεί τι υπάρχει ή τι θεωρείται ότι υπάρχει στο πεδίο της εφαρμογής. Οι οντολογίες καθορίζουν ποιες *βασικές έννοιες* (*concepts*) υπάρχουν σε ένα πεδίο και πως συνδέονται μεταξύ τους. Οπότε για τον ορισμό της σημασίας μιας συσχέτισης, θα πρέπει να τις αντιστοιχίσουμε σε μία οντολογική βάση. Η αντιστοίχηση αυτή θα πρέπει να υποστηρίζει, όσον το δυνατόν περισσότερο, τα συντακτικά στοιχεία των συσχετίσεων. Για την αναπαράσταση υιοθετείται το μοντέλο οντολογία του Bunge, στο οποίο θεωρείται ότι ο κόσμος αποτελείται από *πράγματα* (*things*) τα οποία έχουν φυσική υπόσταση στον κόσμο. Ένα πράγμα κατέχει *ατομικές ιδιότητες* (*individual properties*) κάθε μία από τις οποίες ανταποκρίνεται σε μία *ιδιότητα γενικώς* (*property in general*). Π.χ., όταν ένα `thing` έχει `κόκκινο χρώμα` είναι μια *ατομική ιδιότητα*, ενώ

το χρώμα είναι μια *ιδιότητα γενικώς*. Οι ιδιότητες είναι είτε *εγγενής* (*intrinsic*) είτε *αμοιβαίες* (*mutual*). Οι intrinsic ιδιότητες είναι αυτές που ένα πράγμα κατέχει μόνο του, π.χ., το χρώμα, ενώ οι αμοιβαίες μοιράζονται μεταξύ δυο ή περισσοτέρων πραγμάτων, π.χ., η θερμοκρασία ενός θερμαντήρα (heater) και του περιβάλλοντος αέρα, το δυναμικό του επεξεργαστή και της κύριας μνήμης. Δύο ή περισσότερα πράγματα μπορούν να αλληλεπιδρούν μεταξύ τους. Η αλληλεπίδραση, η οποία ορίζεται με βάση το ιστορικό ενός thing, λαμβάνει χώρα όταν η αλλαγή των ιδιοτήτων του thing εξαρτάται από την ύπαρξη ενός άλλου thing. Οντολογικώς, για να υπάρξει αλληλεπίδραση μεταξύ δύο πραγμάτων A, και B, όπου το A επιδρά στο B, θα πρέπει να υπάρχει μια αμοιβαία ιδιότητα P των A και B. Μια αλλαγή της P στο A σημαίνει αυτόματη αλλαγή και στη P του B.

Στη συνέχεια παρατίθενται οι τρεις μέθοδοι απόδοσης σημασίας στις συσχετίσεις:

- <u>*Λογισμική σημασιολογία των συσχετίσεων*</u>: Οι συσχετίσεις εισάγονται στις γλώσσες προγραμματισμού με την μορφή *δεικτών* (*pointers*) από ένα αντικείμενο σε ένα άλλο. Οι δείκτες χρησιμοποιούνται για την κλήση μεθόδων. Η αλληλεπίδραση-επικοινωνία πραγματοποιείται με την μέθοδο *περάσματος μηνύματος* (*message-passing*). Αυτού του είδους οι συσχετίσεις οι οποίες ονομάζονται εδώ *συσχετίσεις χρήσης* (*use associations*), είναι χρήσιμες για την περιγραφή του λογισμικού, αλλά δεν μπορούν να μεταφερθούν στο πεδίο του Conceptual Modeling για τους παρακάτω λόγους:
    1. Ο μηχανισμός της επικοινωνίας δεν βασίζεται στη μέθοδο περάσματος μηνύματος. Οντολογικά, για την αλληλεπίδραση μεταξύ δυο αντικειμένων A, B, θα πρέπει να αλλάξει κάποιο από τα A, B μία αμοιβαία ιδιότητα P τους.
    2. Ο μηχανισμός περάσματος μηνύματος έχει εξεταστεί σε σχέση με την οντολογία του Bunge και έχει βρεθεί ακατάλληλη για την περιγραφή του πραγματικού κόσμου. Π.χ., το γεγονός ‘η μηχανή στέλνει μήνυμα σε ένα μέρος της για να κινηθεί μόνο του σε νέα τοποθεσία’, δεν ισχύει στον πραγματικό κόσμο από τη στιγμή που η μηχανή δεν μπορεί να στείλει μήνυμα απ’ ευθείας σε ένα μέρος της και το μέρος της μηχανής δεν κινείται μόνο του αλλά με την βοήθεια κάποιου *τελεστή* (*operator*).

3. Τα μηνύματα δεν μπορούν να “μεταφραστούν” σε οντολογικά πράγματα. Η επικοινωνία ενός thing με ένα μήνυμα απαιτεί την ύπαρξη ενός άλλου μηνύματος που να επικοινωνεί με το αρχικό μήνυμα, κ.ο.κ, ορίζοντας με αυτό τον τρόπο μια επαγωγικά άπειρη διαδικασία. Π.χ., έστω ότι το A αλληλεπιδρά με το B μέσω του μηνύματος M_1, αλλά για να συμβεί αυτό, το A θα πρέπει να επικοινωνήσει με το M_1 μέσω ενός άλλου μηνύματος M_2, κ.ο.κ..

- <u>*Συνδετική σημασιολογία των συσχετίσεων*</u>: Οι συσχετίσεις απεικονίζονται με τη μορφή *συνδέσεων μεταξύ των αντικειμένων* (*connections associations*). Η σημασιολογία των συνδέσεων εφαρμόζεται σε ένα πεδίο όπως γίνεται αντιληπτή από τον παρατηρητή, και δεν έχει καμία οντολογική αντιστοιχία. Κάποιος κατασκευαστής μπορεί να θεωρήσει σχετικές κάποιες συμπεριφορές οντοτήτων και να τις μεταφράσει ως σημασιολογική σύνδεση, ενώ κάποιος άλλος να τις θεωρήσει άσχετες ή να τους δώσει άλλο ορισμό σημασιολογικής σύνδεσης. Συνεπώς τέτοιου είδους συσχετίσεις αποτελούν συναρτήσεις, είτε επικοινωνίας-αλληλεπίδρασης είτε εγγενείς ιδιότητες, εξαρτημένες από τον παρατηρητή, και δεν ανταποκρίνονται σε κάτι το οποίο υπάρχει στο πεδίο της εφαρμογής. Οπότε θα πρέπει να αναπαρίστανται με τη μορφή λειτουργιών και όχι με κατασκευαστική συσχέτιση.

- <u>*Conceptual Modeling με Αμοιβαίες Ιδιότητες*</u>: Η αλληλεπίδραση ορίζεται με όρους αμοιβαίων ιδιοτήτων, για την αναπαράσταση των οποίων στο Conceptual Modeling απαιτείται μια γλώσσα κατασκευής. Στη συγκεκριμένη εργασία χρησιμοποιούνται γνωρίσματα κλάσεων συσχέτισης για τους εξής λόγους: *i)* η ιδέα ενός γνωρίσματος ανταποκρίνεται καλά στη οντολογική θεώρηση μιας ιδιότητας, και *ii)* τα γνωρίσματα κλάσεων συσχέτισης γραφικά αναπαριστούν τις συνδέσεις μεταξύ δυο η περισσοτέρων αντικειμένων. Σε αυτό το σημείο θα πρέπει να παρατηρηθεί ότι οι συσχετίσεις και οι κλάσεις συσχέτισης από μόνες τους δεν μπορούν να μεταφραστούν οντολογικά, οπότε δεν θα πρέπει να χρησιμοποιούνται στο Conceptual Modeling. Χρησιμοποιείται, λοιπόν, η γραφική περιγραφή των γνωρισμάτων κλάσης συσχέτισης της UML, το οποίο είναι «αναγκαίο κακό» εάν θέλουμε να

αποφύγουμε την εισαγωγή νέου στοιχείου περιγραφής. Οντολογικά το *ιστορικό των αλληλεπιδράσεων* είναι ιστορικό των αλλαγών των αμοιβαίων ιδιοτήτων. Δυστυχώς δεν υπάρχει κανένα γραφικό μοντέλο αναπαράστασης στις γνωστές αντικειμενοστρεφείς γλώσσες προγραμματισμού, το οποίο να εκφράζει τέτοιες συναρτήσεις. Γι' αυτόν το λόγο προτείνεται μια απλή λεκτική περιγραφή του ιστορικού με την βοήθεια της Prolog. Τέλος, όπως παρατηρήθηκε προηγουμένως, η σημασιολογία των συσχετίσεων είναι άμεσα συνδεδεμένη με τον κατασκευαστή ή τον παρατηρητή, οπότε δεν θα πρέπει να αποτελούν μέρος του πεδίο μοντέλου. Αυτό σημαίνει ότι διαφορετικοί κατασκευαστές μπορούν να ορίσουν διαφορετικά σύνολο συναρτήσεων/λειτουργιών, οι οποίες είναι σχετικές με την οπτικό πρίσμα της κατασκευής τους.

Τέλος, στο [7], παρουσιάζεται ένα γενικό μεθοδολογικό πλαίσιο για την καθοδήγηση του conceptual modeling, εστιάζοντας στο πεδίο του προβλήματος. Αυτό το πλαίσιο ορίζεται λαμβάνοντας υπόψη αρχές σχετιζόμενες με ένα γενικό *conceptualisation*, και της εφαρμογής του στην κατασκευή λογισμικού.

Το *conceptualisation* είναι η χρήση *βασικών εννοιών* (*concepts*) και *σχέσεων* που εμφανίζονται κατά την λύση ενός προβλήματος. Αντίστοιχα τα αφαιρετικά μοντέλα (*conceptual models*) απεικονίζουν ένα μέρος του πραγματικού κόσμου και επιχειρούν να λύσουν το πρόβλημα. Από τη συλλογή των σχετικών πληροφοριών του προβλήματος, τα δύο επόμενα βήματα για την λύση του προβλήματος μέσω υπολογιστή είναι: *i)* η κατασκευή ενός conceptual model στο πεδίο του προβλήματος, και *ii)* η κατασκευή ενός φορμαλιστικού μοντέλου στο πεδίο του υπολογιστή, που βασίζεται στο conceptual model.

Το πρόβλημα των πιο χαρακτηριστικών conceptual models, TELOS, KAOS και EM, που έχουν προταθεί είναι ότι εστιάζουν κυρίως στην λύση του προβλήματος μέσω υπολογιστή, με αποτέλεσμα να συνδέονται με συγκεκριμένες προσεγγίσεις ανάπτυξης. Σε αυτό το πρόβλημα συντελεί και το γεγονός ότι δεν υπάρχει κάποια

τεχνική, η οποία να δηλώνει ρητά πως θα πρέπει να πραγματοποιηθεί το conceptualisation.

Για την αντιμετώπιση των παραπάνω προβλημάτων, κατασκευάζεται ένα μεθοδολογικό πλαίσιο. Η κατασκευή περιλαμβάνει τα βήματα: *i) εύρεση των βασικών εννοιών* (*conceptualization)* και *ii) μοντελοποίηση* (*modeling*).

Το αποτέλεσμα του conceptualisation είναι ένα conceptual model, το οποίο χαρακτηρίζεται από τριπλέτες της μορφής:

(Concepts, Relationships, Functions),

όπου τα concepts απεικονίζουν τις θεωρητικές έννοιες του πραγματικού κόσμου, οι οποίες μπορούν να αλληλεπιδρούν μεταξύ τους δημιουργώντας *συσχετίσεις* (*Relationships*) και είναι σε θέση να εκτελέσουν ορισμένες *λειτουργίες* (*Functions*).

Όσον αφορά στο modeling, ένα μοντέλο είναι η απλοποιημένη αναπαράσταση ενός γεγονότος, συστήματος, και οτιδήποτε άλλο μπορεί να μελετάται. Το μοντέλο δεν αναπαριστά επακριβώς τον πραγματικό κόσμο, αλλά προσπαθεί να λύσει το πρόβλημα σε μικρότερη κλίμακα. Αυτό ακριβώς είναι και ένα από τα προβλήματα του modeling, ότι μια λύση στο επίπεδο του μοντέλου δεν εγγυάται ότι είναι και λύση του προβλήματος στις διαστάσεις του πραγματικού κόσμου.

Το μεθοδολογικό πλαίσιο που προτείνεται εκμεταλλεύεται το γεγονός του παραλληλισμού μεταξύ των δομών της φυσικής γλώσσας και του conceptual modeling. Οι πληροφορίες που συλλέγονται, ταξινομούνται με βάση την λειτουργικότητά τους, σε *στατική* (*static*) και *δυναμική* (*dynamic*) πληροφορία. Η στατική (ή δηλωτική) πληροφορία αποτελείται από δηλώσεις σχετικά με την δομή του πεδίου της πληροφορίας, π.χ., γεγονότα που πιθανόν ισχύουν, γεγονότα (όπως concepts, properties, relationships και constrains) τα οποία μπορούν να χρησιμοποιηθούν σε λειτουργίες. Η δυναμική πληροφορία σχετίζεται με τη συμπεριφορά του πεδίου και χωρίζεται σε δύο υποκατηγορίες: *i)* τη *στρατηγική πληροφορία*, η οποία περιλαμβάνει πληροφορία σχετικά με το τι θα πρέπει να γίνει, πότε και με ποια σειρά, και *ii)* την *τακτική πληροφορία*, όπου δηλώνεται πώς και πότε μπορεί να εισαχθεί μια νέα δηλωτική πληροφορία για το πρόβλημα. Σε πρώτο στάδιο,

τα μέρη της πληροφορίας που συλλέγονται είναι ασύνδετα μεταξύ τους και όχι πλήρως κατανοητά. Στο δεύτερο στάδιο, δημιουργείται ένα conceptual model με βάση τα στοιχεία που έχουμε συλλέξει, ταξινομήσει και ομαδοποιήσει. Σε αυτό ακριβώς το σημείο, εάν είναι απαραίτητη η επιπλέον συλλογή πληροφορίας επιστρέφουμε στο πρώτο στάδιο. Τέλος, το conceptual model έχει κατασκευαστεί και η συλλογή πληροφορίας έχει τερματιστεί. Οι δραστηριότητες που πραγματοποιούνται κατά το conceptualization είναι οι εξής:

1) *Ανάλυση* (*Analysis*), κατά την οποία καθορίζονται τα concepts και οι μεταξύ τους συσχετίσεις στο επίπεδο του προβλήματος, χρησιμοποιώντας τον μηχανισμό της γενίκευσης.
2) *Σύνθεση* (*Synthesis*), η οποία αποτελείται από πράξεις, και αφορά στην οργάνωση και την αναπαράσταση της πληροφορίας που συλλέγεται κατά το στάδιο της ανάλυσης.
3) *Έλεγχος* (*Testing*), όπου ελέγχεται το κατά πόσο το conceptual model ταιριάζει στην πραγματικότητα.

Το πλαίσιο που παρουσιάζεται εδώ, αποτελείται από: *i)* το γενικό μοντέλο που αντιστοιχεί στην γενική γραφική αναπαράσταση του προβλήματος, και *ii)* τα υπομοντέλα και τα στοιχεία των υπομοντέλων, τα οποία αποτελούν τμήματα του μοντέλου. Για κάθε επίπεδο πληροφορίας έχει οριστεί ένα υπομοντέλο, στο οποίο δηλώνεται με κάθε λεπτομέρεια η εννοιολογική πληροφορία**.** Για τη σύνθεση αλλά και για τον έλεγχο της εγκυρότητας του, χρησιμοποιείται η τεχνική των *χαρτών πληροφορίας* (*information maps*), στους οποίους ο άκυκλος γράφος που σχηματίζεται, έχει ως κόμβους τα στοιχεία της δηλωτικής πληροφορίας, και ως ακμές τις συσχετίσεις συμπεράσματος ή υπολογισμού που αναπτύσσονται μεταξύ των δηλωτικών στοιχείων. Με την βοήθεια, λοιπόν, των χαρτών πληροφορίας ελέγχθηκε το προτεινόμενο πλαίσιο ως προς το κατά πόσο είναι εφαρμόσιμο σε κάθε πεδίο, π.χ. στο software engineering, και ως προς το κατά πόσο είναι εφικτό να κατασκευαστεί και να λειτουργήσει.

ΚΕΦΑΛΑΙΟ 3. ΟΡΙΣΜΟΣ ΚΑΙ ΔΟΜΗ ΕΝΟΣ ΣΧΕΔΙΑΣΤΙΚΟΥ ΠΡΟΤΥΠΟΥ ΓΙΑ ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ

Στο προηγούμενο κεφάλαιο είδαμε πως, όλες οι σχετικές μελέτες, παρουσιάζουν τα σχεδιαστικά πρότυπα ως ένα αποδοτικό τρόπο παρουσίασης ενός προβλήματος και της λύσης του. Εκτός της αφαιρετικής ιδιότητάς τους, δηλαδή την όσο το δυνατόν πιο γενική απόδοση ενός προβλήματος, τα σχεδιαστικά πρότυπα αποτελούν έναν κοινό κώδικα επικοινωνίας μεταξύ των σχεδιαστών ενός λογισμικού συστήματος. Σε αυτό το κεφάλαιο θα μελετηθούν τα σχεδιαστικά πρότυπα από πλευράς Βάσεων Δεδομένων.

3.1. Ορισμός και Δομή ενός σχεδιαστικού προτύπου

Για να κατανοήσουμε τον λόγο δημιουργίας ενός σχεδιαστικού προτύπου για τις Βάσεις Δεδομένων, ας θεωρήσουμε το παράδειγμα μιας ταχυδρομικής εταιρίας, όπου καταγράφεται πληροφορία σχετικά με τα γράμματα. Κάθε γράμμα έχει αποστολέα, παραλήπτη και βάρος. Ένα γράμμα μπορεί να είναι απλό (simple), πακέτο (package) ή ταχείας μεταφοράς (express). Ένα πακέτο έχει είδος συσκευασίας (απλή, σκληρή, ενισχυμένη), ενώ για τα express γράμματα κρατάμε τη συμπεφωνημένη ημερομηνία παράδοσής του.

Το παραπάνω παράδειγμα μπορεί να γενικευτεί σε πρόβλημα *γενίκευσης/εξειδίκευσης* (*IsA*), όπου πολλές κατηγορίες αποτελούν εξειδικεύσεις μιας γενικής κατηγορίας, η οποία περιλαμβάνει τα κοινά χαρακτηριστικά τους. Κάθε μια από τις κατηγορίες παρουσιάζει εκτός των κοινών χαρακτηριστικών, επιπλέον χαρακτηριστικά, τα οποία την εξειδικεύουν και την διαφοροποιούν από τις υπόλοιπες. Για τον καθορισμό των σχεδιαστικών λύσεων που μπορούν να απεικονίσουν το πρότυπο, θα πρέπει να λάβουμε υπόψη δύο παράγοντες. Ο πρώτος παράγοντας έχει σχέση με τις ερωτήσεις που πιθανόν να τίθενται προς την βάση. Π.χ., για το παράδειγμα της ταχυδρομικής εταιρίας πιθανότατα να ζητείται:

i. εύρεση όλων των γραμμάτων, ανεξάρτητα από το είδος τους (εύρεση της γενικής κατηγορίας),
ii. εύρεση όλων των γραμμάτων μιας συγκεκριμένης κατηγορίας,
iii. εύρεση συγκεκριμένου γράμματος, οπότε θα πρέπει να πραγματοποιηθεί αναζήτηση σε κάποια συγκεκριμένη κατηγορία.

Ο δεύτερος παράγοντας σχετίζεται με τις αλλαγές που μπορεί να πραγματοποιηθούν τόσο σε επίπεδο στιγμιότυπου (εισαγωγή/διαγραφή/ανανέωση πλειάδων), όσο και σε επίπεδο σχήματος (εισαγωγή/διαγραφή/ανανέωση γνωρισμάτων). Δοθέντος της δομής του σχήματος κάθε σχεδιαστικής λύσης, μελετάται πως συμπεριφέρεται η βάση κατά την εφαρμογή των αλλαγών. Π.χ., στην περίπτωση διαγραφής των χαρακτηριστικών που εξειδικεύουν μια κατηγορία και εάν έχουμε απεικονίσει κάθε κατηγορία με έναν ξεχωριστό πίνακα, θα πρέπει να διαγραφεί όλος ο πίνακας που απεικονίζει την κατηγορία αυτή.

Ο σχεδιαστής, με την βοήθεια της παραπάνω μελέτης, είναι σε θέση να επιλέξει, ανάλογα με το πρόβλημα, την πιο αποδοτική σχεδιαστική λύση, απαλλάσσοντάς τον από την εύρεση και αξιολόγηση μιας νέας σχεδίασης. Με αυτόν τον τρόπο, η σχεδίαση πραγματοποιείται εύκολα, γρήγορα και αποδοτικά, εφόσον είναι γνωστή εκ των προτέρων η συμπεριφορά της βάσης τόσο σε επίπεδο σχήματος όσο και σε επίπεδο στιγμιότυπου. Από τη στιγμή που είναι γνωστή η συμπεριφορά της βάσης, διευκολύνεται και η διατήρηση της, π.χ., εάν σε μια κατηγορία γραμμάτων θεωρηθεί απαραίτητη η εισαγωγή ενός νέου χαρακτηριστικού (εισαγωγή νέων λειτουργικών απαιτήσεων στο σύστημα απεικόνισης), αυτό πραγματοποιείται απλά με την εισαγωγή ενός νέου γνωρίσματος στην αντίστοιχη σχέση της βάσης.

Συμπερασματικά, η δημιουργία ενός σχεδιαστικού προτύπου για τις Βάσεις Δεδομένων έχει ως σκοπό:

1. τον γενικό, αλλά και τυπικό, τρόπο απεικόνισης ενός προβλήματος και του συνόλου των λύσεών του, παρέχοντας κοινό κώδικα επικοινωνίας μεταξύ των σχεδιαστών. Αρχικά, διατυπώνεται ένα συγκεκριμένο πρόβλημα, με την βοήθεια του οποίου γίνεται κατανοητή η σημασία του προτύπου, π.χ., η κατανόηση του προτύπου IsA πραγματοποιείται με την βοήθεια του παραδείγματος της ταχυδρομικής εταιρίας. Στη συνέχεια, γενικεύεται το πρόβλημα, δηλαδή προσδιορίζεται σε ποιες περιπτώσεις μπορεί να χρησιμοποιηθεί το συγκεκριμένο πρότυπο, και δίδεται το σύνολο των σχεδιαστικών λύσεων που επιλύουν το γενικό πρόβλημα. Παρέχοντας την γενική δομή όλων των σχεδιαστικών λύσεων που μπορούν να απεικονίσουν το συγκεκριμένο πρότυπο, η σχεδίαση της λύσης οπουδήποτε παρόμοιου προβλήματος, είναι πιο εύκολη, πιο αποδοτική, αλλά και πιο κατανοητή, ως προς τους σχεδιαστές και τους χρήστες της βάσης.
2. την απεικόνιση όλων των σχεδιαστικών λύσεων σε φυσικό επίπεδο (πίνακες, κλειδιά) και την αξιολόγηση της κάθε μίας από πλευράς κόστους σε χρόνο εκτέλεσης και χώρο αποθήκευσης. Παρέχοντας όλες τις σχεδιαστικές λύσεις καθώς και την αξιολόγησή τους, ο σχεδιαστής είναι σε θέση να επιλέξει την καλύτερη απεικόνιση για το συγκεκριμένο πρόβλημα, γνωρίζοντας εκ των προτέρων τη συμπεριφορά της σχεδίασής του σε φυσικό επίπεδο απεικόνισης. Δηλαδή, είναι σε θέση να γνωρίζει, πόσους και ποιους πίνακες πρέπει να κατασκευάσει, τις εξαρτήσεις που εμφανίζονται μεταξύ των πινάκων, καθώς και την διευκόλυνση στην αναζήτηση που μπορεί να του παρέχει ένα ευρετήριο όταν εφαρμοστεί σε κάποιο πεδίο.
3. την διευκόλυνση της διατήρησης της βάσης, με την απόδοση της συμπεριφοράς κάθε λύσης σε επίπεδο στιγμιότυπου και επίπεδο σχήματος. Η διατήρηση μιας Βάσης Δεδομένων, αφορά τόσο στη διασφάλιση της συνέπειας της βάσης κατά τις εισαγωγές/διαγραφές/ανανεώσεις πλειάδων (επίπεδο στιγμιότυπου), όσο και στην ικανότητά της να είναι επεκτάσιμη και ευέλικτη κατά την εμφάνιση νέων λειτουργικών απαιτήσεων, δηλαδή κατά την εισαγωγή/διαγραφή/ανανέωση πεδίων ή περιορισμών (επίπεδο σχήματος).

Με βάση τα παραπάνω, η αναγκαιότητα δημιουργίας σχεδιαστικών προτύπων για τις Βάσεις Δεδομένων είναι προφανής. Στη συνέχεια του κεφαλαίου παρουσιάζεται ο ορισμός και η δομή ενός σχεδιαστικού προτύπου, καθώς και η κατηγοριοποίηση των προτύπων που παρουσιάζονται αναλυτικά στο επόμενο κεφάλαιο.

3.2. Ορισμός και Δομή ενός σχεδιαστικού προτύπου

Ένα σχεδιαστικό πρότυπο ονοματίζει, γενικεύει και καθορίζει τη σημασιολογία μιας σχεδιαστικής δομής, η οποία εμφανίζεται συχνά κατά τον σχεδιασμό μιας βάσης. Το σχεδιαστικό πρότυπο καθορίζει τις σχέσεις που δημιουργούνται και τις συσχετίσεις που εμφανίζονται μεταξύ τους. Κάθε σχεδιαστικό πρότυπο εστιάζει στην επίλυση ενός συγκεκριμένου προβλήματος, περιγράφοντας υπό ποιες συνθήκες μπορεί να εφαρμοστεί ως λύση και ποιες είναι οι αναμενόμενες συνέπειές του.

Τα τέσσερα βασικά χαρακτηριστικά των σχεδιαστικών προτύπων, γενικά, όπως αυτά παρουσιάζονται στο [8], είναι τα εξής:

1. *το όνομα του προτύπου*. Το όνομα του προτύπου θα πρέπει να περιγράφει με γενικό και συνοπτικό τρόπο το πρόβλημα που επιλύει. Η εύρεση ενός κατανοητού, γενικού και συνοπτικού ονόματος αποτελεί, ίσως, το πιο δύσκολο μέρος της σχεδίασης ενός προτύπου.
2. *το πρόβλημα που επιλύεται*. Ο ακριβής ορισμός του προβλήματος που επιλύει το πρότυπο, καθώς και των περιεχομένων του προβλήματος, αποτελεί σημαντικό στοιχείο για την κατανόηση του προτύπου. Ορίζοντας το πρόβλημα και όλες τις λειτουργικές απαιτήσεις που εμφανίζονται, περιγράφονται οι συνθήκες, υπό τις οποίες, μπορεί να εφαρμοστεί το σχεδιαστικό πρότυπο.
3. *η λύση*. Η απόδοση της λύσης του προβλήματος, δίδεται μέσω της γενικής περιγραφής των στοιχείων του σχεδιαστικού προτύπου καθώς και των συσχετίσεων που λαμβάνουν χώρα μεταξύ των στοιχείων αυτών.
4. *οι συνέπειες*. Η αξιολόγηση της σχεδιαστικής λύσης, περιγράφοντας τα πλεονεκτήματα και τα μειονεκτήματά της, συγκρινόμενης με άλλες εναλλακτικές λύσης, περιγράφει το κατά πόσο μπορεί να εφαρμοστεί ένα πρότυπο, και υπό ποιες προϋποθέσεις.

Βασιζόμενοι στα παραπάνω χαρακτηριστικά που παρουσιάζουν τα πρότυπα σχεδίασης ενός λογισμικού, ορίζουμε την δομή των σχεδιαστικών προτύπων για τις βάσεις δεδομένων. Τα βασικά χαρακτηριστικά ενός προτύπου, προσανατολισμένου στις βάσεις, είναι τα εξής:

1. *Όνομα*. Η επιλογή του ονόματος κάθε προτύπου, πραγματοποιήθηκε προσεκτικά ώστε να απεικονίζει συνοπτικά και με σαφήνεια το πρόβλημα που επιλύεται. Τα ονόματα που επιλέχθηκαν κυρίως, αποτελούν ορισμούς (ή ιδιότητες) των βάσεων δεδομένων, ή αντιστοιχίζονται σε αφαιρετικούς μηχανισμούς, όπως αυτοί παρουσιάστηκαν στο Κεφάλαιο 2.
2. *Κίνητρο & εφαρμογή*. Σε αυτό το σημείο περιγράφεται το πρόβλημα λεκτικά και γραφικά. Η λεκτική περιγραφή του προβλήματος περιλαμβάνει την απόδοση των σχέσεων που λαμβάνουν μέρος, των συσχετίσεων που παρατηρούνται μεταξύ τους, και γενικά όλων των περιορισμών, π.χ., βαθμός πληθικότητας, περιορισμούς αναφορικής ακεραιότητας, κ.α., που εμφανίζονται. Η γραφική αναπαράσταση πραγματοποιείται κυρίως με UML μοντέλα, ή ER σχήματα και αφορούν το συγκεκριμένο πρόβλημα. Παρουσιάζοντας ένα συγκεκριμένο πρόβλημα, μπορούμε να κατανοήσουμε πότε εφαρμόζεται το προτεινόμενο πρότυπο. Το πρότυπο, το οποίο περιγράφεται και αυτό λεκτικά, παρέχει την γενική λύση παρόμοιων προβλημάτων. Επιπλέον παρουσιάζονται και οι τυχόν εναλλακτικές λύσεις που μπορούν να χρησιμοποιηθούν για την επίλυση του προβλήματος.
3. *Δομή*. Η παρουσίαση της δομής του σχήματος του προτύπου, σε φυσικό επίπεδο, πραγματοποιείται με την βοήθεια UML σχημάτων. Το σχήμα του προτύπου εμφανίζεται τόσο για τη γενική περίπτωση, δηλαδή παρουσιάζοντας τις σχέσεις και τις συσχετίσεις με γενικά πεδία και περιορισμούς, όσο και για το συγκεκριμένο πρόβλημα, όπου οι σχέσεις εμφανίζουν τα γνωρίσματα και τους περιορισμούς του προβλήματος. Επιπλέον, για κάθε σχέση του προτύπου παρουσιάζεται ένα στιγμιότυπο της. Είναι δυνατό σε ένα πρότυπο να αντιστοιχίζονται περισσότερες της μιας σχεδιαστικής λύσης, σε αυτή την περίπτωση παρουσιάζονται το σχήμα και το στιγμιότυπο της κάθε μίας από αυτές. Τέλος, θα πρέπει να αναφέρουμε ότι σε αυτό το στάδιο, αναφέρονται και ορισμένες δομές που μπορούν να χρησιμοποιηθούν κατά την κατασκευή σε φυσικό επίπεδο αναπαράστασης,

π.χ., ευρετήρια, triggers, cluster, κ.α., οι οποίες βελτιώνουν την διοργάνωση, την αναζήτηση και την διατήρηση της βάσης.

4. *Συμπεριφορά προτύπου σε επίπεδο στιγμιότυπου*. Σε αυτό το στάδιο μελετάται ο τρόπος με τον οποίον αντιμετωπίζονται οι αλλαγές σε επίπεδο στιγμιότυπου (επίπεδο εγγραφών) από το πρότυπο. Συγκεκριμένα, μελετώνται το κατά πόσο επηρεάζουν οι λειτουργίες *εισαγωγή*/*διαγραφή*/ *ανανέωση μιας πλειάδας* ενός πίνακα του προτύπου, τη συνολική συμπεριφορά του προτύπου. Δηλαδή ποιες είναι οι αλλαγές, οι οποίες πραγματοποιούνται σε επίπεδο πλειάδων με την εφαρμογή μιας εκ των παραπάνω λειτουργιών. Επιπλέον μελετάται, όταν κρίνεται σκόπιμο, η περίπτωση *επιλογής όλης της πληροφορίας* που είναι καταχωρημένη σε όλους τους πίνακες του σχήματος, ή *επιλογή μιας συγκεκριμένης πλειάδας* ενός πίνακα. Μέσω της μελέτης της συμπεριφοράς του προτύπου σε επίπεδο στιγμιότυπου, πραγματοποιείται αξιολόγηση του κόστους διατήρησης της συνέπειας της βάσης (με την εφαρμογή των λειτουργιών εισαγωγή/διαγραφή/ ανανέωση), αλλά και κοστολόγηση της ανάκτησης πληροφορίας, δίδοντας τους τρόπους, με τους οποίους μπορεί να πραγματοποιηθεί η λειτουργία της επιλογής στις συγκεκριμένες σχέσεις.
5. *Συμπεριφορά προτύπου σε επίπεδο σχήματος*. Η μελέτη της συμπεριφοράς του προτύπου σε επίπεδο σχήματος, αφορά στον τρόπο με τον οποίο αντιμετωπίζεται η εφαρμογή των λειτουργιών *εισαγωγή*/*διαγραφή*/*ανανέωση ενός πεδίου ή ενός περιορισμού* μιας σχέσης του προτύπου. Σε αυτή την περίπτωση παρουσιάζεται το κόστος διατήρησης της βάσης κατά την εισαγωγή νέων λειτουργικών απαιτήσεων, δηλαδή το κατά πόσο και σε ποιο βαθμό θα πρέπει να αλλάξει το σχήμα της βάσης ώστε να απεικονίζει τα νέα δεδομένα της μοντελοποίησης του προβλήματος.
6. *Πλεονεκτήματα & Μειονεκτήματα*. Με βάση τις παρατηρήσεις της συμπεριφοράς του προτύπου σε επίπεδο στιγμιότυπου και σχήματος, παρουσιάζονται, συνοπτικά, τα πλεονεκτήματα και τα μειονεκτήματα του προτύπου σε σχέση με τις εναλλακτικές λύσεις που είναι δυνατό να χρησιμοποιηθούν. Η αξιολόγηση πραγματοποιείται κυρίως βάσει του κόστους σε χρόνο εκτέλεσης ορισμένων λειτουργιών ή δομών, που θεωρούνται απαραίτητες κατά τη διατήρηση της βάσης (π.χ., triggers,

λειτουργίες συνένωσης και ένωσης, κ.α.), και του κόστους σε χώρο αποθήκευσης των σχέσεων αλλά και δομών οργάνωσης της βάσης (π.χ., ευρετήρια, clusters, κ.α.), που απαιτούνται για την εφαρμογή της κάθε λύσης.

7. *Κατασκευή*. Ο τρόπος κατασκευής του προτύπου δίδεται μέσω της `SQL-1999` μόνο για πρότυπα που παρουσιάζουν κατασκευαστικό ενδιαφέρον. Υπάρχουν, δηλαδή, περιπτώσεις που παραλείπεται η παρουσίαση της κατασκευής του προτύπου, λόγω απλότητας, δηλαδή περιλαμβάνουν μόνο την κατασκευή πινάκων, κλειδιών και ξένων κλειδιών. Σε ορισμένες περιπτώσεις περιγράφεται η υλοποίηση των λύσεων σε `Oracle10g`.

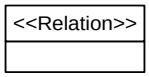
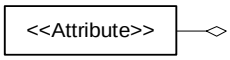
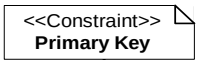
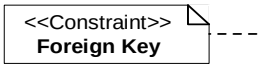
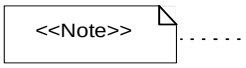
3.3. Οργάνωση των σχεδιαστικών προτύπων

Τα πρότυπα που παρουσιάζονται στο επόμενο κεφάλαιο, μπορούν να κατηγοριοποιηθούν ως εξής:

- *Keys & Values*: Τεχνητό Κλειδί (Artificial Key), Παραγόμενα Πεδία (Derived Attributes).
- *Normal Forms*: Κανονικές μορφές (1NF, 2NF, 3NF), Συνάθροιση-Σύνθεση (Aggregation-Composition), Γενίκευση (IsA), Materialization.
- *Special Cases*: Αξονική Περιστροφή (Pivot Case), Αναδρομικότητα & Γράφοι (Transitive Closure & Graphs), Configuration.

3.4. Συμβολισμοί σχεδίασης δομής προτύπων

Η σχηματική αναπαράσταση της δομής των προτύπων πραγματοποιείται με την βοήθεια `UML` μοντέλων απεικόνισης. Οι συμβολισμοί που παρουσιάζονται στα σχήματα του επομένου κεφαλαίου είναι οι εξής:

Συμβολισμοί	**Περιγραφή**
	Σχέση (πίνακας).
	Γνώρισμα μιας σχέσης.
	Πρωτεύον κλειδί μιας σχέσης, με την μορφή περιορισμού.
	Περιορισμός αναφορικής ακεραιότητας (ξένο κλειδί).
	Σημειώσεις που βοηθάνε στην κατανόηση της σχεδίασης (π.χ., υποψήφια κλειδιά, συναρτησιακές εξαρτήσεις, πλειότιμα πεδία, σύνθετα πεδία, κ.α.).

Σχήμα 3.1 Περιγραφή συμβολισμών, που χρησιμοποιούνται για την αναπαράσταση των προτύπων.

ΚΕΦΑΛΑΙΟ 4. ΣΧΕΔΙΑΣΤΙΚΑ ΠΡΟΤΥΠΑ (DESIGN PATTERNS)

4.1. Τεχνητό Κλειδί (Artificial Key)

4.1.1. Κίνητρο & Εφαρμογή

Ας θεωρήσουμε ότι έχουμε μια βάση δεδομένων της οποίας ένας πίνακας αποθηκεύει τα στοιχεία κάθε χρήστη που γίνεται μέλος σε κάποια ομάδα (π.χ. σε ένα forum). Ο πίνακας έχει την εξής δομή:

```
USER (Name, E-Mail, Age, Job, ...)
```

με υποψήφια κλειδιά: (i) το γνώρισμα `E-Mail`, και (ii) τον συνδυασμό όλων των γνωρισμάτων. Σε μια τέτοια περίπτωση παρατηρούνται τα εξής προβλήματα:

1. Υπάρχει πιθανότητα το υποψήφιο κλειδί να αλλάξει ή να παύσει να ισχύει μετά από ένα χρονικό διάστημα. Π.χ., σε ορισμένες περιπτώσεις το `E-Mail` παύει να ισχύει μετά από 30 ημέρες μη λειτουργίας του.

2. Το υποψήφιο κλειδί E-Mail δεν είναι ακέραιος αριθμός, με αποτέλεσμα η ευρετηριοποίηση αυτού του πίνακα να καθίσταται δύσκολη.
3. Στην περίπτωση του συνδυασμού γνωρισμάτων ως πρωτεύοντος κλειδιού δεν είναι δυνατή η εισαγωγή τιμών NULL σε κανένα από τα γνωρίσματα που αποτελούν το πρωτεύον κλειδί.

Το πιο συνηθισμένο πρόβλημα που προκύπτει κατά τη σχεδίαση της βάσης δεδομένων, αφορά στο χρόνο ζωής του πρωτεύοντος κλειδιού, όπως στο παράδειγμα. Κάθε οντότητα που μοντελοποιείται απεικονίζει μια φυσική έννοια του πραγματικού κόσμου, η οποία έχει κάποιες ιδιότητες. Με το πέρασμα του χρόνου υπάρχει πιθανότητα να αλλάξει η δομή της οντότητας, διαφοροποιώντας το σχήμα της απεικόνισης (διαγραφή ή εισαγωγή πεδίων), η δομή κάποιας ιδιότητας, διαφοροποιώντας το πεδίο ορισμού κάποιου γνωρίσματος, ή ακόμα και να αλλάξει τιμή κάποια από τις ιδιότητες της οντότητας, ανανεώνοντας την τιμή κάποιας εγγραφής του συγκεκριμένου πεδίου. Ας θεωρήσουμε το παράδειγμα όπου η ιδιότητα Ταυτότητα χαρακτηρίζει κάθε οντότητα Άνθρωπος, η ιδιότητα αυτή κάποια στιγμή μπορεί να παύσει να ισχύει (π.χ., να ανακαλυφθεί άλλος τρόπος ταυτοποίησης κάθε ανθρώπου), μπορεί να αλλάξει δομή (π.χ., να μην είναι τύπου συμβολοσειράς) ή ακόμα πιο πιθανόν να αλλάξει τιμή για κάποιον συγκεκριμένο άνθρωπο (στην περίπτωση όπου κάποιος χάνει την ταυτότητά του και εκδίδει μια καινούρια). Οτιδήποτε, λοιπόν, έχει φυσική υπόσταση μπορεί να μεταβληθεί με το χρόνο, οπότε έχει περιορισμένο χρόνο ζωής.

Εκτός από τον περιορισμένο χρόνο ζωής, υπάρχουν και άλλα προβλήματα τα οποία εμφανίζονται στις περισσότερες περιπτώσεις, όπου το γνώρισμα, το οποίο επιλέγεται ως πρωτεύον κλειδί, είναι τύπου συμβολοσειράς (*string*) (π.χ., E-MAIL, license_number, ΑΡΙΘΜΟΣ_ΤΑΥΤ,…). Τα σημαντικότερα προβλήματα που εμφανίζονται στην περίπτωση επιλογής γνωρίσματος string ως πρωτεύον κλειδί, είναι:

- η δύσκολη ευρετηριοποίηση και αναζήτηση πληροφορίας. Ακόμα και στην περίπτωση δημιουργίας μη πυκνού ευρετηρίου, το οποίο δεν αποθηκεύει όλες τις τιμές του πεδίου και συνεπώς δεν καταλαμβάνει πολύ χώρο, η αναζήτηση μιας πληροφορίας τύπου string βασίζεται στη σύγκριση συμβολοσειρών, μια διαδικασία αρκετά χρονοβόρα για το σύστημα.

- η *επανάληψη* (*replication*) της πληροφορίας του πρωτεύοντος κλειδιού μέσω του *ξένου κλειδιού* (*foreign key*) σε πολλούς πίνακες. Ας υποθέσουμε ότι δύο πίνακες συνδέονται μεταξύ τους με αναφορά ξένου κλειδιού, και ότι απαιτείται για την εύρεση κάποιας δεδομένης πλειάδας ο έλεγχος και τον δύο πινάκων. Δεδομένης της δύσκολης αναζήτησης και ευρετηριοποίησης των συμβολοσειρών, η σύγκριση αυτών των γνωρισμάτων απαιτεί αρκετό χρόνο για την πραγματοποίησή της.
- η δύσκολη *συνένωση* (*join*) πινάκων για ανάκτηση πληροφορίας. Η συνένωση δύο πινάκων ως προς δύο γνωρίσματα, ακόμα και τύπου integer, αποτελεί μια χρονοβόρα διαδικασία εκ φύσεως. Είναι προφανές ότι όταν τα γνωρίσματα της συνένωσης είναι τύπου string, με βάση τις προαναφερθέντες ιδιαιτερότητες που εμφανίζουν οι συμβολοσειρές, το σύστημα επιβαρύνεται επιπλέον τόσο σε πόρους όσο και σε χρόνο εκτέλεσης.

Παρόμοια προβλήματα με την επιλογή μιας συμβολοσειράς ως πρωτεύον κλειδί, παρατηρούνται και κατά την επιλογή συνδυασμού κλειδιών, τα οποία εστιάζονται στην περίπτωση δημιουργίας ξένου κλειδιού. Το ξένο κλειδί που δημιουργείται είναι πολύπλοκο, με συνέπεια η συνένωση των πινάκων να είναι χρονοβόρα.

Για την αντιμετώπιση των παραπάνω προβλημάτων, η προτεινόμενη λύση είναι η εισαγωγή ενός τεχνητού κλειδιού. Το *τεχνητό κλειδί* (*artificial key*) είναι ένας ακολουθιακά αυξανόμενος ακέραιος, που ορίζεται από τον σχεδιαστή ως γνώρισμα του πίνακα και το οποίο παίζει το ρόλο του πρωτεύοντος κλειδιού. Στον πραγματικό κόσμο, το γνώρισμα αυτό δεν αποτελεί μέρος της οντότητας που μοντελοποιείται και δεν έχει καμία φυσική υπόσταση. Ο ρόλος του είναι καθαρά βοηθητικός σε περιπτώσεις όπου παρουσιάζονται τα παραπάνω προβλήματα με τα υποψήφια κλειδιά ή όταν δεν υπάρχει καν κάποιο υποψήφιο κλειδί. Ο αυτοματοποιημένος τρόπος δημιουργίας του τεχνητού κλειδιού και καταχώρησης των τιμών του δεν επιτρέπει καμία ανανέωση, από πλευράς χρήστη, στο γνώρισμα αυτό. Η μέγιστη τιμή του γνωρίσματος αυτού προσδιορίζει κάθε φορά το πλήθος των εγγραφών που έχουν καταχωρηθεί συνολικά στον πίνακα, ασχέτως με το πλήθος των εγγραφών που βρίσκονται στο πίνακα, έως εκείνη τη στιγμή.

Η κατασκευή ενός τεχνητού κλειδιού μπορεί να υλοποιηθεί με τους παρακάτω τρόπους:

1. με την δήλωση μιας ακολουθίας ακεραίων αριθμών, τύπου *sequence*, με αρχική τιμή `1` η οποία αυξάνεται κατά `1` κάθε φορά. Αυτή η κατασκευαστική λύση αποτελεί μια, πλήρως, αυτοματοποιημένη διαδικασία και δεν επιτρέπει την παρέμβαση του χρήστη σε καμία περίπτωση. Η δήλωση μιας ακολουθίας καθώς και οι παράμετροί της, δίνονται στην παράγραφο 4.1.6.
2. με την κατασκευή ενός επιπλέον πεδίου σε κάθε πίνακα, τύπου integer. Για γρήγορη αναζήτηση της μέγιστης τιμής του τεχνητού κλειδιού κάθε πίνακα, δημιουργείται ένας *συμβουλευτικός πίνακας* (*lookup table*), στον οποίο καταχωρούνται τα ονόματα των πινάκων που περιλαμβάνουν τεχνητά κλειδιά καθώς και οι μέγιστες τιμές των κλειδιών αυτών. Η αύξηση της τιμής ενός τεχνητού κλειδιού περιλαμβάνει τις παρακάτω διεργασίες:
   i. εύρεση της αντίστοιχης εγγραφής στον συμβουλευτικό πίνακα.
   ii. αύξηση της τιμής κατά `1`.
   iii. ανανέωση της αντίστοιχης πλειάδας του συμβουλευτικού πίνακα.

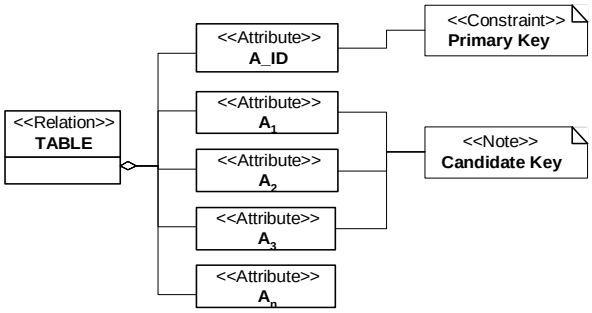
   Η διαδικασία ανανέωσης της τιμής ενός τεχνητού κλειδιού του συμβουλευτικού πίνακα μπορεί να πραγματοποιηθεί αυτόματα από το σύστημα, είτε adhoc από τον χρήστη. Για την διασφάλιση του *ACID*, η διαδικασία της ανανέωσης της τιμής ενός τεχνητού κλειδιού θα πρέπει να συνοδεύεται από τα αντίστοιχα κλειδώματα των εγγραφών στις οποίες πραγματοποιείται η ανανέωση.
3. με την εισαγωγή ενός επιπλέον πεδίου τύπου integer ως τεχνητό κλειδί. Για την εισαγωγή μιας νέας τιμής στο πεδίο θα πρέπει κάθε φορά να ελεγχθούν όλες οι εγγραφές για την εύρεση της μέγιστης τιμής στο συγκεκριμένο πεδίο και μετά αύξηση αυτής της τιμής κατά ένα. Εάν η βάση δεδομένων διαχειρίζεται μόνο από έναν χρήστη, η παραπάνω διαδικασία μπορεί να πραγματοποιηθεί με χρήση της δήλωσης `SELECT` *`max(ARRAY.AT_Key)+1`* `FROM` *`ARRAY`* . Στην περίπτωση όπου περισσότεροι χρήστες διαχειρίζονται την βάση δεδομένων θα πρέπει κάθε φορά, πριν την εφαρμογή οποιασδήποτε νέας εισαγωγής, να κλειδώνονται οι αντίστοιχες εγγραφές ή ο πίνακας. Αυτή η κατασκευαστική λύση συνήθως αποφεύγεται διότι απαιτείται η ανάγνωση όλων των εγγραφών του πίνακα κάθε φορά που λαμβάνει χώρα μία νέα

εισαγωγή σε επίπεδο στιγμιότυπου. Σε αυτή την παράγραφο μελετούνται μόνο οι δύο πρώτες κατασκευαστικές λύσεις.

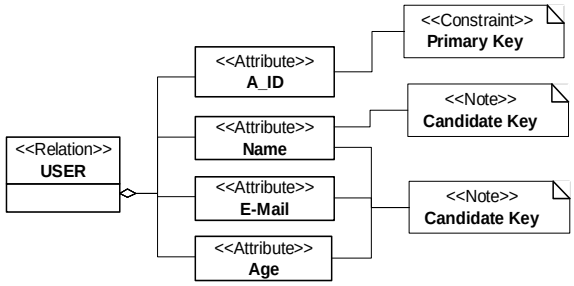
Αν και το τεχνητό κλειδί αποτελεί μια ελκυστική λύση, αρκετοί σχεδιαστές προτείνουν τον περιορισμό της χρήσης του μόνο σε περιπτώσεις όπου δεν υπάρχει άλλο υποψήφιο κλειδί. Π.χ., στην περίπτωση `COUNTRY(Name, Code)`, με υποψήφιο κλειδί το `Code`, εισάγονται το όνομα και ο κωδικός της χώρας. Η εισαγωγή ενός τεχνητού κλειδιού, `COUNTRY(A_ID, Name, Code)`, θα αποτελούσε μια πλεονάζουσα πληροφορία, η οποία επιπλέον δεν είναι κατανοητή στο χρήστη. Π.χ., ο χρήστης είναι σε θέση να καταλάβει τον κωδικό στην καταχώρηση `COUNTRY(France, FR)` αλλά δεν μπορεί να κατανοήσει την τιμή `1` στη `COUNTRY(1, France, FR)`. Μία λύση, για το πρόβλημα της κατανόησης του τεχνητού κλειδιού από πλευράς χρήστη, είναι η εσκεμμένη απόκρυψη της πληροφορίας σχετικά με αυτό. Δηλαδή, στο παραπάνω παράδειγμα του πίνακα `COUNTRY` στον χρήστη θα παρέχεται μόνο η πληροφορία των γνωρισμάτων `Name` και `Code`, αποκρύπτοντας πλήρως την ύπαρξη του πεδίου `A_ID`.

4.1.2. Δομή

Στο Σχήμα 4.1 παρουσιάζεται ένα γενικό σχήμα αναπαράστασης ενός πίνακα, ο οποίος μπορεί να περιλαμβάνει είτε ένα υποψήφιο κλειδί τύπου συμβολοσειράς, είτε ένα υποψήφιο κλειδί με συνδυασμό γνωρισμάτων. Στη συγκεκριμένη σχεδίαση επιλέχθηκε η εισαγωγή ενός τεχνητού κλειδιού `A_ID`. Στο Σχήμα 4.2, όπου αναπαρίσταται η δομή του σχήματος της βάσης του παραδείγματος, παρουσιάζονται τα υποψήφια κλειδιά, όπως αυτά προκύπτουν από την κατανόηση της οντότητας του φυσικού κόσμου που μοντελοποιείται, καθώς επίσης και η επιλογή εισαγωγής τεχνητού κλειδιού ως πρωτεύον κλειδί. Ένα στιγμιότυπο του προηγούμενου σχήματος της βάσης του παραδείγματος δίδεται στο Σχήμα 4.3.



Σχήμα 4.1 Αναπαράσταση δομής σχήματος ενός τυχαίου πίνακα μιας βάσης δεδομένων, ο οποίος περιλαμβάνει ως υποψήφια κλειδιά είτε ένα string, είτε συνδυασμό γνωρισμάτων.

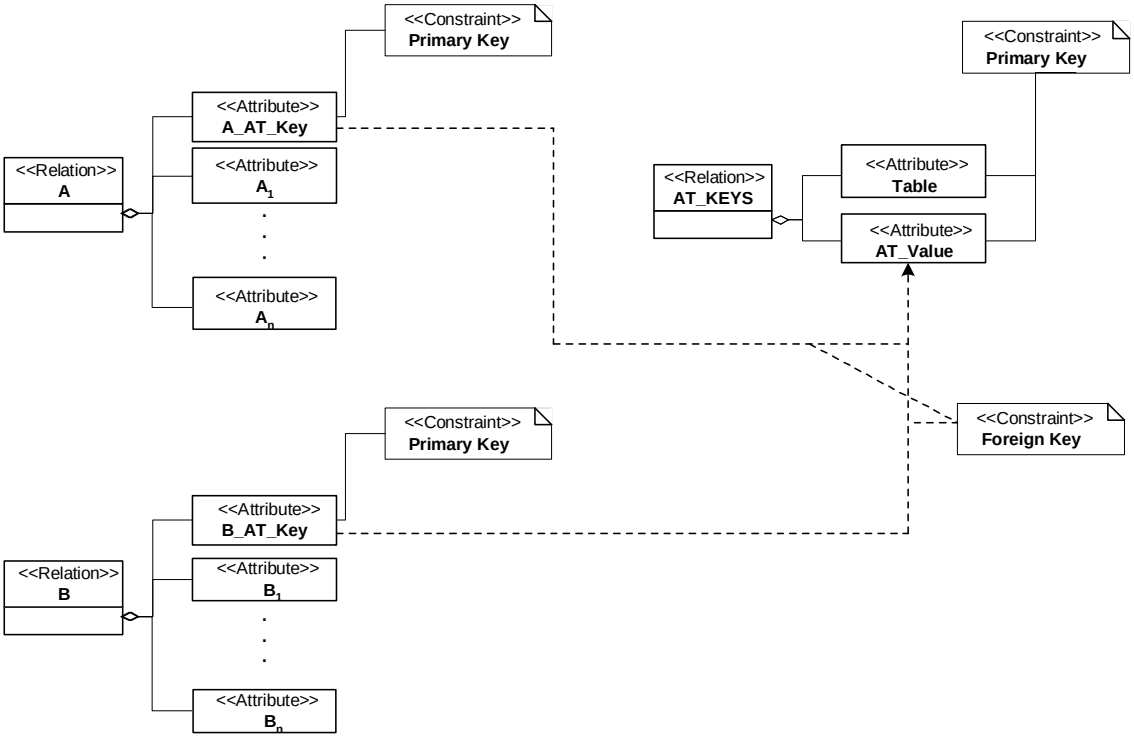


Σχήμα 4.2 Αναπαράσταση δομής σχήματος της βάσης του παραδείγματος με την εισαγωγή τεχνητού κλειδιού.

USER

A_ID	Name	E-Mail	Age
1	Jenny	jstath@cs.uoi.gr	27
2	Tom	Tomy@yahoo.gr	32
3	PVassil	PVassil@cs.uoi.gr	37
4	Mary	mghs@hotmail.com	21
5	Kelly	KPan@yahoo.gr	29

Σχήμα 4.3 Αναπαράσταση δομής στιγμιότυπου της βάσης του παραδείγματος με την εισαγωγή τεχνητού κλειδιού.



Σχήμα 4.4 Αναπαράσταση δομής με την δημιουργία συμβουλευτικού πίνακα καταχώρησης της μέγιστης τιμής του τεχνητού κλειδιού κάθε πίνακα.

Στο Σχήμα 4.4 παρουσιάζεται η δομή του σχήματος μιας βάσης δεδομένων, όπου οι πίνακες `A` και `B` περιλαμβάνουν τα τεχνητά κλειδιά `A_AT_Key` και `B_AT_Key` αντίστοιχα, ενώ στον πίνακα `AT_KEYS` καταχωρούνται οι πίνακες που περιλαμβάνουν τεχνητά κλειδιά καθώς και οι μέγιστες τιμές των κλειδιών. Ο πίνακας `AT_KEYS` κατασκευάστηκε να έχει ως πρωτεύον κλειδί το γνώρισμα `Table`. Εφόσον κάθε πίνακας έχει μοναδικό όνομα σε μια βάση δεδομένων, με την επιλογή του γνωρίσματος `Table` ως πρωτεύον κλειδί εξασφαλίζεται η μοναδικότητα των εγγραφών του πίνακα `AT_KEYS`. Τα πεδία `A1,..,` `An` και `B1,..,` `Bn` αποτελούν γνωρίσματα των πινάκων `A` και `B` αντίστοιχα. Το Σχήμα 4.5 αποτελεί στιγμιότυπο του συμβουλευτικού πίνακα της παραπάνω δομής.

AT_KEYS

Table	AT_Value
A	40
B	30
Emp	100
User	5000

Σχήμα 4.5 Αναπαράσταση στιγμιότυπου του συμβουλευτικού πίνακα αποθήκευσης της μέγιστης τιμής του τεχνητού κλειδιού κάθε πίνακα.

4.1.3. Συμπεριφορά σε επίπεδο στιγμιότυπου

4.1.3.1. Εισαγωγή

Κατά την εισαγωγή μιας νέας εγγραφής, στο πεδίο `A_ID` χρησιμοποιείται ένας ακολουθιακά αυξανόμενος ακέραιος, ο οποίος μπορεί να αρχικοποιηθεί στην τιμή `A_ID = 1` για την πρώτη εγγραφή και κάθε φορά να αυξάνεται κατά ένα. Υπάρχουν δύο τρόποι εισαγωγής μιας πλειάδας στον πίνακα της βάσης, ανάλογα με τον τρόπο κατασκευής του τεχνητού κλειδιού. Στην περίπτωση κατασκευής *αυτοματοποιημένης ακολουθίας* (*sequence*), η εισαγωγή της επόμενης τιμής του τεχνητού κλειδιού πραγματοποιείται αυτόματα από το σύστημα χωρίς επέμβαση από το χρήστη. Ο δεύτερος τρόπος κατασκευής του τεχνητού κλειδιού, προϋποθέτει την πραγματοποίηση μιας *συναλλαγής* (*transaction*). Στην περίπτωση που δεν υποστηρίζεται η δημιουργία sequence, μπορούμε καταχωρούμε σε έναν συμβουλευτικό – ξεχωριστό – πίνακα, τις μέγιστες τιμές των τεχνητών κλειδιών που χρησιμοποιούνται στους διάφορους πίνακες. Ας υποθέσουμε ότι, ο πίνακας `AT_Keys(Table, AT_Value)` αποτελεί συμβουλευτικό πίνακα, όπου καταχωρείται κάθε πίνακας που συμπεριλαμβάνει τεχνητό κλειδί καθώς και η μέγιστη τιμή του στον πίνακα αυτό. Για την εισαγωγή μιας νέας εγγραφής στον πίνακα `Emp` του Σχήματος 4.5, η συναλλαγή ξεκινάει με το κλείδωμα της αντίστοιχης εγγραφής `(Emp, 100)` του συμβουλευτικού πίνακα, ακολουθείται η ανάγνωση της αντίστοιχης τιμής

του τεχνητού κλειδιού και η αύξηση της τιμής αυτής κατά ένα, ενώ σε τελικό βήμα ανανεώνεται η αντίστοιχη εγγραφή του συμβουλευτικού πίνακα σε `(Emp, 101)`.

4.1.3.2. Διαγραφή

Η εφαρμογή της λειτουργίας διαγραφής μιας ή περισσοτέρων εγγραφών δεν παρουσιάζει καμία διαφορά όταν ο πίνακας περιλαμβάνει τεχνητό κλειδί. Το μόνο μειονέκτημα που παρουσιάζεται, κατά την διαγραφή πολλών εγγραφών από τον πίνακα καταχώρησης, είναι η μεγάλη απόκλιση μεταξύ των τιμών του τεχνητού κλειδιού. Π.χ., υπάρχει πιθανότητα μετά την εφαρμογή πολλαπλών διαγραφών πλειάδων, ο πίνακας να αποτελείται από 5 εγγραφές ενώ η μέγιστη τιμή στο πεδίο του τεχνητού κλειδιού να είναι ο αριθμός 50.

4.1.3.3. Ανανέωση

Εν γένει, η ανανέωση της τιμής του τεχνητού κλειδιού κρίνεται σκόπιμο να αποφεύγεται κυρίως για την διασφάλιση της συνέπειας της βάσης. Στην περίπτωση κατασκευής τεχνητού κλειδιού με τη δημιουργία sequence, δεν επιτρέπεται καμία επέμβαση από το χρήστη στο πεδίο του τεχνητού κλειδιού, οπότε δεν υποστηρίζεται η λειτουργία της ανανέωσης. Ακόμα και στην περίπτωση κατασκευής συμβουλευτικού πίνακα, η ανανέωση της τιμής του τεχνητού κλειδιού ενός πίνακα συνίσταται να αποφεύγεται. Η μόνη ανανέωση πραγματοποιείται στις εγγραφές του συμβουλευτικού πίνακα, όπως προαναφέρθηκε.

4.1.4. Συμπεριφορά σε επίπεδο σχήματος

Η δυνατότητα εκτέλεσης των λειτουργιών *εισαγωγή*, *διαγραφή* και *ανανέωση* σε οποιοδήποτε γνώρισμα, πλην του τεχνητού κλειδιού, ή/και σε κάποιον περιορισμό της βάσης δεδομένων, είναι πραγματοποιήσιμη χωρίς να επηρεάζει τη συμπεριφορά του προτύπου σε επίπεδο σχήματος. Εξάλλου ο αυτοματοποιημένος τρόπος χειρισμού του τεχνητού κλειδιού, εκ κατασκευής, δεν επιτρέπει, από πλευράς χρήστη, την εφαρμογή καμίας από τις παραπάνω λειτουργίες στο τεχνητό κλειδί.

4.1.5. Πλεονεκτήματα & Μειονεκτήματα

Με την εισαγωγή του τεχνητού κλειδιού:

- περιορίζεται το φαινόμενο του περιορισμένου χρόνου ζωής που παρατηρείται σε όλα τα γνωρίσματα, τα οποία αποτελούν ιδιότητες της φυσικής οντότητας που μοντελοποιείται. Ορίζοντας ένα τεχνητό κλειδί, το οποίο είναι ανεξάρτητο της οντότητας που μοντελοποιείται, δημιουργούμε ένα γνώρισμα στον πίνακα απεικόνισης το οποίο είναι αμετάβλητο ως προς την δομή και την τιμή. Για να αντιληφθούμε το μέγεθος αυτής της ιδιότητας του τεχνητού κλειδιού, αρκεί να σκεφτούμε την περίπτωση όπου υπάρχουν πίνακες οι οποίοι συνδέονται μεταξύ τους με αναφορές εξωτερικού κλειδιού. Εάν έχουμε επιλέξει ένα φυσικό γνώρισμα ως πρωτεύον κλειδί, οποιαδήποτε αλλαγή παρουσιαστεί θα πρέπει να εφαρμοστεί σε κάθε εμφάνιση του γνωρίσματος αυτού, δηλαδή σε όλους τους πίνακες που το περιλαμβάνουν. Εάν όμως επιλεχθεί ένα τεχνητό κλειδί, λόγω του αυτοματοποιημένου τρόπου εισαγωγής και διαχείρισής του, το παραπάνω φαινόμενο της «αλυσιδωτής» ανανέωσης αποφεύγεται.
- πραγματοποιείται εύκολη και γρήγορη ευρετηριοποίηση και αναζήτηση του πίνακα. Επιλέγοντας ως πρωτεύον κλειδί το τεχνητό κλειδί, έναντι ενός φυσικού κλειδιού το οποίο είτε προσδιορίζεται από μια συμβολοσειρά, είτε αποτελεί συνδυασμό περισσοτέρων γνωρισμάτων, μειώνεται αισθητά ο χώρος που απαιτείται για την αποθήκευση του ευρετηρίου καθώς και ο χρόνος αναζήτησης μέσω του ευρετηρίου.
- η συνένωση πινάκων με βάση το πρωτεύον κλειδί είναι πιο γρήγορη. Είναι προφανές ότι, στην περίπτωση όπου το πρωτεύον κλειδί αποτελείται από συνδυασμό δύο ή περισσότερων γνωρισμάτων, η συνένωση, ακόμα και δύο μόνο πινάκων, με βάση τα γνωρίσματα αυτά είναι αρκετά δύσκολη και καθυστερεί αρκετά η εκτέλεση της. Το πρόβλημα αυτό διογκώνεται όταν κάποιο από τα γνωρίσματα αυτά είναι συμβολοσειρά.
- εισάγεται ένα επιπλέον γνώρισμα στη βάση, το οποίο είναι μεν κατανοητό στο σχεδιαστή και στον κατασκευαστή της βάσης, αλλά όχι και στον χρήστη. Από τη στιγμή που το τεχνητό κλειδί δεν είναι μέρος της οντότητας, δηλαδή η ύπαρξή του δεν της προσδίδει κάποια συγκεκριμένη φυσική ιδιότητα, δεν είναι άμεσα κατανοητή η σημασία του. Ο χρήστης μπορεί απλά να γνωρίζει

την ύπαρξη ενός αριθμού, ο οποίος προσδιορίζει μοναδικά κάθε πλειάδα, τον οποίον δεν μπορεί να μεταβάλλει αλλά ούτε και να ερμηνεύσει σε σχέση με την οντότητα που μοντελοποιείται.

- θα πρέπει να δημιουργηθεί δευτερεύον ευρετήριο στα γνωρίσματα του φυσικού κλειδιού για διασφάλιση της μοναδικότητας των γνωρισμάτων αλλά και για γρήγορη αναζήτηση με βάση τα γνωρίσματα αυτά. Ανεξάρτητα από την ύπαρξη τεχνητού κλειδιού, τα γνωρίσματα του φυσικού κλειδιού της οντότητας θα πρέπει να παρουσιάζουν μοναδικές τιμές για κάθε εγγραφή στο πίνακα απεικόνισης. Π.χ., στην περίπτωση της οντότητας `USER`, το πεδίο `E-mail` αποτελεί φυσικό κλειδί, προσδιορίζοντας μοναδικά κάθε καταχώρηση στο πίνακα καταχωρήσεων, οπότε θα πρέπει, ακόμα και στην περίπτωση εισαγωγής τεχνητού κλειδιού, να διασφαλίζεται, με κάποιον τρόπο, η μοναδικότητα των τιμών του πεδίου `E-mail`. Ένας τρόπος εξασφάλισης της μοναδικότητας των τιμών των γνωρισμάτων του φυσικού κλειδιού, είναι η εισαγωγή ενός δευτερεύοντος ευρετηρίου πάνω στα γνωρίσματα αυτά. Επιπλέον, δεδομένου ότι το τεχνητό κλειδί δεν προσδίδει πληροφορία σχετική με την οντότητα και τις περισσότερες φορές η τιμή του δεν είναι καν προσβάσιμη από πλευράς χρήστη, η πιθανότητα αναζήτησης κάποιας εγγραφής με βάση την τιμή του τεχνητού κλειδιού είναι μικρή. Με την εισαγωγή ενός δευτερεύοντος ευρετηρίου είναι δυνατή η πιο αποδοτική αναζήτηση οποιασδήποτε πληροφορίας σχετικής με την οντότητα η οποία είναι γνωστοποιημένη στον χρήστη.
- θα πρέπει σε κάθε περίπτωση εισαγωγής να βρίσκεται η μέγιστη τιμή του τεχνητού κλειδιού την δεδομένη στιγμή και να αυξάνεται κατά ένα. Η εύρεση της επόμενης μέγιστης τιμής πραγματοποιείται είτε αυτόματα από το σύστημα στην περίπτωση δημιουργία sequence, είτε με την αναζήτηση της στον συμβουλευτικό πίνακα. Κυρίως στην δεύτερη περίπτωση, η εύρεση της μέγιστης τιμής απαιτεί επιπλέον χρόνο, διότι πρέπει να αναζητηθεί η αντίστοιχη εγγραφή του συμβουλευτικού πίνακα
- υπάρχει μεγάλη πιθανότητα να απαιτούνται επιπλέον συνενώσεις πινάκων για την ανάκτηση κάποιας πληροφορίας, η οποία στην περίπτωση επιλογής του φυσικού κλειδιού να αποδίδονταν άμεσα. Π.χ., ας θεωρήσουμε το παράδειγμα

καταχώρησης πληροφορίας σχετικά με τους πελάτες μιας εταιρίας και τις παραγγελίες που κάνει κάθε πελάτης. Οι δύο σχεδιαστικές λύσεις δίνονται από τις παρακάτω σχέσεις:

- *με χρήση συνδυασμού γνωρισμάτων ως πρωτεύον κλειδί.*

  CUSTOMER (Cust_num, …)

  ORDER (Cust_num, Order_Date_Time, …,).

  Στη σχέση πελατών, CUSTOMER, εισάγεται από τον χρήστη ένας χαρακτηριστικός αριθμός, το Cust_num, ο οποίος προσδιορίζει μοναδικά κάθε πελάτη. Στη σχέση παραγγελιών, ORDER, καταχωρείται το χαρακτηριστικό κάθε πελάτη, Cust_num, ως αναφορά ξένου κλειδιού ως προς το αντίστοιχο πεδίο του CUSTOMER, καθώς και η ώρα και ημερομηνία της κάθε παραγγελίας, Order_Date_Time. Ο συνδυασμός των γνωρισμάτων Cust_num, Order_Date_Time αποτελούν το πρωτεύον κλειδί της σχέσης ORDER.

- *με χρήση τεχνητού κλειδιού*

  CUSTOMER (Cust_ID, Cust_num, …)

  ORDER (Order_ID, Cust_ID ,Order_Date_Time, …,).

  Για την αποφυγή του συνδυασμού γνωρισμάτων ως πρωτεύον κλειδί της σχέσης ORDER, εισάγεται το τεχνητό κλειδί Order_ID ενώ το γνώρισμα Cust_num εξακολουθεί να υπάρχει ως αναφορά ξένου κλειδιού. Αντίστοιχα στη σχέση CUSTOMER εισάγεται το τεχνητό κλειδί Cust_ID.

Για την εκτέλεση, π.χ., της ερώτησης «*Εύρεση όλων των πελατών και των παραγγελιών τους ως σήμερα*», ή οποιασδήποτε ερώτησης που συνδυάζει τα γνωρίσματα Cust_num και Order_Date_Time, είναι απαραίτητη η συνένωση των δύο πινάκων για την περίπτωση δημιουργίας τεχνητού κλειδιού. Στην περίπτωση συνδυασμού γνωρισμάτων ως πρωτεύον κλειδί αποφεύγεται η συνένωση των δύο πινάκων, εφόσον η πληροφορία των πελατών καταχωρείται ήδη στον πίνακα παραγγελιών.

4.1.6. Κατασκευή

Στην Oracle10g [13, σελ.991-994] η κατασκευή του τεχνητού κλειδιού πραγματοποιείται με τη δήλωση *CREATE SEQUENCE*, όπως παρουσιάζεται παρακάτω.

```
CREATE SEQUENCE  A_ID
START WITH       1
INCREMENT BY     1
NONMAXVALUE
NOCACHE
NOCYCLE
```

Η εύρεση της τρέχουσας τιμής του τεχνητού κλειδιού πραγματοποιείται με την δήλωση CURRVAL ενώ της επόμενης τιμής με την δήλωση NEXTVAL.

Η εισαγωγή μιας νέας εγγραφής στον πίνακα USER πραγματοποιείται με την εντολή INSERTION INTO *USER* VALUES (A_ID.nextval, 'kmasrk@gmail.com',…).

Η εύρεση της επόμενης τιμής της sequence πραγματοποιείται μέσω της εντολής

```
SELECT A_ID.nextval FROM USER.
```

Τέλος, εάν επιθυμούμε να αρχικοποιήσουμε εκ νέου την ακολουθία, αλλάζοντας κάποια από τις παραμέτρους της, θα πρέπει να την «καταστρέψουμε» DROP SEQUENCE και να την επανακατασκευάσουμε .

4.2. Κανονικές Μορφές (Normal Forms)

Οι κανονικές μορφές είναι ένα είδος διασφάλισης του καλού σχεδιασμού μιας βάσης δεδομένων, όπου αποφεύγεται το φαινόμενο της ασυνέπειας των εγγραφών κατά την εισαγωγή ή/και την ανανέωση, και το φαινόμενο του πλεονασμού κατά το οποίο εισάγεται η ίδια εγγραφή πολλές φορές. Ας υποθέσουμε, ότι ένα γνώρισμα X προσδιορίζει ένα άλλο γνώρισμα Y, δηλαδή λαμβάνει χώρα η συναρτησιακή εξάρτηση X→Y. Όταν μια συναρτησιακή εξάρτηση υφίσταται μεταξύ δύο ή περισσοτέρων γνωρισμάτων, τότε τα γνωρίσματα του αριστερού μέλους της συνάρτησης, στο παράδειγμά μας το X, προσδιορίζουν μοναδικά τα γνωρίσματα του δεξιού μέλους της συνάρτησης, στο παράδειγμά μας το Y. Αυτό σημαίνει ότι, για κάποια συγκεκριμένη τιμή, π.χ., x_1, του πεδίου X θα πρέπει να αντιστοιχίζεται μια και μοναδική τιμή, y_1, του Y. Τότε, τυχόν προβλήματα πλεονασμού ή ασυνέπειας μπορούν να εμφανιστούν σε επίπεδο στιγμιότυπου, όπως παρουσιάζονται στα Σχήμα

4.6 και Σχήμα 4.7. Το πεδίο S δεν προσδιορίζεται από τα X ή Y, και απλά δηλώνει μια οποιαδήποτε ιδιότητα της οντότητας που μοντελοποιείται.

X	Y	S
x_1	y_1	s_1
x_2	y_2	s_2
x_1	**y_1**	**s_3**

Η επανάληψη των τιμών x_1 και y_1 θα παρατηρείται κάθε φορά που θα εισάγεται η τιμή x_1 στο πεδίο X. Αυτό αποτελεί φαινόμενο πλεονασμού.

Σχήμα 4.6 Περίπτωση πλεονασμού.

X	Y	S
x_1	y_1	s_1
x_2	y_2	s_2
x_1	**y_3**	**s_3**

Λόγω της συναρτησιακής εξάρτησης, κάθε φορά που η τιμή του X είναι x_1, θα πρέπει η αντίστοιχη τιμή του Y να είναι y_1. Η εισαγωγή του ζεύγους (x_1, y_3) δημιουργεί ασυνέπεια στις τιμές του πίνακα.

Σχήμα 4.7 Περίπτωση ασυνέπειας τιμών.

Ας θεωρήσουμε, για καλύτερη κατανόηση των φαινόμενων πλεονασμού και ασυνέπειας, ότι έχουμε έναν πίνακα προμηθειών SUPPLY, όπου καταχωρούνται ο κωδικός κάθε προμηθεύτριας εταιρίας, Sup_ID, η πόλη όπου βρίσκεται η εταιρία, Sup_City, καθώς και η ποσότητα, Quantity, κάθε προμήθειας. Ο συγκεκριμένος πίνακας έχει δομή ως εξής:

SUPPLY (Sup_ID, Sup_City, Quantity).

Στη σχέση SUPPLY το πρωτεύον κλειδί είναι το Sup_ID, ενώ παρατηρείται η εξής συναρτησιακή εξάρτηση Sup_ID→Sup_City. Το φαινόμενο του πλεονασμού λαμβάνει χώρα, όταν για κάθε διαφορετική ποσότητα προμήθειας μιας συγκεκριμένης εταιρίας, θα πρέπει να επαναλαμβάνεται η πληροφορία των γνωρισμάτων Sup_ID και Sup_City. Αντίθετα το φαινόμενο της ασυνέπειας εμφανίζεται όταν για τη συγκεκριμένη τιμή κωδικού μιας εταιρίας, εμφανίζεται διαφορετική πόλη. Η αντιστοιχία των γνωρισμάτων της σχέσης SUPPLY με τα παραδείγματα των σχημάτων 4.6 και 4.7 είναι: X ≡ Sup_ID, Y ≡ Sup_City και S ≡ Quantity.

Γενικά, σκοπός των κανονικών μορφών είναι *κάθε γνώρισμα να προσδιορίζεται από το κλειδί, όλο το κλειδί και μόνο από το κλειδί*. Σε κάθε περίπτωση θα πρέπει να διασφαλίζεται η ατομικότητα των γνωρισμάτων (*1NF*) και η εξάλειψη μερικών εξαρτήσεων (*2NF*) και μεταβατικών εξαρτήσεων (*3NF*) από το κλειδί. Η *κανονικοποίηση* (*normalization*) μιας σχεσιακής βάσης δεδομένων είναι μια σειρά βημάτων, που έχουν ως αποτέλεσμα όλοι οι πίνακες της να βρίσκονται και στις τρείς κανονικές μορφές. Η κανονικοποίηση επιτυγχάνεται με την διάσπαση του «προβληματικού» πίνακα σε δύο πίνακες, οι οποίοι βρίσκονται σε κανονική μορφή. Παρακάτω θα παρουσιάσουμε περιπτώσεις πινάκων οι οποίοι δεν βρίσκονται σε κάποια από τις κανονικές μορφές, τα προβλήματα που δημιουργούνται σε κάθε περίπτωση, καθώς επίσης και τον τρόπο κανονικοποίησης αυτών των πινάκων.

4.2.1. Πρώτη Κανονική Μορφή (First Normal Form, 1NF)

4.2.1.1. Κίνητρο & Εφαρμογή

Ένας πίνακας βρίσκεται σε 1NF όταν όλα τα γνωρίσματά του είναι ατομικά, δηλαδή δεν περιλαμβάνει πλειότιμα ή σύνθετα γνωρίσματα. Παράδειγμα ενός πίνακα ο οποίος δεν βρίσκεται σε 1NF αποτελεί ο παρακάτω πίνακας:

```
EMP (E_ID, Name(First, Middle, Last), Address (Number, Street, City,
                                Code), {Phone}),
```

όπου τα γνωρίσματα `Name` και `Address` είναι σύνθετα, διότι μπορούν να απλοποιηθούν στα επιμέρους γνωρίσματα των παρενθέσεων `First`, `Middle` και `Last` όσον αφορά το γνώρισμα `Name`, και `Number`, `Street`, `City` και `Code` για το `Address`. Το πεδίο `Phone` αποτελεί πλειότιμο γνώρισμα, διότι ένας εργαζόμενος μπορεί να έχει κανέναν, έναν, δύο ή και περισσότερους τηλεφωνικούς αριθμούς. Τα προβλήματα που δημιουργούνται όταν ένας πίνακας δεν βρίσκεται σε 1NF είναι:

1. Η αναζήτηση και η ευρετηριοποίηση των σύνθετων και των πλειότιμων γνωρισμάτων είναι πολύ δύσκολη. Σε αυτό το σημείο θα πρέπει να αναφερθεί ότι τα παραπάνω γνωρίσματα είναι τύπου, κυρίως, συμβολοσειράς. Η αναζήτηση πληροφορίας με βάση τα πεδία αυτά είναι αρκετά χρονοβόρα, διότι θα πρέπει σε κάθε περίπτωση να ελεγχθεί ολόκληρος ο πίνακας καταχωρήσεων

και επιπλέον σε κάθε εγγραφή να πραγματοποιείται έλεγχος *υποσυμβολοσειράς* (*substring*), δηλαδή ελέγχεται εάν η πληροφορία περιλαμβάνεται σε κάποια εγγραφή ως μέρος της συμβολοσειράς. Ας υποθέσουμε ότι αναζητούμε πληροφορία σχετικά με την πόλη, π.χ. την `California`, του πίνακα του παραδείγματος. Τότε θα πρέπει για κάθε εγγραφή του πεδίου `Address` να ελέγχουμε εάν η λέξη `California` περιλαμβάνεται ως υποσύνολο, δηλαδή εάν υπάρχει η λέξη ως μέρος της συμβολοσειράς. Με την κατασκευή ενός ευρετηρίου σε κάποιο από αυτά τα γνωρίσματα, εκτός του ότι δεσμεύεται επιπλέον χώρος για την αποθήκευση του ευρετηρίου, δεν επιλύεται το πρόβλημα της αναζήτησης ενός μέρους της πληροφορίας.

2. Ο απαιτούμενος χώρος αποθήκευσης των σύνθετων και ειδικότερα των πλειότιμων γνωρισμάτων είναι πολύ μεγάλος. Είναι προφανές ότι για την καταχώρηση τιμών των πεδίων `Address` και `Phone` θα πρέπει να δεσμευθεί αρκετός αποθηκευτικός χώρος εφόσον αποτελούν συμβολοσειρές μεγάλου μήκους. Δεδομένου ότι τα πεδία αυτά δεν ανήκουν στο κλειδί, υπάρχει και η πιθανότητα εμφάνισης τιμής `NULL` σε κάποια εγγραφή, οπότε σε αυτή την περίπτωση δεσμεύεται άσκοπα χώρος αποθήκευσης.

Η κανονικοποίηση ενός πίνακα, ο οποίος δεν βρίσκεται σε 1NF, πραγματοποιείται με την διάσπαση και την αντικατάσταση του σύνθετου γνωρίσματος από πιο απλά γνωρίσματα. Στην περίπτωση του πλειότιμου γνωρίσματος η κανονικοποίηση πραγματοποιείται με την διάσπαση του αρχικού πίνακα σε δύο πίνακες ως εξής:

1) Από τον αρχικό πίνακα απομακρύνεται το πλειότιμο γνώρισμα και εισάγεται σε έναν νέο πίνακα.
2) Το πρωτεύον κλειδί του αρχικού πίνακα εισάγεται στον νέο πίνακα ως αναφορά ξένου κλειδιού.

Στην παράγραφο 4.2.1.2 παρουσιάζονται δύο παραπλήσιες σχεδιαστικές λύσεις, οι οποίες διαφοροποιούνται μόνο στην επιλογή του πρωτεύοντος κλειδιού το νέου πίνακα.

Ένας εναλλακτικός τρόπος σχεδίασης, για την απομάκρυνση του πλειότιμου γνωρίσματος, αποτελεί η παράθεση τόσων σε πλήθος πεδίων όσο είναι το πλήθος του μέγιστου συνόλου του πλειότιμου γνωρίσματος. Π.χ., εάν το μέγιστο πλήθος των

τηλεφωνικών αριθμών, που μπορούν να εμφανιστούν σε μια πλειάδα, είναι πέντε, τότε θα μπορούσαμε να σχεδιάσουμε πέντε πεδία της μορφής (Phone1,Phone2,Phone3,Phone4,Phone5). Τα μειονεκτήματα που παρουσιάζει αυτού του είδους ο σχεδιασμός είναι τα εξής:

- Η ύπαρξη πολλών NULL τιμών σε αυτά τα πεδία. Δεδομένου ότι τα πεδία σχεδιάστηκαν ώστε να καλύπτουν το μέγιστο, σε πλήθος, σύνολο (τιμές) του πλειότιμου πεδίου, τα άλλα σύνολα θα περιέχουν ίσα ή λιγότερα στοιχεία (εμφανίσεις τηλεφωνικών αριθμών). Όσες εγγραφές έχουν λιγότερους από πέντε τηλεφωνικούς αριθμούς, καταχωρούν τιμές NULL στα υπόλοιπα πεδία. Αν το πλήθος των εγγραφών, που καταχωρούν λιγότερους από πέντε τηλεφωνικούς αριθμούς, είναι αρκετά μεγάλο είναι προφανές, ότι με αυτή τη σχεδίαση, έχουμε δεσμεύσει πολύ αποθηκευτικό χώρο, άσκοπα.
- Η πιθανότητα εμφάνισης μιας πλειάδας με περισσότερα στοιχεία από αυτά που σχεδιάστηκαν ως πεδία. Ακόμα και στην περίπτωση κατασκευής πεδίων ανάλογα με το μέγιστο πλήθος στοιχείων ενός συνόλου, μπορεί να εμφανιστεί κάποια πλειάδα η οποία να περιέχει περισσότερα στοιχεία. Π.χ., ενώ έχουμε σχεδιάσει πέντε πεδία καλύπτοντας το μέγιστο πλήθος τηλεφωνικών αριθμών, που έχουν εμφανιστεί μέχρις στιγμής, δεν αποκλείεται να εμφανιστεί μια καταχώρηση έξη τηλεφωνικών αριθμών. Μία λύση του συγκεκριμένου προβλήματος μπορεί να αποτελέσει η επιλογή και καταχώρηση πέντε εκ των έξη στοιχείων του πεδίου, αν και αυτό δεν συμβαδίζει πλήρως με την έννοια της βάσης δεδομένων, στη οποία θα πρέπει να είναι δυνατή η καταχώρηση όλης της σχετικής, με μια οντότητα, πληροφορίας.
- Η δημιουργία μεγάλων και σύνθετων ερωτήσεων για την ανάκτηση πληροφορίας. Αν υποθέσουμε ότι ζητείται η εύρεση ενός τηλεφωνικού αριθμού, τότε η αναζήτηση της πληροφορίας αυτής θα πρέπει να πραγματοποιηθεί και στα πέντε (5) γνωρίσματα του πεδίου PHONE. Η αντίστοιχη ερώτηση θα πρέπει να περιλαμβάνει υπό-ερωτήσεις αναζήτησης για κάθε πεδίο, οι οποίες συνδέονται μεταξύ τους με τον όρο OR. Είναι προφανές ότι, μια τέτοιου είδους ερώτηση είναι αρκετά μεγάλη, οπότε και αρκετά χρονοβόρα εφόσον προϋποθέτει, στη χειρότερη περίπτωση, τον έλεγχο όλων των εγγραφών όλων των πεδίων.

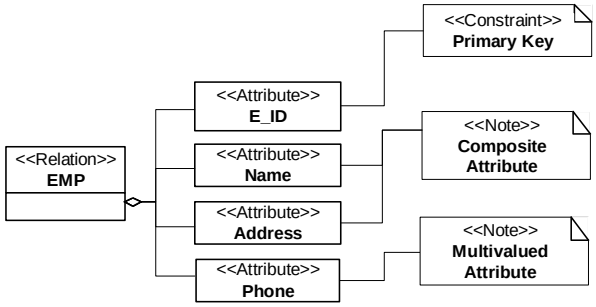
Για τους παραπάνω λόγους, ο σχεδιασμός με παράθεση πεδίων, τα οποία καλύπτουν το μέγιστο πλήθος στοιχείων του πλειότιμου γνωρίσματος, δεν συνίσταται.

4.2.1.2. Δομή

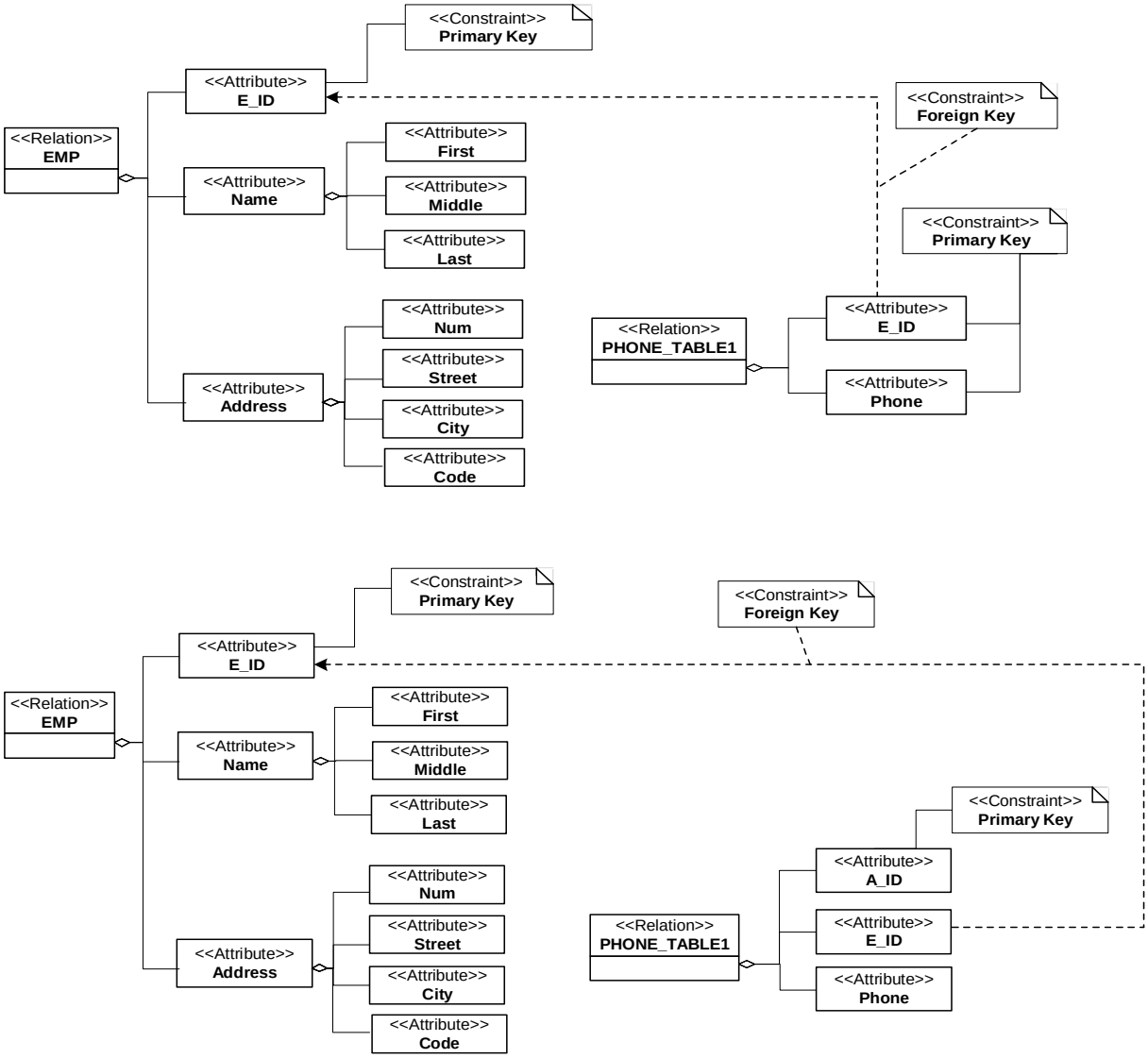
Στο Σχήμα 4.8 παρουσιάζεται η δομή του σχήματος της σχέσης `EMP` , η οποία παραβιάζει την 1NF, ενώ στο Σχήμα 4.10 παρουσιάζεται ένα στιγμιότυπό της. Για την επίλυση των προβλημάτων που δημιουργούν τα σύνθετα γνωρίσματα, η κανονικοποίηση προτείνει τη διάσπαση των σύνθετων γνωρισμάτων σε πιο απλά γνωρίσματα. Το Σχήμα 4.9 παρουσιάζει την αναπαράσταση της δομής του σχήματος της σχέσης `EMP` μετά την κανονικοποίηση ώστε η βάση να βρίσκεται σε 1NF, ενώ το Σχήμα 4.11 παρουσιάζει ένα στιγμιότυπο της κανονικοποιημένης βάσης δεδομένων ως προς τα σύνθετα γνωρίσματα.

Για την απομάκρυνση των πλειότιμων γνωρισμάτων υπάρχουν δύο σχεδιαστικές λύσεις:

1. Η πρώτη λύση περιλαμβάνει την διάσπαση του αρχικού πίνακα σε δύο υποπίνακες, ο ένας εκ των οποίων διατηρεί όλα τα γνωρίσματα του αρχικού πλην του πλειότιμου γνωρίσματος. Ο δεύτερος πίνακας περιλαμβάνει το πρωτεύον γνώρισμα του αρχικού πίνακα, με την μορφή ξένου κλειδιού, και το μέχρι πρότινος πλειότιμο γνώρισμα. Ως πρωτεύον κλειδί του δεύτερου πίνακα ορίζεται ο συνδυασμός του ξένου κλειδιού και του πεδίου που αποτελούσε πλειότιμο γνώρισμα στον αρχικό πίνακα.
2. Η δεύτερη σχεδιαστική λύση διαφοροποιείται ως προς την πρώτη, στον ορισμό του πρωτεύοντος κλειδιού του δεύτερου πίνακα. Σε αυτή την περίπτωση, μπορούμε να αποφύγουμε τον συνδυασμό γνωρισμάτων, με την εισαγωγή ενός τεχνητού κλειδιού.



Σχήμα 4.8 Αναπαράσταση δομής του σχήματος της βάσης του παραδείγματος, στο οποίο παραβιάζεται η 1NF.



Σχήμα 4.9 Δύο εναλλακτικές αναπαραστάσεις της δομής του σχήματος της βάσης του παραδείγματος όπως προκύπτει μετά την κανονικοποίηση σε 1NF.

Ο πίνακας EMP του Σχήματος 4.10 παρουσιάζει ένα στιγμιότυπο του παραδείγματος. Τα προβλήματα με τις μεγάλες συμβολοσειρές, που παρατηρούνται στα σύνθετα γνωρίσματα, είναι εμφανή.

EMP

E_ID	**Name**	**Address**	**Phone**
10	Jenny Stathos	13 Broadway Str. N.Y. 11234	(112)223-3444, (112)332-4233, (112)223 -2233
110	Sally Thomson	83 Hallway Colorado 18273	(291)983-2836
293	James M. Jean	623A California Av. California 2123	

Σχήμα 4.10 Αναπαράσταση δομής στιγμιότυπου της βάσης του παραδείγματος, στο οποίο παραβιάζεται η 1NF.

EMP

E_ID	**Name**			**Address**				**Phone**
	First	**Middle**	**Last**	**Num**	**Street**	**City**	**Code**	
10	Jenny		Stathos	13	Broadway Str.	N.Y.	11234	(112)2235444, (112)3324233, (112)2232233
110	Sally		Thomson	83	Hallway	Colorado	18273	(291)9832836
293	James	M.	Jean	623A	California Av.	California	2123	

Σχήμα 4.11 Αναπαράσταση δομής στιγμιότυπου της βάσης του παραδείγματος όπου λύνεται το πρόβλημα του σύνθετου γνωρίσματος.

Το Σχήμα 4.12 παρουσιάζει στιγμιότυπο της βάσης όπως προκύπτει μετά την απομάκρυνση του πλειότιμου γνωρίσματος. Οι δύο σχεδιαστικές λύσεις διαφοροποιούνται μόνο στον ορισμό του πρωτεύοντος κλειδιού του δεύτερου πίνακα, γεγονός που αναπαρίσταται με τους δύο εναλλακτικούς πίνακες PHONE_TABLE1 και PHONE_TABLE2.

EMP

E_ID	Name			Address			
	First	Middle	Last	Num	Street	City	Code
10	Jenny		Stathos	13	Broadway Str.	N.Y.	11234
110	Sally		Thomson	83	Hallway	Colorado	18273
293	James	M.	Jean	623A	California Av.	California	2123

PHONE_TABLE1

E_ID	Phone
10	(112)223-5444
10	(112)332-4233
10	(112)223-2233
110	(291)983-2836

PHONE_TABLE2

A_ID	E_ID	Phone
1	10	(112)223-5444
2	10	(112)332-4233
3	10	(112)223-2233
4	110	(291)983-2836

Σχήμα 4.12 Αναπαράσταση στιγμιότυπων των δύο εναλλακτικών σχεδιασμών της βάσης του παραδείγματος όπου λύνεται το πρόβλημα του πλειότιμου γνωρίσματος.

4.2.1.3. Συμπεριφορά σε επίπεδο στιγμιότυπου

Εισαγωγή

Η εισαγωγή μιας νέας εγγραφής θα μπορούσαμε να πούμε ότι δεν διαφοροποιείται κατά πολύ με την κανονικοποίηση του πίνακα, όσον αφορά τουλάχιστον στα σύνθετα γνωρίσματα. Ενώ στον αρχικό πίνακα συμπληρώνονταν ένα μόνο σύνθετο πεδίο, με τον εναλλακτικό, κανονικοποιημένο, τρόπο συμπληρώνονται περισσότερα πεδία τα όποια έχουν παρόμοιο μήκος συμβολοσειράς με το αρχικό.

Εν αντιθέσει, σημαντική βελτίωση παρατηρείται στην περίπτωση του πλειότιμου γνωρίσματος. Αρχικά διατηρούνταν ένας αρκετά μεγάλος χώρος για αποθήκευση του πλειότιμου γνωρίσματος, «σπαταλώντας» τον σε περιπτώσεις κενής εισαγωγής, π.χ. κάποιος δεν έχει τηλέφωνο. Επιπλέον δεν εξαλείφονταν η περίπτωση μη επαρκούς χώρου για κάποια εγγραφή, π.χ., εάν είχαμε υποθέσει ότι κάποιος έχει το πολύ τέσσερις τηλεφωνικούς αριθμούς στην περίπτωση όπου κάποιος έχει περισσότερους ο

αντίστοιχος χώρος που έχει δεσμευθεί για το πεδίο Phone δεν θα επαρκούσε. Μετά την κανονικοποίηση του αρχικού πίνακα αυτό το πρόβλημα παύει να υφίσταται, εφόσον κάθε νέα εισαγωγή στο πεδίο Phone αντιστοιχεί πλέον σε μία νέα εγγραφή του νέου πίνακα.

Διαγραφή

Η διαδικασία της διαγραφής μιας πλειάδας πραγματοποιείται με τον ίδιο τρόπο όταν η διάσπαση λαμβάνει χώρα μόνο σε επίπεδο γνωρισμάτων και όχι πίνακα, δηλαδή όταν ο αρχικός πίνακας περιλαμβάνει μόνο την περίπτωση σύνθετου γνωρίσματος.

Διαφοροποίηση στην λειτουργία διαγραφής παρατηρείται κατά την ύπαρξη πλειότιμου γνωρίσματος. Π.χ., η διαγραφή ενός εργαζόμενου από τον πίνακα EMP, αρχικά, σήμαινε απλά διαγραφή μιας πλειάδας του πίνακα, ενώ με την κανονικοποιημένη μορφή του πίνακα θα πρέπει αναγκαστικά να εξεταστούν και οι δύο πίνακες και να διαγραφούν οι αντίστοιχες πλειάδες. Αυτό φυσικά απαιτεί επιπλέον χρόνο αναζήτησης και διαγραφής.

Ανανέωση

Στον αρχικό, μη κανονικοποιημένο, πίνακα οποιαδήποτε ανανέωση σε σύνθετο γνώρισμα, συνεπάγονταν την επανεγγραφή ενός μέρους της πλειάδας που ανανεώνεται. Π.χ., εάν θέλαμε να εισάγουμε το ενδιάμεσο όνομα M. στην πλειάδα του εργαζομένου Jenny Stathos, στον αρχικό πίνακα θα έπρεπε να δηλώσουμε εκ νέου την τιμή Name = “Jenny M. Stathos”. Μετά την κανονικοποίηση, η ανανέωση της αντίστοιχης πλειάδας πραγματοποιείται με την εισαγωγή της τιμής M. στο πεδίο Middle.

Ανάλογα πραγματοποιείται και η ανανέωση σε ένα πλειότιμο γνώρισμα, αποφεύγοντας τον κίνδυνο μη επαρκής μνήμης. Η ενημέρωση του πεδίου Phone πραγματοποιείται απλά με την ενημέρωση μιας πλειάδας στον πίνακα καταχωρήσεων τηλεφωνικών αριθμών κάθε εργαζομένου, χωρίς πλέον να απαιτείται η επανεγγραφή όλων των τηλεφωνικών αριθμών. Με άλλα λόγια για την ανανέωση ενός τηλεφωνικού αριθμού, π.χ. του εργαζομένου Jenny Stathos, εάν ο πίνακας δεν βρισκόταν σε 1NF θα έπρεπε να ξαναγράψουμε όλους τους τηλεφωνικούς αριθμούς

που αντιστοιχούν σε αυτόν μεταβάλλοντας μόνο έναν, Phone = {(112)223-5444, (112)332-4233, (112)223 –4938 }. Από τη στιγμή που ο πίνακας είναι σε 1NF η ανανέωση αυτή απλά σημαίνει εύρεση του παλιού τηλεφωνικού αριθμού στον PHONE_TABLE1 ή PHONE_TABLE2 και ενημέρωση της τιμής της αντίστοιχης πλειάδας, Phone((112)223 -2233) = (112)223 –4938.

4.2.1.4. Συμπεριφορά σε επίπεδο σχήματος

Σε επίπεδο σχήματος, οποιαδήποτε *εισαγωγή*, *διαγραφή* ή *ανανέωση* ενός γνωρίσματος δεν επηρεάζει σημαντικά τη συμπεριφορά του προτύπου σε επίπεδο σχήματος. Ο μόνος έλεγχος, ο οποίος θα πρέπει να πραγματοποιηθεί, μετά την εφαρμογή μιας από τις παραπάνω λειτουργίας, είναι να τηρούνται οι περιορισμοί που έχουν τεθεί, π.χ. περιορισμός κλειδιού, περιορισμός αναφορικής ακεραιότητας. Πιο συγκεκριμένα, η λειτουργία ανανέωσης σε οποιοδήποτε γνώρισμα δεν έχει καμία επίπτωση στη συμπεριφορά της βάσης, λαμβάνοντας υπόψη τους περιορισμούς σχεδίασης της βάσης. Κατά την λειτουργία εισαγωγής γνωρισμάτων απλά θα πρέπει να ελέγχονται οι περιορισμοί σχεδίασης, π.χ. τυχόν δημιουργία νέου πρωτεύοντος κλειδιού. Η διαγραφή γνωρίσματος μπορεί να επιφέρει αλλαγές στους περιορισμούς, π.χ. κατάργηση του κλειδιού και δημιουργία νέου, ή και διαγραφή κάποιου πίνακα, π.χ. εάν διαγραφτεί το πεδίο Phone δεν υφίσταται ο PHONE_TABLE1 ή ο PHONE_TABLE2, ανάλογα με τον σχεδιασμό που επιλέχθηκε.

4.2.1.5. Κατασκευή

Η SQL-1999 [15] «επιτρέπει» την κατασκευή πλειότιμων [σελ. 42] και σύνθετων γνωρισμάτων [σελ. 43], αναιρώντας με αυτόν τον τρόπο την 1NF. Για τα πλειότιμα γνωρίσματα κατασκευάζεται ένας *κατασκευαστής τύπου* (*type constructor*) ARRAY, έχοντας ως γενική δομή: *dt* ARRAY [*mc*]. Το dt αντιπροσωπεύει τον τύπο των δεδομένων που θα αποθηκεύονται στον ARRAY, ενώ το mc αντιπροσωπεύει την μέγιστη πληθικότητα του ARRAY, δηλαδή το πλήθος των στοιχείων αποθηκεύονται. Π.χ., στην περίπτωση του γνωρίσματος Phone της σχέσης EMP, θα μπορούσαμε να κατασκευάσουμε τον type constructor: *INTEGER/ VARCHAR(25)* ARRAY [5], στον οποίον αποθηκεύονται ως και 5 τηλεφωνικοί αριθμοί. Για τα σύνθετα γνωρίσματα

κατασκευάζεται ένας *τύπος ανώνυμης γραμμής* (*Anonymous Row Type*) με γενική δομή: ROW (*fld*$_1$, *fld*$_2$,…, *fld*$_n$), όπου το fld$_i$ ορίζει το πεδίο κάθε γραμμής δηλαδή το *όνομα* (*fldname*) και τον *τύπο* (*fldtype*). Π.χ., στην περίπτωση του πεδίου Address της σχέσης EMP, θα μπορούσαμε να ορίσουμε τον παρακάτω τύπο γραμμής:

ROW (*Num* INTEGER, *Street* CHARACTER VARAYING(30),
City CHARACTER VARAYING(20), *Code* CHARACTER VARAYING(10)).

Παρόμοια, η Oracle10g [13, σελ. 1086-1103], μέσω της δήλωσης CREATE TYPE επιτρέπει:

i. την κατασκευή μεταβλητού μεγέθους τύπο πίνακα, *VARRAY*, για τον ορισμό πλειότιμων γνωρισμάτων. Π.χ., για την καταγραφή των τηλεφωνικών αριθμών μπορούμε να δηλώσουμε:

   CREATE TYPE *phone_list* AS VARRAY(5) OF VARCHAR(25),

   στον οποίον αποθηκεύονται το πολύ 5 τηλεφωνικοί αριθμοί, έχοντας ο καθένας έως 25 ψηφία.

ii. την κατασκευή *τύπου αντικειμένου* (*object type*), ο οποίος ουσιαστικά είναι ένας *τύπος ορισμένος από τον χρήστη* (*user defined type*) και μπορεί να «ομαδοποιήσει» συγκεκριμένα πεδία. Π.χ., στην περίπτωση της διεύθυνσης της σχέσης EMP του παραδείγματος, θα μπορούσαμε να ορίσουμε τον τύπο:

   CREATE TYPE *address* AS OBJECT(*Num* INTEGER,
   Street VARCHAR(30),*City* VARCHAR(20),*Code* VARCHAR(10)).

4.2.2. Δεύτερη Κανονική Μορφή (Second Normal Form, 2NF)

4.2.2.1. Κίνητρο & Εφαρμογή

Η δεύτερη κανονική μορφή έχει ως σκοπό την απομάκρυνση της πλεονάζουσας πληροφορίας και την αποφυγή, ως ένα βαθμό, του φαινόμενου της ασυνέπειας των δεδομένων ενός πίνακα. Όταν κάποιο γνώρισμα εξαρτάται μόνο από ένα μέρος του πρωτεύοντος κλειδιού, στην περίπτωση που το κλειδί αποτελείται από περισσότερα του ενός γνωρίσματα, τότε λέγεται ότι υπάρχει *μερική εξάρτηση* από το πρωτεύον κλειδί και παραβιάζεται η 2NF. *Ένας πίνακας βρίσκεται σε 2NF όταν: i) βρίσκεται σε 1NF και ii) δεν υπάρχουν μερικές εξαρτήσεις γνωρισμάτων από το πρωτεύον κλειδί.* Ας

θεωρήσουμε το παράδειγμα, όπου καταχωρούνται οι υπηρεσίες που παρέχουν ορισμένες εταιρείες σε κάποιες πόλεις. Η σχέση έχει την δομή:

SUPPLY (Sup_ID, Comp_ID, SupCity, CompName, Date).

Πρωτεύον κλειδί αποτελεί ο συνδυασμός των γνωρισμάτων Sup_ID και Comp_ID. Το γνώρισμα SupCity αποθηκεύει πληροφορία σχετική με την πόλη στην οποία παρέχεται η υπηρεσία, το CompName αντιστοιχεί στο όνομα της εταιρίας που παρέχει την υπηρεσία και τέλος στο γνώρισμα Date αποθηκεύεται η ημερομηνία της παροχής υπηρεσίας.

Υπάρχουν δύο γνωρίσματα, τα SupCity και CompName, τα οποία παραβιάζουν την 2NF σύμφωνα με τις παρακάτω εξαρτήσεις, εφόσον προσδιορίζονται από ένα, μόνο, μέρος του πρωτεύοντος κλειδιού:

Sup_ID → SupCity και Comp_ID → CompName.

Τα σημαντικότερα προβλήματα τα οποία προκαλεί η παραβίαση της 2NF είναι:

1. Το φαινόμενου του πλεονασμού.
2. Το φαινόμενο της ασυνέπειας.
3. Προβληματική συμπεριφορά σε επίπεδο στιγμιότυπου. Εκτός από το φαινόμενο της ασυνέπειας, το οποίο στην ουσία έχει να κάνει με προβληματική *εισαγωγή* κάποιας εγγραφής, παρουσιάζονται προβλήματα και στις λειτουργίες *διαγραφή* και *ανανέωση* εγγραφών. Το πρόβλημα εστιάζεται στο γεγονός ότι μια διαγραφή/ανανέωση τιμής κάποιας εγγραφής μπορεί να πυροδοτήσει μια σειρά από άλλες διαγραφές/ανανεώσεις περισσοτέρων εγγραφών. Η συμπεριφορά της βάσης σε επίπεδο στιγμιότυπου μελετάται εκτενέστερα στην παράγραφο 4.2.2.3.

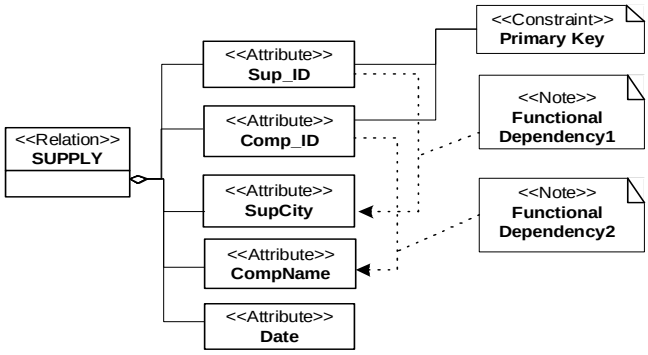
Για την αντιμετώπιση των παραπάνω προβλημάτων η λύση είναι η κανονικοποίηση του πίνακα της βάσης δεδομένων ώστε να βρίσκεται σε 2NF. Η διάσπαση του πίνακα πραγματοποιείται με τα εξής βήματα:

1) Ελέγχονται οι εξαρτήσεις και ομαδοποιούνται τα γνωρίσματα που προσδιορίζονται από το ίδιο μέρος του κλειδιού.

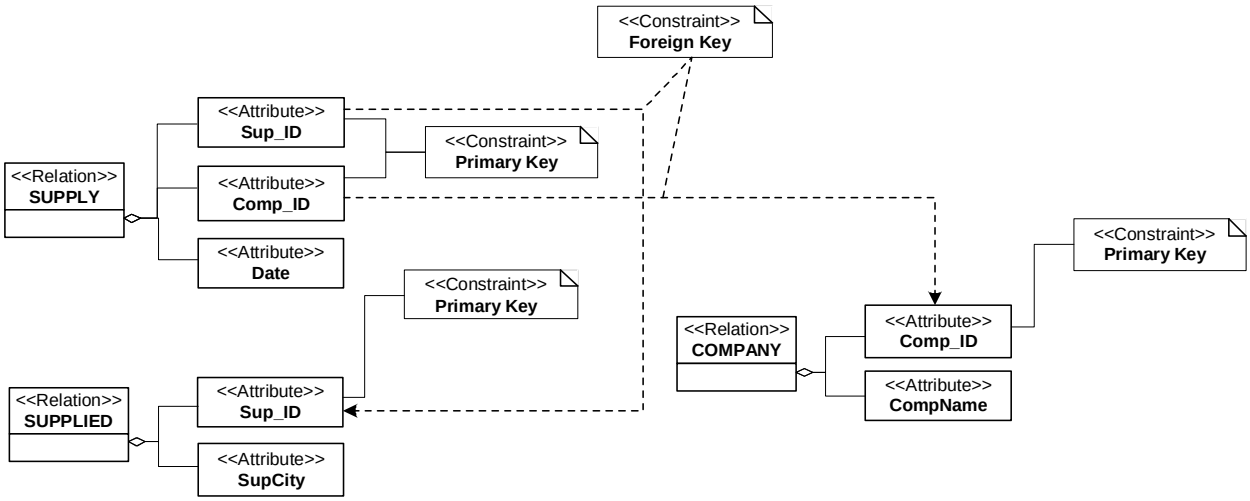
2) Η κάθε ομάδα γνωρισμάτων απομακρύνεται από τον αρχικό πίνακα και τοποθετείται σε έναν καινούριο πίνακα.
3) Κλειδί κάθε νέου πίνακα αποτελεί το μέρος του αρχικά πρωτεύοντος κλειδιού που προσδιόριζε τα γνωρίσματα.
4) Κάθε γνώρισμα του αρχικού πρωτεύοντος κλειδιού, που συμμετέχει σε κάποια συναρτησιακή εξάρτηση, αποτελεί, μετά την κανονικοποίηση, ξένο κλειδί ως προς το αντίστοιχο γνώρισμα των πινάκων που προκύπτουν μετά την διάσπαση.

4.2.2.2. Δομή

Στο Σχήμα 4.13 παρουσιάζεται το σχήμα του πίνακα του παραδείγματος επισημαίνοντας τις συναρτησιακές εξαρτήσεις που δημιουργούν το πρόβλημα, ενώ στο Σχήμα 4.14 παρουσιάζεται η τελική μορφή του σχήματος του πίνακα, όπως προκύπτει μετά την κανονικοποίηση του.



Σχήμα 4.13 Αναπαράσταση του σχήματος του πίνακα του παραδείγματος που δεν βρίσκεται σε 2NF.



Σχήμα 4.14 Αναπαράσταση του σχήματος των πινάκων που προκύπτουν μετά την κανονικοποίηση σε 2NF του αρχικού πίνακα του παραδείγματος.

Στο στιγμιότυπο του πίνακα SUPPLY του παραδείγματος, όπως παρουσιάζεται στο Σχήμα 4.15, εμφανίζεται το φαινόμενο του πλεονασμού (σκίαση των αντίστοιχων τιμών) και το φαινόμενο της ασυνέπειας (χρωματισμός των αντίστοιχων τιμών με γκρι και εμφάνιση άσπρων γραμμάτων) των τιμών του πίνακα. Ο πλεονασμός είναι ένα πρόβλημα το οποίο μπορεί εύκολα να επιλυθεί, αρκεί να παρατηρηθεί ότι είναι άσκοπο να αποθηκεύεται κάθε φορά και το όνομα της εταιρίας για κάθε παροχή υπηρεσίας, από τη στιγμή που ο κωδικός της εταιρίας Comp_ID αρκεί. Παρόμοια θα πρέπει να αντιμετωπιστεί και η περίπτωση με τα γνωρίσματα Sup_ID και SupCity, από τη στιγμή που κάθε κωδικός χαρακτηρίζει μόνο μία πόλη. Η ασυνέπεια, από την άλλη πλευρά, μπορεί μόνο να περιοριστεί με το να αποφεύγεται η άσκοπα επαναλαμβανόμενη πληροφορία των γνωρισμάτων που εξαρτώνται μερικώς από το πρωτεύον κλειδί.

SUPPLY

Sup_ID	Comp_ID	SupCity	CompName	Date
10	**200**	**Arta**	**IST**	10/02/99
11	**200**	Ioannina	**IST**	23/04/02
02	201	Patra	D-Mode	09/09/05
10	**200**	**Arta**	**IST**	12/02/03
10	**200**	**Arta**	**IST**	23/04/02
11	201	**Patra**	D-Mode	09/09/05

Σχήμα 4.15 Αναπαράσταση στιγμιότυπου του πίνακα του παραδείγματος που δεν βρίσκεται σε 2NF.

Η κανονικοποίηση του πίνακα αντιμετωπίζει τα παραπάνω προβλήματα, και πραγματοποιείται με την διάσπαση του παραπάνω πίνακα σε 3 νέους πίνακες. Ο πρώτος πίνακας, SUPPLY του Σχήματος 4.16, περιλαμβάνει όλα τα γνωρίσματα πλην αυτών, SupCity και CompName, που εξαρτώνται μερικώς από το κλειδί. Από τη στιγμή που τα δύο γνωρίσματα δεν προσδιορίζονται από το ίδιο μέρος του πρωτεύοντος κλειδιού, τοποθετούνται σε δύο διαφορετικούς πίνακες. Σε κάθε έναν από τους δύο νέους πίνακες, SUPPLIED και COMPANY, εισάγεται ένα ακόμα γνώρισμα, το μέρος του πρωτεύοντος κλειδιού του αρχικού πίνακα το οποίο ικανοποιεί την εκάστοτε συναρτησιακή εξάρτηση, και αποτελεί κλειδί του νέου πίνακα. Ο πίνακας SUPPLIED του Σχήματος 4.16 αντιστοιχεί στη συναρτησιακή εξάρτηση Sup_ID → SupCity, ενώ ο πίνακας COMPANY στη συναρτησιακή εξάρτηση Comp_ID → CompName.

SUPPLY

Sup_ID	Comp_ID	Date
10	200	10/02/99
11	200	23/04/02
02	201	09/09/05
10	200	12/02/03
10	200	23/04/02
11	201	09/09/05

SUPPLIED

Sup_ID	SupCity
10	Arta
11	Ioannina
02	Patra
11	Patra

COMPANY

Comp_ID	CompName
200	IST
201	D-Mode

Σχήμα 4.16 Αναπαράσταση στιγμιότυπων των τριών νέων πινάκων όπως προκύπτουν μετά την κανονικοποίηση του αρχικού πίνακα σε 2NF.

4.2.2.3. Συμπεριφορά σε επίπεδο στιγμιότυπου

Εισαγωγή

Με την κανονικοποίηση του πίνακα σε 2NF αποφεύγεται μεν, το φαινόμενο του πλεονασμού, προκαλείται δε, η εισαγωγή τιμών σε περισσότερους του ενός πίνακες. Στο παράδειγμά μας, η εισαγωγή μιας πλειάδα στον SUPPLY, ενδεχομένως να προκαλέσει εισαγωγή ανάλογων πλειάδων στους SUPPLIED και COMPANY. Ας θεωρήσουμε το παράδειγμα εισαγωγής της πλειάδας (12, 203, Iraklio, NetWave, 02/08/06) που αντιστοιχεί στον αρχικό πίνακα. Σε αυτή την περίπτωση θα πρέπει να εισαχθεί η καταχώρηση (12, 203, 02/08/06) στον SUPPLY και εν συνεχεία να εισαχθούν οι καταχωρήσεις (12, Iraklio) και (203, NetWave) στους πίνακες SUPPLIED και COMPANY, αντίστοιχα. Εάν κάποια από τις πλειάδες (12, Iraklio) ή (203, NetWave) υπάρχει ήδη καταχωρημένη στον αντίστοιχο πίνακα, είναι προφανές ότι η πληροφορία δεν επαναλαμβάνεται. Παρατηρείται, λοιπόν, ότι, όσον αφορά στους πίνακες που είναι σε 2NF, η εισαγωγή μιας πλειάδας συνεπάγεται, στη χειρότερη περίπτωση, ενημέρωση όλων των πινάκων, εν αντιθέσει με τον αρχικό πίνακα όπου ενημερώνεται μόνο ένας πίνακας.

Διαγραφή

Η διαγραφή μιας πλειάδας από τον EMP, αρχικά, δεν παρουσιάζει το φαινόμενο της «αλυσιδωτής» διαγραφής πλειάδων και από τους άλλους κανονικοποιημένους πίνακες. Με τον όρο «αλυσιδωτή» διαγραφή εννοούμε ότι η διαγραφή μιας εγγραφής ενός πίνακα, προκαλεί, λόγω της ύπαρξης ξένων κλειδιών, τη διαγραφή σχετιζόμενων εγγραφών και στους άλλους πίνακες. Επιστρέφοντας στο παράδειγμά μας και στο στιγμιότυπο του κανονικοποιημένου σε 2NF πίνακα, η διαγραφή της πλειάδας (10, 200, 23/04/02) δεν συνεπάγεται την διαγραφή των πλειάδων (10, Arta) και (200, IST) από τους SUPPLIED και COMPANY αντίστοιχα. Οι πλειάδες μπορεί να διατηρηθούν στους πίνακες ακόμα και αν δεν υπάρχει καμία πλειάδα στον πίνακα SUPPLY με τιμές Sup_ID = 10 ή/και Dept_ID = 200. Με αυτό τον τρόπο η διαγραφή ακόμα και της τελευταίας πλειάδας του πίνακα SUPPLY όπου εμφανίζονται οι τιμές Sup_ID = 10 ή/και Dept_ID = 200 δεν συνεπάγεται την απομάκρυνση της

πληροφορίας σχετικά με τις πόλεις ή τις εταιρίες στις οποίες αντιστοιχούν οι κωδικοί. Το φαινόμενο του «χασίματος» πληροφορίας με την διαγραφή μιας πλειάδας, εμφανίζεται στο μη κανονικοποιημένο πίνακα, εάν φανταστούμε ότι υπάρχει μόνο μια πλειάδα όπου εμφανίζονται οι τιμές `Sup_ID = 10` ή/και `Comp_ID = 200` και η πλειάδα αυτή διαγραφεί.

Στην περίπτωση, όμως, που η διαγραφή παρατηρηθεί σε οποιονδήποτε από τους πίνακες `SUPPLIED` και `COMPANY`, αυτό έχει ως συνέπεια την «αλυσιδωτή» διαγραφή αρκετών πλειάδων από τον πίνακα `SUPPLY`. Π.χ., ας υποθέσουμε ότι διαγράφεται η πλειάδα `(10, Arta)` του πίνακα `SUPPLIED`, θα πρέπει να ενημερωθεί και ο πίνακας `SUPPLY`, διαγράφοντας όλες τις πλειάδες στις οποίες το γνώρισμα `Sup_ID` έχει την τιμή `10`. Ανάλογα αποτελέσματα προκαλεί η λειτουργία της διαγραφής όταν εφαρμόζεται σε γνώρισμα του πρωτεύοντος κλειδιού ή σε κάποιο γνώρισμα που εξαρτάται από μέρος του κλειδιού, στο μη κανονικοποιημένο πίνακα.

Ανανέωση

Η λειτουργία της ανανέωσης όταν εφαρμόζεται σε κάποιο γνώρισμα το οποίο δεν ανήκει στο κλειδί και δεν εξαρτάται από μέρος του κλειδιού, επιφέρει τα ίδια αποτελέσματα είτε πρόκειται για κανονικοποιημένο είτε μη κανονικοποιημένο πίνακα. Η ανανέωση, π.χ., οποιαδήποτε πλειάδας του γνωρίσματος `Date` και στις δύο περιπτώσεις πραγματοποιείται απλά την αλλαγή μιας και μόνο τιμής του γνωρίσματος.

Αυτό που αξίζει να μελετηθεί είναι η περίπτωση ανανέωσης της τιμής ενός γνωρίσματος που είτε ανήκει στο κλειδί είτε προσδιορίζεται μοναδικά από αυτό. Αρχικά, ας θεωρήσουμε ότι η ανανέωση πραγματοποιείται σε κάποιο από τα γνωρίσματα του πρωτεύοντος κλειδιού, π.χ. το `Sup_ID` αλλάζει τιμή από `10` σε `91` αλλά εξακολουθεί να αντιστοιχεί στην πόλη `Arta`. Η δεδομένη ανανέωση προκαλεί μια σειρά από ανανεώσεις στον πίνακα είτε αυτός είναι σε 2NF είτε όχι, μεταβάλλοντας την τιμή του γνωρίσματος σε όλες τις πλειάδες που εμφανίζεται. Διαφοροποιείται λίγο η αντιμετώπιση της ανανέωσης όταν πραγματοποιείται σε γνώρισμα που εξαρτάται από μέρος του κλειδιού. Π.χ., ας θεωρήσουμε ότι θέλουμε

να αντιστοιχίσουμε στον κωδικό `Sup_ID = 10` την πόλη `SupCity = Xania`, η ανανέωση αυτή στον κανονικοποιημένο πίνακα πραγματοποιείται απλά με την αλλαγή μιας τιμής στον πίνακα `SUPPLIED`. Η συγκεκριμένη ανανέωση αποτελεί πιο πολύπλοκη και χρονοβόρα διαδικασία στην περίπτωση του μη κανονικοποιημένου πίνακα, αφού θα πρέπει να εξεταστούν όλες οι εγγραφές του πίνακα για την εύρεση όλων των πλειάδων με ζεύγος τιμών `Sup_ID = 10, SupCity = Arta.`

4.2.2.4. Συμπεριφορά σε επίπεδο σχήματος

Θα εξετάσουμε τη συμπεριφορά του προτύπου κατά την *εισαγωγή*/ *διαγραφή*/ *ανανέωση* ενός γνωρίσματος κυρίως υπό το πρίσμα των συναρτησιακών εξαρτήσεων. Πότε, δηλαδή, μπορεί να δημιουργηθεί ή να καταργηθεί μια μερική εξάρτηση κατά την εφαρμογή των παραπάνω λειτουργιών.

Εισαγωγή

Εν γένει, η εισαγωγή ενός νέου γνωρίσματος στο σχήμα της σχέσης δεν επηρεάζει τη συμπεριφορά του προτύπου, αρκεί να μην προκύπτει μερική εξάρτηση του γνωρίσματος αυτού από το κλειδί.

Στην περίπτωση εμφάνισης εξάρτησης, η λύση είναι η κανονικοποίηση μέσω των βημάτων που έχουν προαναφερθεί. Πιο συγκεκριμένα, εάν το γνώρισμα που εισάγεται προσδιορίζεται από μέρος του κλειδιού το οποίο ήδη εμφανίζεται ως ξένο κλειδί σε διαφορετικό πίνακα, το νέο γνώρισμα εισάγεται απ’ ευθείας στον πίνακα αυτό. Σε αντίθετη περίπτωση, παρατηρείται η δημιουργία μιας νέας συναρτησιακής εξάρτησης, η οποία έχει ως συνέπεια την κανονικοποίηση του πίνακα μετά την εισαγωγή του νέου γνωρίσματος. Η νέα κανονικοποίηση πραγματοποιείται στα πλαίσια της κανονικοποίησης ενός πίνακα σύμφωνα με τους κανόνες της 2NF.

Διαγραφή

Η διαγραφή ενός γνωρίσματος το οποίο δεν εμφανίζεται σε κάποια συναρτησιακή

εξάρτηση δεν παρουσιάζει κάποιον ενδιαφέρον για μελέτη της συμπεριφοράς του προτύπου σε επίπεδο σχήματος.

Αντιθέτως, η διαγραφή ενός γνωρίσματος που λαμβάνει μέρος σε μια συναρτησιακή εξάρτηση μπορεί να επηρεάσει αρκετά τη συμπεριφορά του σχήματος της βάσης, έως και σε σημείο διαγραφής ολόκληρου πίνακα. Ας θεωρήσουμε, π.χ., ότι διαγράφεται το γνώρισμα SupCity του SUPPLIED του παραδείγματός μας, σε αυτή την περίπτωση η διαγραφή αυτή συνεπάγεται την διαγραφή ολόκληρου του SUPPLIED. Με την διαγραφή του SupCity, παύει να υφίσταται η συναρτησιακή εξάρτηση Sup_ID → SupCity, οπότε δεν υπάρχει και λόγος ύπαρξης του πίνακα SUPPLIED.

Ανανέωση

Και κατά την λειτουργία της ανανέωσης ενός πεδίου, θα εστιάσουμε την μελέτη μας στην περίπτωση ανανέωσης ενός πεδίου, το οποίο λαμβάνει μέρος σε μια συναρτησιακή εξάρτηση. Η ανανέωση οποιουδήποτε άλλου πεδίου, πλην του πρωτεύοντος κλειδιού κάθε πίνακα, δεν επηρεάζει τη συμπεριφορά του προτύπου σε επίπεδο σχήματος.

Πρώτα θα μελετήσουμε την περίπτωση ανανέωσης ενός γνωρίσματος που ανήκει στο κλειδί. Εάν ανανεωθεί ένα πεδίο το οποίο ανήκει στο σύνολο του πρωτεύοντος κλειδιού, αυτό έχει ως αποτέλεσμα την ανανέωση και των πεδίων που εξαρτώνται (προσδιορίζονται) από το πεδίο αυτό. Π.χ., ας υποθέσουμε ότι ανανεώνεται το γνώρισμα Comp_ID και θέτεται ως τιμή DEFAULT = 200, γνωστοποιώντας με αυτό τον τρόπο ότι π.χ. έχει γίνει συγχώνευση όλων των εταιριών σε μια εταιρία –λίγο απίθανο αλλά όχι αδύνατο-. Η ανανέωση, λοιπόν, στο πεδίο Comp_ID θα πρέπει να επιβληθεί και σε όλα τα γνωρίσματα που προσδιορίζονται από αυτό, στο παράδειγμά μας στο πεδίο CompName.

Με το ίδιο σκεπτικό, είναι προφανές το αποτέλεσμα της ανανέωσης γνωρίσματος που εξαρτάται από μέρος του κλειδιού. Π.χ., ανανεώνοντας το γνώρισμα CompName στην τιμή DEFAULT = IST″ είναι λογικό ότι θα πρέπει να ενημερωθεί αντίστοιχα και το γνώρισμα Comp_ID τόσο του πίνακα COMPANY, όσο και του πίνακα EMP. Αν

κοιτάξουμε όμως πιο προσεκτικά, θα παρατηρήσουμε ότι πλέον, μετά την εφαρμογή της ανανέωσης, ο πίνακας COMPANY, του παραδείγματος, εμφανίζει το φαινόμενο του πλεονασμού, διατηρώντας πλειάδες με την ίδια τιμή σε κάθε ζεύγος (Comp_ID, DeptName). Σε αυτό το σημείο, η διαγραφή των πλεοναζουσών πλειάδων του COMPANY είναι αναγκαία.

4.2.3. Τρίτη Κανονική Μορφή (Thinrd Normal Form, 3NF)

4.2.3.1. Κίνητρο & Εφαρμογή

Ο λόγος ύπαρξης της τρίτης κανονικής μορφής δεν διαφέρει από τον λόγο ύπαρξης της δεύτερης. Ένας πίνακας, λέγεται ότι, βρίσκεται σε 3NF όταν: *i) βρίσκεται σε 2NF και ii) όταν κάθε γνώρισμα εξαρτάται από το κλειδί και μόνο το κλειδί.* Ο περιορισμός *ii)*απομακρύνει περιπτώσεις όπου κάποιο γνώρισμα εξαρτάται μέσω μιας μεταβατικής εξάρτησης από το κλειδί. Για λόγους πληρότητας, υπενθυμίζεται ότι εάν ισχύουν οι συναρτησιακές εξαρτήσεις: $X \rightarrow A$, $A \rightarrow B$, με την εφαρμογή του *μεταβατικού κανόνα*, δημιουργείται η *μεταβατική* συναρτησιακή εξάρτηση $X \rightarrow B$.

Ας υποθέσουμε ότι έχουμε μια σχέση, όπου καταχωρείται πληροφορία σχετικά με τους υπαλλήλους μιας εταιρίας, καταχωρώντας τον κωδικό και το ονοματεπώνυμο του κάθε εργαζομένου, καθώς και τον κωδικό και το όνομα του κάθε τμήματος στο οποίο εργάζεται. Η σχέση έχει την παρακάτω δομή:

EMPLOYEE (Emp_ID, Emp_LastName, Dept_ID, Dept_Name).

Οι συναρτησιακές εξαρτήσεις που προκύπτουν από την παραπάνω σχέση είναι οι εξής:

Emp_ID → Emp_LastName, Emp_ID → Dept_ID, Dept_ID → Dept_Name.

Ο πίνακας δεν βρίσκεται σε 3NF λόγω της τελευταίας εξάρτησης, από τη στιγμή που το Dept_ID δεν αποτελεί κλειδί του πίνακα. Αν παρατηρήσουμε πιο προσεκτικά θα ανακαλύψουμε ότι επί της ουσίας υπάρχει μεταβατική εξάρτηση του γνωρίσματος Dept_Name από το πρωτεύον κλειδί Emp_ID, με συνδυασμό των δυο τελευταίων εξαρτήσεων.

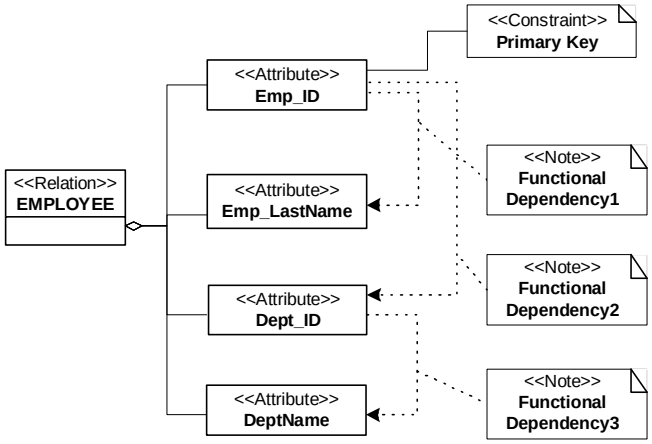
Τα προβλήματα που εμφανίζονται σε έναν πίνακα που δεν είναι σε 3NF, είναι τα ίδια με αυτά που μελετήθηκαν αναλυτικά στη 2NF. Η απομάκρυνση του πλεονασμού και ο περιορισμός, έως ένα βαθμό τουλάχιστον, του φαινομένου της ασυνέπειας, παρέχεται μέσω της κανονικοποίησης του πίνακα σε 3NF. Η κανονικοποίηση ενός πίνακα ώστε να υποστηρίζει την 3NF πραγματοποιείται με τα εξής βήματα:

1) Εύρεση των μεταβατικών εξαρτήσεων και γενικά των εξαρτήσεων μεταξύ γνωρισμάτων όπου το αριστερό μέλος δεν αποτελεί πρωτεύον κλειδί. Οι εξαρτήσεις, στις οποίες στο αριστερό μέλος τους εμφανίζεται το ίδιο γνώρισμα, ομαδοποιούνται.
2) Για κάθε μια από τις ομάδες που δημιουργήθηκαν, κατασκευάζεται ένας νέος πίνακας, ο οποίος περιλαμβάνει ως πεδία όλα τα γνωρίσματα των συναρτησιακών εξαρτήσεων. Τα γνωρίσματα που εμφανίζονται στο δεξί μέλος κάθε εξάρτησης απομακρύνονται από τον αρχικό πίνακα.
3) Σε κάθε νέο πίνακα ορίζεται ως πρωτεύον κλειδί το αριστερό μέλος της αντίστοιχης συναρτησιακής εξάρτησης.
4) Τα γνωρίσματα του πρωτεύοντος κλειδιού του πίνακα που διασπάστηκε αποτελούν ξένα κλειδιά προς τα αντίστοιχα γνωρίσματα των νέων πινάκων που προέκυψαν κατά την κανονικοποίηση.

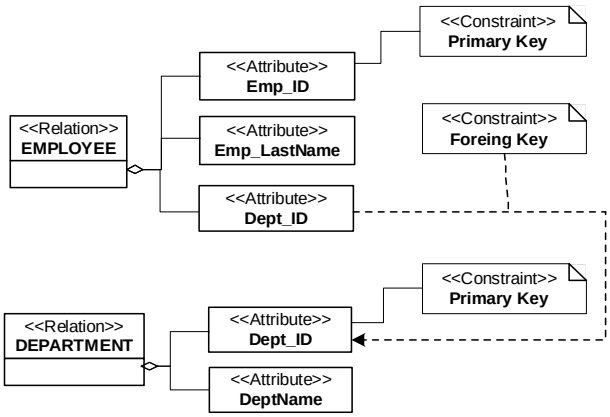
Είναι προφανές, ότι το φαινόμενο της ασυνέπειας των τιμών δεν μπορεί να εξαλειφθεί δια παντός, με την κανονικοποίηση, όμως, σε 3NF ο έλεγχος για τυχόν ασυνέπειες στις τιμές των δεδομένων πραγματοποιείται πιο γρήγορα και πιο αποδοτικά. Απομακρύνοντας, αρχικά, την πιθανότητα καταχώρησης πλεονάζουσας πληροφορίας, ελαχιστοποιείται η πιθανότητα λανθασμένης εισαγωγής ή ανανέωσης κάποιας τιμής γνωρίσματος. Ακόμα και στην περίπτωση όπου μια λάθος καταχώρηση προκαλέσει ασυνέπεια στη βάση, με την ελαχιστοποίηση της πληροφορίας που καταχωρείται γενικά στους κανονικοποιημένους πίνακες, και ιδιαίτερα σε αυτόν που προκύπτει από την «προβληματική» εξάρτηση, η εύρεση μιας τέτοια καταχώρησης μπορεί να γίνει ευκολότερα, εφόσον το πλήθος των πλειάδων είναι αρκετά πιο μικρό.

4.2.3.2. Δομή

Στο Σχήμα 4.17 παρουσιάζεται η δομή του σχήματος της σχέσης EMPLOYEE και οι συναρτησιακές εξαρτήσεις μεταξύ των γνωρισμάτων, επισημαίνοντας όλες όσες παρουσιάζουν πρόβλημα για την κανονικοποίηση. Στο Σχήμα 4.18 παρουσιάζεται η δομή του σχήματος της σχέσης σε 3NF.



Σχήμα 4.17 Αναπαράσταση του σχήματος του πίνακα του παραδείγματος που δεν βρίσκεται σε 3NF.



Σχήμα 4.18 Αναπαράσταση του σχήματος του πίνακα του παραδείγματος μετά την κανονικοποίηση σε 3NF.

Το Σχήμα 4.19 παρουσιάζει ένα στιγμιότυπο του πίνακα EMPLOYEE του παραδείγματος, στο οποίο εντοπίζεται τόσο το φαινόμενο του πλεονασμού (σκίαση

των τιμών) όσο και το φαινόμενο της ασυνέπειας (χρωματισμός των τιμών με γκρι και εμφάνιση άσπρων γραμμάτων).

EMPLOYEE

Emp_ID	Emp_LastName	Dept_ID	DeptName
01	Mallow	**20**	**IST**
02	Stamos	**20**	**IST**
03	Johnson	201	D-Mode
04	Stathos	**20**	**IST**
05	Jackson	**12**	**IST**

Σχήμα 4.19 Αναπαράσταση στιγμιότυπου του πίνακα του παραδείγματος που δεν είναι σε 3NF.

Κανονικοποιώντας τον πίνακα, απομακρύνεται το γνώρισμα Dept_Name το οποίο δημιουργεί το πρόβλημα, και τοποθετείται μαζί με το Dept_ID σε έναν νέο πίνακα, ικανοποιώντας και τη συναρτησιακή εξάρτηση που υπάρχει μεταξύ τους. Επιπλέον, σύμφωνα με τους κανόνες της κανονικοποίησης, το Dept_ID παίζει το ρόλο του κλειδιού στο νέο πίνακα. Το στιγμιότυπο του κανονικοποιημένου πίνακα παρουσιάζεται στο Σχήμα 4.20.

EMPLOYEE

Emp_ID	Emp_LastName	Dept_ID
01	Mallow	20
02	Stamos	20
03	Johnson	201
04	Stathos	20
05	Jackson	12

DEPARTMENT

Dept_ID	DeptName
20	IST
201	D-Mode
12	**IST**

Σχήμα 4.20 Αναπαράσταση στιγμιότυπου των δύο πινάκων του παραδείγματος όπως προκύπτουν μετά την κανονικοποίηση σε 3NF.

4.2.3.3. Συμπεριφορά σε επίπεδο στιγμιότυπου

Η συμπεριφορά της βάσης σε επίπεδο στιγμιότυπου για τις λειτουργίες *εισαγωγή*, *διαγραφή* και *ανανέωση* όσον αφορά στην κανονικοποίηση σε 3NF δεν παρουσιάζει

καμία διαφορά με αυτή σε 2NF. Επιγραμματικά αναφέρεται ότι κατά την διαδικασία της εισαγωγής μιας νέας πλειάδας στον νέου πίνακα που περιλαμβάνει τα πεδία των εξαρτήσεων, στο παράδειγμά μας ο πίνακας DEPARTMENT, δεν συνεπάγεται εισαγωγή της αντίστοιχης τιμής στα αντίστοιχα γνωρίσματα του EMPLOYEE. Αντίθετα για οποιαδήποτε εισαγωγή μιας πλειάδας στον πίνακα EMPLOYEE θα πρέπει πρώτα να ελέγχεται και εάν δεν υπάρχει η αντίστοιχη καταχώρηση να ενημερώνεται ο πίνακας DEPARTMENT.

Παρόμοια, η διαγραφή μιας πλειάδας από τον πίνακα με τις αναφορές ξένων κλειδιών, στο παράδειγμά μας ο πίνακας EMPLOYEE, δεν προκαλεί το φαινόμενο της «αλυσιδωτής» διαγραφής πλειάδων του πίνακα DEPARTMENT. Εν αντιθέσει, οποιαδήποτε διαγραφή πραγματοποιηθεί στον πίνακα DEPARTMENT συνεπάγεται άμεση διαγραφή των αντίστοιχων πλειάδων από τον πίνακα EMPLOYEE.

Τέλος, η ανανέωση οποιουδήποτε γνωρίσματος, πλην των πρωτευόντων κλειδιών των πινάκων, δεν επηρεάζει τη συμπεριφορά της βάσης σε επίπεδο στιγμιότυπου. Η ανανέωση στα πεδία των πρωτευόντων κλειδιών επιφέρει ανανέωση της τιμής του αντίστοιχου γνωρίσματος, όπου αυτή εμφανίζεται. Π.χ., ας υποθέσουμε ότι πραγματοποιείται μια ανανέωση του πεδίου Dept_ID =20 στην τιμή Dept_ID =100. Τότε είναι απαραίτητη η ενημέρωση των αντίστοιχων καταχωρήσεων όλων των πινάκων, στην περίπτωσή μας ο EMPLOYEE, που περιλαμβάνουν το πεδίο Dept_ID ως ξένο κλειδί.

4.2.3.4. Συμπεριφορά σε επίπεδο σχήματος

Η συμπεριφορά του προτύπου σε επίπεδο σχήματος, στην περίπτωση της 3NF, δεν παρουσιάζει καμία διαφορά με την περίπτωση της 2NF. Επιγραμματικά και εδώ αναφέρεται, ότι κατά την εισαγωγή και την διαγραφή ενός γνωρίσματος, θα πρέπει να λαμβάνονται υπόψη οι τυχόν συναρτησιακές εξαρτήσεις που προκύπτουν (κατά την εισαγωγή γνωρίσματος) ή που αναιρούνται (κατά τη διαγραφή γνωρίσματος). Η δημιουργία μιας νέας εξάρτησης, οδηγεί σε εκ νέου κανονικοποίηση του πίνακα, ενώ η αναίρεση μιας εξάρτησης, ίσως επιφέρει κατάργηση ενός πίνακα. Η ανανέωση ενός πεδίου μπορεί να προκαλέσει την ανανέωση των αντίστοιχων τιμών περισσοτέρων

του ενός πίνακα, όταν το πεδίο είναι κλειδί, ή ενός πίνακα όταν το γνώρισμα δεν αποτελεί κλειδί. Επιπλέον κατά την λειτουργία της ανανέωσης ενός γνωρίσματος κλειδιού, θα πρέπει να πραγματοποιηθεί έλεγχος για τυχόν ύπαρξη, και απομάκρυνση, πλεονάζουσας πληροφορίας.

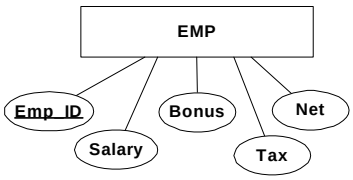
4.2.4. Πλεονεκτήματα & Μειονεκτήματα

- Έχει ήδη αναφερθεί, ότι ο σκοπός της κανονικοποίησης είναι η αποφυγή του φαινομένου του πλεονασμού και ο περιορισμός, ως ένα βαθμό, του φαινομένου της ασυνέπειας που παρουσιάζεται στις πλειάδες μιας βάσης δεδομένων. Έχοντας αναπτύξει αρκετά και τα δύο φαινόμενα, και μελετώντας τα μέσα από παραδείγματα, είναι προφανές ότι με την κανονικοποίηση σε 3NF επιτυγχάνεται ο στόχος.
- Ένα άλλο πλεονέκτημα της κανονικοποίησης, το οποίο αποτελεί και απόρροια του φαινομένου του πλεονασμού, είναι η μείωση του χώρου αποθήκευσης που απαιτείται για την καταχώρηση των πλειάδων. Ομαδοποιώντας και διασπώντας την πληροφορία ανάλογα με τις εξαρτήσεις που λαμβάνουν χώρα ανάμεσα στα γνωρίσματα, αποφεύγεται η επανάληψη πληροφορίας σε έναν πίνακα και συνεπώς η άσκοπη δέσμευση αποθηκευτικού χώρου.
- Μειονέκτημα της κανονικοποίησης αποτελεί το γεγονός ότι για την ανάκτηση της προηγούμενης πληροφορίας του παρέχεται από τον μη κανονικοποιημένο πίνακα, είναι απαραίτητη η εφαρμογή της λειτουργίας της συνένωσης πινάκων. Π.χ., ας υποθέσουμε ότι ζητείται η πληροφορία «*να βρεθούν τα ονόματα όλων των εργαζομένων του τμήματος X*» της σχέσης `EMPLOYEE` του παραδείγματος. Σε αυτή την περίπτωση θα πρέπει να πραγματοποιηθεί συνένωση των δυο πινάκων, `EMPLOYEE` και `DEPARTMENT`, του Σχήματος 4.9 για την εύρεση της συγκεκριμένης πληροφορίας. Στον μη κανονικοποιημένο πίνακα η εύρεση πραγματοποιούνταν απλά με την αναζήτηση των αντίστοιχων πλειάδων που ικανοποιούσαν τη συνθήκη `Dept_ID = X`.

4.3. Αξονική Περιστροφή (Pivot Case)

4.3.1. Κίνητρο & Εφαρμογή

Για την κατανόηση της περίπτωσης του *pivot* (*αξονική περιστροφή*) ας θεωρήσουμε το παράδειγμα καταγραφής των οικονομικών στοιχείων των εργαζομένων μιας εταιρείας. Η αναπαράσταση του σχετικού πίνακα στο ER μοντέλο, η οποία παρουσιάζεται στο Σχήμα 4.21, δίνει μια γενική εικόνα του προβλήματος. Πιο αναλυτικά, στον πίνακα EMP καταχωρείται ο μισθός (Salary), τα πιθανά επιδόματα (Bonus), ο φόρος (Tax) που αναλογεί, και το καθαρό εισόδημα (Net) του κάθε εργαζομένου. Παρατηρούμε ότι τα παραπάνω γνωρίσματα ανήκουν στην γενική κατηγορία οικονομικά (Financials).



Σχήμα 4.21 Αναπαράσταση του παραδείγματος στο ER.

Σε περιπτώσεις, λοιπόν, όπου κάποιες από τις ιδιότητες της οντότητας που μοντελοποιείται αποτελούν μέρη μιας γενικής ιδιότητας, η αναπαράστασή της στο σχεσιακό μοντέλο μπορεί να πραγματοποιηθεί με δυο διαφορετικούς τρόπους.

- *Flat design*: παράθεση των ιδιοτήτων ως ξεχωριστά πεδία στη σχέση που αναπαριστά την οντότητα. Για το συγκεκριμένο παράδειγμα το flat design στο σχεσιακό σχήμα θα έχει την εξής δομή:

  EMP (Emp_ID, Salary, Bonus, Tax, Net).

- *Encoded design*: δημιουργία δύο σχέσεων. Στη μια σχέση, η οποία αποτελεί τη *βασική σχέση* (*master relation* ή *master table*), καταχωρούνται οι ιδιότητες με κωδικούς και οι αντίστοιχες τιμές τους, ενώ στη δεύτερη σχέση, η οποία παίζει τον ρόλο συμβουλευτικού πίνακα (*lookup relation* ή *lookup table*), καταχωρούνται οι κωδικοί με τις περιγραφές τους. Η δομή του παραδείγματος EMP στο σχεσιακό σχήμα με encoded design είναι:

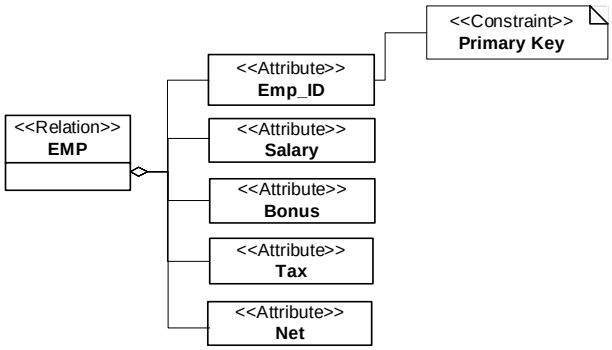
EMP (Emp_ID, Amt_ID, Amt_Value) (*master relation*)

AMOUNT_TYPES (Amt_ID, Amt_Description) (*lookup relation*)

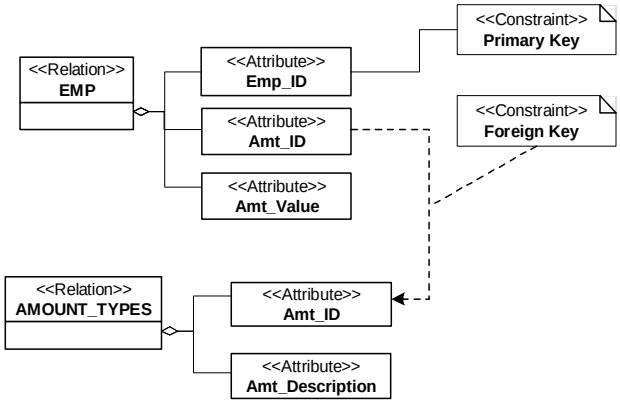
Η μετάβαση από το encoded design στο flat ονομάζεται pivot. Σε αυτή την ενότητα θα μελετήσουμε και θα συγκρίνουμε τη συμπεριφορά των δυο σχεδιαστικών μεθόδων σε επίπεδο σχήματος και στιγμιότυπου.

4.3.2. Δομή

Στα Σχήματα 4.22 και 4.23 παρουσιάζεται η δομή του σχήματος του flat design και του encoded design αντίστοιχα.



Σχήμα 4.22 Αναπαράσταση του σχήματος του παραδείγματος με χρήση του προτύπου flat design.



Σχήμα 4.23 Αναπαράσταση του σχήματος του παραδείγματος με χρήση του προτύπου encoded design.

Στο Σχήμα 4.24 παρουσιάζεται ένα στιγμιότυπο του παραδείγματος με flat design, ενώ στο Σχήμα 4.25 ένα στιγμιότυπο με encoded design.

EMP

Emp_ID	Salary	Bonus	Tax	Net
01	1500	300	500	1300
02	2000	500	700	1800
03	1500		500	1000
04	1000			1000

Σχήμα 4.24 Αναπαράσταση στιγμιότυπου του παραδείγματος με χρήση του προτύπου flat design.

Παρατηρώντας τα παρακάτω στιγμιότυπα προκύπτει ότι, το flat design εκτείνεται σε πλάτος, δηλαδή παρατίθενται όλα τα πεδία ονομαστικά, ενώ το encoded design εκτείνεται σε μήκος, εφόσον εισάγονται πολλές εγγραφές για κάθε υπάλληλο στον βασικό πίνακα.

EMP

Emp_ID	Amt_ID	Amt_Value
01	1	1500
01	2	300
01	3	500
01	4	1300
02	1	2000
02	2	500
02	3	700
02	4	1800
03	1	1500
03	3	500
03	4	1000
04	1	1000
04	4	1000

AMOUNT_TYPES

Amt_ID	Amt_Description
1	Salary
2	Bonus
3	Tax
4	Net

Σχήμα 4.25 Αναπαράσταση στιγμιότυπου του παραδείγματος με χρήση του προτύπου encoded design.

4.3.3. Συμπεριφορά σε επίπεδο στιγμιότυπου

Θεωρώντας ως αρχικό σχήμα της βάσης, την απεικόνιση στο ER μοντέλο του Σχήματος 4.21, σε αυτή την ενότητα θα μελετήσουμε τη συμπεριφορά των δύο σχεδιαστικών προτύπων σε επίπεδο στιγμιότυπου. Εκτός από τις συνήθεις λειτουργίες, *εισαγωγή*, *διαγραφή* και *ανανέωση* μιας πλειάδας, επιπλέον θα μελετηθεί η λειτουργία της *επιλογής* μιας πλειάδας του σχήματος της βάσης.

4.3.3.1. Επιλογή

Ας θεωρήσουμε το παράδειγμα επιλογής ενός υπαλλήλου και την εμφάνιση όλων των οικονομικών του στοιχείων. Μια τέτοια λειτουργία παρουσιάζει ενδιαφέρον κυρίως στο encoded πρότυπο, όπου δεν αντιστοιχεί σε μια απλή επιλογή μιας πλειάδας από των πίνακα των καταχωρήσεων. Ουσιαστικά ας σκεφτούμε την λειτουργία της επιλογής ως το pivot case, δηλαδή την μετάβαση από το encoded πρότυπο στο flat, ή ακριβέστερα στην αναπαράσταση της οντότητας στο ER μοντέλο. Στην περίπτωση που το αρχείο, στο οποίο αποθηκεύονται οι πλειάδες του πίνακα καταχωρήσεων, δεν παρουσιάζει καμία οργάνωση σε φυσικό επίπεδο, δηλαδή αποτελεί *αρχείο σωρού*, το pivot αποτελεί δύσκολη και χρονοβόρα διαδικασία (Σχήμα 4.26). Δυνατότητα βελτίωσης παρουσιάζεται εάν δημιουργήσουμε ένα *ευρετήριο ομαδοποίησης* (*clustered index*) στα γνωρίσματα του πρωτεύοντος κλειδιού. Το cluster index υλοποιείται με την δομή ενός *B+ δέντρου*, όπου στα φύλλα του δέντρου είναι αποθηκευμένα τα δεδομένα. Το πλεονέκτημα αυτής της δομής, εκτός του ότι στα φύλλα του δέντρου έχουμε την ακριβή πληροφορία και όχι κάποιον δείκτη σε αυτή, είναι ότι τα δεδομένα καταχωρούνται σε φυσικό επίπεδο με βάση την ταξινόμηση του clustered index. Εάν, λοιπόν, εφαρμοστεί ένας clustered index πάνω στα πεδία (`Emp_ID, Amt_ID`) του encoded προτύπου οι καταχωρήσεις του βασικού πίνακα θα παρουσιάζουν την μορφή του Σχήματος 4.25 σε επίπεδο στιγμιότυπου. Υπό αυτή τη μορφή, από το βασικό πίνακα μπορούμε πιο εύκολα να μεταβούμε στο πίνακα του flat προτύπου, δηλαδή να ανακατασκευάσουμε την πληροφορία. Τέλος, αναφέρεται ότι σε περίπτωση όπου η σειρά εμφάνισης των γνωρισμάτων, κατά την ανακατασκευή της πληροφορίας, δεν είναι ίδια με τη σειρά καταχώρησης των γνωρισμάτων στον συμβουλευτικό πίνακα θα πρέπει να εισαχθεί μια *λίστα ταξινόμησης* (*ordering list*) στους κωδικούς των ομαδοποιημένων γνωρισμάτων.

EMP

Emp_ID	Amt_ID	Amt_Value
01	1	1500
02	2	500
02	4	1800
03	3	500
03	4	1000
04	4	1000
01	2	300
02	1	2000
03	1	1500
04	1	1000
01	3	500
02	3	700
01	4	1300

Σχήμα 4.26 Αναπαράσταση στιγμιότυπου της σχέσης EMP για το encoded design, με δομή αρχείου σωρού σε φυσικό επίπεδο αποθήκευσης.

4.3.3.2. Εισαγωγή

Ας θεωρήσουμε το παράδειγμα εισαγωγής ενός νέου υπαλλήλου, σκεπτόμενοι πάντα τον πίνακα EMP του Σχήματος 4.21. Στο flat πρότυπο μια τέτοια εισαγωγή αντιστοιχεί σε μια και μόνο εισαγωγή πλειάδας στον αντίστοιχο πίνακα, εν αντιθέσει με το encoded πρότυπο όπου αντιστοιχεί σε πολλαπλές εισαγωγές στον master πίνακα. Το μειονέκτημα που παρουσιάζει το encoded πρότυπο σε αυτό το σημείο είναι προφανές, εφόσον επιβάλει την εισαγωγή τόσων πλειάδων όσα είναι και τα οικονομικά πεδία, στα οποία συμμετέχει ο νέος υπάλληλος. Το μη προφανές μειονέκτημα του flat προτύπου προκύπτει, αν σκεφτούμε την περίπτωση όπου ο νέος υπάλληλος δεν συμμετέχει στα περισσότερα πεδία των οικονομικών. Σε αυτή την περίπτωση παρατηρείται το φαινόμενο των πολλαπλών τιμών NULL και συνεπώς άσκοπη δέσμευση αποθηκευτικού χώρου για το flat πρότυπο, φαινόμενο που αποφεύγεται στο encoded πρότυπο.

4.3.3.3. Διαγραφή

Ανάλογα με την λειτουργία της εισαγωγής, συμπεριφέρονται τα δυο πρότυπα και στην περίπτωση διαγραφής ενός υπαλλήλου. Στο flat πρότυπο για την διαγραφή ενός υπαλλήλου από τον πίνακα, διαγράφεται μόνο η αντίστοιχη πλειάδα, ενώ στο encoded πρότυπο θα πρέπει να διαγραφούν όλες οι πλειάδες που αντιστοιχούν στον συγκεκριμένο υπάλληλο.

4.3.3.4. Ανανέωση

Η λειτουργία της ανανέωσης ενός πεδίου μιας πλειάδας δεν παρουσιάζει σημαντικές διαφορές στον τρόπο εφαρμογής της στα δύο πρότυπα. Η μόνη διαφορά που παρατηρείται ανάμεσα στα δυο πρότυπα είναι ότι για την ανανέωση ενός πεδίου που έχει κωδικοποιηθεί, στο encoded πρότυπο, προϋποθέτεται η εύρεση του αντίστοιχου κωδικού στον συμβουλευτικό πίνακα, και εν συνεχεία η εύρεση της πλειάδας και η ανανέωση της στο βασικό πίνακα. Π.χ., έστω ότι στον υπάλληλο με `Emp_ID = 10` πραγματοποιούνταν ανανέωση του `Bonus = 1000`, σε αυτή την περίπτωση θα πρέπει πρώτα να αναζητηθεί το `Amt_ID` του `Bonus` στον πίνακα lookup και με βάση αυτό να βρεθεί και ανανεωθεί η αντίστοιχη πλειάδα του υπαλλήλου στο βασικό πίνακα.

Σε αυτό το σημείο θα πρέπει να παρατηρηθεί ότι πιθανώς κάποια ανανέωση των τιμών των οικονομικών πεδίων μπορεί να επιφέρει ανανέωση και άλλων οικονομικών πεδίων, δηλαδή εάν ανανεωθεί ο βασικός μισθός, `Salary`, κάποιου υπαλλήλου προφανώς και ανανεώνεται τόσο ο φόρος, `Tax`, όσο και το καθαρό του εισόδημα, `Net`. Σε αυτή την ενότητα σκοπίμως θα παραβλέψουμε το οικονομολογικό μέρος των ανανεώσεων, διότι θα μελετηθούν εκτενέστερα στην ενότητα 4.4 Derived values.

4.3.4. Συμπεριφορά σε επίπεδο σχήματος

Σκεπτόμενοι πάντα το `ER` μοντέλο του Σχήματος 4.21, ας υποθέσουμε ότι ζητούμενο είναι η εισαγωγή ενός νέου πεδίου, π.χ., το `Bonus2`, στο σχήμα της βάσης. Η συγκεκριμένη πράξη, αντιστοιχεί στην εισαγωγή ενός νέου γνωρίσματος στο σχεσιακό σχήμα του flat προτύπου. Στο encoded πρότυπο, από την άλλη, αυτή η πράξη πραγματοποιείται διαφορετικά, απλά με την εισαγωγή μιας νέας πλειάδας στον

συμβουλευτικό πίνακα. Με άλλα λόγια αν επιθυμούμε να εισάγουμε το πεδίο Bonus2, ως ένα νέο πεδίο οικονομικών, απλά εισάγουμε μια νέα πλειάδα στον πίνακα Amount_Types (Σχήμα 4.27), χωρίς να απαιτείται καμία ανανέωση του βασικού πίνακα.

AMOUNT_TYPES

Amt_ID	Amt_Description
1	Salary
2	Bonus
3	Tax
4	Net
5	Bonus2

Σχήμα 4.27 Αναπαράσταση στιγμιότυπου του συμβουλευτικού πίνακα για το encoded design μετά την εισαγωγή ενός νέου πεδίου.

Αντίστοιχα, η διαγραφή ενός πεδίου του ER μοντέλου, στο flat πρότυπο αναπαρίσταται με την διαγραφή του συγκεκριμένου γνωρίσματος από το σχήμα της βάσης, ενώ στο encoded πρότυπο με την διαγραφή της αντίστοιχης πλειάδας από τον συμβουλευτικό πίνακα. Παρατηρούμε, λοιπόν, ότι η εισαγωγή/διαγραφή ενός πεδίου αντιμετωπίζεται στο flat πρότυπο με τον κλασικό τρόπο εισαγωγής/διαγραφής γνωρίσματος στη βάση, μεταβάλλοντας το σχήμα της. Εν αντιθέσει, στο encoded πρότυπο οι αλλαγές αυτές πραγματοποιούνται χωρίς αλλαγή του σχήματος της βάσης, απλά με την εισαγωγή/διαγραφή της αντίστοιχης πλειάδας του συμβουλευτικού πίνακα.

Τέλος, για τη λειτουργία της ανανέωσης ενός πεδίου ας θεωρήσουμε το παράδειγμα ανανέωσης των τιμών του γνωρίσματος Bonus όπου θέτονται όλες Bonus = 1000. Αυτή αποτελεί την χειρότερη περίπτωση ανανέωσης πεδίου τουλάχιστον όσον αφορά στο encoded πρότυπο, όπου πραγματοποιούνται οι εξής διαδικασίες:

1. Εύρεση του κωδικού Amt_ID που αντιστοιχεί στο πεδίο Bonus στον lookup πίνακα.
2. Εύρεση όλων των πλειάδων με το αντίστοιχο Amt_ID και αντικατάσταση των τιμών Amt_Value = 1000.

3. Εισαγωγή νέων εγγραφών για τους υπαλλήλους που δεν έχουν καταχώρηση για το πεδίο `Bonus`.

Για την διευκόλυνση της εφαρμογής του 3^{ου} βήματος είναι απαραίτητη η ύπαρξη ενός από τα παρακάτω:

- Clustered index πάνω στο πρωτεύον κλειδί του βασικού πίνακα, ώστε και ο πίνακας να είναι ταξινομημένος και σε φυσικό επίπεδο, με βάση τα γνωρίσματα του πρωτεύοντος κλειδιού.
- Πρωτεύον ευρετήριο στο πρωτεύον κλειδί, `(Emp_ID, Amt_ID)`, του βασικού πίνακα.
- Ευρετήριο στο γνώρισμα `Emp_ID` και λίστα ταξινόμησης στο `Amt_ID`.

4.3.5. Πλεονεκτήματα & Μειονεκτήματα

- Βασικό πλεονέκτημα του flat προτύπου είναι η ευκολία εφαρμογής των λειτουργιών εισαγωγή, διαγραφή και ανανέωση σε επίπεδο στιγμιότυπου σε σχέση με το encoded πρότυπο. Όσον αφορά στο flat πρότυπο αρχικά πραγματοποιείται εύρεση της αντίστοιχης πλειάδας και μετά εφαρμόζεται η οποιαδήποτε αλλαγή (διαγραφή/ανανέωση). Ακόμα και στην περίπτωση εισαγωγής μιας νέας πλειάδας το κέρδος του flat προτύπου είναι η πιο γρήγορη εύρεση της αντίστοιχης θέσης, ταξινομώντας το αρχείο με βάση το πρωτεύον κλειδί, εφόσον το αρχείο περιέχει λιγότερες εγγραφές. Εν αντιθέσει, στο encoded πρότυπο η εφαρμογή των λειτουργιών διαγραφή ή/και ενημέρωση επιβάλει πρωτίστως την εύρεση του κατάλληλου κωδικού από τον συμβουλευτικό πίνακα και εν συνεχεία την εύρεση και μετατροπή της αντίστοιχης πλειάδας, με βάση τον κωδικό αυτό, στον κυρίως πίνακα. Τέλος, κατά την λειτουργία της εισαγωγής μιας νέας πλειάδας στο encoded πρότυπο θα πρέπει, και σε αυτή την περίπτωση, να πραγματοποιηθεί εύρεση της κατάλληλης θέσης στην οποία θα πρέπει να εισαχθεί η πλειάδα με βάση το πρωτεύον κλειδί, ώστε το αρχείο να είναι ταξινομημένο. Η μόνη διαφορά είναι ότι επειδή το πλήθος των πλειάδων είναι πολύ μεγαλύτερο, στο encoded πρότυπο, η αναζήτηση της θέσης απαιτεί περισσότερο χρόνο.

- Μειονέκτημα, με όσα προελέχθησαν, του flat προτύπου αποτελεί η πιθανότητα εμφάνισης πολλών NULL τιμών, γεγονός που αποφεύγεται στο encoded πρότυπο. Δεδομένου ότι, εξ' αρχής, έχει δεσμευτεί χώρος για το κάθε πεδίο-γνώρισμα του flat προτύπου, η εμφάνιση NULL τιμών σε κάποια από αυτά προκαλεί άσκοπη δέσμευση αποθηκευτικού χώρου. Το φαινόμενο αυτό αποφεύγεται στο encoded πρότυπο από τη στιγμή όπου στον master πίνακα καταχωρούνται μόνο πλειάδες στις οποίες αντιστοιχίζεται τιμή σε κάποιον συγκεκριμένο κωδικό.
- Ένα επίσης σημαντικό πλεονέκτημα του encoded προτύπου είναι η εύκολη «αλλαγή» του σχήματος της βάσης. Ενώ στο flat πρότυπο η εισαγωγή ή η διαγραφή μιας ιδιότητας (πεδίου) της οντότητας που μοντελοποιείται (σχηματική αναπαράσταση στο ER μοντέλο) συνεπάγεται την εισαγωγή ή την διαγραφή αντίστοιχα κάποιου γνωρίσματος του πίνακα της βάσης, στο encoded πρότυπο αυτού του είδους οι αλλαγές πραγματοποιούνται απλά με την εισαγωγή ή την διαγραφή της αντίστοιχης πλειάδας στον συμβουλευτικό πίνακα. Ουσιαστικά οποιαδήποτε αλλαγή των ιδιοτήτων της οντότητας που μοντελοποιείται, αντιμετωπίζεται στο flat πρότυπο ως αλλαγή σε επίπεδο σχήματος, ενώ στο encoded ως αλλαγή σε επίπεδο στιγμιότυπου.
- Το βασικότερο μειονέκτημα του encoded προτύπου υπόκεινται στην δυσκολία ανακατασκευής της πληροφορίας σε σχέση με το ER μοντέλο αναπαράστασης. Όπως προαναφέρθηκε, σημαντική βελτίωση αποτελεί η χρήση clustered index στα γνωρίσματα του πρωτεύοντος κλειδιού, ταξινομώντας και σε φυσικό επίπεδο τις καταχωρήσεις.
- Συγκρινόμενα τα δύο πρότυπα, το flat μοντέλο αποτελεί πιο κατανοητή, από τους χρήστες του συστήματος, μοντελοποίηση μιας οντότητας.
- Με βάση τα χαρακτηριστικά του κάθε προτύπου, το flat πρότυπο αποδίδει καλύτερα σε *web-based* περιβάλλοντα, όπου οποιαδήποτε αλλαγή στο σχήμα της βάσης δεδομένων οδηγεί σε ανανέωση της εφαρμογής με την οποία επικοινωνεί ένας χρήστης με τον server. Αντίθετα το encoded πρότυπο, λόγω του ότι η οποιαδήποτε αλλαγή στο σχήμα της βάσης μεταβιβάζεται σε αλλαγή σε επίπεδο στιγμιότυπου, μπορεί να εφαρμοστεί σε περιβάλλοντα *client-server*. Η ιδιαιτερότητα ενός περιβάλλοντος client-server έγκειται στο γεγονός

ότι μια αλλαγή στο σχήμα της βάσης δεδομένων συνεπάγεται αυτόματα αλλαγή τόσο της εφαρμογής από πλευράς server, μέσω της οποίας επικοινωνεί ένας client με τον server, όσο και της εφαρμογής η οποία είναι από την πλευρά του client, εφαρμογή η οποία είναι εγκατεστημένη στον client. Εφόσον η αλλαγή του σχήματος της βάσης, στο encoded πρότυπο πραγματοποιείται με την εισαγωγή, διαγραφή ή ανανέωση κάποιας πλειάδας του lookup πίνακα, για client-server περιβάλλοντα το encoded μοντέλο αποτελεί μια πολύ καλή επιλογή σχεδίασης.

4.4. Παραγόμενα Γνωρίσματα (Derived Attributes)

4.4.1. Κίνητρο & Εφαρμογή

Ας θεωρήσουμε το παράδειγμα καταχώρησης των οικονομικών κάθε εργαζομένου μιας εταιρίας, όπως αυτό παρουσιάστηκε και στην προηγούμενη ενότητα. Στον πίνακα αυτό καταχωρούνται πληροφορίες σχετικά με το μεικτό μισθό (`Salary`), το πιθανό επίδομα (`Bonus`), το φόρο (`Tax`), το «καθαρό» εισόδημα (`Net`) του κάθε εργαζομένου και η χρονολογία καταγραφής. Η δομή του παραπάνω πίνακα δίνεται από τη σχέση:

`EMP_FINANCIALS (Emp_ID, Salary, Bonus, Net, Date).`

Ένα γνώρισμα ονομάζεται *παραγόμενο* (*derived*), όταν οι τιμές του μπορούν να προκύψουν από τις τιμές άλλων γνωρισμάτων με την εφαρμογή κάποιας συνάρτησης υπολογισμού. Στο παράδειγμά μας, το καθαρό εισόδημα κάθε εργαζομένου μπορεί να υπολογιστεί με βάση τις τιμές των πεδίων `Salary` και `Bonus` μέσω της συνάρτηση `Net=(Salary+Bonus)/1.19`. Η συνάρτηση αυτή απεικονίζει το τρέχον φορολογικό σύστημα, όπου το `19%` του μεικτού εισοδήματος ενός εργαζομένου, αντιστοιχίζεται στο φόρο προστιθέμενης αξίας (`ΦΠΑ`).

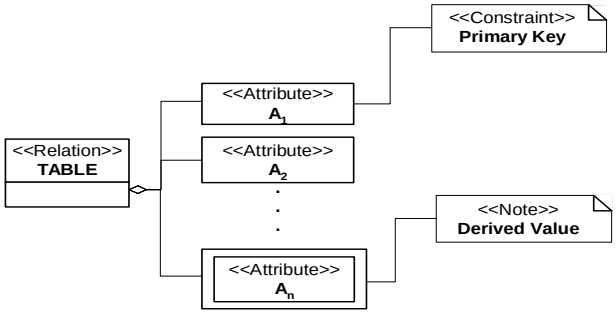
Ο περιορισμός, των παραγόμενων γνωρισμάτων, είναι το ότι θα πρέπει πάντα οι τιμές τους να είναι συνεπείς με τη συνάρτηση υπολογισμού που ίσχυε τη στιγμή της τελευταίας μεταβολής τους. Για την κατανόηση του περιορισμού αυτού ας

υποθέσουμε ότι στον πίνακα, EMP_FINANCIALS, του παραδείγματος μας θέλουμε να καταχωρούνται τα οικονομικά των εργαζομένων για τα τελευταία 10 χρόνια. Ας υποθέσουμε επίσης, ότι το φορολογικό σύστημα, ήτοι η συνάρτηση υπολογισμού του παραγόμενου πεδίου, αλλάζει το 2000. Σε αυτή την περίπτωση θα πρέπει οι νέες τιμές του πεδίου Net να υπολογίζονται βάσει της νέας συνάρτησης, ενώ οι παλιές τιμές (πριν από το 2000) να παραμείνουν ως έχουν.

Ο παραπάνω περιορισμός θα πρέπει να ισχύει ακόμα και όταν προστεθεί στη συνάρτηση υπολογισμού ένα επιπλέον πεδίο το οποίο θα πρέπει να ληφθεί υπόψη κατά τον υπολογισμό του παραγόμενου πεδίου. Ας υποθέσουμε ότι, στον πίνακα EMP_FINANCIALS εισάγεται ένα νέο πεδίο, Bonus2, το οποίο αντιπροσωπεύει ένα επιπλέον επίδομα που δίδεται σε ορισμένους εργαζομένους. Η συνάρτηση υπολογισμού του Net θα πρέπει να μεταβληθεί έτσι ώστε να λαμβάνει υπόψη το νέο πεδίο, π.χ., Net=(Salary+Bonus+Bonus2)/1.19. Και σε αυτή την περίπτωση, οι νέες εγγραφές θα υπολογίζονται με βάση την νέα συνάρτηση υπολογισμού, ενώ οι παλιές τιμές του πεδίου Net θα πρέπει να μείνουν αμετάβλητες, εφόσον αντιστοιχούν στην παλιά συνάρτηση υπολογισμού.

4.4.2. Δομή

Η πρώτη σχεδιαστική λύση, και ίσως η πιο απλή στην κατασκευή της, περιλαμβάνει την εισαγωγή όλων των παραγόμενων πεδίων στον πίνακα καταχώρησης. Στο Σχήμα 4.28 παρουσιάζεται μια γενική απεικόνιση ενός πίνακα που συμπεριλαμβάνει παραγόμενο γνώρισμα A_n, το οποίο είναι ο συνδυασμός των A_2, A_3, ..., A_{n-1} πεδίων του TABLE. Ένα στιγμιότυπο της σχέσης EMP_FINANCIALS παρουσιάζεται στο Σχήμα 4.29.



Σχήμα 4.28 Αναπαράσταση δομής γενικού σχήματος της πρώτης σχεδιαστικής λύσης, με εισαγωγή επιπλέον πεδίου, για το παραγόμενο γνώρισμα.

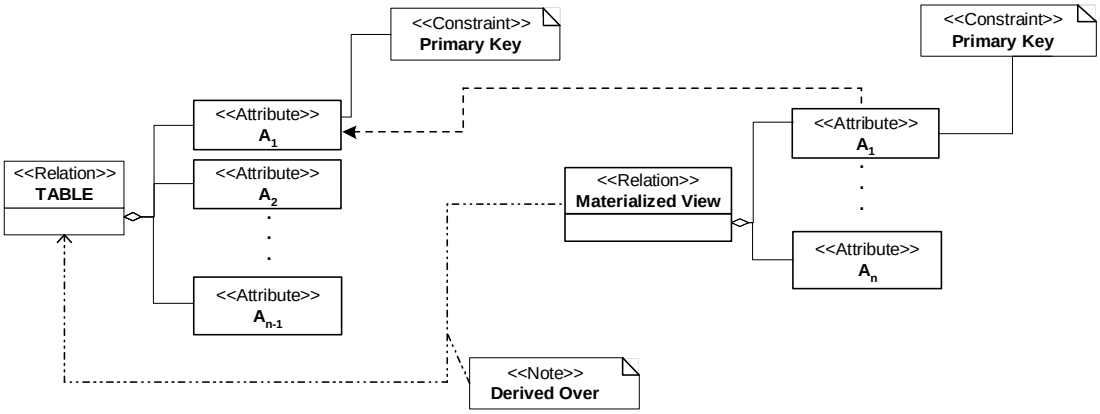
EMP_FINANCIALS

Emp_ID	Salary	Bonus	Net	Date
1	1000.00	0	840.34	08/09/07
2	2000.00	1000.0	2521.00	08/09/07

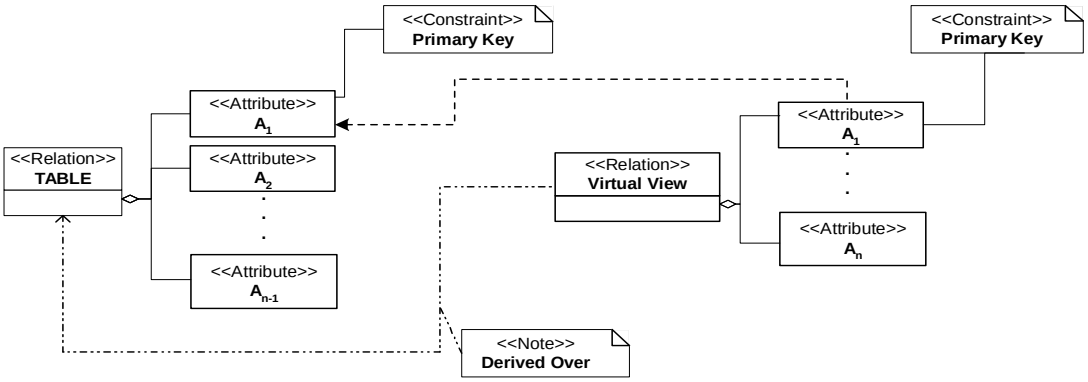
Σχήμα 4.29 Αναπαράσταση στιγμιότυπου της σχέσης EMP_FINANCIALS με χρήση της πρώτης σχεδιαστικής λύσης.

Η δεύτερη και τρίτη σχεδιαστική παραλείπουν την ύπαρξη του παραγόμενου πεδίου, εφόσον αυτό μπορεί να προκύψει, με χρήση μιας δεδομένης συνάρτησης υπολογισμού, από τα άλλα πεδία. Για την παραγωγή, κάθε φορά, των τιμών του παραγόμενου πεδίου μπορεί να χρησιμοποιηθεί μια *όψη* (*view*). Μια όψη είναι μια σχέση που παράγεται από άλλες σχέσεις ή όψεις με την χρήση SQL ερωτήσεων. Ανάλογα με τον αν η όψη αποθηκεύεται κάπου στην βάση δεδομένων ή όχι, υπάρχουν δύο τρόποι κατασκευής. Η *υλοποιήσιμη όψη* (*materialized view*), η οποία αποτελεί τη δεύτερη σχεδιαστική λύση απεικόνισης των παραγόμενων πεδίων, αποθηκεύει τα δεδομένα που προκύπτουν από την εφαρμογή μιας ερώτησης (SQL query). Η *εικονική όψη* (*virtual view*), η οποία αποτελεί την τρίτη σχεδιαστική λύση, δεν περιέχει στοιχεία αλλά λειτουργεί ως μακροεντολή. Κάθε φορά που τίθεται μια ερώτηση στην όψη, αυτή μεταφράζεται σε ερώτηση των υποκείμενων σχέσεων ή όψεων. Π.χ., μια materialized view για το EMP_FINANCIALS θα μπορούσε να περιλαμβάνει τα πεδία MV(Emp_ID,Net,Date). Αντίστοιχα η virtual view θα μπορούσε να είναι της μορφής VV(Emp_ID,NET,Date), χωρίς όμως να αποθηκεύονται οι τιμές κανενός από τα πεδία της. Στα σχήματα 4.30 και 4.31

παρουσιάζεται η δομή του γενικού σχήματος της δεύτερης και τρίτης σχεδιαστικής λύσης, ενώ ένα στιγμιότυπο του πίνακα EMP_FINANCIALS με χρήση της δεύτερης σχεδιαστικής λύσης δίνεται στο Σχήμα 4.32.



Σχήμα 4.30 Αναπαράσταση δομής γενικού σχήματος της δεύτερης σχεδιαστικής λύσης, με δημιουργία materialized view.



Σχήμα 4.31 Αναπαράσταση δομής γενικού σχήματος της τρίτης σχεδιαστικής λύσης, με δημιουργία virtual view.

EMP_FINANCIALS

Emp_ID	Salary	Bonus	Date
1	1000.00	0	08/09/07
2	2000.00	1000.0	08/09/07

M.VIEW

Emp_ID	Net
1	840.34
2	2521.00

Σχήμα 4.32 Αναπαράσταση στιγμιότυπου του πίνακα EMP_FINANCIALS με χρήση της δεύτερης σχεδιαστικής μεθόδου (κατασκευή materialized view).

Μέχρι στιγμής αναφερθήκαμε στην περίπτωση ύπαρξης ενός, μόνο, παραγόμενου πεδίου. Παρόμοια αντιμετωπίζεται και η περίπτωση ύπαρξης περισσοτέρων του ενός παραγόμενων πεδίων. Στην πρώτη σχεδιαστική λύση, ο πίνακας που κατασκευάζεται περιλαμβάνει όλα τα παραγόμενα πεδία ως γνωρίσματα. Διαφοροποίηση παρατηρείται κατά τη δημιουργία κατασκευής όψεων, όπου είναι δυνατή η επιλογή μιας εκ των παρακάτω λύσεων:

i. κατασκευή μίας μόνο όψης (materialized ή virtual ανάλογα με τη σχεδιαστική λύση), η οποία υπολογίζει όλα τα παραγόμενα πεδία.
ii. κατασκευή ξεχωριστών όψων για κάθε παραγόμενο πεδίο.

Κατά την περίπτωση κατασκευής μιας μόνο όψης, κυρίως materialized, η οποιαδήποτε αλλαγή ενός παραγόμενου πεδίου έχει ως αντίκτυπο την ενημέρωση όλης της materialized view. Π.χ., εάν η materialized view περιλαμβάνει δύο παραγόμενα πεδία, η αλλαγή του υπολογισμού οποιουδήποτε εκ των δύο πεδίων, οδηγεί στην ανακατασκευή της όψης και τον επαναϋπολογισμό και του άλλου πεδίου. Η συμπεριφορά του προτύπου της materialized view τόσο σε επίπεδο στιγμιότυπου όσο και σε επίπεδο σχήματος, παρουσιάζεται στις επόμενες παραγράφους.

Με την κατασκευή ξεχωριστών όψεων, παρατηρούμε ότι:

1. για την περίπτωση materialized view, απαιτείται επιπλέον αποθηκευτικός χώρος και εφαρμογή λειτουργιών συνένωσης για την εξαγωγή όλης της πληροφορίας. Π.χ., στην περίπτωση ύπαρξης δύο παραγόμενων πεδίων, η κάθε materialized view θα περιλαμβάνει τουλάχιστον δύο γνωρίσματα (το πρωτεύον κλειδί και το παραγόμενο πεδίο), παρουσιάζοντας επανάληψη πληροφορίας. Εκτός του επιπλέον αποθηκευτικού χώρου που απαιτείται, θα πρέπει να πραγματοποιηθεί συνένωση των materialized views και του βασικού πίνακα για την ανάκτηση όλης της πληροφορίας.
2. για την περίπτωση virtual view, απαιτείται η ανάγνωση όλων των πλειάδων του ίδιου πίνακα, για κάθε virtual view. Π.χ., με την δημιουργία δυο διαφορετικών virtual views για την παραγωγή δύο παραγόμενων πεδίων, θα πρέπει να διατρέξουμε όλες τις πλειάδες του ίδιου πίνακα δύο φορές.

Οι τρεις προτεινόμενες σχεδιαστικές λύσεις, που παρουσιάζονται σε αυτή την ενότητα, λαμβάνουν υπόψη τις τυχόν αλλαγές που μπορούν να εφαρμοστούν στη συνάρτηση υπολογισμού του παραγόμενου πεδίου. Η πρώτη σχεδιαστική λύση

περιλαμβάνει την κατασκευή ξεχωριστού πεδίου, στον πίνακα καταχώρησης, για κάθε παραγόμενο γνώρισμα , ενώ η δεύτερη και η τρίτη στηρίζεται στον υπολογισμό των τιμών του γνωρίσματος αυτού. Στην επόμενη παράγραφο περιγράφονται αναλυτικά οι τρείς σχεδιαστικές λύσεις.

4.4.3. Συμπεριφορά σε επίπεδο στιγμιότυπου

Στις προηγούμενες παραγράφους, εσκεμμένως, αποφεύχθηκε να αναφερθεί οτιδήποτε σχετικά με την αποθήκευση και τον τρόπο εφαρμογής της συνάρτησης υπολογισμού. Η συνάρτηση υπολογισμού θα πρέπει να είναι αποθηκευμένη κάπου στην βάση δεδομένων, ώστε να είναι δυνατή η εφαρμογή της κάθε φορά που πραγματοποιείται *εισαγωγή* ή *ανανέωση* μιας πλειάδας του πίνακα καταχωρήσεων. Για την αποθήκευση, αλλά και την πιστοποίηση της σωστής εφαρμογής μιας συνάρτησης υπολογισμού, κατασκευάζεται ένα *έναυσμα* (*trigger*). Ο trigger είναι αποθηκευμένος κώδικας σε `SQL` ή `PL/SQL` γλώσσα, ο οποίος μπορεί να «πυροδοτείται» κάθε φορά που πραγματοποιείται μια λειτουργία *εισαγωγής, διαγραφής* ή *ανανέωση*, ανάλογα με τον σχεδιασμό του. Η χρηστικότητα του trigger έγκειται στο γεγονός ότι πριν ή μετά μια λειτουργία, ελέγχεται η συνέπεια των εγγραφών που εισήχθησαν ή ανανεώθηκαν, σύμφωνα με τη συνάρτηση υπολογισμού.

Στην περίπτωση κατασκευής ενός πίνακα που περιλαμβάνει ως γνώρισμα το παραγόμενο πεδίο, κάθε φορά που πραγματοποιείται μια εισαγωγή ή ανανέωση μιας εγγραφής πυροδοτείται ο αντίστοιχος trigger, ο οποίος υπολογίζει την τιμή του παραγόμενου πεδίου με βάση τη συνάρτηση υπολογισμού. Με τη χρήση του trigger διασφαλίζεται η συνέπεια των τιμών του παραγόμενου πεδίου. Μια εναλλακτική μέθοδος υλοποίησης του trigger και της υλοποίησης εισαγωγής ή/και ανανέωσης, αποτελεί η δυνατότητα εισαγωγής/ανανέωσης μιας τιμής του πεδίου άμεσα από τον χρήστη. Ο trigger σε αυτή την περίπτωση, πιστοποιεί την ορθότητα της τιμής μετά το πέρας της λειτουργίας. Εάν το αποτέλεσμα υπολογισμού του trigger διαφέρει από την τιμή εισαγωγής του παραγόμενου πεδίου, ο trigger εμφανίζει *μήνυμα λάθους* (*exception*) ενημερώνοντας τον χρήστη για την λανθασμένη εισαγωγή.

Στην περίπτωση της υλοποιήσιμης όψης, το σύστημα δημιουργεί αυτόματα triggers, οι οποίοι εφαρμόζονται κατά την εισαγωγή, διαγραφή και ανανέωση μιας πλειάδας. Το μόνο που θα πρέπει να δηλωθεί κατά την κατασκευή μιας materialized view είναι ο τρόπος ανανέωσης των πλειάδων που περιλαμβάνει η όψη. Η `Oracle10g` [13] παρέχει την δυνατότητα της *σταδιακής ανακατασκευής* (*Incremental Rebuild* ή *Fast Refresh*) ή της *ολοκληρωτικής ανακατασκευής* (*Full Rebuild* ή *Complete Refresh*) μιας materialized view. Με τη σταδιακή ανακατασκευή, όλες οι αλλαγές (εισαγωγή, διαγραφή, ανανέωση μιας πλειάδας) που εφαρμόζονται στον *βασικό πίνακα* (*master relation - table*), από τον οποίον προκύπτει η όψη, αποθηκεύονται πρωτίστως στο *ημερολόγιο της όψης* (*materialized view log*) και στη συνέχεια εφαρμόζονται μόνο στις αντίστοιχες πλειάδες της όψης. Εν αντιθέσει, η ολοκληρωτική ανακατασκευή της όψης προκαλεί την επαναδημιουργία ολόκληρης της όψης κάθε φορά που πραγματοποιείται μια αλλαγή στις πλειάδες του βασικού πίνακα. Και στις δύο περιπτώσεις εξασφαλίζεται ότι οι πλειάδες της όψης είναι σε συνέπεια με τις αντίστοιχες πλειάδες του βασικού πίνακα, καθώς και ότι οι τιμές του παραγόμενου πεδίου προκύπτουν από την εφαρμογή της συνάρτησης υπολογισμού. Τέλος, το κάθε πότε μπορεί και πρέπει να ανακατασκευάζεται μια materialized view είναι στην ευχέρεια του σχεδιαστή. Μια όψη μπορεί να ανακατασκευάζεται είτε περιοδικά είτε αυτόματα, δυνατότητες που δίδονται κατά τον σχεδιασμό μιας materialized view από την `Oracle10g`.

Στην περίπτωση κατασκευής virtual view, δεν είναι δυνατή η εφαρμογή καμίας από τις λειτουργίες εισαγωγή/διαγραφή/ανανέωση στην όψη, εφόσον ο πίνακας, που αντιπροσωπεύει η όψη, είναι εικονικός και δεν περιέχει δεδομένα.

4.4.4. Συμπεριφορά σε επίπεδο σχήματος

4.4.4.1. Εισαγωγή, Διαγραφή, Ανανέωση Γνωρίσματος

Σε αυτή την παράγραφο θα μελετήσουμε την περίπτωση, όπου η αλλαγή του σχήματος της βάσης (εισαγωγή/διαγραφή/ανανέωση ενός γνωρίσματος) επηρεάζει τη συνάρτηση υπολογισμού ενός παραγόμενου πεδίου. Π.χ., ας θεωρήσουμε ότι

εισάγεται ένα νέο πεδίο, Bonus2,το οποίο λαμβάνει μέρος στον υπολογισμό του παραγόμενου γνωρίσματος Net. Σε αυτή την περίπτωση η συνάρτηση υπολογισμού θα πρέπει να αλλάξει ώστε να περιλαμβάνει και το νέο πεδίο.

Ο πιο απλός τρόπος αντιμετώπισης της παραπάνω περίπτωσης εισαγωγής, είναι η δημιουργία νέου πίνακα, EMP_FINANCIALS_2, ο οποίος περιλαμβάνει το νέο γνώρισμα. Επιπλέον δημιουργείται μια νέα συνάρτηση υπολογισμού, η οποία εφαρμόζεται στα πεδία του νέου πίνακα. Θεωρώντας ως βασικό τον νέο πίνακα με δομή, EMP_FINANCIALS_2(Emp_ID,Salary,Bonus,Bonus2,Date), μπορούμε να επιλέξουμε οποιαδήποτε από τις τρεις σχεδιαστικές μεθόδους, δηλαδή είτε να συμπεριλάβουμε σε αυτόν το παραγόμενο γνώρισμα είτε να κατασκευάσουμε μια όψη. Η προηγούμενη συνάρτηση υπολογισμού εξακολουθεί να ισχύει για τα πεδία του πίνακα EMP_FINANCIALS. Με αυτόν τον τρόπο καταχωρείται πληροφορία και στους δύο πίνακες. Το μειονέκτημα αυτής της λύσης είναι η περίπτωση αναζήτησης όλων των στοιχείων από όλους τους πίνακες, π.χ., όλα τα φορολογικά στοιχεία όλων των εργαζομένων, όπου θα πρέπει να εφαρμοστεί η λειτουργία της ένωσης (UNION) των σχετικών πινάκων. Δεδομένου ότι η ένωση μπορεί να εφαρμοστεί μόνο ανάμεσα σε πίνακες με ίσο αριθμό γνωρισμάτων και ίδιο πεδίο ορισμού στα αντίστοιχα πεδία, τότε θα πρέπει κατά την δήλωση της SQL ερώτησης να εισαχθούν τιμές NULL στα πλεονάζοντα πεδία. Π.χ., έστω ότι ζητείται πληροφορία σχετικά με τα οικονομικά στοιχεία όλων των εργαζομένων για κάθε χρόνο. Σε αυτή την περίπτωση θα πρέπει να πραγματοποιηθεί ένωση των πινάκων EMP_FINANCIALS και EMP_FINANCIALS_2, η οποία αντιστοιχίζεται στην παρακάτω SQL ερώτηση:

```
(select * from EMP_FINANCIALS_2)
                UNION
(select Emp_ID,Salary,Bonus,NULL,Date from EMP_FINANCIALS).
```

Ένας άλλος τρόπος αντιμετώπισης της περίπτωσης εισαγωγής/διαγραφής/ανανέωσης ενός γνωρίσματος, το οποίο λαμβάνει μέρος στη συνάρτηση υπολογισμού, περιλαμβάνει τη δημιουργία νέας συνάρτησης υπολογισμού.

Στην περίπτωση κατασκευής ενός πίνακα που συμπεριλαμβάνει το παραγόμενο γνώρισμα, δηλαδή της πρώτης σχεδιαστικής λύσης, η νέα συνάρτηση υπολογισμού εφαρμόζεται μόνο στις νέες εγγραφές αφήνοντας τις παλιές ως έχουν. Π.χ., στην

περίπτωση εισαγωγής του πεδίου Bonus2 στον πίνακα EMP_FINANCIALS, η συνάρτηση υπολογισμού του ενημερώνεται, λαμβάνοντας υπόψη και τις τιμές του νέου πεδίου, ως εξής: Net=(Salary+Bonus+Bonus2)/1.19. Θέτοντας ως τιμή Default για το πεδίο Bonus2 την τιμή μηδέν είναι προφανές ότι, η συνάρτηση επαληθεύεται για όλες τις πλειάδες του πίνακα. Με αυτόν τον τρόπο είναι προφανές ότι δεν απαιτείται η ανανέωση των παλαιών πλειάδων του πίνακα, εφόσον αυτές είναι συνεπείς με την νέα συνάρτηση υπολογισμού (θέτοντας Bonus2 = 0). Ανάλογα, εάν διαγραφεί ένα γνώρισμα, το οποίο λαμβάνει μέρος κατά τον υπολογισμό του παραγόμενου πεδίου, δημιουργείται μία νέα συνάρτηση. Η συνάρτηση αυτή εφαρμόζεται μόνο στις εγγραφές που πρόκειται να εισαχθούν από τη στιγμή της διαγραφής και ύστερα, αφήνοντας αμετάβλητες τις προηγούμενες. Η νέα συνάρτηση αποτελεί ανανέωση της ήδη υπάρχουσας συνάρτησης.

Παρόμοια, αντιμετωπίζεται η εφαρμογή των λειτουργιών εισαγωγή/διαγραφή/ ανανέωση σε μία materialized view. Αντί της τροποποίησης του αντίστοιχου trigger (μέθοδος που μπορεί επίσης να εφαρμοστεί και σε αυτή την περίπτωση), τροποποιείται ο κώδικας υλοποίησης της όψης. Η δημιουργία μιας νέας συνάρτησης επηρεάζει μόνο ορισμένες πλειάδες, σύμφωνα με κάποιο πεδίο διαχωρισμού, αφήνοντας αμετάβλητες τις υπόλοιπες πλειάδες. Π.χ., η εισαγωγή του πεδίου Bonus2 δημιουργεί μια νέα συνάρτηση υπολογισμού για τις πλειάδες με τιμή στο πεδίο διαχωρισμού Date = 2007 και μετά. Όλα τα δεδομένα που αντιστοιχούν σε εγγραφές με Date = 2006 και πριν δεν μεταβάλλονται. Σε αυτό το σημείο θα πρέπει να παρατηρηθεί ότι ο μόνος λόγος, ο οποίος επιβάλλει την ανακατασκευή της όψης, είναι όταν περιλαμβάνει το πεδίο το οποίο διαγράφεται. Π.χ., εάν η όψη είναι της μορφής: MV(Emp_ID, Net, Bonus, Date), τότε η διαγραφή του πεδίου Bonus από το σχήμα του βασικού πίνακα οδηγεί σε πλήρη ανακατασκευή της όψης.

Στην περίπτωση δημιουργίας εικονικής όψης, η εισαγωγή/διαγραφή/ανανέωση ενός γνωρίσματος του πίνακα, που υπολογίζεται στη συνάρτηση, οδηγεί στον επανα-προσδιορισμό της όψης. Από τη στιγμή που η όψη ορίζεται με βάση κάποια συγκεκριμένη συνάρτηση υπολογισμού και δεδομένου ότι δεν αποθηκεύεται κανένα στοιχείο σε φυσικό επίπεδο, η οποιαδήποτε αλλαγή στον κώδικα ορισμού της δημιουργεί μια τελείως διαφορετική όψη. Το πρόβλημα που παρατηρείται, με την

ανακατασκευή της virtual view, είναι η απώλεια της πληροφορίας που εξάγονταν με την βοήθεια της προηγούμενης όψης. Π.χ., για την προσθήκη του πεδίου `Bonus2` και την αλλαγή της συνάρτησης υπολογισμού, η προηγούμενη εικονική όψη που βασιζόταν στη μη ανανεωμένη συνάρτηση χάνεται. Η λύση σε αυτό το πρόβλημα είναι ο διαφορετικός ορισμός του ονόματος της κάθε όψης, ώστε να διατηρείται και η κάθε συνάρτηση υπολογισμού. Το μειονέκτημα, και σε αυτή την περίπτωση, είναι η αναγκαιότητα της λειτουργίας `UNION` όλων των όψεων για την εξαγωγή καθολικών αποτελεσμάτων. Π.χ., εάν ζητηθούν όλα τα φορολογικά στοιχεία ενός εργαζομένου για όλα τα έτη, τότε θα πρέπει να πραγματοποιηθεί ένωση όλων των αντίστοιχων όψεων.

4.4.4.2. Εισαγωγή, Διαγραφή, Ανανέωση Περιορισμού

Αλλαγή στη συνάρτηση υπολογισμού μπορεί, επίσης, να εφαρμοστεί και χωρίς να αλλάξει το σχήμα της βάσης δεδομένων. Π.χ., έστω ότι το φορολογικό σύστημα αλλάζει για το `2007`, διαφοροποιώντας τις κλίμακες μισθοδοσίας. Οι μισθωτοί της πρώτης κλίμακας παρουσιάζουν εισόδημα χαμηλότερου των `12.000 euro` και δεν φορολογούνται. Στην δεύτερη κλίμακα αντιστοιχούν οι μισθωτοί με εισοδήματα έως `30.000 euro` και φορολογούνται με βάση τη συνάρτηση `Σ1`, και τέλος όλοι οι υπόλοιποι με βάση τη συνάρτηση `Σ2`. Θα πρέπει για το συγκεκριμένο παράδειγμα να επιλέγονται οι κατάλληλες πλειάδες με `Date=2007` και `Salary` τις τρεις συγκεκριμένες κατηγορίες μισθών, `Salary<=12.000,Salary<=30.000` και να εφαρμόζεται η αντίστοιχη φορολογική φόρμουλα υπολογισμού του πεδίου `Net`.

Η αντιμετώπιση της εφαρμογής διαφορετικών συναρτήσεων με βάση κάποιο/α πεδίο/α της βάσης δεδομένων, και για τις τρεις σχεδιαστικές λύσεις δεν διαφέρει από την αντιμετώπιση αλλαγής του σχήματος. Στην περίπτωση κατασκευής ενός πίνακα που συμπεριλαμβάνει το παραγόμενο γνώρισμα, τροποποιείται ο κώδικας του trigger ώστε να περιλαμβάνει τις διαφορετικές συναρτήσεις ανάλογα με τα πεδία της συνθήκης επιλογής (`WHERE` *`Clause`*). Η materialized view τροποποιεί κατάλληλα τον κώδικα της ώστε να περιλαμβάνει όλους τους τρόπους παραγωγής των παραγόμενων πεδίων, ενώ τέλος η τροποποίηση της virtual view οδηγεί στη δημιουργία μιας νέας όψης.

4.4.5. Πλεονεκτήματα & Μειονεκτήματα

1η μέθοδος: Εισαγωγή επιπλέον πεδίου στον πίνακα καταχωρήσεων.

- Δημιουργία triggers, οι οποίοι υπολογίζουν και καταχωρούν, κάθε φορά, την παραγόμενη τιμή. Αν και η ύπαρξη του trigger εξασφαλίζει τη συνέπεια της βάσης δεδομένων, για κάθε εισαγωγή/διαγραφή/ανανέωση μιας πλειάδας του πίνακα, ο κώδικας κατασκευής του παρουσιάζεται πιο δύσκολος, πιο εκτενής και πιο πολύπλοκος σε σχέση με αυτόν της κατασκευής μιας όψης.

2η μέθοδος: Κατασκευή materialized view

- Η κατασκευή των triggers, οπότε και ο έλεγχος για την ορθότητα των εγγραφών καταχώρησης ενός πίνακα, πραγματοποιείται αυτόματα από το σύστημα.
- Ο κώδικας κατασκευής μιας materialized view είναι πιο απλός σε σχέση με αυτόν ενός trigger, διότι βασίζεται σε απλές `SQL` δηλώσεις.
- Μια materialized view παρουσιάζει μεγάλο αποθηκευτικό κόστος, δεδομένου ότι η πληροφορία που αποθηκεύεται σε αυτή υπάρχει ήδη αποθηκευμένη ως μέρος του βασικού πίνακα (π.χ., τουλάχιστον το πρωτεύον κλειδί `Emp_ID` του πίνακα `EMP_FINANCIALS` θα εμφανίζεται και στην όψη).
- Το κόστος διατήρησης μιας materialized view είναι αρκετά μεγάλο. Για να είναι συνεπής κάθε φορά η όψη ως προς τον αντίστοιχο βασικό πίνακα, δηλαδή για να είναι σε αντιστοιχία οι καταχωρήσεις της όψης με αυτές του πίνακα από τον οποίο προκύπτει, θα πρέπει να ανανεώνεται τακτικά. Η ανανέωση μιας materialized view αποτελεί μια αρκετά δύσκολη και χρονοβόρα διαδικασία, η οποία επιβαρύνει το σύστημα με φόρτο που ενδεχομένως δεν μπορεί να διαχειριστεί εύκολα.

3η μέθοδος: Κατασκευή virtual view

- Πλεονέκτημα της virtual view αποτελεί το μηδενικό αποθηκευτικό κόστος της, εφόσον δεν αποθηκεύεται κανένα αποτέλεσμα στο δίσκο.
- Λόγω της μη αποθήκευσης αποτελεσμάτων, η virtual view παρουσιάζει κόστος στο χρόνο εκτέλεσης, δεδομένου ότι η όψη «δημιουργείται» κάθε φορά που εκτελείται μια ερώτηση προς τον συγκεκριμένο πίνακα.

- Σημαντικό επίσης μειονέκτημα της μεθόδου αποτελεί το γεγονός ότι η virtual view δεν είναι «ανθεκτική» στις αλλαγές που μπορεί να εμφανιστούν στη συνάρτηση υπολογισμού. Η οποιαδήποτε αλλαγή της συνάρτησης με την οποία παράγονται οι τιμές του παραγόμενου πεδίου, οδηγεί στην δημιουργία μιας νέας όψης. Οπότε, για την εξαγωγή συνολικών αποτελεσμάτων (π.χ., συνολικά φορολογικά στοιχεία μιας εταιρίας ή ενός εργαζομένου για το EMP_FINANCIALS) θα πρέπει να εφαρμοστεί η λειτουργία της ένωσης στις όψεις που αντιστοιχούν σε όλες τις διαφορετικές συναρτήσεις υπολογισμού. Επιπλέον, η διαδικασία της ένωσης μπορεί να περιλαμβάνει αρκετά προβλήματα, π.χ., άνισος αριθμός πεδίων των σχέσεων που ενώνονται.

4.4.6. Κατασκευή

Σε αυτή την παράγραφο παρουσιάζονται ενδεικτικά ένα παράδειγμα κατασκευής trigger εισαγωγής στον πίνακα EMP_FINANCIALS της πρώτης σχεδιαστικής λύσης και ένα παράδειγμα κατασκευής μιας materialized view για την δεύτερη λύση. Η Oracle10g παρέχει πληροφορίες για την κατασκευή triggers [13, σελ.16-75, σελ.16-86], materialized views [13, σελ.15-4, σελ.15-24], virtual views [13, σελ.17-31, σελ.17-41] και stored procedures [13, σελ.15-49, σελ.15-51].

Κατασκευή Trigger Εισαγωγής:

```
CREATE TRIGGER ComputeNet BEFORE INSERT ON EMP_FINANCIALS
FOR EACH ROW
BEGIN
SET  :new.Net = (:new.Salary + new.Bonus)/1.19
INSERT INTO EMP_FINANCIALS VALUES
( :new.Emp_ID, :new.Salary, new.Bonus, :new.Net, :new.Date )
END
```

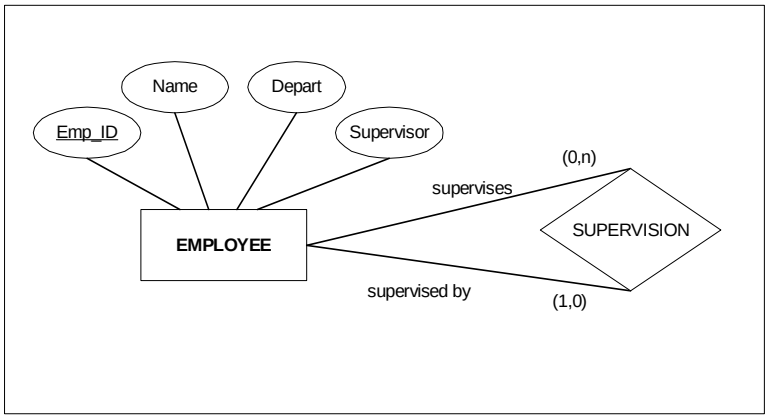
Κατασκευή Materialized View:

```
CREATE MATERIALIZED VIEW ComputeNet(ComputeNet _id, ComputeNet.Net)
REFRESH FAST AS
SELECT Emp_ID, (Salary+Bonus)/1.19
FROM EMP_FINANCIALS
```

4.5. Αναδρομική Κλειστότητα (Transitive or Recursive Closure) & Γράφοι

4.5.1. Κίνητρο & Εφαρμογή

Η *αναδρομική κλειστότητα* ορίζεται ως η πράξη που εφαρμόζεται σε μια *αναδρομική συσχέτιση*. Μια συσχέτιση του ER μοντέλου καλείται αναδρομική όταν ο ίδιος τύπος οντοτήτων συμμετέχει σε αυτή περισσότερες από μία φορές, έχοντας κάθε φορά διαφορετικό ρόλο. Π.χ., ας υποθέσουμε ότι καταγράφεται η πληροφορία των υπαλλήλων μιας εταιρίας. Στην κατηγορία των υπαλλήλων κατατάσσονται και οι προϊστάμενοι. Η αναπαράσταση στο ER μοντέλο της αναδρομικής συσχέτισης SUPERVISION παρουσιάζεται στο Σχήμα 4.33. Μέσω της SUPERVISION κάθε πλειάδα της EMPLOYEE (σε ρόλο υφισταμένου) συσχετίζεται με μία άλλη πλειάδα της EMPLOYEE (σε ρόλο προϊσταμένου).



Σχήμα 4.33 Αναπαράσταση του παραδείγματος στο ER μοντέλο.

Η αντίστοιχη σχέση της οντότητας EMPLOYEE έχει την εξής δομή:

EMPLOYEE (Emp_ID, Name, Depart, Supervisor).

Το γνώρισμα Supervisor της EMPLOYEE αποτελεί αναφορά ξένου κλειδιού στο γνώρισμα του πρωτεύοντος κλειδιού Emp_ID, αναπαριστώντας με αυτόν τον τρόπο τη αναδρομική συσχέτιση SUPERVISION του ER μοντέλου.

Παράδειγμα εφαρμογής μιας αναδρομικής πράξης αποτελεί η εύρεση όλων των υφισταμένων ενός προϊσταμένου σε όλα τα επίπεδα. Δηλαδή για έναν προϊστάμενο ζητούνται όλοι οι, άμεσα, υφιστάμενοι του, οι υφιστάμενοι των υφισταμένων του

κ.ο.κ.. Με αυτόν τον τρόπο δημιουργείται ένα *δέντρο συσχέτισης προγόνου-απογόνου*, όπου ο κόμβος πρόγονος αντιστοιχεί στον τύπο σχέσης που συμμετέχει στη συσχέτιση με πληθικότητα (0,N) (ρόλος προϊσταμένου για το παράδειγμά μας), ενώ ο κόμβος απόγονος αντιστοιχεί στον τύπο σχέσης που συμμετέχει με πληθικότητα (0,1) (ρόλος εργαζομένου). Οι ακμές του δέντρου αναπαριστούν την αναδρομική συσχέτιση.

Η προφανής λύση του προβλήματος εύρεσης της αναδρομικής κλειστότητας είναι η εφαρμογή της λειτουργίας της *συνένωσης* της σχέσης EMPLOYEE, η οποία εμφανίζεται και στα δύο μέλη της συνένωσης. Στη μια συμμετέχει με τον ρόλο του εργαζόμενου ενώ στην άλλη με το ρόλο του προϊσταμένου. Τέτοιου τύπου συνενώσεις καλούνται *self joins*. Για την πάραξη του τελικού αποτελέσματος θα πρέπει να εφαρμοστεί επιπλέον η λειτουργία της ένωσης (Union) όλων των αποτελεσμάτων. Π.χ., ας υποθέσουμε ότι ζητούνται οι συσχετίσεις προϊστάμενος – υφιστάμενος μόνο για τα δύο πρώτα επίπεδα, δηλαδή ζητάμε όλους τους άμεσα υφισταμένους ενός προϊσταμένου και για κάθε έναν από αυτούς όλους τους άμεσα υφισταμένους του. Οι αντίστοιχες πράξεις της σχεσιακής άλγεβρας για την παραπάνω ερώτηση είναι οι εξής:

$$\text{RESULT1(Emp\_ID)} = \pi_{\text{Emp\_ID}} \left(\text{EMPLOYEE} \underset{\text{Emp\_ID = Supervisor}}{\bowtie} \text{EMPLOYEE} \right)$$

$$\text{RESULT2(Emp\_ID)} = \pi_{\text{Emp\_ID}} \left(\text{RESULT1} \underset{\text{Emp\_ID = Supervisor}}{\bowtie} \text{EMPLOYEE} \right)$$

$$\text{RESULT} = \text{RESULT1} \cup \text{RESULT2}$$

Είναι προφανές ότι η μέθοδος των self joins δεν αποτελεί καλή λύση στην περίπτωση που δεν γνωρίζουμε επακριβώς το βάθος του δέντρου, δηλαδή ποιος είναι ο συνολικός αριθμός των επιπέδων της αναδρομής. Οπότε, ζητούμενο αποτελεί η αναπαράσταση όλων των δέντρων απεικόνισης της αναδρομικής συσχέτισης, εφαρμόζοντας την αναδρομική κλειστότητα, για όλα τα επίπεδα (ανεξάρτητα από το πλήθος τους). Στην επόμενη παράγραφο παρουσιάζεται μια αποδοτική μέθοδος, για τον υπολογισμό της αναδρομικής κλειστότητας.

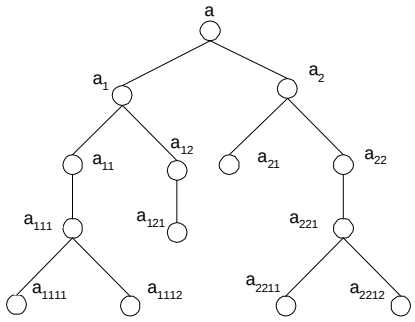
4.5.2. Δομή & Κατασκευή

Ας θεωρήσουμε τον γράφο του Σχήματος 4.34 όπου απεικονίζεται η αναδρομική συσχέτιση με βήμα 1, δηλαδή η άμεση συσχέτιση των δύο ρόλων. Συγκεκριμένα για το παράδειγμα EMPLOYEE, το δέντρο απεικονίζει τους άμεσα υφισταμένους κάθε εργαζομένου. Η πληροφορία που απεικονίζεται στο δέντρο καταχωρείται σε έναν πίνακα με δομή

PARENT (Child, Parent),

όπου τα πεδία Child και Parent αντιπροσωπεύουν τους ρόλους υφιστάμενος και προϊστάμενος, αντίστοιχα.

Στην περίπτωση που μια σχέση συμμετέχει στην αναδρομική συσχέτιση με πληθικότητα (0,N), αντί (0,1), επιλέγουμε ως πρωτεύον κλειδί τον συνδυασμό των πεδίων Child και Parent, διότι υπάρχει πιθανότητα εμφάνισης κύκλου, π.χ., ένας εργαζόμενος μπορεί να έχει περισσότερους του ενός προϊσταμένου. Με την ύπαρξη κύκλου ο ορισμός δέντρο συσχέτισης προγόνου-απογόνου, αντικαθίσταται από τον γενικότερο ορισμό *γράφος συσχέτισης*, ο οποίος μελετάται στην παράγραφο 4.5.6.



Σχήμα 4.34 Δέντρο αναπαράστασης της αναδρομής με βήμα 1.

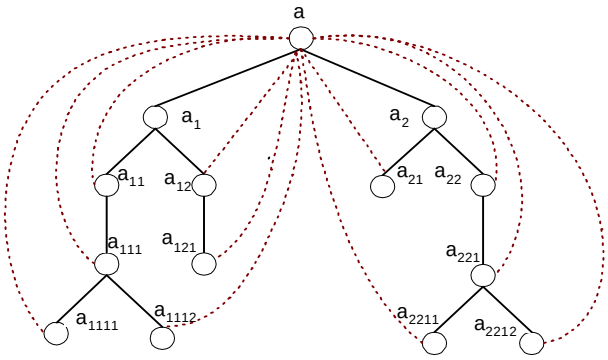
Επιπλέον, κατασκευάζεται ένας πίνακας, στον οποίον καταγράφεται όλη η πληροφορία που παράγεται με την εφαρμογή της αναδρομικής κλειστότητας. Στον πίνακα με δομή:

ANCESTORS (Node, Descendant, Length),

περιγράφονται όλα τα δυνατά μονοπάτια του δέντρου, με βήμα από ένα (Length = 1) ως το ύψος του δέντρου (Length = μεγαλύτερο μονοπάτι από την πηγή ως ένα φύλλο). Τα πεδία Node και Descendant είναι σε άμεση αντιστοιχία με τα πεδία Parent και Child του πίνακα PARENT.

Στο Σχήμα 4.35 παρουσιάζεται το δέντρο που αντιστοιχεί στον πίνακα ANCESTORS, όπου οι διακεκομμένες γραμμές αναπαριστούν τις συσχετίσεις που προκύπτουν μετά την εφαρμογή της αναδρομικής κλειστότητας. Στο Σχήμα 4.36 παρουσιάζονται στιγμιότυπα των πινάκων PARENT και ANCESTORS για τη σχέση EMPLOYEE.

Ο αλγόριθμος με την βοήθεια του οποίου κατασκευάζεται ο πίνακας ANCESTORS δίνεται στη συνέχεια με την μορφή PL/SQL δηλώσεων.



Σχήμα 4.35 Γράφος αναπαράστασης της αναδρομικής κλειστότητας.

Μέθοδος κατασκευής του πίνακα απεικόνισης της αναδρομικής κλειστότητας:

```
create procedure transitive _closure () as
      counter number;
      BEGIN
            insert into ANCESTORS (ANC,DES,LENGTH)
                  (     SELECT P , C , 1
                        FROM  PARENTS     );
            LOOP
                  counter :=0;
                  INSERT INTO ANCESTORS2 (Anc2,Des2,Length2)
                  (     SELECT P.P , A.DES, A.LENGTH + 1
                        FROM ANCESTORS A, PARENTS P
                        WHERE A.ANC = P.c
                              AND A.LENGTH IN
                              (     SELECT MAX (LENGTH)
                                    FROM ANCESTORS )  );
                  INSERT INTO ANCESTORS
                  (     SELECT *
                        FROM ANCESTORS2   );

                  counter:= SQL%ROWCOUNT;
                  if counter=0 then exit;end if;
                  delete from ancestors2;
            END LOOP;
      end;
```

PARENT

Child	Parent
a_1	a
a_2	a
a_{11}	a_1
a_{12}	a_1
a_{21}	a_2
a_{22}	a_2
a_{111}	a_{11}
a_{121}	a_{12}
a_{221}	a_{22}
a_{1111}	a_{111}
a_{1112}	a_{111}
a_{2211}	a_{221}
a_{2212}	a_{221}

ANCESTORS

Node	Descend	Length
a	a_1	1
a	a_2	1
a_1	a_{11}	1
a_1	a_{12}	1
a_2	a_{21}	1
a_2	a_{22}	1
a_{11}	a_{111}	1
a_{12}	a_{121}	1
a_{22}	a_{221}	1
a_{111}	a_{1111}	1
a_{111}	a_{1112}	1
a_{221}	a_{2211}	1
a_{221}	a_{2212}	1
a	a_{11}	2
a	a_{12}	2
a	a_{21}	2
a	a_{22}	2
a_1	a_{111}	2
a_1	a_{121}	2
a_2	a_{221}	2
a_{11}	a_{1111}	2
a_{11}	a_{1112}	2
a_{22}	a_{2211}	2
a_{22}	a_{2212}	2
a	a_{111}	3
a	a_{121}	3
a	a_{221}	3
a_1	a_{1111}	3
a_1	a_{1112}	3
a_2	a_{2211}	3
a_2	a_{2212}	3
a	a_{1111}	4
a	a_{1112}	4
a	a_{2211}	4
a	a_{2212}	4

Σχήμα 4.36 Αναπαράσταση στιγμιότυπων των σχέσεων PARENT και ANCESTORS για το δέντρο του παραδείγματος.

4.5.3. Συμπεριφορά σε επίπεδο στιγμιότυπου

Όλες οι παρακάτω λειτουργίες υποθέτουμε ότι εφαρμόζονται στο γράφο του Σχήματος 4.34, για καλύτερη κατανόηση του αποτελέσματος τους. Φυσικά, αυτή η υπόθεση δεν βλάπτει την γενικότητα των αλγορίθμων που δίδονται παρακάτω.

4.5.3.1. Επιλογή

Η μελέτη της λειτουργίας της επιλογής εστιάζει σε τρεις περιπτώσεις:

1. Επιλογή της πηγής του δέντρου: η εύρεση του κόμβου της πηγής πραγματοποιείται εύκολα με αναζήτηση είτε στον πίνακα PARENT είτε στον πίνακα ANCESTORS. Π.χ., πηγή του δέντρου απεικόνισης της σχέσης EMPLOYEE αποτελεί ο διευθυντής (προϊστάμενος όλων) της εταιρίας του παραδείγματος. Στην περίπτωση όπου στο πίνακα PARENT έχει εισαχθεί εγγραφή της πηγής (π.χ., (a,NULL)) τότε πραγματοποιείται εύρεση της αντίστοιχης εγγραφής. Σε αντίθετη περίπτωση τίθεται η SQL ερώτηση:

   ```
   SELECT Parent FORM PARENT AS P

   WHERE P.PARENT NOT IN (SELECT E.Child FROM PARENT AS E),
   ```

   με την βοήθεια της οποίας αναζητούνται οι εγγραφές του πίνακα PARENT, όπου οι τιμές του πεδίου Parent δεν εμφανίζονται στο πεδίο Child.
2. Επιλογή των φύλλων του δέντρου: η εύρεση των φύλλων, που αντιστοιχεί στην εύρεση όλων των εργαζομένων που δεν είναι προϊστάμενοι κανενός για το παράδειγμα EMPLOYEE, πραγματοποιείται εύκολα με την αναζήτηση όλων των πλειάδων του πίνακα PARENT, όπου οι τιμές του πεδίου Child δεν εμφανίζονται στο πεδίο Parent. Η SQL ερώτηση επιλογής των φύλλων είναι η εξής:

   ```
   SELECT Child FORM PARENT AS P

   WHERE P.Child NOT IN (SELECT E.Parent FROM PARENT AS E),
   ```

   με την βοήθεια της οποίας αναζητούνται οι εγγραφές του πίνακα PARENT, όπου οι τιμές του πεδίου Parent δεν εμφανίζονται στο πεδίο Child.
3. Επιλογή όλων των δυνατών μονοπατιών: ουσιαστικά η επιλογή όλων των δυνατών μονοπατιών του δέντρου αντιστοιχεί στην εφαρμογή της

αναδρομικής κλειστότητας. Η πληροφορία αυτή παρέχεται από τον πίνακα ANCESTORS.

4.5.3.2. Εισαγωγή εγγραφής – κόμβου

Σε αυτό το σημείο θα μελετήσουμε τον τρόπο εισαγωγής ενός κόμβου στο δέντρου απεικόνισης του σχήματος 4.34. Ο κόμβος αυτός μπορεί να εισαχθεί ως:

1. Πηγή (r). Τα βήματα του αλγόριθμου εισαγωγής είναι τα εξής:
   i. Εύρεση της πηγής του δέντρου.
   ii. Εισαγωγή της πλειάδας PARENT(a,r).
   iii. Εισαγωγή της πλειάδας ANCESTORS(a,r,1).
   iv. Θέτουμε Length = 1.
   v. Εύρεση των παιδιών του a, δηλαδή της πλειάδας ANCESTORS(r,Descendant,Length).
   vi. Για κάθε Descendant εισάγεται η πλειάδα ANCESTORS(r,Descendant,Length+1).
   vii. Θέτουμε Length = Length +1.
   viii. Μετάβαση στο βήμα v., εκτός και αν δεν υπάρχει Descendant οπότε και τερματίζεται η διαδικασία.

Σημείωση: Στην περίπτωση που ο γράφος αποτελείται από δυο ασύνδετους υπογράφους, με a και b να είναι οι πηγές του κάθε υποδέντρου, τότε θα πρέπει να εισαχθεί στο βήμα *i.* και η πλειάδα PARENT(b,r). Τέλος θα πρέπει να εφαρμοστούν όλα υπόλοιπα βήματα και για τις πλειάδες που αντιστοιχούν στον υπογράφο με πηγή το b, θέτοντας όπου a το b.

2. Φύλλο (a_{211}). Εφόσον γνωρίζουμε τον κόμβο-πατέρα, στον οποίον εισάγεται ο κόμβος, τότε τα βήματα εισαγωγής του κόμβου είναι τα εξής:
   i. Εισαγωγή της πλειάδας PARENT(a_{21},a_{211}).
   ii. Εισαγωγή της πλειάδας ANCESTORS(a_{21},a_{211},1).
   iii. Θέτουμε Length = 1.

iv. Εύρεση του πατέρα του κόμβου a_{21} στον πίνακα ANCESTORS για το δεδομένο Length. Δηλαδή εύρεση της ANCESTORS(Node,a_{21},Length).

v. Εισαγωγή της πλειάδας ANCESTORS(Node,a_{211},Length+1).

vi. Θέτουμε Length = Length+1.

vii. Έλεγχος εάν το Node αποτελεί ρίζα, οπότε και τερματίζεται ο αλγόριθμος. Αλλιώς μετάβαση στο βήμα iv.

3. Κόμβος (e). Υποθέτουμε ότι εισέρχεται ένας κόμβος (εσωτερικός) ανάμεσα στους κόμβους a_{22} και a_{221}. Δηλαδή το κόμβος εισέρχεται ως απόγονος του a_{22} και ταυτόχρονα ως πατέρας του a_{221}. Τα βήματα που αλγορίθμου εισαγωγής είναι τα παρακάτω:

   i. Εισαγωγή των πλειάδων PARENT(a_{22},e), PARENT(e,a_{221}) και διαγραφή της πλειάδας PARENT(a_{22},a_{221}).

   ii. Εισαγωγή των ANCESTORS(a_{22},e,1), ANCESTORS(e,a_{221},1) και διαγραφή της πλειάδας ANCESTORS(a_{22},a_{221},1).

   iii. Εύρεση των προγόνων (Node) του απογόνου (a_{221}) του e:

      I. Θέτουμε Length = 2.

      II. Εύρεση της πλειάδας ANCESTORS(Node,a_{221},Length), που αντιστοιχεί στον πρόγονο του a_{221} για το δεδομένο Length.

      III. Εισαγωγή των πλειάδων ANCESTORS(Node,e,Length) και ANCESTORS(Node,a_{221},Length+1).

      IV. Διαγραφή της ANCESTORS(Node,a_{221},Length).

      V. Θέτουμε Length = Length +1.

      VI. Μετάβαση στο II, εκτός και αν ο Node αποτελεί ρίζα οπότε και τερματίζεται το βήμα iii.

   iv. Εύρεση όλων των απογόνων (Descendant) του απογόνου (a_{221}) του e:

      I. Θέτουμε Length =2.

      II. Εύρεση όλων των απογόνων του a_{221}, δηλαδή όλων των πλειάδων ANCESTORS(a_{221},Descendant,Length).

      III. Για κάθε μια από τις παραπάνω πλειάδες:

a. Εισάγονται οι πλειάδες
   ANCESTORS(e,Descendant,Length) και
   ANCESTORS(a_{221},Descendant,Length +1).
b. Διαγράφονται οι ANCESTORS(a_{221},Descendant,Length).

IV. Θέτουμε Length = Length +1.

V. Μετάβαση στο II, εκτός και αν δεν υπάρχει πλέον Descendant οπότε και τερματίζεται το βήμα iv καθώς και όλος ο αλγόριθμος.

4.5.3.3. Διαγραφή εγγραφής – κόμβου

Παρόμοια με την εισαγωγή, και στην περίπτωση διαγραφής ενός κόμβου θα μελετήσουμε τις περιπτώσεις όπου ο κόμβος που διαγράφεται είναι:

1. Πηγή (a): Τα βήματα του αλγορίθμου διαγραφής της πηγής του δέντρου αναπαράστασης της αναδρομικής συσχέτισης είναι τα εξής:
   i. Διαγραφή των εγγραφών PARENT(a_1,a) και PARENT(a_2,a) οι οποίες αποτελούν τις συνδέσεις της πηγής με τα παιδιά της.
   ii. Θέτουμε Length = 1.
   iii. Διαγραφή των πλειάδων ANCESTORS(a,Descendant,Length), στις οποίες εμφανίζεται ο a στο πεδίο Node.
   iv. Θέτουμε Length = Length +1.
   v. Μετάβαση στο βήμα iii, εκτός και αν το Length>max(Length) οπότε και τερματίζει η διαδικασία.

2. Φύλλο (a_{21}):
   i. Διαγραφή της πλειάδας στην οποία εμφανίζεται ο a_{21} ως τιμή στο πεδίο Child, δηλαδή της PARENT(a_{21},Parent).
   ii. Θέτουμε Length = 1.
   iii. Διαγραφή των πλειάδων ANCESTORS(Node,a_{21},Length), στις οποίες εμφανίζεται ο a_{21} στο πεδίο Descendant.
   iv. Θέτουμε Length = Length +1.
   v. Μετάβαση στο βήμα iii, εκτός και αν το Node αποτελεί ρίζα του δέντρου οπότε και τερματίζει η διαδικασία.

3. Κόμβος (a_{11}):
    i. Εύρεση του πατέρα, `Parent`, του a_{11} στον πίνακα `PARENT`, δηλαδή της πλειάδας `PARENT(`a_{11}`,Parent)`.
    ii. Εύρεση των παιδιών, `Child`, του a_{11}, δηλαδή των πλειάδων `PARENT(Child,` a_{11}`)`.
    iii. Για κάθε απόγονο του a_{11}, εισάγεται η πλειάδα `PARENT(Child,Parent)` και διαγράφονται οι πλειάδες των βημάτων i. και ii.
    iv. Εύρεση του πατέρα και των παιδιών του a_{11}, δηλαδή των πλειάδων `ANCESTORS(Node,`a_{11}`,1)` και `ANCESTORS(`a_{11}`,Descendant,1)`.
    v. Για κάθε απόγονο, που προέκυψε από το βήμα iv., εισάγονται οι πλειάδες `ANCESTORS(Node,Descendant,1)`και διαγράφονται οι πλειάδες του βήματος iv.
    vi. Θέτουμε `Length =2`.
    vii. Εύρεση του προγόνου του a_{11}, `ANCESTORS(Node,`a_{11}`,Length)`.
    viii. Εύρεση των παιδιών του a_{11}, `ANCESTORS(`a_{11}`,Descendant,Length)`
    ix. Εύρεση των πλειάδων που συσχετίζουν τον πατέρα και οι απόγονοι του a_{11}, `ANCESTORS(Node,Descendant,Length+1)`. Ενημέρωση των πλειάδων σε `ANCESTORS(Node,Descendant,Length)`.
    x. Διαγραφή των πλειάδων `ANCESTORS(`a_{11}`,Descendant,Length)` και `ANCESTORS(Node,`a_{11}`,Length)`.
    xi. Θέτουμε `Length = Length +1`.
    xii. Έλεγχος εάν ο `Node` αποτελεί ρίζα και αν `Length>max(Length)`. Εάν η συνθήκη ελέγχου είναι αληθής τερματίζεται η διαδικασία, αλλιώς μεταβαίνουμε στο βήμα vii.

4.5.3.4. Ανανέωση εγγραφής – κόμβου

Η ανανέωση ενός κόμβου του δέντρου αναδρομής αντιμετωπίζεται ως διαγραφή και επαναεισαγωγή του κόμβου σε νέα θέση.

4.5.4. Συμπεριφορά σε επίπεδο σχήματος

Η εισαγωγή οποιουδήποτε γνωρίσματος σε κάποιον από τους πίνακες PARENT ή ANCESTORS δεν επηρεάζει τη συμπεριφορά του προτύπου σε επίπεδο σχήματος. Η διαγραφή ή ανανέωση ενός γνωρίσματος, το οποίο συμμετέχει στην κατασκευή του γράφου συσχέτισης προγόνου-απογόνου, δεν είναι επιτρεπτή.

4.5.5. Πλεονεκτήματα & Μειονεκτήματα

Με την εφαρμογή του αλγορίθμου της αναδρομικής κλειστότητας και την δημιουργία του αντίστοιχου δέντρου παρατηρούμε ότι:

- η αναζήτηση οποιαδήποτε πληροφορίας δίδεται μέσω του πίνακα ANCESTORS**.** Στον ANCESTORS περιέχονται τα δυνατά αποτελέσματα της αναδρομικής κλειστότητας, οπότε μπορούμε εύκολα να απαντήσουμε σε ερωτήσεις για οποιοδήποτε επίπεδο της αναδρομής.
- αποφεύγεται η εφαρμογή self joins. Η εφαρμογή self joins αποτελεί χρονοβόρα διαδικασία, η οποία θα πρέπει να εφαρμόζεται κάθε φορά που τίθεται μια ερώτηση αναδρομής. Επιπλέον, θα πρέπει απαραιτήτως να γνωρίζουμε το πλήθος των επιπέδων αναδρομής, ώστε να γνωρίζουμε και το πλήθος των self joins που θα πρέπει να πραγματοποιηθούν.
- ο υπολογισμός συναθροιστικών συναρτήσεων είναι εύκολος. Εφόσον είναι γνωστά όλα τα μονοπάτια του δέντρου, μπορούμε να εφαρμόσουμε μια συναθροιστική συνάρτηση (SUM, AVG, κ.α.) στους κόμβους του μονοπατιού που μας ενδιαφέρει κάθε φορά. Ας υποθέσουμε ότι ζητείται ο υπολογισμός του συνολικού βάρους του υποδέντρου κάθε κόμβου του σχήματος 4.35. Δηλαδή, θέλουμε να υπολογίσουμε το συνολικό βάρος όλου του δέντρου, καθώς και τα επιμέρους βάρη κάθε υποδέντρου. Ο υπολογισμός προϋποθέτει την κατασκευή ενός πίνακα καταχωρήσεων με δομή:

  Nodes (Node, Weight, Sum_Weight),

  όπου αποθηκεύεται κάθε κόμβος (Node) με το βάρος (Weight) του ενώ στο πεδίο Sum_Weight, καταχωρείται το συνολικό βάρος του υποδέντρου με ρίζα το κόμβο αυτό. Υπάρχουν δύο τρόποι εφαρμογής της συναθροιστικής συνάρτησης SUM στο δέντρο:

1. Εφόσον έχουμε κατασκευάσει τον πίνακα ANCESTORS, μπορούμε για κάθε κόμβο να βρούμε όλους τους απογόνους του για όλα τα μήκη μονοπατιού και να αθροίσουμε τα βάρη τους. Τα βήματα του αλγορίθμου είναι τα εξής:
   i. Για κάθε κόμβο n, με `length=1`
   ii. Για κάθε απόγονο des, θέσε
   `n.Sum_Weigth = n.Weight + des.Weight.`
2. Για να αποφύγουμε την πολλαπλή ανάγνωση του πίνακα ANCESTORS μπορούμε να υπολογίζουμε το Sum_Weigth κάθε κόμβου κατά την κατασκευή του ANCESTORS. Αρχικά θα πρέπει να θέσουμε `Sum_Weight = Weight` για κάθε κόμβο. Με την εισαγωγή κάθε πλειάδας του τύπου `(n, des, length)` στον πίνακα ANCESTORS, υπολογίζεται η συνάρτηση:
   `n.Sum_Weigth = n.Sum_Weight + des.Weight.`

- απαιτείται επιπλέον αποθηκευτικός χώρος. Παρατηρώντας το Σχήμα 4.36 συμπεραίνουμε ότι, ο χώρος που απαιτείται για την αποθήκευση των αποτελεσμάτων της εφαρμογής της αναδρομικής κλειστότητας είναι αρκετά μεγάλος. Είναι προφανές ότι όσο μεγαλύτερος είναι ο πίνακας PARENT τόσο περισσότερος είναι και ο χώρος που απαιτείται για τον ANCESTORS.
- απαιτείται ενημέρωση του ANCESTORS κατά την εισαγωγή/διαγραφή ενός κόμβου. Η εισαγωγή ή διαγραφή ενός κόμβου από το δέντρο οδηγεί στην ενημέρωση ενός μεγάλου μέρους του δέντρου, και κατά συνέπεια σε πολλές αλλαγές στον πίνακα ANCESTORS. Για την ενημέρωση του ANCESTORS απαιτείται αρκετός χρόνος διότι θα πρέπει, πρωτίστως να αναζητηθούν οι κόμβοι οι οποίοι επηρεάζονται, και εν συνεχεία να εφαρμοστούν οι αντίστοιχες αλλαγές σε αυτούς. Μία εναλλακτική λύση είναι ο πλήρης επαναϋπολογισμός του πίνακα ANCESTORS, αποφεύγοντας τις πολλαπλές και χρονοβόρες ενημερώσεις. Το μέγεθος του δέντρου και το πλήθος των κόμβων που επηρεάζονται, αποτελούν σημαντικούς παράγοντες με βάση τους οποίους θα πρέπει να πραγματοποιείται η επιλογή της καλύτερης εκ των δύο λύσεων.

4.5.6. Μοντελοποίηση Γράφων

Σε όλη την παραπάνω ενότητα αναφερθήκαμε στα δέντρα συσχέτισης, τα οποία αποτελούν ειδική περίπτωση των γράφων συσχέτισης. Για να θεωρηθεί ένας κατευθυνόμενος γράφος ως δέντρο θα πρέπει να ισχύουν και οι δύο παρακάτω ιδιότητες:

i. Κάθε κόμβος έχει το πολύ έναν πατέρα. Δηλαδή σε κάθε κόμβο του κατευθυνόμενου γράφου θα πρέπει να προσπίπτει μια ακμή, εκτός από την πηγή που δεν έχει πατέρα.
ii. Δεν υπάρχει κύκλος.

Η αναπαράσταση ενός γράφου σε μια βάση δεδομένων πραγματοποιείται με την βοήθεια δύο πινάκων. Στον πρώτο πίνακα καταχωρούνται οι κόμβοι του γράφου, και η δομή είναι η εξής:

NODES (Node_ID, Description),

όπου το πεδίο Node_ID αποτελεί πρωτεύον κλειδί της σχέσης NODES.

Στον δεύτερο πίνακα καταχωρούνται οι ακμές, όπως αυτές εμφανίζονται στο γράφο, και η δομή του έχει τη μορφή:

EDGES (From_N_ID, To_N_ID),

με πρωτεύον κλειδί το συνδυασμό των πεδίων From_N_ID, To_N_ID.

Ο πίνακας EDGES αποτελείται από δύο πεδία, καθένα από τα οποία αποτελεί αναφορά ξένου κλειδιού στο πεδίο Node_ID του πίνακα NODES. Τα γνωρίσματα From_N_ID και To_N_ID της σχέσης EDGES βρίσκονται σε αντιστοιχία με τα πεδία Ancestor και Descendant της σχέσης ANCESTORS.

Η εύρεση όλων των κόμβων-πηγών, δηλαδή όλων των κόμβων στους οποίους δεν προσπίπτουν ακμές, πραγματοποιείται με την παρακάτω SQL ερώτηση:

```
SELECT E.From_N_ID FROM EDGES AS E
WHERE E.From_N_ID NOT IN (SELECT To_N_ID FROM EDGES).
```

Η εύρεση όλων των κόμβων-φύλλων, δηλαδή των κόμβων στους οποίους προσπίπτουν μόνο ακμές δίνεται μέσω της παρακάτω SQL ερώτησης:

```
SELECT E.To_N_ID FROM EDGES AS E
WHERE E.To_N_ID NOT IN (SELECT From_N_ID FROM EDGES).
```

Η αναζήτηση της ύπαρξης ενός μονοπατιού ανάμεσα σε δύο δεδομένους κόμβους μπορεί να πραγματοποιηθεί:

i. μέσω των πινάκων της αναδρομικής κλειστότητας, εφόσον έχει ελεγχθεί ότι ο γράφος δεν περιλαμβάνει κύκλο. Η `SQL-1999` δίδει έναν τρόπο ελέγχου για την ύπαρξη κύκλου σε έναν γράφο, και παρουσιάζεται στην επόμενη ενότητα.
ii. με την κατασκευή μιας stored procedure, η οποία να δέχεται ως παραμέτρους δύο κόμβους και να παράγει το μήκος του μονοπατιού ή τους ενδιάμεσους κόμβους.
iii. Με την βοήθεια του αντικειμένου `cursor` [15, σελ. 447-467]. Ο cursor είναι ένας *δείκτης* (*pointer*) της SQL, ο οποίος «κινείται» μέσα στο σύνολο των εγγραφών που απαρτίζουν το αποτέλεσμα μιας ερώτησης. Κάθε φορά που δημιουργείται ένας cursor, εφαρμόζεται κάποια ερώτηση επιλογής εγγραφών. Οι εγγραφές εξάγονται στο αποτέλεσμα μια-μια κάθε φορά, με την βοήθεια μιας *επαναληπτικής διαδικασίας* (*loop*). Κάθε μια από τις εγγραφές που παράγονται, μπορούμε να τη τροποποιήσουμε (Update/Delete), ή να εκτελέσουμε για κάθε μια από αυτές κάποια άλλη, εσωτερική, επαναληπτική διαδικασία αναζήτησης εγγραφών. Η δεύτερη περίπτωση αποτελεί ένα είδος αναδρομής. Κάθε φορά που τερματίζει η εσωτερική επανάληψη, ο cursor «δείχνει» στην τελευταία εγγραφή για την οποία πραγματοποιήθηκε η επανάληψη, και προχωράει στην επόμενη πλειάδα του αποτελέσματος της αρχικής ερώτησης.

Εφαρμόζοντας οποιαδήποτε από τις παραπάνω λύσεις εύρεσης ενός μονοπατιού θα πρέπει να είμαστε πολύ προσεκτικοί λόγω της πιθανής ύπαρξης κύκλου στο γράφο. Η επαναληπτική διαδικασία της κάθε τεχνικής θα πρέπει να ελέγχει το κατά πόσο η εκάστοτε πλειάδα που παράγεται υπάρχει ήδη, οπότε και να τερματίζει η διαδικασία.

4.5.7. Κατασκευή

Στην SQL-1999 είναι δυνατός ο ορισμός της αναδρομικής κλειστότητας με χρήση της σύνταξης WITH RECURSIVE [15, σελ.344-353]. Η μέθοδος εφαρμογής της αναδρομικής κλειστότητας είναι η ίδια με αυτή που αναπτύχθηκε παραπάνω, δηλαδή για την παραγωγή όλων των μονοπατιών εφαρμόζεται συνένωση πινάκων και ένωση των αποτελεσμάτων. Η απεικόνιση της αναδρομικής κλειστότητας του σχήματος 4.34, σε SQL-1999, πραγματοποιείται ως εξής:

```
WITH RECURSIVE ANCESTORS (Node, Descendant, Length) AS
 (     SELECT Parent, Child, 1
       FROM PARENT
   UNION ALL
       SELECT P.Parent, A.Descnendant, A.Length+1
       FROM PARENT P, ANCESTORS A
       WHERE P.Child = A.Node)
  CYCLE Node, Descendant
  SET cyclemark to ‘Y’ DEFAULT ‘N’
  USING cyclepath
```

Στο παραπάνω παράδειγμα παρουσιάζεται και ο τρόπος ελέγχου ύπαρξης κύκλου στον γράφο της αναδρομής. Ο έλεγχος πραγματοποιείται με τη χρήση των δηλώσεων:

- CYCLE <cycle node list>: όπου εισάγονται τα πεδία τα οποία χρησιμοποιούνται για την αναγνώριση του κύκλου,
- SET *name* to <cycle mark value> DEFAULT <non-cycle mark value>: όπου δημιουργείται ένα νέο πεδίο στον πίνακα αποτελεσμάτων, θέτοντας σε κάθε νέα πλειάδα που παράγεται, την τιμή non-cycle mark value. Στις πλειάδες που ήδη υπάρχουν στο αποτέλεσμα θέτεται η τιμή cycle mark value.
- USING <path column>: όπου ένα νέο γνώρισμα δημιουργείται με τύπο ARRAY, στο οποίο αποθηκεύονται οι πλειάδες που παράγονται από την αναδρομή.

Στις δηλώσεις SET και USING δεν μπορεί να ανατεθεί όνομα γνωρίσματος, το οποίο ήδη χρησιμοποιείται στην κατασκευή της αναδρομής, π.χ., τα ονόματα Node, Child, κ.λ.π. Κατά την εκτέλεση της αναδρομής, όσες πλειάδες παράγονται αποθηκεύονται

τόσο στο αποτέλεσμα της ερώτησης, όσο και στο πεδίο cyclepath. Εάν η παραγόμενη πλειάδα δεν υπάρχει ήδη στο cyclepath, τότε στο πεδίο cyclemark ανατίθεται η τιμή 'N'. Σε αντίθετη περίπτωση, όπου μια πλειάδα υπάρχει ήδη στο cyclepath, μεταβάλλεται η τιμή του πεδίου cyclemark της πλειάδας από 'N' σε 'Y', γεγονός που υποδηλώνει την ύπαρξη κύκλου στο γράφο.

4.6. Συνάθροιση - Σύνθεση (Aggregation - Composition)

4.6.1. Κίνητρο & Εφαρμογή

Πολλές φορές οι όροι *συνάθροιση* (*aggregation*) και *σύνθεση* (*composition*) συγχέονται, εσφαλμένως, μεταξύ τους. Πριν μελετήσουμε τους τρόπους αναπαράστασής τους στις Σχεσιακές Βάσεις Δεδομένων, να παρουσιάσουμε τις διαφορές ανάμεσα στις δύο συσχετίσεις.

Τα βασικά χαρακτηριστικά ενός aggregation είναι τα εξής:

1. Το aggregation αποτελεί μια *λογική συνάθροιση* αντικειμένων (*parts*) σε ένα σύνολο (*whole*). Με τον όρο λογική συνάθροιση εννοούμε οποιαδήποτε συσχέτιση η οποία πραγματοποιείται με βάση κανόνες λογικής. Π.χ., ας θεωρήσουμε ένα μαγαζί με παιχνίδια το οποίο περιλαμβάνει κούκλες. Οι κούκλες αποτελούν μέρος (part) των παιχνιδιών σε λογικό επίπεδο.
2. Δεδομένου ότι η συσχέτιση του πραγματοποιείται σε λογικό επίπεδο, δεν ισχύει εν γένει η ιδιότητα της μεταβατικότητας. Π.χ., όταν μια ορχήστρα αποτελείται από μουσικούς, και μέρος του κάθε μουσικού αποτελούν τα χέρια του, δεν μπορούμε να ισχυριστούμε ότι μια ορχήστρα αποτελείται από τα χέρια των μουσικών.
3. Ένα part κατά το aggregation μπορεί να συμμετέχει σε ένα ή περισσότερα διαφορετικά aggregations. Δηλαδή τα parts ενός aggregation μπορεί να είναι είτε *αποκλειστικά* (*exclusive*), δηλαδή κάθε συστατικό να συμμετέχει μόνο σε ένα whole, είτε *διαμοιραζόμενα* (*shared*), όπου τα συστατικά μπορούν να συμμετέχουν σε περισσότερα whole.

4. Ο κατευθυνόμενος γράφος, που αναπαριστά ένα aggregation, δεν μπορεί να περιέχει κύκλο. Δηλαδή ένα part δεν είναι δυνατό να περιλαμβάνει το whole του.
5. Η εξάρτηση μεταξύ parts και whole μπορεί να είναι είτε *whole-to-part*, όπου η ύπαρξη του whole εξαρτάται από την ύπαρξη του part, είτε *part-to-whole*, όπου η ύπαρξη των parts εξαρτάται από την ύπαρξη του whole. Με άλλα λόγια, στην περίπτωση του whole-to-part, όταν διαγράφεται ένα part αυτόματα παύει να ισχύει και το whole που δημιουργείται από αυτό, ενώ αντίθετα στην περίπτωση του part-to-whole, με την διαγραφή του whole διαγράφονται και όλα τα parts από τα οποία αποτελείται.

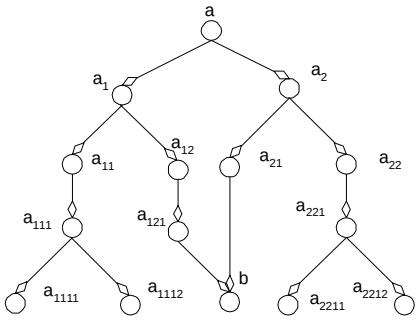
Το composition αποτελεί ένα είδος aggregation, όπου ισχύουν ορισμένοι περιορισμοί. Τα χαρακτηριστικά στα οποία διαφοροποιείται το composition από το aggregation είναι τα εξής:

1. Το whole αποτελεί *φυσική σύνθεση* των parts. Για να γίνει κατανοητή η διαφορά μεταξύ φυσικής και λογικής συσχέτισης, ας θεωρήσουμε το παράδειγμα ενός κτηρίου το οποίο αποτελείται από δωμάτια, η συσχέτιση αυτή πραγματοποιείται σε φυσικού επίπεδο.
2. Δεδομένου του ότι η συσχέτιση του composition πραγματοποιείται με την φυσική σύνθεση ενός αντικειμένου (whole) από τα συστατικά του (parts), ισχύει η μεταβατική ιδιότητα. Π.χ., ένα κτήριο αποτελείται από δωμάτια, τα δωμάτια αποτελούνται από τοίχους, άρα το κτήριο αποτελείται από τοίχους.
3. Ένα part από τη στιγμή που δημιουργεί ένα whole, ανήκει αποκλειστικά σε αυτό το αντικείμενο. Δηλαδή, στο composition τα parts είναι exclusive.
4. Υπάρχει part-to-whole εξάρτηση μεταξύ των parts και του whole που συνθέτουν. Π.χ., από τη στιγμή που γκρεμίζεται το κτήριο, αυτόματα παύουν όλα τα συστατικά από τα οποία απαρτίζεται, π.χ., τα δωμάτια και οι τοίχοι.

Συμπληρωματικά θα πρέπει να αναφέρουμε ότι αντίστοιχα με τους όρους parts και whole, δίδονται και οι όροι *components* και *composite*.

Όπως προαναφέρθηκε, η σχηματική αναπαράσταση ενός aggregation-composition δίδεται μέσω ενός γράφου, όπου οι κόμβοι του αποτελούν τα components και τα

composites που δημιουργούνται και οι ακμές αντιπροσωπεύουν τη συσχέτιση του aggregation-composition. Ας θεωρήσουμε το γενικό παράδειγμα του aggregation, Σχήμα 4.37, όπου το τελικό composite a αποτελείται από τα components a_1 και a_2, τα οποία με τη σειρά τους αποτελούν composites των a_{11}, a_{12} (για το a_1) και a_{21}, a_{22} (για το a_2). Επιπλέον ένα component μπορεί να συμπεριλαμβάνεται σε περισσότερα του ενός composites, π.χ., το composite b αποτελεί μέρος τόσο του a_{121} όσο και του a_{21}. Στην περίπτωση composition δεν θα μπορούσε να υπάρχει το component b, λόγω της αποκλειστικότητας των συστατικών.



Σχήμα 4.37 Γράφος αναπαράστασης του aggregation.

4.6.2. Δομή

Στην γενική περίπτωση ενός aggregation, όπως αυτή παρουσιάζεται μέσω του γράφου του Σχήματος 4.37, η αναπαράσταση ενός γράφου στο σχεσιακό μοντέλο πραγματοποιείται μέσω των σχέσεων `Nodes(Node_ID, Description)` και `EDGES(From,To)`, όπως αυτοί παρουσιάστηκαν στην παράγραφο 4.5.6. Επιγραμματικά, υπενθυμίζουμε ότι ο πίνακας `Nodes` καταχωρεί τους κόμβους, ενώ ο πίνακας `EDGES` τις ακμές οι οποίες αντιπροσωπεύουν την ύπαρξη συσχέτισης (aggregation ή composition) μεταξύ δύο κόμβων. Για λόγους πληρότητας επισημαίνουμε ότι λόγω της αποκλειστικής συμμετοχής ενός component σε ένα composite, ο γράφος του Σχήματος 4.37, στην περίπτωση composition, απλοποιείται σε δέντρο συσχέτισης. Στο Σχήμα 4.38 παρουσιάζεται ένα στιγμιότυπο της σχέσης `EDGES`, που αντιστοιχεί στο γράφο του Σχήματος 4.37.

EDGES

From	To
a	a_1
a	a_2
a_1	a_{11}
a_1	a_{12}
a_2	a_{21}
a_2	a_{22}
a_{11}	a_{111}
a_{12}	a_{121}
a_{22}	a_{221}
a_{111}	a_{1111}
a_{111}	a_{1112}
a_{221}	a_{2211}
a_{221}	a_{2212}
a_{121}	b
a_{21}	b

Σχήμα 4.38 Αναπαράσταση στιγμιότυπου της σχέσης EDGES για τον γράφο του παραδείγματος.

4.6.3. Συμπεριφορά σε επίπεδο στιγμιότυπου

Σε επίπεδο στιγμιότυπου η οποιαδήποτε λειτουργία *εύρεσης*/*εισαγωγής*/*διαγραφής*/*ανανέωσης* μιας πλειάδας των πινάκων NODES και EDGES ισοδυναμεί με την εφαρμογή της αντίστοιχης λειτουργίας σε κάποιον κόμβο του γράφου. Όλες οι παραπάνω λειτουργίες ακολουθούν τους κανόνες, που αναφέρθηκαν κατά την μοντελοποίηση των γράφων στην προηγούμενη ενότητα.

4.6.3.1. Εύρεση μονοπατιού

Η εύρεση της πηγής και των φύλλων του γράφου μελετήθηκε στην προηγούμενη ενότητα. Σε αυτό το σημείο το μόνο, ίσως, σημαντικό προς μελέτη αποτελεί η εύρεση των μονοπατιών ανάμεσα σε δύο κόμβους.

1. Εύρεση ενός μονοπατιού ανάμεσα σε δύο κόμβους: Η αναζήτηση ενός μονοπατιού ανάμεσα σε δύο κόμβους όπως μελετήθηκε στην

προηγούμενη παράγραφο, μπορεί να πραγματοποιηθεί είτε με την βοήθεια της αναδρομικής κλειστότητας, είτε με την εφαρμογή self joins (εάν είναι γνωστό το πλήθος των ενδιάμεσων κόμβων), με την κατασκευή μιας store procedure/function, είτε με την κατασκευή ενός cursor. Η λύση της αναδρομικής κλειστότητας είναι δυνατό να εφαρμοστεί και στην περίπτωση του aggregation καταχρηστικά. Μπορούμε λοιπόν να αναζητήσουμε ένα μονοπάτι μεταξύ δύο κόμβων ανεξάρτητα από το εάν πραγματικά υφίσταται λογική συνάθροιση μεταξύ τους. Π.χ., στην περίπτωση της ορχήστρα, η οποία αποτελείται από μουσικούς και τμήμα των μουσικών αποτελούν τα χέρια τους, μπορούμε να καταχωρήσουμε την πληροφορία (`ορχήστρα, χέρια`) στον πίνακα `EGDES`, χωρίς να υπάρχει λογική συσχέτιση μεταξύ τους.

2. Εύρεση όλων των μονοπατιών: Για την περίπτωση του composition για την αναζήτηση όλων των μονοπατιών, μπορούμε να εφαρμόσουμε την αναδρομική κλειστότητα. Λόγω του ότι στο aggregation δεν ισχύει η μεταβατικότητα, θα πρέπει να καταφύγουμε στην κατασκευή μιας επαναληπτικής διαδικασίας, π.χ., μιας store procedure ή ενός cursor. Όπως και προηγουμένως, μπορούμε καταχρηστικά να εφαρμόσουμε την αναδρομική κλειστότητα και κατά το aggregation.

4.6.3.2. Εισαγωγή ενός κόμβου

Στη γενική περίπτωση του aggregation, όπου η σχηματική αναπαράσταση δίδεται μέσω ενός κατευθυνόμενου άκυκλου γράφου, η εισαγωγή ενός κόμβου στο γράφο, πραγματοποιείται με την εισαγωγή μιας εγγραφής στο πίνακα `NODES` και των αντίστοιχων εγγραφών στον πίνακα `EDGES`. Στην περίπτωση, και μόνο, του aggregation είναι δυνατό ένας κόμβος να υπάρχει ως component στον πίνακα `NODES` και απλά να συμμετέχει σε ένα διαφορετικό aggregation. Τότε ενημερώνεται μόνο ο πίνακας `EDGES`.

4.6.3.3. Διαγραφή ενός κόμβου

Σε αυτό το σημείο θα πρέπει να γίνει διαχωρισμός της συμπεριφοράς ενός aggregation και ενός composition.

- *Aggregation*: Η διαγραφή ενός κόμβου πραγματοποιείται ανάλογα με το ρόλο του κόμβου στο γράφο απεικόνισης, και με τις ιδιότητες του. Ο κόμβος μπορεί να συμμετέχει στο γράφο ως:
    1. Πηγή. Εάν υπάρχει εξάρτηση part-to-whole, τότε η διαγραφή του τελικού component οδηγεί στην διαγραφή όλου του γράφου. Σε αντίθετη περίπτωση διαγράφεται μόνο ο συγκεκριμένος κόμβος από τον πίνακα `NODES` και οι ακμές με τις οποίες συμμετέχει στο γράφο από τον `EDGES`. Το αποτέλεσμα αυτής της διαγραφής οδηγεί στη δημιουργία δύο ξεχωριστών υπογράφων.
    2. Φύλλο. Η διαγραφή οποιουδήποτε component, ανεξάρτητα από την εξάρτηση μεταξύ composite-component, πραγματοποιείται απλά με την διαγραφή της αντίστοιχης πλειάδας από τον πίνακα `NODES` και των πλειάδων που αντιστοιχούν στις ακμές του από τον `EDGES`.
    3. Εσωτερικός κόμβος. Στο γράφο απεικόνισης ένας εσωτερικός κόμβος αντιπροσωπεύει ένα composite το οποίο αποτελεί component ενός άλλου aggregation, π.χ., ο κόμβος a_1. Και σε αυτή την περίπτωση, το αποτέλεσμα της διαγραφής εξαρτάται από την εξάρτηση που εμφανίζεται μεταξύ component και composite. Εάν η εξάρτηση είναι whole-to-part, θα πρέπει να διαγραφεί ολόκληρος ο γράφος απεικόνισης. Εάν η εξάρτηση είναι part-to-whole, διαγράφεται μόνο ο υπογράφος με πηγή τον συγκεκριμένο κόμβο. Διαγραφή του υποδέντρου σημαίνει τη διαγραφή όλων των κόμβων και των αντίστοιχων ακμών τους, που εμφανίζονται στο υποδέντρο. Η διαγραφή οποιουδήποτε κόμβου του υποδέντρου από τον πίνακα `NODES` θα πρέπει να πραγματοποιηθεί μόνο εφόσον ο κόμβος αυτός δεν συμμετέχει σε άλλο aggregation.
- *Composition*: Το composition υποστηρίζει μόνο την εξάρτηση part-to-whole. Οπότε ανάλογα με τον ρόλο του κόμβου στο γράφο, η διαγραφή μπορεί να εφαρμοστεί:

1. στην πηγή του γράφου. Σε αυτή την περίπτωση διαγράφεται όλος ο γράφος απεικόνισης και καταστρέφονται οι αντίστοιχοι πίνακες NODES και EDGES.
2. σε ένα φύλλο του γράφου. Η διαγραφή ενός component δεν οδηγεί στη διαγραφή ολόκληρου του γράφου. Απλά διαγράφεται το component και το composite που δημιουργεί και από τους δύο πίνακες NODES και EDGES.
3. σε έναν εσωτερικό κόμβο. Ένας εσωτερικός κόμβος αποτελεί composite, ως πηγή του υπογράφου του, και ταυτόχρονα component του συνολικού composite του γράφου. Με αυτό το σκεπτικό η διαγραφή του από τον γράφο απεικόνισης οδηγεί στη διαγραφή όλων των components από τα οποία απαρτίζεται, αφήνοντας αναλλοίωτο τον υπόλοιπο γράφο.

4.6.4. Συμπεριφορά σε επίπεδο σχήματος

Σε επίπεδο σχήματος η εισαγωγή οποιουδήποτε γνωρίσματος στο σχήμα της βάσης δεν μεταβάλλει τη συμπεριφορά του προτύπου. Η διαγραφή ή ανανέωση δεν είναι επιτρεπτή σε κανένα από τα γνωρίσματα που συμμετέχουν στο ορισμό της συσχέτισης *πρόγονος-απόγονος* ή *από-προς*, όπως αυτά ορίσθηκαν στους πίνακες NODES και EDGES.

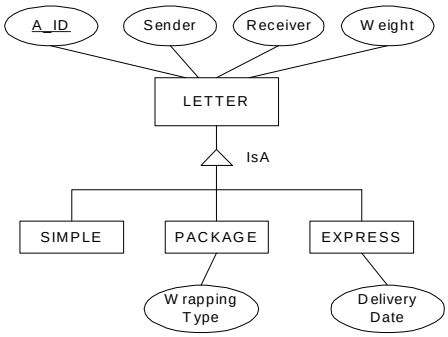
4.7. Γενίκευση - Εξειδίκευση (IsA)

4.7.1. Κίνητρο & Εφαρμογή

Ο τύπος *IsA* (ή *IS-A-SUBCLASS-OF*) αντιπροσωπεύει τη συσχέτιση μεταξύ μιας υπερκλάσης και μίας, τουλάχιστον, υποκλάσης. Στο επεκτεταμένο μοντέλο (ER) ο τύπος IsA αναπαριστά τη διαδικασία της *γενίκευσης /εξειδίκευσης*. Η γενίκευση είναι η διαδικασία κατά την οποία διάφορες κλάσεις γενικεύονται σε μια αφηρημένη κλάση υψηλότερου επιπέδου, η οποία περιλαμβάνει τα αντικείμενα αυτών των κλάσεων. Η εξειδίκευση αποτελεί αντίστροφη διαδικασία της γενίκευσης, όπου μια

κλάση αντικειμένων κατατάσσεται σε πιο εξειδικευμένες υποκλάσεις. Ουσιαστικά η εξειδίκευση αποτελεί εννοιολογική εκλέπτυνση, ενώ η γενίκευση εννοιολογική σύνθεση μια κλάσης.

Ας θεωρήσουμε το παράδειγμα μιας εταιρίας μεταφοράς γραμμάτων. Ένα γράμμα μπορεί να είναι απλό (SIMPLE), πακέτο (PACKAGE) ή ταχείας μεταφοράς (EXPRESS). Κάθε γράμμα έχει αποστολέα, παραλήπτη και βάρος. Επιπλέον, ένα πακέτο έχει είδος συσκευασίας (απλή, σκληρή, ενισχυμένη), ενώ για τα express γράμματα κρατάμε τη συμπεφωνημένη ημερομηνία παράδοσής του. Η αναπαράσταση του παραδείγματος στο ER μοντέλο δίνεται στο Σχήμα 4.39.



Σχήμα 4.39 Αναπαράσταση του παραδείγματος στο ER μοντέλο.

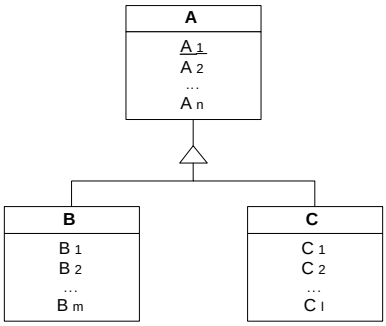
Παρατηρούμε ότι:

1. η κλάση LETTER αποτελεί υπερκλάση (πιο γενική, αφηρημένη κλάση) των κλάσεων SIMPLE, PACKAGE, EXPRESS.
2. τα γνωρίσματα της υπερκλάσης *κληρονομούνται* (δηλαδή αποτελούν επίσης γνωρίσματα) στις υποκλάσεις SIMPLE, PACKAGE, EXPRESS.
3. κάθε μία από τις υποκλάσεις είναι δυνατό να περιλαμβάνει δικά της γνωρίσματα, τα οποία χαρακτηρίζουν μόνο αυτή.

Η απεικόνιση του παραπάνω μοντέλου σε σχεσιακό σχήμα, μπορεί να πραγματοποιηθεί με πέντε (5) σχεδιαστικές λύσεις, οι οποίες μελετούνται στην επόμενη παράγραφο.

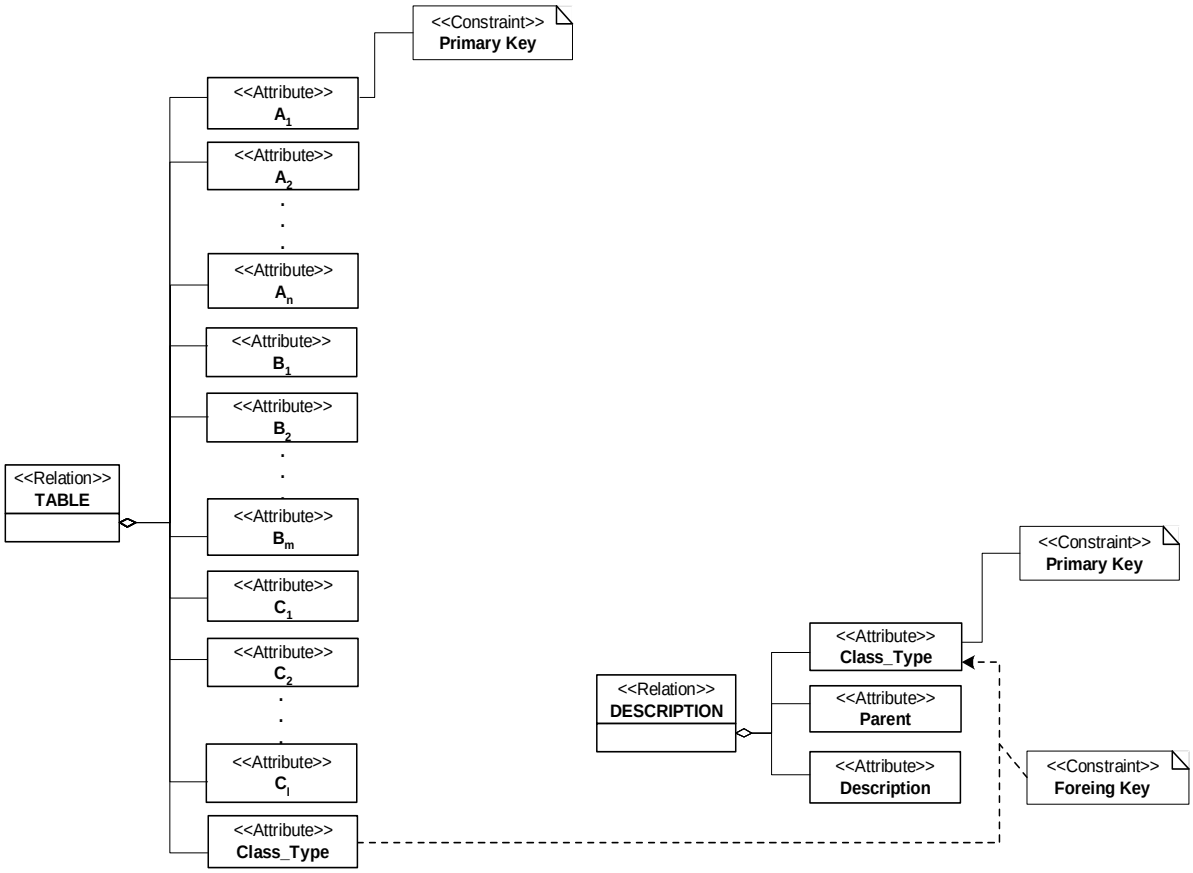
4.7.2. Δομή

Στην παράγραφο αυτή δίνονται σχηματικά όλες οι προτεινόμενες λύσεις τόσο στην γενική τους αναπαράσταση όσο και συγκεκριμένα για το παράδειγμα LETTER. Για την γενική αναπαράσταση της ιδιότητας IsA, ας θεωρήσουμε ότι έχουμε μια υπερκλάση $A(\underline{A_1}, A_2, \ldots, A_n)$ και τις δυο υποκλάσεις της $B(B_1, B_2, .., B_m)$ και $C(C_1, \ldots, C_l)$. Το γνώρισμα A_1 αποτελεί κλειδί της υπερκλάσης, και λόγω της ιδιότητας της κληρονομικότητας και κλειδί των υποκλάσεων. Στο Σχήμα 4.40 η UML αναπαράσταση της γενικής περίπτωσης IsA.



Σχήμα 4.40 Αναπαράσταση σε UML της γενικής περίπτωσης IsA.

Η πρώτη σχεδιαστική λύση προτείνει την ύπαρξη ενός πίνακα, ο οποίος να περιλαμβάνει όλα τα γνωρίσματα όλων των κλάσεων, Σχήμα 4.41. Η ύπαρξη του πεδίου Class_Type συνεισφέρει στην κατάταξη κάθε εγγραφής σε κάποια από τις κλάσεις. Π.χ., στο Σχήμα 4.42 κάθε μία εγγραφή του πίνακα ALL_LETTERS, ανάλογα με την κλάση στην οποία ανήκει, συμπληρώνει τιμές και στα αντίστοιχα πεδία. Στο βοηθητικό πίνακα DESCRIPTION καταχωρούνται όλες οι κλάσεις και οι ιεραρχίες τους, π.χ., καταχωρείται το γεγονός ότι η κλάση PACKAGE αποτελεί υποκλάση της LETTER. Σε αυτό το σημείο θα πρέπει να τονιστεί ότι λόγω της μη ύπαρξης ξεχωριστού πεδίου για την υποκλάση SIMPLE, θεωρείται ότι δεν αποτελεί ξεχωριστό τμήμα της υπερκλάσης, γι' αυτό το λόγο και παραλείπεται τόσο σε επίπεδο σχήματος όσο και σε επίπεδο στιγμιότυπου.



Σχήμα 4.41 Αναπαράσταση δομής σχήματος της γενικής περίπτωσης IsA εφαρμόζοντας την πρώτη σχεδιαστική λύση (*ένας πίνακας να περιλαμβάνει τα γνωρίσματα όλων των κλάσεων*).

ALL_LETTERS

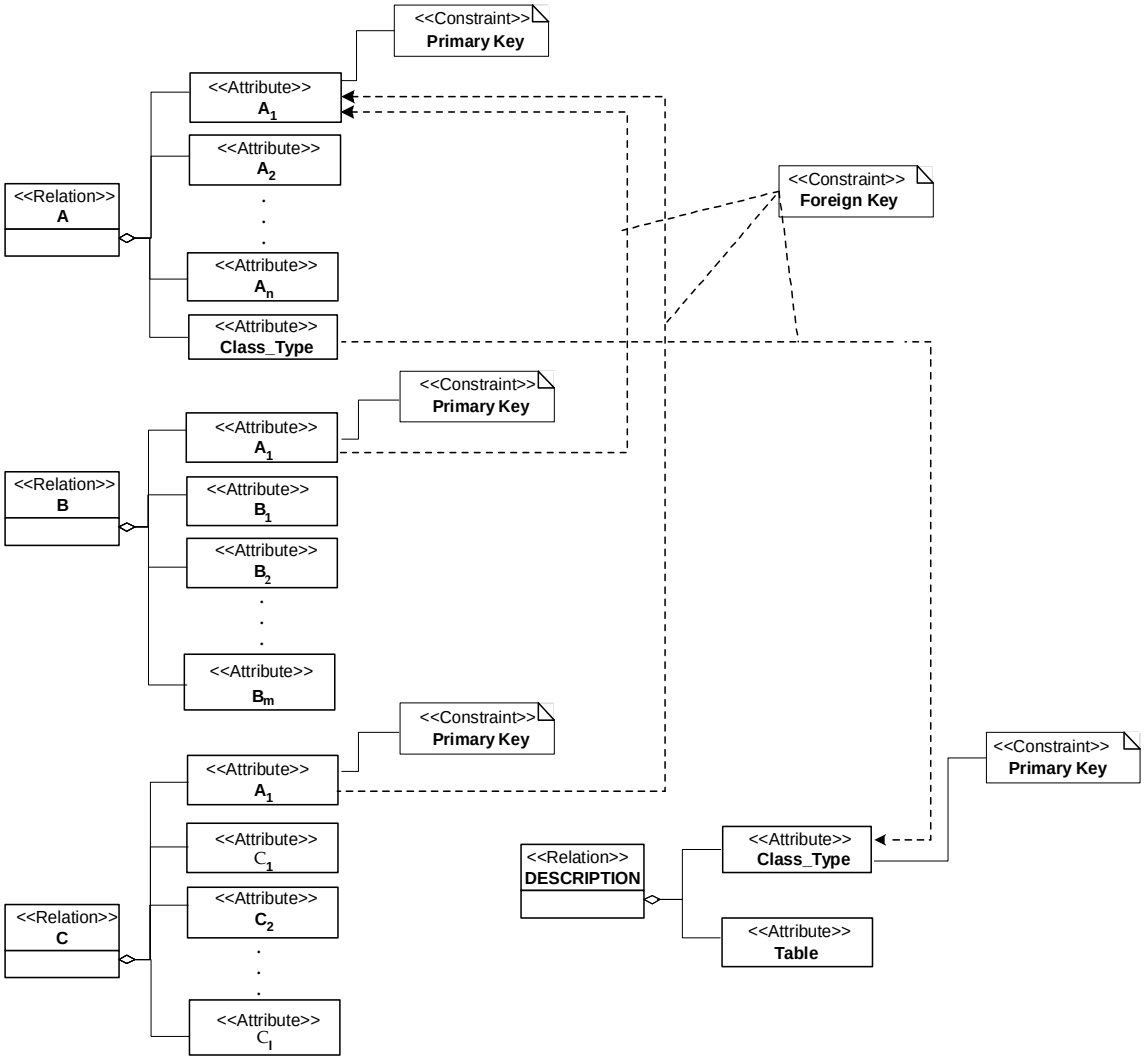
A_ID	Sender	Receiver	Weight	Wrapping Type	Delivery Date	Class_Type
1	Johnson	Thomson	10	Simple		2
2	Stathos	Robins	2			1
3	Henry	Tomas	3		13/10/07	3

DESCRIPTION

Class_Type	Parent	Description
1	1	ALL_LETTERS (including Simple)
2	1	Package
3	1	Express

Σχήμα 4.42 Αναπαράσταση στιγμιότυπου του παραδείγματος με χρήση της πρώτης σχεδιαστικής μεθόδου.

Η δεύτερη σχεδιαστική λύση, που παρουσιάζεται στο Σχήμα 4.43, περιλαμβάνει την δημιουργία διαφορετικού πίνακα για κάθε υποκλάση, η οποία περιέχει ξεχωριστά γνωρίσματα. Κάθε πίνακας περιλαμβάνει τα γνωρίσματα της αντίστοιχης κλάσης, συμπεριλαμβανομένου του πρωτεύοντος κλειδιού της υπερκλάσης, ως ξένο κλειδί. Ένα τυχαίο στιγμιότυπο της λύσης παρουσιάζεται στο Σχήμα 4.44



Σχήμα 4.43 Αναπαράσταση δομής σχήματος της γενικής περίπτωσης IsA εφαρμόζοντας την δεύτερη σχεδιαστική λύση (*δημιουργία ενός πίνακα για την υπερκλάση, ο οποίος λειτουργεί ως βοηθητικός πίνακας αναζήτησης, και ενός πίνακα για κάθε υποκλάση*).

DESCRIPTION

Class_Type	Table
1	ALL_LETTERS
2	PACKAGE
3	EXPRESS

ALL_LETTERS

A_ID	Sender	Receiver	Weight	Class_Type
1	Johnson	Thomson	10	2
2	Stathos	Robins	2	1
3	Henry	Tomas	3	3

PACKAGE

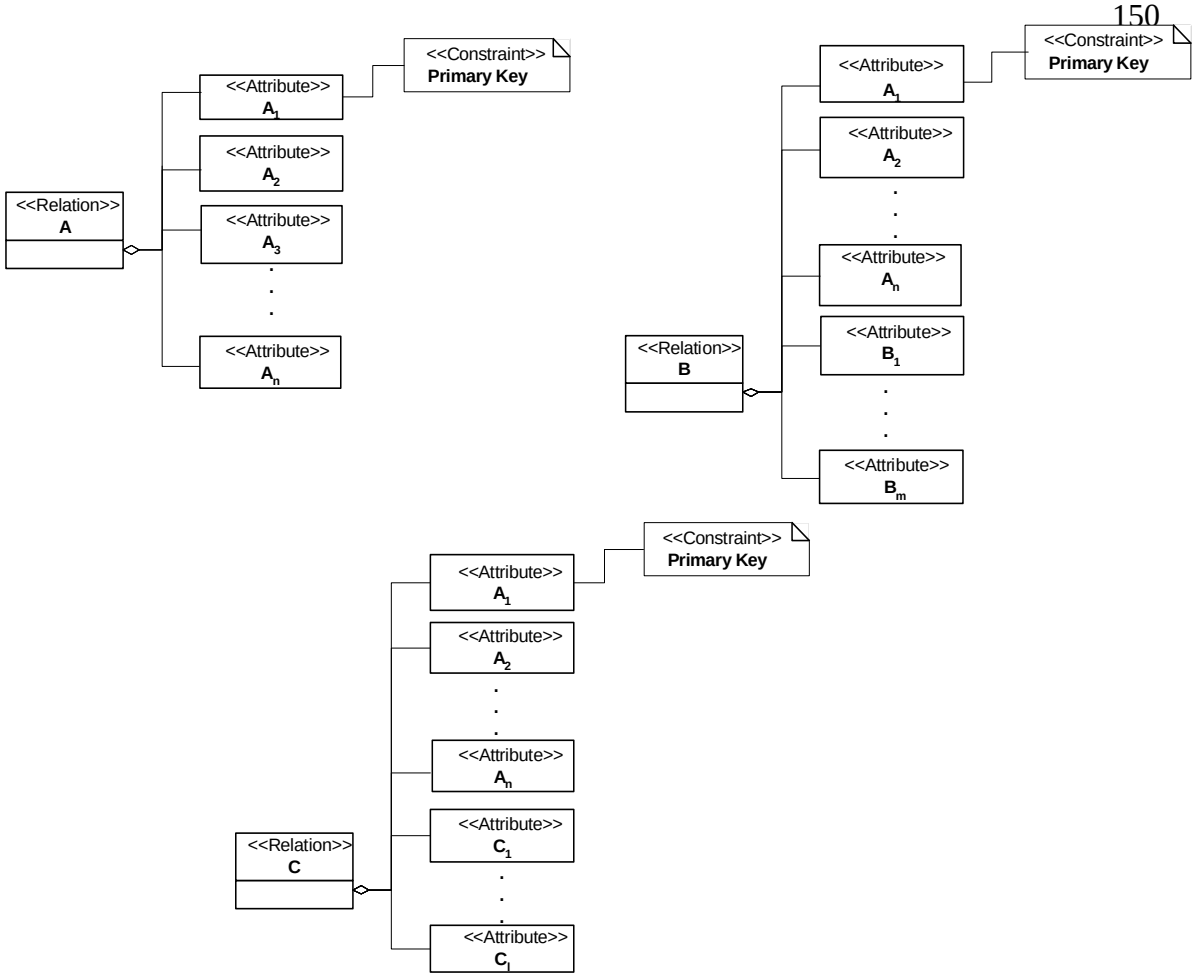
A_ID	Wrapping Type
1	Simple

EXPRESS

A_ID	Delivery Date
3	13/10/07

Σχήμα 4.44 Αναπαράσταση στιγμιότυπου του παραδείγματος με χρήση της δεύτερης σχεδιαστικής μεθόδου.

Η τρίτη σχεδιαστική λύση, Σχήμα 4.45, παρουσιάζει ξεχωριστούς πίνακες για κάθε υποκλάση, οι οποίοι συμπεριλαμβάνουν όλα τα γνωρίσματα της υπερκλάσης. Στιγμιότυπο της λύσης αυτής αποτελεί το Σχήμα 4.46.



Σχήμα 4.45 Αναπαράσταση δομής σχήματος της γενικής περίπτωσης IsA εφαρμόζοντας την τρίτη σχεδιαστική λύση (*δημιουργία ξεχωριστών πινάκων για κάθε κλάση χωρίς περιορισμούς αναφορικής ακεραιότητας*).

ALL_LETTERS

A_ID	Sender	Receiver	Weight
2	Stathos	Robins	2

PACKAGE

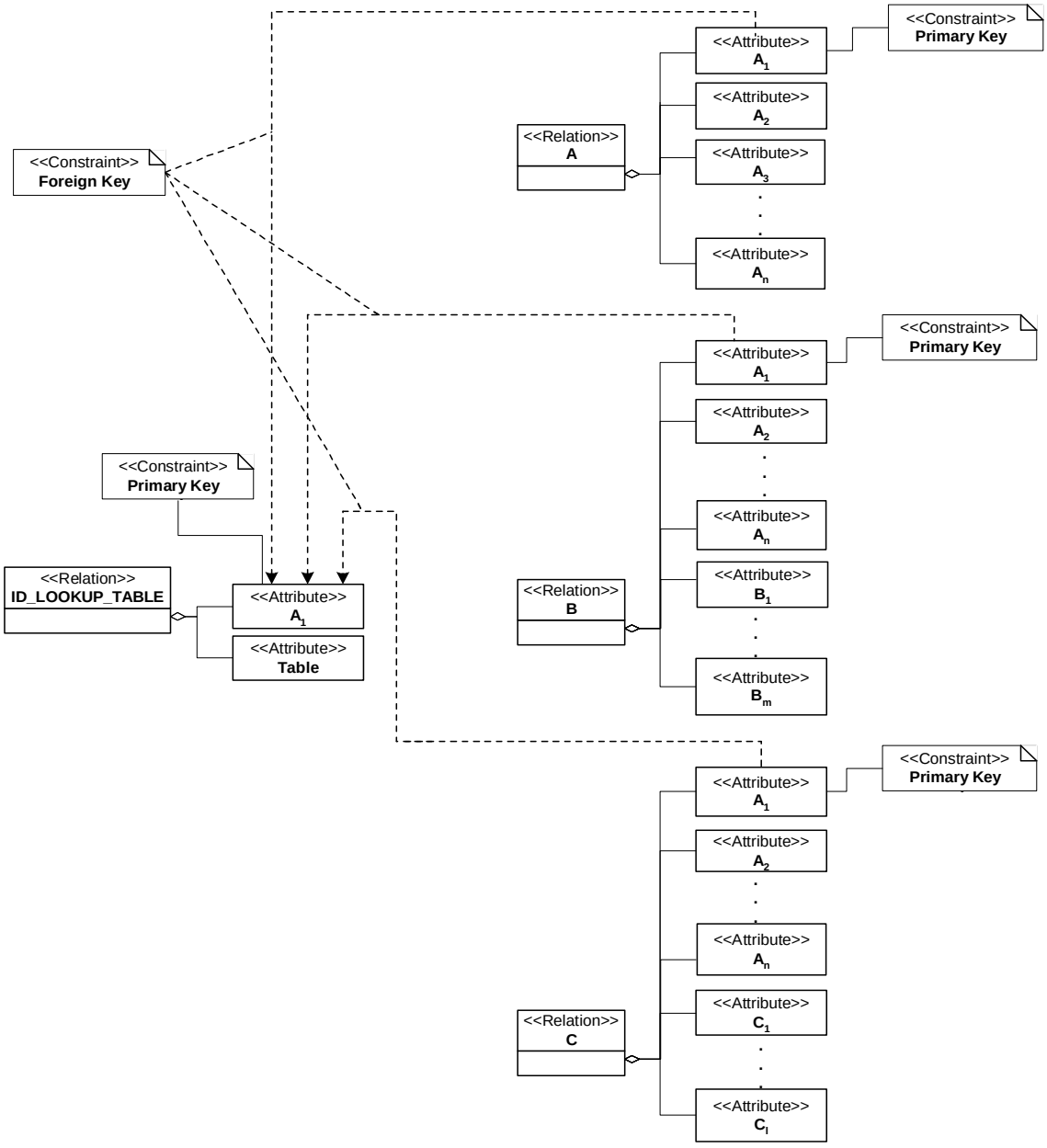
A_ID	Sender	Receiver	Weight	Wrapping Type
1	Johnson	Thomson	10	Simple

EXPRESS

A_ID	Sender	Receiver	Weight	Delivery Date
3	Henry	Tomas	3	13/10/07

Σχήμα 4.46 Αναπαράσταση στιγμιότυπου του παραδείγματος με χρήση της τρίτης σχεδιαστικής μεθόδου.

Στο Σχήμα 4.47 παρουσιάζεται η τέταρτη σχεδιαστική, όπου κάθε κλάση αντιστοιχίζεται σε έναν ξεχωριστό πίνακα και επιπλέον κάθε υποκλάση συμπεριλαμβάνει όλα τα γνωρίσματα της υπερκλάσης. Ο πίνακας `ID_LOOKUP_TABLE`, καταχωρεί τα αναγνωριστικά όλων των πινάκων, και επιπλέον καταδεικνύει σε ποιον πίνακα εμφανίζεται κάθε αναγνωριστικό, κάνοντας την αναζήτηση πιο αποδοτική. Στο Σχήμα 4.48 παρουσιάζεται ένα στιγμιότυπο της λύσης αυτής.



Σχήμα 4.47 Αναπαράσταση δομής σχήματος της γενικής περίπτωσης IsA εφαρμόζοντας την τέταρτη σχεδιαστική λύση (*ξεχωριστοί πίνακες για κάθε κλάση με περιορισμούς αναφορικής ακεραιότητας σε έναν βοηθητικό πίνακα*).

ID_LOOKUP

A_ID	Table
2	ALL_LETTERS
1	PACKAGE
3	EXPRESS

ALL_LETTERS

A_ID	Sender	Receiver	Weight
2	Stathos	Robins	2

PACKAGE

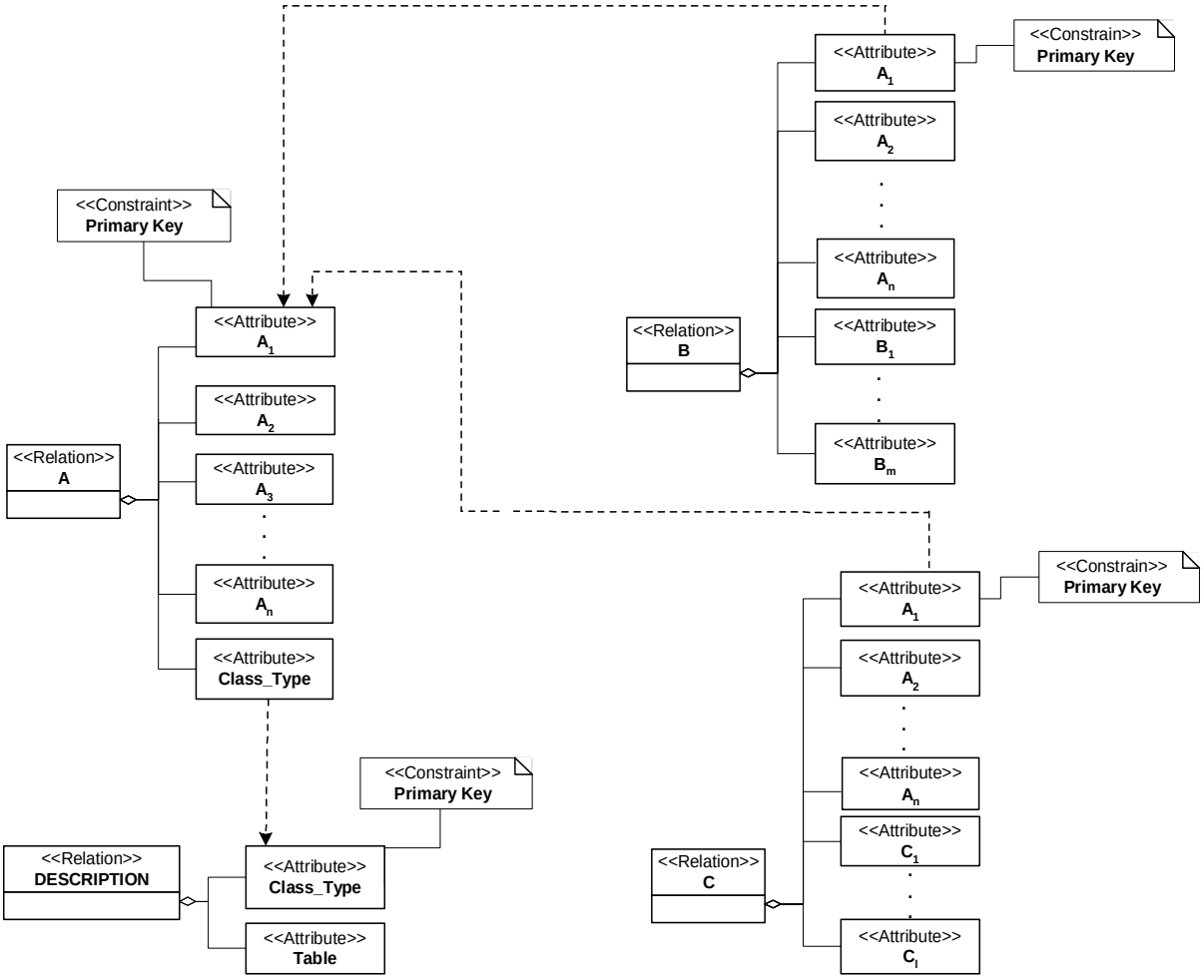
A_ID	Sender	Receiver	Weight	Wrapping Type
1	Johnson	Thomson	10	Simple

EXPRESS

A_ID	Sender	Receiver	Weight	Delivery Date
3	Henry	Tomas	3	13/10/07

Σχήμα 4.48 Αναπαράσταση στιγμιότυπου του παραδείγματος με χρήση της τέταρτης σχεδιαστικής μεθόδου.

Η τελευταία λύση αποτελεί, και αυτή, υβρίδιο της δεύτερης και τρίτης σχεδιαστική λύσης. Όπως παρουσιάζεται και στο Σχήμα 4.49, η υπερκλάση αποτελεί ξεχωριστό πίνακα και κάθε υποκλάση περιλαμβάνει όλα τα γνωρίσματα της υπερκλάσης. Η μόνη διαφορά με την τέταρτη σχεδιαστική λύση είναι ότι τα ξένα κλειδιά των υποκλάσεων αναφέρονται στο πρωτεύον κλειδί του πίνακα της υπερκλάσης. Στο Σχήμα 4.50 δίνεται ένα στιγμιότυπο της πέμπτης σχεδιαστικής λύσης.



Σχήμα 4.49 Αναπαράσταση δομής σχήματος της γενικής περίπτωσης IsA εφαρμόζοντας την πέμπτη σχεδιαστική λύση (*δημιουργία ξεχωριστών πινάκων για κάθε υποκλάση με περιορισμούς αναφορικής ακεραιότητας στον πίνακα της υπερκλάσης, και ενός βοηθητικού πίνακα*).

DESCRIPTION

Class_Type	Table
1	ALL_LETTERS
2	PACKAGE
3	EXPRESS

ALL_LETTERS

A_ID	Sender	Receiver	Weight	Class_Type
1	Johnson	Thomson	10	2
2	Stathos	Robins	2	1
3	Henry	Tomas	3	3

PACKAGE

A_ID	Sender	Receiver	Weight	Wrapping Type
1	Johnson	Thomson	10	Simple

EXPRESS

A_ID	Sender	Receiver	Weight	Delivery Date
3	Henry	Tomas	3	13/10/07

Σχήμα 4.50 Αναπαράσταση στιγμιότυπου του παραδείγματος με χρήση της πέμπτης σχεδιαστικής μεθόδου.

Σε αυτό το σημείο θα πρέπει να παρατηρηθεί ότι στην περίπτωση όπου η υπερκλάση είναι *εικονική* (*virtual*) στις τρεις τελευταίες σχεδιαστικές λύσεις απαλείφεται ο πίνακας LETTER, ο οποίος αντιπροσωπεύει την υπερκλάση. Για λόγους πληρότητας αναφέρεται ότι, μια υπερκλάση είναι εικονική όταν δεν έχει αμιγώς δικά της στιγμιότυπα, αλλά κάθε στιγμιότυπό της εμφανίζεται, υποχρεωτικά, ως στιγμιότυπο ακριβώς μιας υποκλάσης της.

4.7.3. Συμπεριφορά σε επίπεδο στιγμιότυπου

4.7.3.1. Επιλογή μιας εγγραφής

Σε αυτή την παράγραφο θα μελετήσουμε τη συμπεριφορά κάθε σχεδιαστικής λύσης κατά την αναζήτηση μιας εγγραφής, με βάση το πεδίο του πρωτεύοντος κλειδιού.

Η αναζήτηση στην πρώτη σχεδιαστική λύση (ένας πίνακας να περιλαμβάνει τα γνωρίσματα όλων των κλάσεων), πραγματοποιείται εύκολα, εφόσον όλα τα γνωρίσματα βρίσκονται στον ίδιο πίνακα.

Η δεύτερη σχεδιαστική λύση (ξεχωριστοί πίνακες για κάθε υποκλάση και ένας, βοηθητικός, πίνακας για την υπερκλάση) απαιτεί τη συνένωση δύο πινάκων, του πίνακα της υπερκλάσης και του πίνακα της υποκλάσης στην οποία ανήκει η συγκεκριμένη εγγραφή. Εάν δεν είναι γνωστό σε ποια υποκλάση ανήκει η συγκεκριμένη εγγραφή, τότε θα πρέπει να προηγηθεί η εύρεσή της μέσω του βοηθητικού πίνακα. Π.χ., εάν ζητείται η εγγραφή με `A_ID = 1`, τότε θα πρέπει να αναζητηθεί στον γενικό πίνακα `LETTER` προσδιορίζοντας την κλάση στην οποία ανήκει. Εφόσον διαπιστωθεί ότι ανήκει στην κλάση `PACKAGE`, θα πρέπει να πραγματοποιηθεί συνένωση των πινάκων `LETTER` και `PACKAGE` ώστε να εξαχθεί όλη η πληροφορία της δεδομένης εγγραφής, η οποία είναι κατανεμημένη στους δύο πίνακες.

Η τρίτη σχεδιαστική λύση (ξεχωριστοί πίνακες για κάθε κλάση χωρίς περιορισμούς αναφορικής ακεραιότητας), μπορεί να πραγματοποιήσει εύκολα μια αναζήτηση αρκεί να είναι γνωστός ο πίνακας στον οποίον βρίσκεται η εγγραφή. Σε αυτή την περίπτωση η εξαγωγή της συνολικής πληροφορίας είναι εύκολη, εφόσον περιλαμβάνεται στον αντίστοιχο πίνακα. Π.χ., εάν γνωρίζουμε ότι το `A_ID` αναφέρεται σε πακέτο, τότε απλά αναζητούμε τη συγκεκριμένη πλειάδα στον πίνακα `PACKAGE`. Ο πίνακας περιέχει όλα τα γνωρίσματα της κλάσης `PACKAGE`, συμπεριλαμβανομένων και των γνωρισμάτων της υπερκλάσης `LETTER`. Εάν όμως δεν είναι γνωστός ο πίνακας εύρεσης, τότε θα πρέπει να πραγματοποιηθεί αναζήτηση σε όλους τους πίνακες.

Η τέταρτη σχεδιαστική λύση (ξεχωριστοί πίνακες για κάθε κλάση με περιορισμούς αναφορικής ακεραιότητας σε έναν βοηθητικό πίνακα), πραγματοποιεί μια αναζήτηση σε δύο στάδια. Πρώτα βρίσκεται η κλάση (ο πίνακας) στη οποία ανήκει η συγκεκριμένη καταχώρηση του πίνακα `ID_LOOKUP_TABLE`. Στη συνέχεια η εύρεση όλης της πληροφορίας αποτελεί εύκολη διαδικασία εφόσον κάθε πίνακας περιλαμβάνει όλα τα σχετικά γνωρίσματα.

Τέλος και η πέμπτη σχεδιαστική λύση (ξεχωριστοί πίνακες για κάθε υποκλάση με περιορισμούς αναφορικής ακεραιότητας στον πίνακα της υπερκλάσης και ένας βοηθητικός πίνακας), προϋποθέτει την γνώση του αντίστοιχου πίνακα καταχώρησης. Εάν δεν είναι γνωστή η κλάση στην οποία ανήκει το συγκεκριμένο στιγμιότυπο, τότε είναι απαραίτητη η εκ των προτέρων εύρεση του αντίστοιχου πίνακα στον οποίο είναι αποθηκευμένη η εγγραφή. Η εύρεση όλης της πληροφορίας είναι και σε αυτή την περίπτωση εύκολη, διότι κάθε πίνακας περιέχει όλα τα πεδία της αντίστοιχης κλάσης αλλά και της υπερκλάσης.

4.7.3.2. Επιλογή όλων των εγγραφών

Λίγο πιο σύνθετη είναι η περίπτωση αναζήτησης όλων των εγγραφών που είναι καταχωρημένες στους διαφορετικούς πίνακες. Σε αυτή την παράγραφο παρατίθενται οι τρόποι, με τους οποίους μπορούμε να ανακατασκευάσουμε από τους πίνακες, της κάθε λύσης, έναν που να περιέχει όλη την πληροφορία. Ο πίνακας `ALL_LETTERS` της πρώτης λύσης αποτελεί το αποτέλεσμα της ανακατασκευής των διαφόρων πινάκων, κάθε προτεινόμενης λύσης, σε έναν ενιαίο πίνακα.

Η δεύτερη σχεδιαστική λύση προϋποθέτει την εφαρμογή *πλήρους εξωτερικής συνένωσης* (*Full Outer Join*) όλων των πινάκων που αντιστοιχούν σε όλες τις κλάσεις.

Η τρίτη και τέταρτη σχεδιαστική λύση, για την ανακατασκευή του ενιαίου πίνακα, απαιτούν την εφαρμογή της λειτουργίας *ένωσης* (*Union*) όλων των σχετικών πινάκων.

Τέλος στην πέμπτη σχεδιαστική λύση, λόγο της ύπαρξης περιορισμών αναφορικής ακεραιότητας από τους πίνακες των υποκλάσεων προς τον πίνακα της υπερκλάσης, θα πρέπει να πραγματοποιηθεί *ένωση* (*Union*) των πινάκων των υποκλάσεων και στη συνέχεια εφαρμογή *εξωτερικής συνένωσης* (*Outer Join*) με τον πίνακα της υπερκλάσης.

Σε αυτό το σημείο σημειώνεται ότι, εάν ορίσουμε κάποια *εικονική όψη* (*virtual view*), για οποιοδήποτε σχεδιαστική λύση, η ερώτηση πραγματοποιείται εύκολα. Όπως έχουμε προαναφέρει, μια εικονική όψη δεν λαμβάνει περιεχόμενα κατά τη στιγμή του ορισμού της αλλά τη στιγμή που διατυπώνεται μια ερώτηση προς αυτή. Με αυτόν τον τρόπο, η εικονική όψη είναι πάντα ενημερωμένη, διότι οποιαδήποτε ανανέωση των πλειάδων των βασικών πινάκων πάνω στους οποίους ορίζεται η όψη, αντανακλάται στην όψη. Ορίζοντας, λοιπόν, μια εικονική όψη για την ανάκτηση εγγραφών όλων των πινάκων, ανάλογα με τη σχεδιαστική λύση που έχει επιλεγεί, μπορούμε να θέσουμε οποιαδήποτε ερώτηση στην όψη αποφεύγοντας τον προσδιορισμό συνενώσεων (Joins) και ενώσεων (Unions) των βασικών πινάκων. Η κατασκευή της όψης μπορεί να πραγματοποιηθεί είτε με βάση τα κοινά χαρακτηριστικά όλων των κλάσεων (γνωρίσματα της υπερκλάσης), είτε με βάση όλα τα χαρακτηριστικά των κλάσεων (ανάκτηση όλης της πληροφορίας που είναι καταχωρημένη). Π.χ., ας θεωρήσουμε ότι έχουμε επιλέξει την τρίτη σχεδιαστική λύση (ξεχωριστοί πίνακες για κάθε κλάση χωρίς περιορισμούς αναφορικής ακεραιότητας) και έχουμε ορίσει την παρακάτω εικονική όψη, η οποία δημιουργεί έναν εικονικό πίνακα με τα γνωρίσματα όλων των ξεχωριστών πινάκων, παρέχοντας όλη την πληροφορία που είναι αποθηκευμένη σε όλους τους πίνακες:

```
CREATE VIEW ALL_CLASSES_ATTR AS
(    ( SELECT A1, A2, ..., An, NULL AS B1, …, NULL AS Bm, NULL AS C1, …,
            NULL AS Cl
     FROM A )
  UNION
  ( SELECT A1, A2, ..., An, B1, …, Bm, NULL AS C1, …, NULL AS Cl
    FROM B  )   )
UNION
 ( SELECT A1, A2, ..., An, NULL AS B1, …, NULL AS Bm, C1, …, Cl
 FROM C ).
```

Ας υποθέσουμε, επιπλέον, ότι ζητείται η επιλογή όλων των πλειάδων που ικανοποιούν τη συνθήκη A_n = 'x'. Με την κατασκευή της όψης ALL_CLASSES_ATTR, θέτουμε την ερώτηση στη όψη:

SELECT * FROM ALL_CLASSES_ATTR WHERE A_n = 'x',

αποφεύγοντας, με αυτόν τον τρόπο, την εφαρμογή της ερώτησης σε όλους τους πίνακες και εν συνεχεία την ένωση των αποτελεσμάτων.

4.7.3.3. Εισαγωγή, Διαγραφή, Ανανέωση Πλειάδας

Η εφαρμογή οποιασδήποτε λειτουργίας σε επίπεδο στιγμιότυπου πραγματοποιείται εύκολα στον ενιαίο πίνακα της πρώτης σχεδιαστικής λύσης. Ακόμα και στην περίπτωση όπου ένα στιγμιότυπο αλλάζει κλάση, δηλαδή εάν έχει καταχωρηθεί ως στιγμιότυπο της PACKAGE, μπορεί εύκολα να διαγραφεί και να εισαχθεί σε νέα κλάση με διαφορετικές τιμές στα αντίστοιχα πεδία, π.χ., ως στιγμιότυπο της EXPRESS.

Η εισαγωγή/διαγραφή μιας εγγραφής στη δεύτερη σχεδιαστική λύσης πραγματοποιείται αρχικά με την εισαγωγή/διαγραφή της εγγραφής στο πίνακα της υπερκλάσης και στη συνέχεια την «προώθηση» της λειτουργίας στον αντίστοιχο πίνακα της υποκλάσης της οποία αποτελεί στιγμιότυπο. Π.χ., για την εισαγωγή ενός πακέτου, θα πρέπει πρωτίστως να καταχωρηθεί η εγγραφή στον πίνακα LETTER και στη συνέχεια να καταχωρηθεί και στον πίνακα PACKAGE. Επιπλέον η αλλαγή της κλάσης στην οποία ανήκει ένα στιγμιότυπο, πραγματοποιείται με την διαγραφή της αντίστοιχης πλειάδας από τον πίνακα της υποκλάσης, την εισαγωγής της στον πίνακα της άλλης υποκλάσης και τέλος την ανανέωση του πεδίου Class_Type του πίνακα της υπερκλάσης.

Η εισαγωγή/διαγραφή/ανανέωση μιας πλειάδας στην περίπτωση ύπαρξης ξεχωριστών πινάκων για κάθε κλάση χωρίς περιορισμούς αναφορικής ακεραιότητας πραγματοποιείται εύκολα, εφόσον υπάρχουν ξεχωριστοί πίνακες χωρίς αναφορές ξένων κλειδιών. Η αλλαγή κλάσης επιτυγχάνεται με την διαγραφή της πλειάδας από τον έναν πίνακα και την εισαγωγής της στον άλλον.

Η τέταρτη και πέμπτη σχεδιαστική λύση παρουσιάζουν ανάλογη συμπεριφορά με αυτή της δεύτερης. Για την τέταρτη λύση οποιαδήποτε εισαγωγή/διαγραφή πραγματοποιείται στον πίνακα `ID_LOOKUP_TABLE` και προωθείται στον αντίστοιχο πίνακα της κλάσης στην οποία εμφανίζεται η εγγραφή ως στιγμιότυπο. Αντίστοιχα, κάθε πλειάδα που εισάγεται ή διαγράφεται από τον πίνακα `LETTER` της τελευταίας λύσης, εισάγεται ή διαγράφεται και από τον αντίστοιχο πίνακα της υποκλάσης. Η αλλαγή κλάσης, και στις δύο λύσεις, περιλαμβάνει την διαγραφή της εγγραφής από τον έναν πίνακα, και στην εισαγωγή της στον άλλο, συμπεριλαμβανομένης και της ανανέωσης της τιμής του πεδίου `Table` ή `Class_Type`, για την τέταρτη και πέμπτη λύση αντίστοιχα.

Σε αυτό το σημείο θα πρέπει να παρατηρηθεί ότι, μόνο για λόγους συντομίας οι λειτουργίες εφαρμόζονται στον πίνακα της υπερκλάσης `ALL_LETTERS` ή στον βοηθητικό πίνακα `ID_LOOKUP_TABLE`. Οποιαδήποτε αλλαγή μπορεί να εμφανιστεί σε κάθε πίνακα. Η αλλαγή αυτή, προωθείται και στον πίνακα της υποκλάσης για τη δεύτερη και πέμπτη σχεδιαστική λύση, ή στον γενικό πίνακα για την τέταρτη λύση.

4.7.4. Συμπεριφορά σε επίπεδο σχήματος

4.7.4.1. Εισαγωγή, Διαγραφή Γνωρίσματος

Στην πρώτη σχεδιαστική λύση, ένας πίνακας περιλαμβάνει όλα τα γνωρίσματα, η εισαγωγή/διαγραφή ενός γνωρίσματος μιας κλάσης αντιστοιχεί στην εισαγωγή/διαγραφή του αντίστοιχου πεδίου στον πίνακα καταχώρησης.

Στη δεύτερη σχεδιαστική λύση, όπου εμφανίζονται διαφορετικοί πίνακες για κάθε κλάση και οι αντίστοιχοι των υποκλάσεων περιλαμβάνουν μόνο το πρωτεύον κλειδί της υπερκλάσης, η εισαγωγή/διαγραφή ενός γνωρίσματος μιας κλάσης περιορίζεται στην εφαρμογή της λειτουργίας στον αντίστοιχο πίνακα. Π.χ., η εισαγωγή ενός γνωρίσματος στο πίνακα της υπερκλάσης δεν επηρεάζει τους πίνακες των υποκλάσεων, εφόσον ο καθένας από αυτούς περιλαμβάνει μόνο το πρωτεύον κλειδί της υπερκλάσης.

Στην τρίτη σχεδιαστική λύση, η οποία παρουσιάζει ξεχωριστούς πίνακες για κάθε κλάση χωρίς περιορισμούς αναφορικής ακεραιότητας, η εισαγωγή/διαγραφή ενός γνωρίσματος μιας οποιασδήποτε κλάσης περιορίζεται μόνο στον αντίστοιχο πίνακα.

Στην τέταρτη και πέμπτη λύση η μόνη διαφορά παρατηρείται στην περίπτωση όπου το πεδίο που εισάγεται ή διαγράφεται αποτελεί γνώρισμα της υπερκλάσης. Τότε θα πρέπει η συγκεκριμένη λειτουργία να προωθηθεί και στους πίνακες που αντιπροσωπεύουν τις υποκλάσεις.

4.7.4.2. Εισαγωγή ή Διαγραφή Κλάσης

Η εισαγωγή/διαγραφή μιας κλάσης στην πρώτη σχεδιαστική λύση οδηγεί στην εισαγωγή/διαγραφή των αντίστοιχων πεδίων από τον ενιαίο πίνακα και ανανέωση του πίνακα `DESCRIPTION`.

Στην δεύτερη λύση για την εισαγωγή μιας κλάσης δημιουργείται νέος πίνακας, του οποίου το πρωτεύον κλειδί αποτελεί αναφορά ξένου κλειδιού στον πίνακα της υπερκλάσης, και εν συνεχεία εισάγεται μια εγγραφή στον πίνακα των περιγραφών `DESCRIPTION`. Η διαγραφή μιας κλάσης αντιστοιχεί στη διαγραφή του αντίστοιχου πίνακα από το σχήμα της βάσης, και στην απομάκρυνση όλων των αντίστοιχων εγγραφών που εμφανίζονται στον πίνακα της υπερκλάσης μέσω της αναφοράς ξένου κλειδιού. Τέλος διαγράφεται η αντίστοιχη εγγραφή του πίνακα `DESCRIPTION`.

Η εισαγωγή ή διαγραφή μιας κλάσης, στην τρίτη και πέμπτη σχεδιαστική λύση, εφαρμόζεται με την εισαγωγή ή διαγραφή ενός πίνακα.

Παρόμοια αντιμετωπίζει την εισαγωγή/διαγραφή μιας κλάσης και η τέταρτη λύση, με τη διαφορά ότι θα πρέπει να ενημερωθεί ο πίνακας `DESCRIPTION`.

4.7.5. Πλεονεκτήματα & Μειονεκτήματα

Σε αυτή την παράγραφο θα μελετήσουμε τα πλεονεκτήματα και τα μειονεκτήματα κάθε μιας από τις προτεινόμενες σχεδιαστικές μεθόδους.

1η μέθοδος: Δημιουργία ενός πίνακα που περιλαμβάνει τα γνωρίσματα όλων των κλάσεων.

- Εμφάνιση πολλών τιμών NULL. Είναι προφανές ότι μια εγγραφή που αντιστοιχεί σε μια κλάση που είναι ψηλά στην ιεραρχία, θα παρουσιάζει τιμές μόνο στα αντίστοιχα πεδία της κλάσης και NULL σε όλα τα άλλα πεδία.
- Είναι δύσκολος ο έλεγχος ορθότητας καταχώρησης μιας εγγραφής σε κάποια κλάση. Για να πιστοποιήσουμε ότι μια εγγραφή ανήκει σε μια κλάση θα πρέπει να έχει τιμές μόνο στα συγκεκριμένα πεδία της κλάση και NULL στα υπόλοιπα. Ο έλεγχος όλων των εγγραφών του πίνακα καταχώρησης στηρίζεται στον έλεγχο κάθε εγγραφής, πράγμα το οποίο απαιτεί αρκετά πολύπλοκο και δύσκολο στη συντήρηση κώδικα.
- Δεν απαιτείται *ανακατασκευή* (*restructuring*) του πίνακα για την εύρεση όλης της πληροφορίας. Λόγω του ότι ο πίνακας περιλαμβάνει όλα τα γνωρίσματα όλων των κλάσεων, η ανάκτηση της πληροφορίας ως απάντηση στην SQL ερώτηση SELECT * FROM *ALL_TABLES*, στηρίζεται απλά τη συγκέντρωση όλων των εγγραφών του πίνακα, χωρίς την απαίτηση λειτουργιών συνένωσης.
- Η εισαγωγή/διαγραφή μιας κλάσης οδηγεί σε αλλαγή του σχήματος της βάσης. Εισάγοντας μια νέα κλάση θα πρέπει να εισαχθούν όλα τα γνωρίσματά της ως πεδία του πίνακα, γεγονός που δεν αποτελεί και τόσο εύκολη διαδικασία.

2η μέθοδος: Δημιουργία ενός πίνακα για την υπερκλάση, η οποία λειτουργεί ως βοηθητικός πίνακας αναζήτησης, και από έναν πίνακα για κάθε υποκλάση.

- Αποφεύγονται οι NULL τιμές. Κάθε πίνακας περιλαμβάνει μόνο τα γνωρίσματα της κλάσης του, και κάθε εγγραφή καταχωρείται στον πίνακα της υπερκλάσης και στον αντίστοιχο πίνακα της υποκλάσης όπου ανήκει.
- Για την ανασύνθεση όλης της πληροφορίας των κλάσεων απαιτείται ανακατασκευή ενός πίνακα με όλα τα γνωρίσματα. Η ανακατασκευή πραγματοποιείται με την εφαρμογή της λειτουργίας πλήρους εξωτερικής συνένωσης όλων πινάκων.
- Η αναζήτηση μιας πλειάδας απαιτείται εφαρμογή της λειτουργία της συνένωσης των αντίστοιχων πινάκων.

- Η εισαγωγή/διαγραφή μιας πλειάδας πιθανόν να οδηγεί στην εφαρμογή της λειτουργίας και σε άλλους πίνακες. Διαγράφοντας μια πλειάδα από τον πίνακα μιας υποκλάσης, θα πρέπει να ενημερωθεί και ο πίνακας της υπερκλάσης λόγω ύπαρξης περιορισμού αναφορικής ακεραιότητας. Αν διαγραφεί πλειάδα της υπερκλάσης, η οποία δεν ανήκει σε καμία υποκλάση, τότε δεν πραγματοποιείται προώθηση της διαγραφής.

<u>3η μέθοδος: Δημιουργία ξεχωριστών πινάκων.</u>

- Αποφεύγεται η απαίτηση εφαρμογής των λειτουργιών εισαγωγή/διαγραφή μιας πλειάδας σε περισσότερους του ενός πίνακες. Λόγω μη ύπαρξης περιορισμός αναφορικής ακεραιότητας μεταξύ των πινάκων, κάθε φορά που εφαρμόζεται μια εισαγωγή/διαγραφή μιας πλειάδας ενός πίνακα δεν χρειάζεται να ενημερωθούν και άλλοι πίνακες. Οι λειτουργίες αφορούν μόνο τον αντίστοιχο πίνακα.
- Εύκολη αναζήτηση μιας πλειάδας ή όλης της πληροφορίας μιας κλάσης, εάν είναι γνωστή εκ των προτέρων η κλάση στην οποία θα πρέπει να πραγματοποιηθεί η αναζήτηση. Π.χ., εάν γνωρίζουμε ότι η εγγραφή που αναζητείται ανήκει στην κλάση PACKAGE, τότε πραγματοποιείται η αναζήτηση μόνο στον αντίστοιχο πίνακα. Σε αυτή την περίπτωση παρουσιάζεται σημαντική βελτίωση του χρόνου αναζήτησης.
- Δέσμευση αποθηκευτικού χώρου για επαναλαμβανόμενη πληροφορία. Σε κάθε έναν από τους πίνακες των υποκλάσεων περιλαμβάνονται όλα τα γνωρίσματα της υπερκλάσης, προκαλώντας επανάληψη της πληροφορίας. Η δέσμευση χώρου για τα πεδία αυτά αποτελεί άσκοπη δέσμευση του αποθηκευτικού χώρου.
- Η αναζήτηση μιας συγκεκριμένης πλειάδας είναι δύσκολη. Επειδή οι πίνακες δεν συνδέονται περιορισμούς αναφορικής ακεραιότητας, για την αναζήτηση μιας πληροφορίας θα πρέπει να είναι γνωστός ο πίνακας στο οποίον είναι καταχωρημένη. Ελλείψει αυτής της γνώσης θα πρέπει να ελέγχουν όλοι οι πίνακες μέχρι να βρεθεί η πληροφορία.
- Η ανάκτηση όλης της πληροφορίας είναι δύσκολη. Όπως δείξαμε προηγουμένως, για την ανάκτηση όλης της πληροφορίας θα πρέπει να

πραγματοποιηθεί `UNION` μεταξύ όλων των πινάκων. Ο περιορισμός του `UNION` για συμβατότητα των πεδίων των πινάκων στους οποίους εφαρμόζεται, θα πρέπει να ισχύει σε κάθε περίπτωση.

4η μέθοδος: Ξεχωριστοί πίνακες για κάθε κλάση και ένας βοηθητικός πίνακας.

- Η μέθοδος αυτή αποτελώντας υβρίδιο των μεθόδων 2 και 3, παρουσιάζει το κυριότερο μειονέκτημα της 3ης μεθόδου, τη δέσμευση χώρου για επαναλαμβανόμενη πληροφορία.
- Με την ύπαρξη του βοηθητικού πίνακα διευκολύνεται η αναζήτηση μιας συγκεκριμένης πλειάδας, εφόσον παρέχεται η γνώση του πίνακα που την περιέχει. Επιπλέον, εφόσον τα γνωρίσματα της υπερκλάσης συμπεριλαμβάνονται σε κάθε πίνακα υποκλάσης, είναι δυνατή η εύρεση όλης της πληροφορίας χωρίς την εφαρμογή της συνένωσης.
- Η ανάκτηση όλης της πληροφορίας απαιτεί την εφαρμογή της λειτουργίας `UNION` μεταξύ όλων των πινάκων.

5η μέθοδος: Ξεχωριστοί πίνακες για κάθε υποκλάση και ένας βοηθητικός.

- Απαιτείται μεγάλος αποθηκευτικός χώρος δεδομένου του ότι η πληροφορία επαναλαμβάνεται σε κάθε πίνακα της υποκλάσης.
- Η ανάκτηση μιας συγκεκριμένης πλειάδας πραγματοποιείται εύκολα, εφόσον παρέχεται πληροφορία σχετικά με τον πίνακα καταχώρησης κάθε πλειάδας από τον `DESCRIPTION.`
- Η ανάκτηση όλης της πληροφορίας απαιτεί την εφαρμογή ένωσης των πινάκων των υποκλάσεων και στη συνέχεια εξωτερική συνένωση με τον πίνακα της υπερκλάσης. Για την εφαρμογή της ένωσης θα πρέπει να πρώτα να μετατραπούν οι πίνακες έτσι ώστε να είναι συμβατοί ως προς την πράξη.

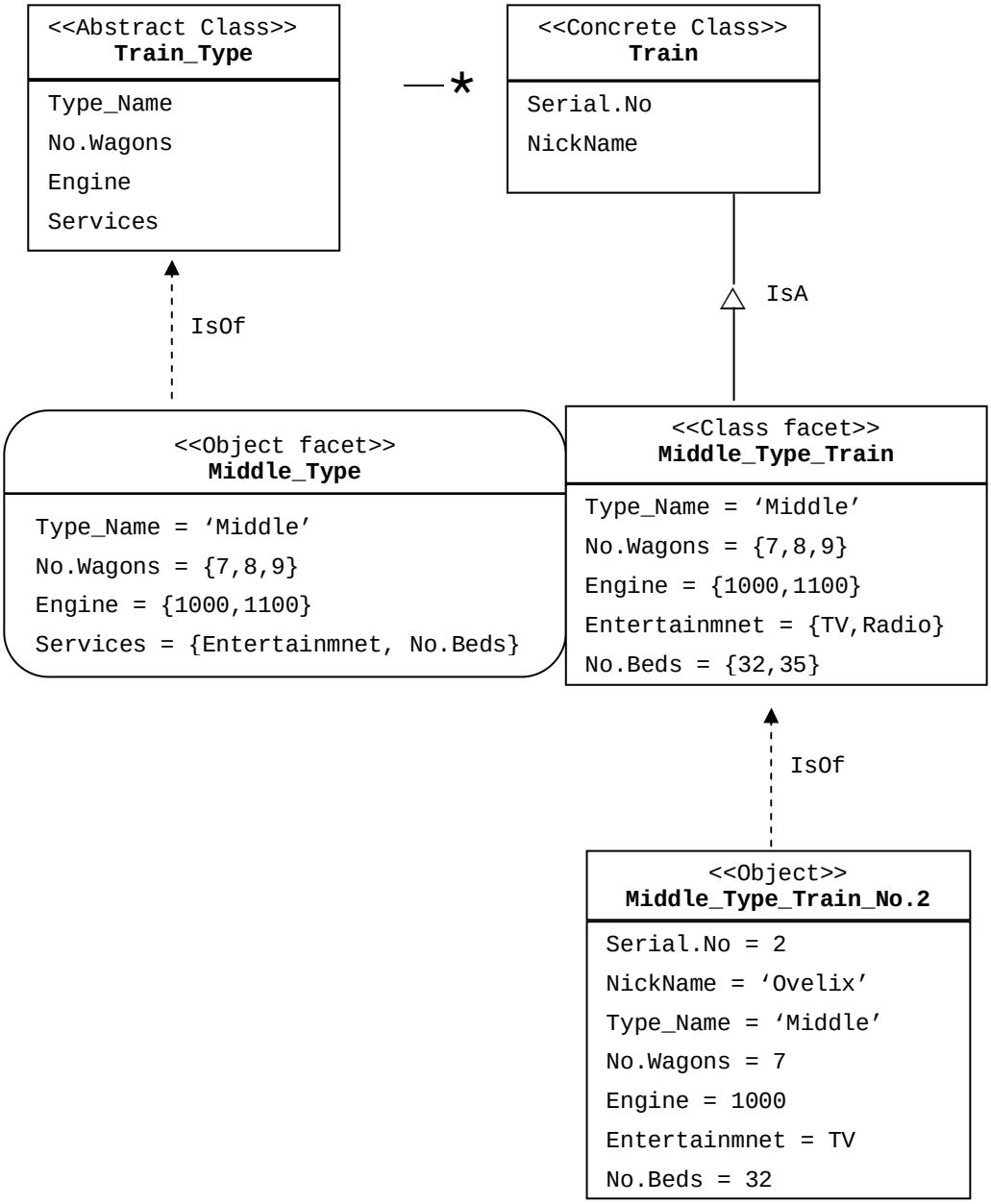
4.8. Materialization

4.8.1. Κίνητρο & Εφαρμογή

Το *materialization* είναι μια δυαδική συσχέτιση ανάμεσα σε μια πολύ γενική (αφαιρετική) σχέση και μια πιο ειδική (συγκεκριμένη). Ο τύπος του materialization, όπως δίδεται στο [3], είναι της μορφής:

`Abstract` — ∗ `Concrete`.

Ας θεωρήσουμε το παράδειγμα μιας σιδηροδρομικής εταιρίας που διαθέτει τρένα. Υπάρχουν τρεις κατηγορίες αμαξοστοιχιών (μεγάλου, μεσαίου και μικρού μεγέθους) και κάθε τρένο ανήκει σε μια κατηγορία. Κάθε κατηγορία έχει ένα εύρος τιμών για το επιτρεπτό μέγεθος συρμού και ένα προκαθορισμένο σύνολο διακριτών τιμών για την ισχύ της μηχανής. Επιπλέον κάθε κατηγορία παρέχει διάφορες υπηρεσίες. Με βάση τη μελέτη που παρουσιάζεται στα [3] και [Dahl98], το σχήμα 4.51 αποτελεί απεικόνιση του παραδείγματος. Το `Type_Middle` είναι στιγμιότυπο της `Train_Type` και αποτελεί αντικείμενο (`object facet`) της κλάσης `Middle_Type_Train`. Η `Middle_Type_Train` καλείται `class facet` και είναι εξειδίκευση της `Concrete` κλάσης. Τέλος η κλάση `Overlie` αποτελεί στιγμιότυπο της `class facet`.



Σχήμα 4.51 Αναπαράσταση του σχήματος του παραδείγματος σε UML, με βάση την προσέγγιση του [2].

Σημαντικός παράγοντας, ο οποίος θα πρέπει να ληφθεί υπόψη πριν την αναπαράσταση του materialization στις σχεσιακές βάσεις δεδομένων, αποτελεί ο τρόπος προώθησης γνωρισμάτων από την Abstract στη Concrete κλάση. Δανειζόμενοι την ορολογία από την UML, κατατάσσουμε τα γνωρίσματα της Abstract κλάσης σε τρεις κατηγορίες:

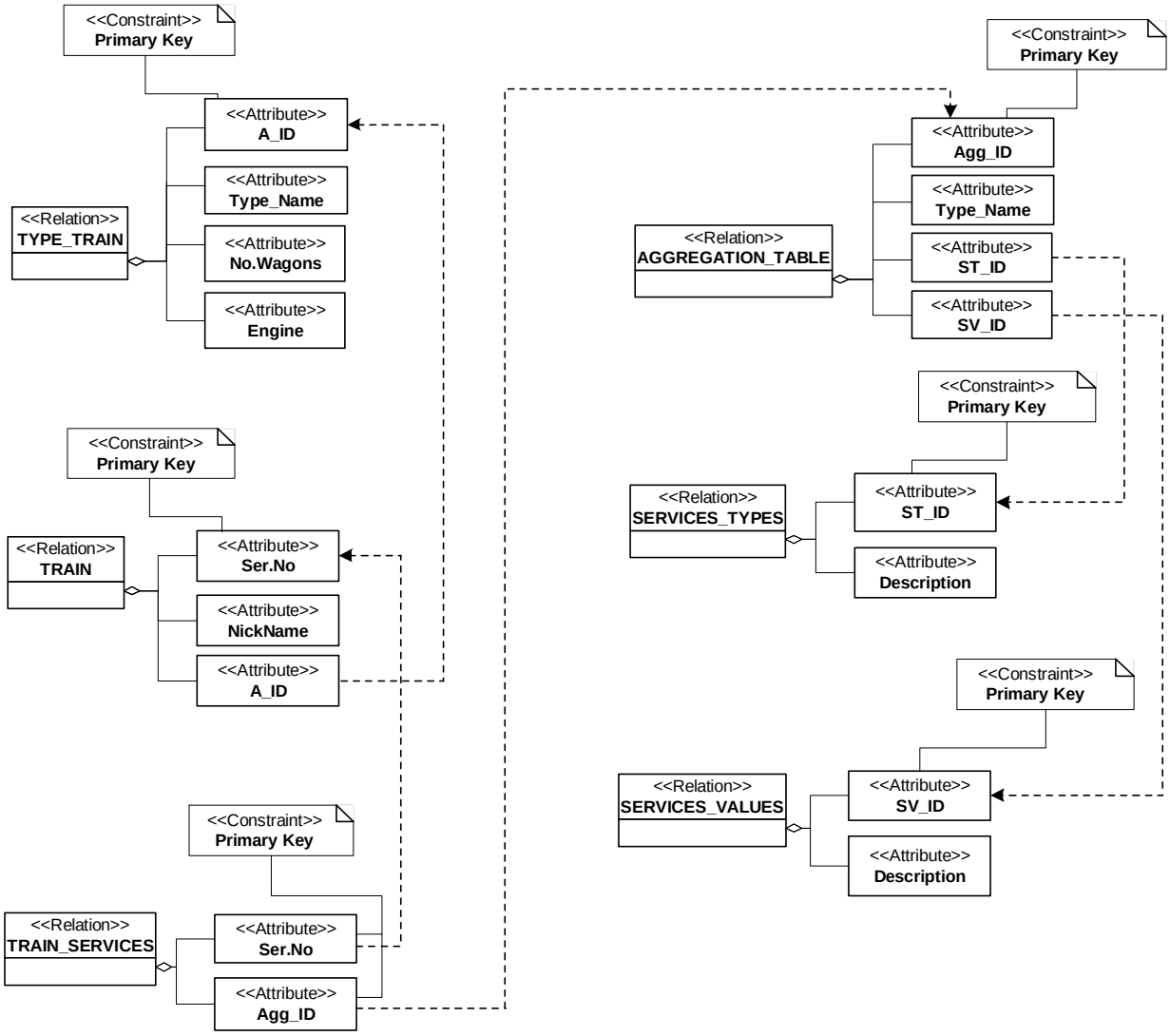
1. *στατικά γνωρίσματα* (*static attributes*). Μόνιμα γνωρίσματα, των οποίων οι τιμές προωθούνται στο object facet, στο class facet και στο στιγμιότυπο της class facet. Π.χ., το γνώρισμα `Type_Name = 'middle'`.
2. *γνωρίσματα πεδίου* (*domain attributes*). Οι τιμές των γνωρισμάτων δίνονται μέσα από ένα σύνολο και προωθούνται (κληρονομικά) από το object facet στο αντίστοιχο class facet. Ένα στιγμιότυπο της class facet λαμβάνει μια από τις παραπάνω τιμές για το συγκεκριμένο πεδίο. Π.χ., το γνώρισμα `No.Wagons` της `Middle_Type_Train` παρέχει τρεις δυνατές τιμές `No.Wagons = {7,8,9}`, ενώ στο στιγμιότυπο της, `Ovelix`, το αντίστοιχο πεδίο λαμβάνει μια τιμή εξ' αυτών `No.Wagons = 7`.
3. *μεταγνωρίσματα* (*meta-attributes*). Με τον όρο μετα-γνώρισμα, αναφερόμαστε σε ένα πεδίο της `Abstract` κλάσης, το οποίο αντιστοιχίζεται σε ένα σύνολο πλειότιμων γνωρισμάτων. Π.χ., το πεδίο `Services`. Κάθε στιγμιότυπο της `Abstract` κλάσης, δηλαδή κάθε object facet, στο πεδίο αυτό έχει ως τιμή ένα σύνολο πλειότιμων γνωρισμάτων. Π.χ., για το object facet, `Middle_Type`, στο πεδίο `Services` αντιστοιχίζεται ένα σύνολο γνωρισμάτων, `Services = {Entertainment, No.Beds}`. Τα γνωρίσματα αυτά είναι πλειότιμα γνωρίσματα. Στο class facet, `Middle_Type_Train`, με βάση τις τιμές του πεδίου `Services` του αντίστοιχου object facet, δημιουργούνται ξεχωριστά πεδία, των οποίων οι τιμές δίδονται μέσω ενός συνόλου τιμών. Π.χ., για στο ξεχωριστό πεδίο `Entertainment` του `Middle_Type_Train`, αντιστοιχίζεται το σύνολο τιμών `Entertainment={TV, Radio}`. Μία εξ' αυτών των τιμών εμφανίζεται ως τιμή στο αντίστοιχο πεδίο ενός στιγμιότυπου της class facet. Δηλαδή, στο `Ovelix` το πεδίο `Entertainment` έχει τιμή `TV`.

Βάσει της παραπάνω μελέτης των γνωρισμάτων που εμφανίζονται στην `Abstract` κλάση, παρουσιάζονται στην επόμενη παράγραφο δυο σχεδιαστικές λύσεις της συσχέτισης materialization.

4.8.2. Δομή

Οι δύο σχεδιαστικές λύσεις που παρουσιάζονται, δημιουργούνται από την διαφορετική προσέγγιση όσον αφορά στην απεικόνιση των μεταγνωρισμάτων.

Στην πρώτη σχεδιαστική λύση, Σχήμα 4.52, κατασκευάζεται ένας πίνακας, TRAIN_TYPE, ο οποίος περιέχει όλες τις κατηγορίες των τρένων. Ο πίνακας αυτός περιλαμβάνει όλα τα γνωρίσματα της Abstract κλάσης, πλην του μεταγνωρίσματος Services. Για το μετα-γνώρισμα κατασκευάζονται δυο πίνακες, ο SERVICES_TYPES και ο SERVICES_VALUES. Στον SERVICES_TYPES παρουσιάζονται όλοι οι τύποι υπηρεσιών που παρέχει κάθε κατηγορία, ενώ στον SERVICES_VALUES εμφανίζονται οι τιμές του κάθε τύπου υπηρεσιών. Για τη συσχέτιση κάθε κατηγορίας τρένου, με τις αντίστοιχες υπηρεσίες που παρέχονται σε αυτή, δημιουργείται ένας πίνακας συνάθροισης, AGGREGATION_TABLE. Ο πίνακας TRAIN, καταχωρεί όλα τα τρένα κάθε κατηγορίας. Η συσχέτιση κάθε τρένου, του πίνακα TRAIN, με τις αντίστοιχες υπηρεσίες, του πίνακα AGGREGATION_TABLE, που παρέχονται σε αυτό, πραγματοποιείται μέσω ενός βοηθητικού πίνακα, TRAIN_SERVICES. Το Σχήμα 4.53 παρουσιάζει ένα στιγμιότυπο της πρώτης σχεδιαστικής λύσης.



Σχήμα 4.52 Αναπαράσταση δομής σχήματος της πρώτης σχεδιαστικής λύσης (*ξεχωριστοί πίνακες των μεταγνωρισμάτων της* `Abstract` *κλάσης και ένας συναθροιστικός βοηθητικός πίνακας*).

TYPE_TRAIN

A_ID	Type_Name	No.Wagons	Engine
…	…	…	…
10	Middle	7	1000
11	Middle	7	1100
12	Middle	8	1000
13	Middle	8	1100
14	Middle	9	1000
14	Middle	9	1100
…	…	…	…

TRAIN

Ser.No	NickName	A_ID
112	Ovelix	10
113	Asterix	14
188	Delton	13

SERVICES_ TYPES

ST_ID	Description
1	Entertainment
2	No.Beds

AGGREGATION_TABLE

Agg_ID	Type_Name	ST_ID	SV_ID
1	Middle	1	1
2	Middle	1	2
3	Middle	2	3
4	Middle	2	4

SERVICES_VALUES

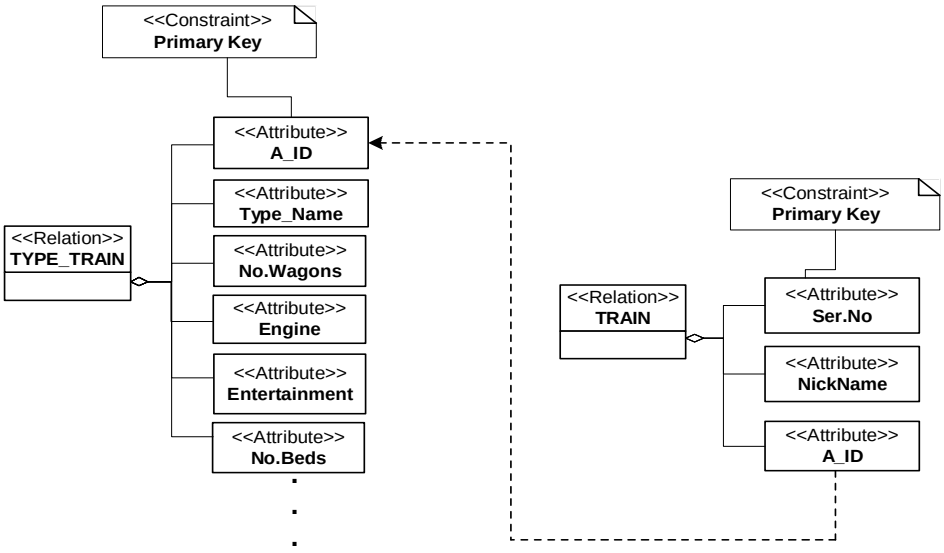
SV_ID	Description
1	TV
2	Radio
3	32
4	35

TRAIN_SERVICES

Ser.No	Agg_ID
112	1
112	3

Σχήμα 4.53 Αναπαράσταση στιγμιότυπων των πινάκων της πρώτης σχεδιαστικής λύσης.

Η δεύτερη σχεδιαστική λύση παρουσιάζει την κατασκευή του πίνακα TRAIN_TYPE, ο οποίος περιλαμβάνει όλα τα γνωρίσματα της Abstract κλάσης. Για τα μεταγνωρίσματα, δημιουργούνται ξεχωριστά πεδία, ανάλογα με τις τιμές που έχει κάθε μεταγνώρισμα στα object facet. Τέλος κατασκευάζεται και ο πίνακας TRAIN, ο οποίος καταχωρεί όλα τα τρένα. Η δομή του σχήματος της δεύτερης σχεδιαστικής λύσης παρουσιάζεται στο Σχήμα 4.54, ενώ στο Σχήμα 4.55 παρουσιάζεται ένα στιγμιότυπό της.



Σχήμα 4.54 Αναπαράσταση δομής του σχήματος της δεύτερης σχεδιαστικής λύσης (*ξεχωριστά πεδία για κάθε τιμή των μεταγνωρισμάτων*).

TYPE_TRAIN

A_ID	**Type_Name**	**No.Wagons**	**Engine**	**Entertainment**	**No.Beds**	**…**
…	…	…	…	…	…	…
100	Middle	7	1000	TV	32	NULL
101	Middle	7	1000	TV	35	NULL
102	Middle	7	1000	Radio	32	NULL
103	Middle	7	1000	Radio	35	NULL
…	…	…	…	…	…	…

TRAIN

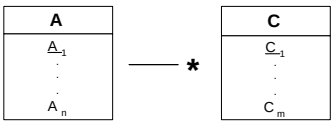
Ser.No	**NickName**	**A_ID**
112	Ovelix	100
113	Asterix	103

Σχήμα 4.55 Αναπαράσταση στιγμιότυπων των πινάκων της δεύτερης σχεδιαστικής λύσης.

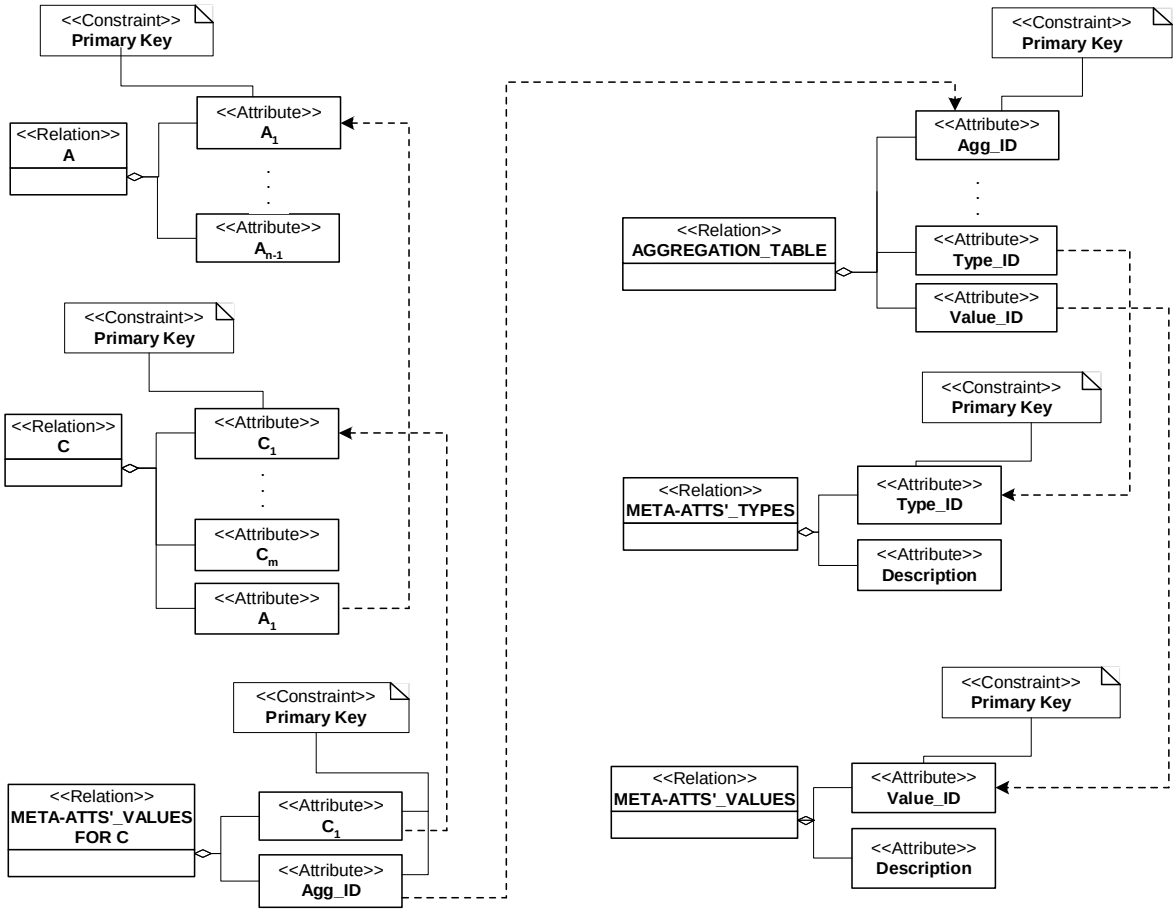
Σε αυτό το σημείο θα πρέπει να αναφέρουμε ότι και στις δύο σχεδιαστικές λύσεις επιλέξαμε την κατασκευή τεχνητού κλειδιού, για να εξασφαλίσουμε την 1NF στα πλειότιμα γνωρίσματα `No.Wagons` και `Engine`. Όπως έχουμε προαναφέρει είναι

δυνατόν να παραβιάσουμε την 1NF με την κατασκευή type constructor ARRAY ή VARRAY (βλ. Ενότητα 4.2 1NF, παράγραφο 4.2.1.5 Κατασκευή).

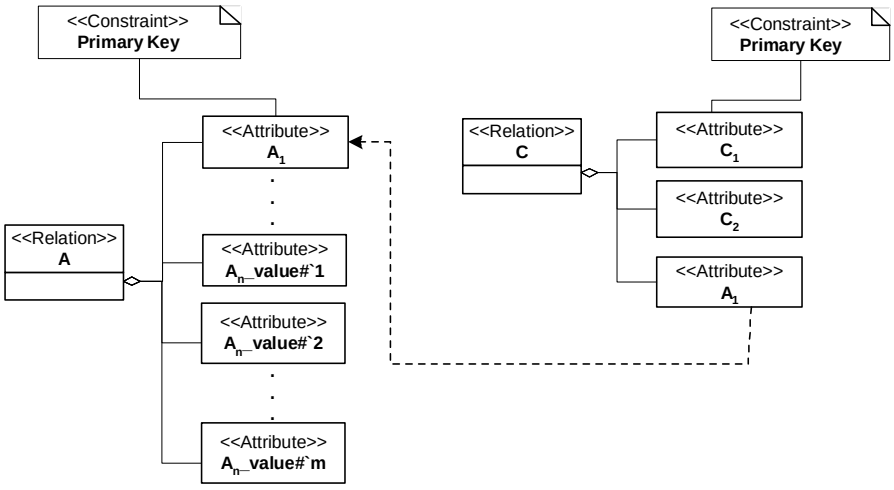
Πριν μελετήσουμε τη συμπεριφορά του προτύπου, ας θεωρήσουμε την γενική περίπτωση του materialization ανάμεσα στις κλάσεις A και C (Σχήμα 4.56), απεικονίζοντας την Abstract κλάση και την Concrete κλάση του materialization, αντίστοιχα. Το πεδίο A_1 αποτελεί πρωτεύον κλειδί, ενώ το πεδίο A_n αποτελεί μεταγνώρισμα της κλάσης A. Το πεδίο C_1 αποτελεί πρωτεύον κλειδί της κλάσης C. Στα Σχήματα 4.57. και 4.58, αναπαρίσταται η δομή του σχήματος της γενικής περίπτωσης με χρήση των δύο σχεδιαστικών λύσεων.



Σχήμα 4.56 Αναπαράσταση σε UML της γενικής περίπτωσης materialization.



Σχήμα 4.57 Αναπαράσταση δομής σχήματος της γενικής περίπτωσης materialization με χρήση της πρώτης σχεδιαστικής λύσης (*ξεχωριστοί πίνακες των μεταγνωρισμάτων της* `Abstract` *κλάσης και ένας συναθροιστικός βοηθητικός πίνακας*) .



Σχήμα 4.58 Αναπαράσταση δομής σχήματος της γενικής περίπτωσης materialization με χρήση της δεύτερης σχεδιαστικής λύσης (*ξεχωριστά πεδία για κάθε τιμή των μεταγνωρισμάτων*).

4.8.3. Συμπεριφορά σε επίπεδο στιγμιότυπου

4.8.3.1. Επιλογή

1η σχεδιαστική λύση: *ξεχωριστοί πίνακες των μεταγνωρισμάτων της `Abstract` κλάσης και ένας συναθροιστικός βοηθητικός πίνακας.* Η αναζήτηση οποιασδήποτε πληροφορίας απαιτεί την εφαρμογή της λειτουργίας της συνένωσης των αντίστοιχων πινάκων. Π.χ., για την αναζήτηση πληροφορίας σχετικά με το τρένο `Ovelix`, θα πρέπει να πραγματοποιηθούν οι παρακάτω πράξεις στη σχεσιακή άλγεβρα:

$$\texttt{RESULT1= TRAIN\_TYPE} \underset{\text{A\_ID = A\_ID}}{\triangleright\!\triangleleft} \texttt{TRAIN}$$

$$\texttt{RESULT2(Agg\_ID)= TRAIN} \underset{\text{Ser.No = Ser.No}}{\triangleright\!\triangleleft} \texttt{TRAIN\_SERVICES}$$

$$\texttt{RESULT3(Service\_Type\_ID,Service\_Value\_ID)= AGGREGATION\_TABLE} \underset{\text{Agg\_ID = Agg\_ID}}{\triangleright\!\triangleleft} \texttt{RESULT2}$$

$$\texttt{RESULT4(Type\_Description)= RESULT3} \underset{\text{Service\_Type\_ID = Service\_Type\_ID}}{\triangleright\!\triangleleft} \texttt{SERVICES\_TYPES}$$

$$RESULT5(Value\_Description) = RESULT3 \triangleright\!\triangleleft_{Service\_Value\_ID = Service\_Value\_ID} SERVICES\_VALUES$$

$$FINAL\_RESULT = RESULT1 \cup (RESULT4 \cup RESULT5)$$

2η σχεδιαστική λύση: *ξεχωριστά πεδία για κάθε τιμή των μεταγνωρισμάτων*. Η αναζήτηση οποιασδήποτε πληροφορίας σχετικά με τις κατηγορίες των τρένων δίδεται άμεσα από τον πίνακα TRAIN_TYPE. Για την αναζήτηση πληροφορίας σχετικά με ένα συγκεκριμένο τρένο, π.χ., το Ovelix, είναι απαραίτητη η ανακατασκευή του πίνακα, η οποία πραγματοποιείται με την βοήθεια των λειτουργιών της συνένωσης των πινάκων TRAIN_TYPE και TRAIN.

4.8.3.2. Εισαγωγή, Διαγραφή, Ανανέωση

Στην 1η σχεδιαστική λύση, όπου υπάρχουν ξεχωριστοί πίνακες των μεταγνωρισμάτων της Abstract κλάσης και ένας συναθροιστικός βοηθητικός πίνακας, η εισαγωγή/διαγραφή μιας πλειάδας σε κάποιον από τους πίνακες οδηγεί στην εισαγωγή/διαγραφή των αντίστοιχων πλειάδων και στους πίνακες στους οποίους συμμετέχει ο πίνακας μέσω αναφοράς ξένου κλειδιού. Π.χ., η εισαγωγή μιας νέας κατηγορίας τρένων (ουσιαστικά εισαγωγή πολλών πλειάδων), προκαλεί την ενημέρωση των πινάκων TRAIN_TYPE, AGGREGATION_TABLE και πιθανόν και των πινάκων SERVICES_TYPES και SERVICES_VALUES, εάν εισάγονται νέοι τύπου υπηρεσιών για τη συγκεκριμένη κατηγορία. Αντίστοιχα, η διαγραφή μιας κατηγορίας, οδηγεί στην ενημέρωση όλων των πινάκων. Η ανανέωση μιας πλειάδας σημαίνει διαγραφή της και επαναεισαγωγή της.

Στη 2η σχεδιαστική λύση, με ξεχωριστά πεδία για κάθε τιμή των μεταγνωρισμάτων), η εισαγωγή ή διαγραφή μιας πλειάδας στον πίνακα TRAIN δεν προκαλεί καμία αλλαγή στις πλειάδες του πίνακα TRAIN_TYPE. Αντίθετα η διαγραφή μιας πλειάδας του πίνακα TRAIN_TYPE, προκαλεί την διαγραφή όλων των πλειάδων του TRAIN που αντιστοιχίζονται σε αυτή, μέσω της αναφοράς ξένου κλειδιού. Η ανανέωση της τιμής

ενός πεδίου, πλην του πρωτεύοντος κλειδιού, και στους δύο πίνακες δεν επηρεάζει τον άλλον πίνακα.

4.8.4. Συμπεριφορά σε επίπεδο σχήματος

Η εισαγωγή ή διαγραφή οποιουδήποτε πεδίου ενός πίνακα, πλην αυτών που συμμετέχουν και σε άλλους πίνακες (π.χ., πρωτεύοντα κλειδιά και ξένα κλειδιά), δεν μεταβάλλουν τη συμπεριφορά του προτύπου σε επίπεδο σχήματος, στην περίπτωση όπου επιλέγεται η κατασκευή ξεχωριστών πίνακων για τα μεταγνωρίσματα της `Abstract` κλάσης και ενός συναθροιστικού πίνακας.

Αν και στην πρώτη σχεδιαστική λύση μπορούμε εύκολα να εισάγουμε ή να διαγράψουμε μια κατηγορία τρένων, στη δεύτερη λύση, όπου κατασκευάζονται ξεχωριστά πεδία για κάθε τιμή των μεταγνωρισμάτων, μια τέτοια λειτουργία επιφέρει αλλαγή στο σχήμα του προτύπου. Π.χ., για την διαγραφή μιας κατηγορίας, εκτός της διαγραφής των αντίστοιχων πλειάδων, θα πρέπει να διαγράψουμε από τον πίνακα `TRAIN_TYPE` τα πεδία των `Services` που παρέχει αυτή η κατηγορία. Π.χ., εάν θέλουμε να διαγράψουμε την μεσαία κατηγορία, θα πρέπει να διαγραφτούν και τα πεδία `Entertainment` και `No.Beds`. Στην περίπτωση τα γνωρίσματα αυτά αποτελούν μέρος και άλλων κατηγοριών, τότε η διαγραφή τους αποφεύγεται. Δηλαδή, εάν το πεδίο `Entertainment` αποτελεί γνώρισμα εκτός της μεσαίας κατηγορίας και της μεγάλης κατηγορίας τρένων, τότε δεν είναι δυνατή τη διαγραφή του γνωρίσματος.

4.8.5. Πλεονεκτήματα & Μειονεκτήματα

1η σχεδιαστική λύση (*ξεχωριστοί πίνακες των μεταγνωρισμάτων της* `Abstract` *κλάσης και ένας συναθροιστικός βοηθητικός πίνακας*):

- Η αναζήτηση οποιασδήποτε πληροφορίας απαιτεί την εφαρμογή πολλών συνενώσεων. Π.χ., στην περίπτωση αναζήτησης πληροφορίας σχετικά με ένα τρένο, όπως παρουσιάστηκε στην παράγραφο 4.8.3.1, θα πρέπει να συνενωθούν όλοι οι πίνακες καταχώρησης. Υπενθυμίζεται ότι η συνένωση αποτελεί την πιο «ακριβή» λειτουργία, ως προς το χρόνο εκτέλεσής της.

- Η εισαγωγή/διαγραφή/ανανέωση μιας πλειάδας ενός πίνακα, οδηγεί στην ενημέρωση των πινάκων με τους οποίους συσχετίζεται ο πίνακας.
- Η εισαγωγή/διαγραφή/ανανέωση μιας κατηγορίας πραγματοποιείται απλά με την εισαγωγή/διαγραφή των αντίστοιχων πλειάδων στους αντίστοιχους πίνακες, χωρίς να επηρεάζει το σχήμα του προτύπου.

2η σχεδιαστική λύση (*ξεχωριστά πεδία για κάθε τιμή των μεταγνωρισμάτων*):

- Η αναζήτηση μιας πληροφορίας, στη χειρότερη περίπτωση, απαιτεί τη συνένωση των δύο πινάκων. Χειρότερη περίπτωση αναζήτησης αποτελεί η αναζήτηση πληροφορίας σχετικά με κάποιο τρένο.
- Εμφάνιση NULL τιμών. Εφόσον ο πίνακας περιλαμβάνει ως πεδία όλους τους τύπους υπηρεσιών που παρέχονται από κάθε κατηγορία τρένου, είναι προφανές ότι στα πεδία τα οποία δεν ανήκουν σε κάποια κατηγορία θα εμφανίζονται NULL τιμές.
- Η εισαγωγή ή διαγραφή μιας κατηγορίας οδηγεί στη αλλαγή του σχήματος του προτύπου.

4.9. Configuration

4.9.1. Κίνητρο & Εφαρμογή

Ας υποθέσουμε ότι ζητείται η μηχανογράφηση μιας γραμματείας ενός Τμήματος Πανεπιστημίου και συγκεκριμένα το κομμάτι που αφορά την εγγραφή φοιτητών σε μαθήματα. Κάθε φοιτητής χαρακτηρίζεται από κωδικό, όνομα και έτος εγγραφής. Κάθε μάθημα χαρακτηρίζεται από κωδικό, όνομα και διδασκόμενο βιβλίο. Το βασικό ενδιαφέρον μας είναι να μπορέσουμε να καταγράψουμε τα μαθήματα στα οποία πρέπει/μπορεί να εγγραφεί ένας φοιτητής. Τα μαθήματα που διατίθενται, καθώς και οι απαιτήσεις (μαθήματα που πρέπει να πάρει ένας φοιτητής) για τη λήψη του πτυχίου, αλλάζουν με την πάροδο του χρόνου. Π.χ., για τους φοιτητές που εγράφησαν το 1990, θα πρέπει για την λήψη πτυχίου τους να πάρουν τα μαθήματα Y_1, Y_2, Y_3 υποχρεωτικά και 2 εκ των E_1, E_2, E_3, E_4, E_5. Το 1995 αλλάζει ο οδηγός σπουδών και απαιτούνται

για την λήψη πτυχίου τα υποχρεωτικά μαθήματα Y_1, Y_2, Y_4 και 3 εκ των E_2, E_3, E_4, E_5.

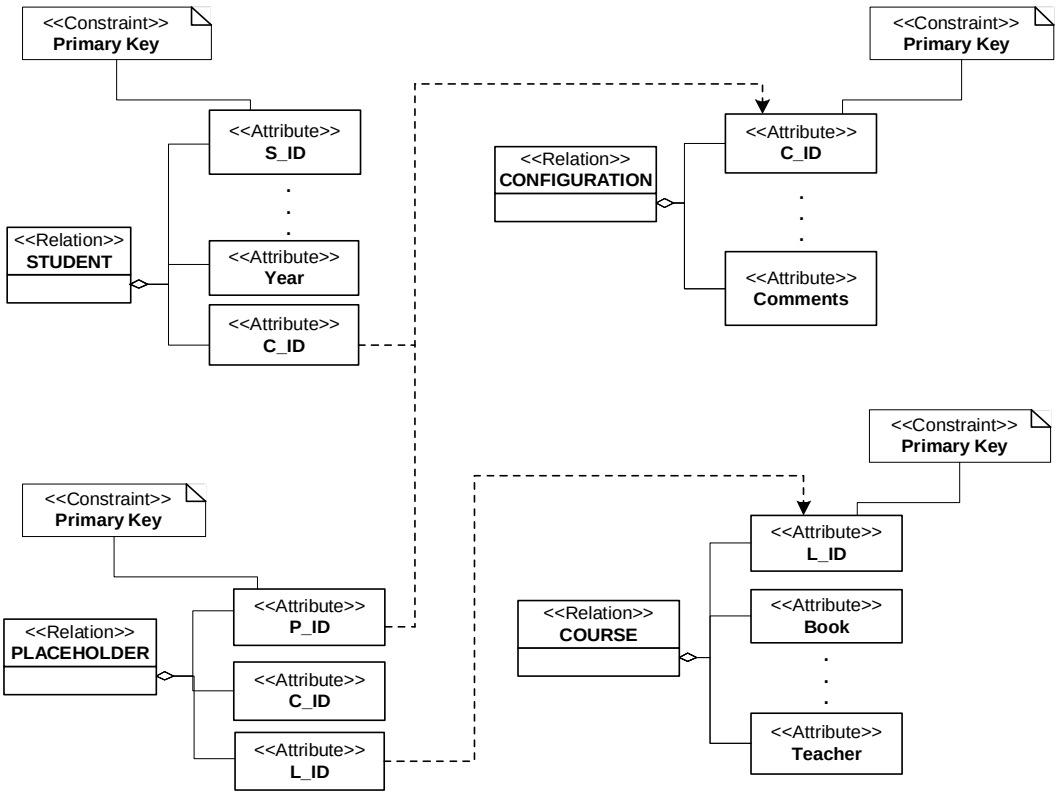
Ζητούμενο, λοιπόν, αποτελεί η σχεδίαση ενός σχεσιακού σχήματος το οποίο:

1. να χωρίζει του φοιτητές σε ομάδες ανάλογα με το χρόνο εγγραφής τους,
2. να αντιστοιχίζει σε κάθε μια ομάδα ένα πρόγραμμα σπουδών,
3. και κυρίως να είναι «ανθεκτικό» στην εξέλιξη των κανόνων στο χρόνο, δηλαδή να καταγράφονται όλα τα διαφορετικά προγράμματα σπουδών.

Το πρότυπο *Configuration* αποτελεί τη σχεδίαση του σχεσιακού σχήματος ενός προβλήματος, όπου οι κανόνες μεταβάλλονται με το χρόνο δημιουργώντας μεταβλητό πλήθος γνωρισμάτων σε έναν πίνακα, π.χ., τα διαφορετικά μαθήματα κάθε οδηγού σπουδών.

4.9.2. Δομή

Για τη σχηματική απεικόνιση του προτύπου, κατασκευάζεται ένας πίνακας, `CONFIGURATION`, στον οποίον καταχωρούνται οι διαφορετικές ομάδες κατάταξης. Στον πίνακα `PLACEHOLDER` καταχωρούνται όλες οι ιδιότητες κάθε ομάδας κατάταξης. Π.χ., για το παράδειγμα της γραμματείας, στον `CONFIGURATION` καταχωρείται το πλήθος των διαφορετικών οδηγών, ενώ στο `PLACEHOLDER` καταχωρούνται όλα τα μαθήματα (με την μορφή κωδικών) που αντιστοιχούν σε κάθε ένα πρόγραμμα σπουδών. Για τη συσχέτισης των σχέσεων `STUDENT` και `CONFIGURATION`, μπορούμε είτε να εισάγουμε το γνώρισμα `Year` στον πίνακα `CONFIGURATION`, είτε να εισάγουμε το πρωτεύον κλειδί της `CONFIGURATION` στον πίνακα `STUDENT`, με αναφορά ξένου κλειδιού. Εμείς επιλέξαμε την δεύτερη λύση, δηλαδή να εισάγουμε το `C_ID` του `CONFIGURATION` στον πίνακα `STUDENT`. Στο Σχήμα 4.59 παρουσιάζεται η δομή του σχήματος του προτύπου Configuration για το παράδειγμα της γραμματείας, ενώ στο Σχήμα 4.60 παρουσιάζεται ένα στιγμιότυπό του.



Σχήμα 4.59 Αναπαράσταση δομής σχήματος του προτύπου Configuration για το παράδειγμα της γραμματείας ενός τμήματος πανεπιστημίου.

STUDENT

S_ID	…	Year	C_ID
100	…	1990	1
200	…	1995	2
300	…	1997	2
400	…	1998	3

CONFIGURATION

C_ID	…	Comments
1	…	Obligatory: Y_1, Y_2, Y_3 and 2 of E_1, E_2, E_3, E_4, E_5
2	…	Obligatory: Y_1, Y_2, Y_4 and 3 of E_2, E_3, E_4, E_5
3	…	Switch: E_3 to Y_2

PLACEHOLDER

P_ID	C_ID	L_ID
1	1	Y_1
2	1	Y_2
3	1	Y_3
4	1	E_1
5	1	E_2
6	1	E_3
7	1	E_4
8	1	E_5
…	…	…

COURSE

L_ID	…	Teacher
Y_1	…	Tom
Y_2	…	John
Y_3	…	Peter
E_1	…	Mary
E_2		Tom
…	…	…

Σχήμα 4.60 Αναπαράσταση στιγμιότυπου του προτύπου Configuration για το παράδειγμα της γραμματείας ενός τμήματος πανεπιστημίου.

4.9.3. Συμπεριφορά σε επίπεδο στιγμιότυπου

4.9.3.1. Επιλογή

Η αναζήτηση πληροφορίας σχετικά με το πρόγραμμα σπουδών ενός φοιτητή ανάλογα με το χρόνο εισαγωγής του, δίδεται με την εφαρμογή της λειτουργίας της συνένωσης των πινάκων STUDENT και PLACEHOLDER με βάση το πεδίο C_ID. Η SQL ερώτηση τίθεται ως εξής:

```
SELECT S_ID, …,Year, L_ID,… ,Teacher
FROM STUDENT S, COURSE C ,PLACEHOLDER P
WHERE S.S_ID=term AND S.C_ID = P.C_ID AND P.L_ID = C.L_ID.
```

Η επιλογή όλων των προγραμμάτων σπουδών και των μαθημάτων που αντιστοιχούν σε κάθε ένα από αυτά, δίδεται άμεσα από τον πίνακα PLACEHOLDER.

4.9.3.2. Εισαγωγή, Διαγραφή και Ανανέωση εγγραφής

Η εισαγωγή/διαγραφή μιας πλειάδας στον πίνακα CONFIGURATION οδηγεί στην «προώθηση» της λειτουργίας στους πίνακες PLACEHOLDER και STUDENT, λόγω ύπαρξης περιορισμού αναφορικής ακεραιότητας. Π.χ., ας θεωρήσουμε ότι διαγράφεται η πλειάδα (2,1995, ...) από τον πίνακα CONFIGURATION, τότε θα πρέπει να διαγραφούν όλες οι πλειάδες των πινάκων PLACEHOLDER και STUDENT με C_ID = 2. Τέλος η ανανέωση οποιασδήποτε τιμής πεδίου, πλην του πρωτεύοντος κλειδιού, του πίνακα CONFIGURATION δεν μεταβάλλει τη συμπεριφορά του προτύπου σε επίπεδο στιγμιότυπου.

Η εισαγωγή/διαγραφή/ανανέωση μιας πλειάδας του πίνακα COURSE, προκαλεί την εισαγωγή μιας νέας πλειάδας στον πίνακα CONFIGURATION. Π.χ., εάν εισαχθεί κάποιο νέο μάθημα, αυτό έχει ως συνέπεια την δημιουργία ενός νέου οδηγού σπουδών που να το περιλαμβάνει. Η ανανέωση οποιουδήποτε άλλου πεδίου της σχέσης COURSE, πλην του L_ID δεν επηρεάζει τη συμπεριφορά του προτύπου σε επίπεδο στιγμιότυπου.

Τέλος, η εισαγωγή/διαγραφή/ανανέωση μιας πλειάδας του πίνακα STUDENT δεν επηρεάζει τη συμπεριφορά του προτύπου. Ακόμα και στην περίπτωση όπου διαγράφεται και η τελευταία πλειάδα με C_ID = ‘1’, δεν «απαιτείται» η διαγραφή της αντίστοιχης πλειάδας του Configuration.

4.9.4. Συμπεριφορά σε επίπεδο σχήματος

Σε επίπεδο σχήματος είναι δυνατή η εφαρμογή εισαγωγής/διαγραφής ενός πεδίου σε όλους τους πίνακες, πλην των πεδίων του πρωτεύοντος κλειδιού κάθε πίνακα και των πεδίων που αποτελούν αναφορές ξένου κλειδιού. Σε αυτή την περίπτωση η συμπεριφορά του προτύπου δεν μεταβάλλεται.

4.9.5. Πλεονεκτήματα & Μειονεκτήματα

Το πρότυπο Configuration παρουσιάζει τις εξής ιδιότητες:

- χωρίζει ένα σύνολο στοιχείων σε ξένα μεταξύ τους υποσύνολα, όπου το καθένα περιλαμβάνει τους δικούς του κανόνες.
- αποτελεί μια δομή κατάταξης των υποσυνόλων, «ανθεκτική» στις αλλαγές των κανόνων μέσα στο χρόνο. Η δομή αυτή καταχωρεί όλους τους κανόνες που ισχύουν για κάθε υποσύνολο χωρίς να τους μεταβάλει, ούτε να τους παύει.
- η αναζήτηση μιας πληροφορίας σχετικά με τους κανόνες που ισχύουν σε κάθε υποσύνολο, απαιτεί τη συνένωση των αντίστοιχων πινάκων. Υπενθυμίζεται ότι η λειτουργία της συνένωσης έχει αρκετά μεγάλο υπολογιστικό κόστος.
- ο πίνακας, στον οποίο αποθηκεύονται οι κανόνες που ισχύουν σε κάθε υποσύνολο κατάταξης, εκτείνεται σε βάθος. Π.χ., ο πίνακας PLACEHOLDER, στον οποίον καταχωρούνται τα μαθήματα κάθε οδηγού σπουδών, περιλαμβάνει πολλές πλειάδες.

ΚΕΦΑΛΑΙΟ 5. ΜΕΤΡΙΚΕΣ ΠΟΙΟΤΗΤΑΣ ΛΟΓΙΣΜΙΚΟΥ

Για την αξιολόγηση των σχεδιαστικών λύσεων που προτείνονται για κάθε πρότυπο στηριζόμαστε, κυρίως, στις *μετρικές* (*metrics*), που ορίζονται στο [1]. Σκοπός της αξιολόγησης είναι η αποτίμηση του κόστους σχεδίασης και διατήρησης μιας Βάσης Δεδομένων ανάλογα με την εκάστοτε σχεδιαστική λύση. Επιπλέον, με την βοήθεια των μετρικών, είναι κανείς σε θέση να συγκρίνει τις υπάρχουσες, αλλά και τις μελλοντικές, λύσεις ενός προτύπου. Οι μετρικές που χρησιμοποιούνται εδώ είναι: το *μέγεθος* (*size*), η *συνδεσιμότητα* (*coupling*), η *συνεκτικότητα* (*cohesion*) και το *μέγεθος στοιχείων* (*data volume*). Στην ενότητα 5.1 παρουσιάζεται η τεχνική της μοντελοποίησης, στην ενότητα 5.2. ορίζονται οι μετρικές, με τις οποίες θα πραγματοποιηθούν τα πειράματα, ενώ στην ενότητα 5.3 παρουσιάζεται η πειραματική μελέτη των προτύπων.

5.1. Τεχνική μοντελοποίησης

Στο [1], ορίζεται ένα *σύστημα* (*system), S,* ως ένας γράφος της μορφής `S=(E,R)`, όπου `E` είναι το σύνολο των κόμβων του, και `R` το σύνολο των ακμών μεταξύ των κόμβων. Μία *μονάδα* (*module*), *m,* είναι ένα υποσύνολο των κόμβων του συστήματος. Γενικά, οι μονάδες μεταξύ τους μπορεί να είναι επικαλυπτόμενες, δηλαδή ένας κόμβος να ανήκει σε δύο μονάδες. Στην περίπτωση όπου οι μονάδες διαμερίζουν τους κόμβους ενός συστήματος, δηλαδή οι μονάδες του συστήματος δεν παρουσιάζουν επικάλυψη,

το σύστημα ονομάζεται *σύστημα μονάδων* (*modular system*), `MS`. Τέλος, οι ακμές χωρίζονται σε δύο κατηγορίες: i) σε *intermodule* ακμές, όπου οι ακμές συνδέουν κόμβους διαφορετικών μονάδων, και ii) σε *intramodule* ακμές, όπου οι ακμές εφαρμόζονται ανάμεσα σε κόμβους της ίδιας μονάδας. Οι intermodule ακμές μιας μονάδας κατηγοριοποιούνται, επιπλέον, σε: i) *εισερχόμενες* (*incoming*), όταν η ακμή προσπίπτει σε κόμβο της μονάδας, και ii) *εξερχόμενες* (*outgoing*), όταν η ακμή ξεκινά από κόμβο της μονάδας.

Με βάση τους παραπάνω ορισμούς και προσανατολιζόμενοι στην περίπτωση σχεδίασης προτύπων, μια Βάση Δεδομένων αποτελεί ένα σύστημα, *S*, το οποίο αναπαρίσταται γραφικά μέσω ενός γράφου `S=(E,R)`. Κόμβοι, `E`, του συστήματος είναι:

i. Οι σχέσεις.
ii. Τα γνωρίσματα των σχέσεων.
iii. Οι περιορισμοί που τίθενται κατά τη σχεδίαση, όπως περιορισμός πρωτεύοντος κλειδιού (`PK`), περιορισμός αναφορικής ακεραιότητας (`FK`), περιορισμός μοναδικότητας (`Unique Value-UV`), περιορισμός μη μηδενική τιμή (`Not Null-NN`), κ.τ.λ..
iv. Οι `SQL` επερωτήσεις (`Q`), που θέτονται προς τη βάση.
v. Οι όψεις (`Materialized Views- MV`, και `Virtual Views -VV`), που τυχόν δημιουργούνται για διευκόλυνση της αναζήτησης.
vi. Οι αποθηκευμένες διαδικασίες (`Stored Procedure-SP`), αποτελούμενες από `PL/SQL` δηλώσεις.
vii. Οι συναρτησιακές εξαρτήσεις (`functional dependencies-fd`), που εμφανίζονται μεταξύ των γνωρισμάτων μιας σχέσης παραβιάζοντας την 3NF.

Οι ακμές, `R`, του γράφου αναπαριστούν τις συσχετίσεις που υπάρχουν μεταξύ των κόμβων του συστήματος. Η κατεύθυνση μιας ακμής δηλώνει τον προσδιορισμό (ή την εξάρτηση) του κόμβου στον οποίο προσπίπτει από τον κόμβο από τον οποίο ξεκινά. Οι συσχετίσεις που εμφανίζονται κατά τη σχεδίαση των προτύπων είναι:

i. Συσχετίσεις σχήματος, αναπαριστώντας τη συσχέτιση μεταξύ μιας σχέσης και των πεδίων της.

ii. Συσχετίσεις περιορισμών, αναπαριστώντας τις συσχετίσεις μεταξύ των πεδίων και των περιορισμών που τίθενται σ' αυτά. Οι ακμές που αναπαριστούν τις συσχετίσεις περιορισμών, ξεκινούν από τους κόμβους των περιορισμών και καταλήγουν στους κόμβους των γνωρισμάτων, στους οποίους εφαρμόζονται. Στην περίπτωσης ύπαρξης αναφορικής ακεραιότητας, ένα γνώρισμα μιας σχέσης προσδιορίζει κάποιο γνώρισμα μιας άλλης σχέσης. Σχηματικά, η ύπαρξη ξένου κλειδιού αναπαριστάται με την δημιουργία ενός μονοπατιού. Το μονοπάτι ξεκινά από τον κόμβο του γνωρίσματος που προσδιορίζει και καταλήγει στον κόμβο του γνωρίσματος που προσδιορίζεται. Ο κόμβος του περιορισμού αποτελεί τον ενδιάμεσο κόμβο του μονοπατιού.

iii. Συσχετίσεις ερωτήσεων, όπου απεικονίζονται οι εξαρτήσεις μιας ερώτησης από τα γνωρίσματα των σχέσεων της βάσης. Μια SQL ερώτηση εξαρτάται από τα πεδία τα οποία εμφανίζονται στις γραμμές `SELECT`, `FROM`, `WHERE`, `GROUP BY`, και `HAVING`, οπότε κάθε ακμή ξεκινά από έναν κόμβο γνωρίσματος μιας σχέσης και προσπίπτει στο κόμβο της ερώτησης. Στην περίπτωση, όπου στη γραμμή `SELECT` εμφανίζονται όλα τα γνωρίσματα της σχέσης, τότε για λόγους απλότητας, ο κόμβος της ερώτησης συνδέεται απευθείας με το κόμβο της σχέσης. Τέλος, στην περίπτωση όπου μια ερώτηση ενημερώνει έναν πίνακα, εισάγοντας ή διαγράφοντας πλειάδες του, η συσχέτιση αναπαρίσταται με μία ακμή από τον κόμβο της ερώτησης προς τον κόμβο του πίνακα.

iv. Συσχετίσεις όψεων, οι οποίες απεικονίζουν τις εξαρτήσεις μιας όψης από τα γνωρίσματα των σχέσεων της βάσης. Οι συσχετίσεις όψεων και ερωτήσεων δεν διαφοροποιούνται πουθενά, δεδομένου ότι οι όψεις αποτελούνται από SQL ερωτήσεις. Στη συνέχεια του κεφαλαίου δεν θα γίνεται διαχωρισμός μεταξύ τους, εκτός αν είναι απαραίτητο.

v. Συσχετίσεις διαδικασίας, όπου συνδέει μια stored procedure με τους πίνακες, οι οποίοι εμφανίζονται στο σώμα της, εισάγοντας, διαγράφοντας ή διαβάζοντας εγγραφές από αυτούς. Οι συσχετίσεις διαδικασίας, επί της ουσίας, είναι ίδιες με αυτές των ερωτήσεων. Η μόνη διαφορά που παρουσιάζεται, και γι' αυτό διαφοροποιούνται οι δύο περιπτώσεις, είναι ότι η διαδικασία δέχεται παραμέτρους εισόδου. Οι ακμές, που παρουσιάζουν τις παραμέτρους εισόδου και εξόδου μιας διαδικασίας θα μπορούσαν να απεικονιστούν με τη σημείωση

των λέξεων (IN, OUT), αντίστοιχα, πάνω στις ακμές. Στη μελέτη που πραγματοποιείται εδώ οι διαδικασίες δεν έχουν παραμέτρους.

vi. Συσχετίσεις εξάρτησης, οι οποίες συνδέουν μια συνάρτηση με τα πεδία που μετέχουν σ' αυτή. Οι κόμβοι των γνωρισμάτων, τα οποία προσδιορίζουν ένα γνώρισμα, ενώνονται με ακμές που προσπίπτουν στον κόμβο της εξάρτησης. Ο κόμβος του γνωρίσματος που προσδιορίζεται, συνδέεται μέσω μιας ακμής που ξεκινά τον κόμβο της εξάρτησης και καταλήγει σε αυτόν.

Στο Σχήμα 5.1 δίδεται η γραφική αναπαράσταση των παραπάνω συσχετίσεων.

Για χάριν απλότητας των σχημάτων, η σχηματική αναπαράσταση οποιασδήποτε ερώτησης, όψης, καθώς και διαδικασίας, θα πραγματοποιείται με την απεικόνιση ενός μόνο κόμβου για κάθε περίπτωση.

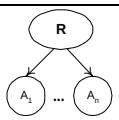
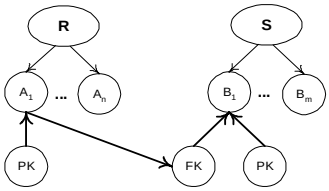
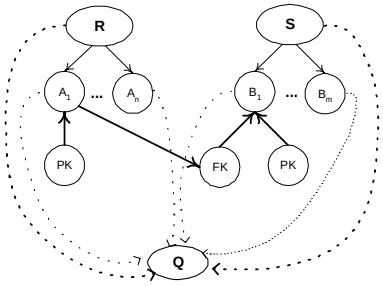
Βάσει των παραπάνω ορισμών, μια μονάδα ενός συστήματος απεικόνισης μιας βάσης δεδομένων, μπορεί να είναι:

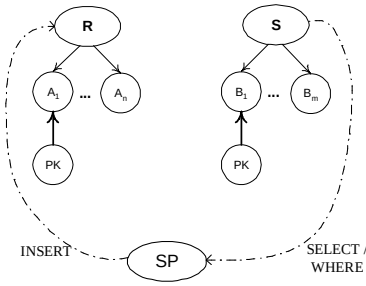
i. Μια σχέση, συμπεριλαμβανομένων των πεδίων της, των περιορισμών και των εξαρτήσεων, που πιθανόν εμφανίζονται μεταξύ των πεδίων της.
ii. Μια ερώτηση, αποκρύπτοντας τη λεπτομερή αναπαράστασή της.
iii. Μια όψη, αποκρύπτοντας τη λεπτομερή αναπαράστασή της.
iv. Μια stored procedure, αποκρύπτοντας τη λεπτομερή αναπαράστασή της.

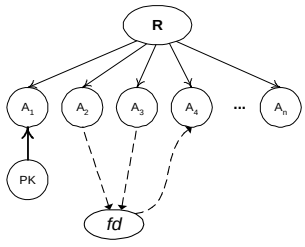
Σε αυτό το σημείο θα πρέπει να ορίσουμε το *σύστημα προτύπου* (*pattern's system*), *PS*, ως ένα σύστημα μονάδων που αποτελείται από μονάδες σχέσεων και διαδικασιών, οι οποίες εμφανίζονται κατά τη σχεδίαση ενός προτύπου. Διαισθητικά, το σύστημα προτύπου απεικονίζει το σχήμα της βάσης δεδομένων. Το σύστημα προτύπου, σε συνδυασμό με τις ερωτήσεις που θέτονται σε αυτό, αποτελούν το συνολικό σύστημα S. Στη συγκεκριμένη εργασία, ομαδοποιούνται οι μονάδες σε συστήματα μονάδων, μόνο εάν οι μονάδες αυτές επιτελούν μια συγκεκριμένη λειτουργία, ως μέρος του συστήματος προτύπου.

Τέλος ορίζουμε την πράξη της *ένωσης* (*union*) δύο μονάδων. Ας θεωρήσουμε δύο μονάδες m1=(Em1,Rm1) και m2=(Em2,Rm2). Ενώνοντας τις δύο μονάδες, προκύπτει μια νέα μονάδα m'=(Em',Rm'), με Em'=Em1∪Em2 και Rm'=Rm1∪Rm2. Οι τυχόν

εξωτερικές ακμές από την μια μονάδα προς την άλλη θεωρούνται πλέον εσωτερικές ακμές της παραγόμενης μονάδας `m'`. Οι υπόλοιπες εξωτερικές ακμές των `m1`, `m2` προς άλλες μονάδες του συστήματος, διατηρούνται ως έχουν. Για την περίπτωση ενός συστήματος Βάσης Δεδομένων, η ένωση δύο μονάδων (για κάθε τύπου μονάδας, όπως αυτοί έχουν οριστεί προηγουμένως) οδηγεί στη δημιουργία: i) συστημάτων μονάδων, ii) του συστήματος προτύπου, ή/και iii) του συστήματος.

	Ακμές σχήματος (σχηματική αναπαράσταση με απλές ακμές) μεταξύ της σχέσης R και των πεδίων της A_1 ,…, A_n. Η κατεύθυνση των ακμών δηλώνει ότι η ύπαρξη των γνωρισμάτων εξαρτάται από την ύπαρξη του πίνακα.
	Ακμές αντιστοίχισης (σχηματική αναπαράσταση με έντονες ακμές) μεταξύ πεδίων και περιορισμών (PK, FK), που τίθενται σε αυτά. Το πεδίο $S.B_1$ είναι πρωτεύον κλειδί της S, αλλά και έχει περιορισμό αναφορικής ακεραιότητας ως προς το $R.A_1$. Η κατεύθυνση της ακμής του ξένου κλειδιού δηλώνει ότι η πληροφορία του πρωτεύοντος κλειδιού της S εξαρτάται από την πληροφορία του κλειδιού της σχέσης R.
	Ακμές ερώτησης (σχηματική αναπαράσταση με ακμές τελείων) μεταξύ των πεδίων και των πινάκων που μετέχουν στη ερώτηση: `SELECT * FROM R, S WHERE R.A1 = S.B1.` Οι κόμβοι των γνωρισμάτων που εμφανίζονται στην ερώτηση συνδέονται με ακμές που προσπίπτουν στο κόμβο της ερώτησης (αραιές γραμμές με τελείες). Στο παραπάνω παράδειγμα, η ερώτηση θα πρέπει να συνδεθεί με όλα τα πεδία των σχέσεων R, S. Επειδή στη συγκεκριμένη εργασία μελετάμε μόνο τις περιπτώσεις επιλογής όλων των πεδίων μιας ή όλων των εγγραφών που είναι καταχωρημένες στη βάση, απλοποιούμε την απεικόνιση ζωγραφίζοντας ακμές από τους κόμβους των σχέσεων (που συμμετέχουν) προς τον κόμβο της ερώτησης.

	Ακμές διαδικασίας (σχηματική αναπαράσταση με ακμές γραμμών και τελείες) μεταξύ των πεδίων ή/και των πινάκων που εμφανίζονται στο σώμα της διαδικασίας:
	`CREATE PROCEDURE SP()AS`
	`INSERT INTO R(SELECT * FROM S WHERE S.B1 = term).`
	Και σε αυτή τη περίπτωση ισχύει η απλοποίηση της απεικόνισης του γράφου, ζωγραφίζοντας ακμές ανάμεσα στις σχέσεις που μετέχουν σε μια SP με όλα τα γνωρίσματά τους, και στο κόμβο της SP. Στο συγκεκριμένο παράδειγμα διαχωρίζονται οι ακμές του SELECT/FROM/WHERE ως εισερχόμενες στην SP, ενώ η ακμή INSERT ως εξερχόμενη. Με αυτό τον τρόπο δηλώνουμε ότι η SP έχει έξοδο μόνο στη περίπτωση INSERT/DELETE εγγραφών από έναν πίνακα.

	Ακμές εξάρτησης (σχηματική αναπαράσταση με διακεκομμένες γραμμές) μεταξύ μιας συναρτησιακής εξάρτησης *fd* και των πεδίων που μετέχουν σ' αυτή. Τα πεδία A_2, A_3 προσδιορίζουν το πεδίο A_4. Οπότε, στο κόμβο της εξάρτησης προσπίπτουν οι ακμές από τους κόμβους A_2, A_3, ενώ στον κόμβο A_4 προσπίπτει η ακμή που ξεκινά από τον κόμβο της εξάρτησης.

Σχήμα 5.1 Αναπαράσταση των συσχετίσεων που εμφανίζονται στα συστήματα απεικόνισης των σχεδιαστικών προτύπων.

5.2. Ορισμός μετρικών

Στη συνέχεια ορίζουμε τις μετρικές που θα χρησιμοποιήσουμε, παρουσιάζοντας: i) τη γενική και διαισθητική περιγραφή τους, και ii) αποδείξεις ορθότητας με βάση τις ιδιότητες του [1] όταν αυτές υφίστανται. Οι μετρικές Cohesion και Data_Volume δεν αποδεικνύονται σύμφωνα με τις παραπάνω ιδιότητες, διότι το Cohesion ορίζεται διαφορετικά για τις Βάσεις Δεδομένων, ενώ το Data_Volume αποτελεί μια μετρική που δεν παρουσιάζεται στο [1]. Πριν προχωρήσουμε στην παρουσίαση των μετρικών, θα πρέπει να σημειωθεί ότι, ο τρόπος υπολογισμού της εκάστοτε μετρικής για την περίπτωση των ερωτήσεων εξετάζεται ξεχωριστά, λόγω του ότι παραλείπεται η λεπτομερής περιγραφή τους.

Μέγεθος (Size). Το μέγεθος αποτελεί μια μέτρηση των στοιχείων που συγκροτούν το σύστημα. Διαισθητικά, η μετρική του μεγέθους παρουσιάζει την «προσπάθεια» που απαιτείται για τη σχεδίαση και τη συντήτηση ενός συστήματος. Εν γένει, όσο πιο μεγάλο είναι το μέγεθος του συστήματος, τόσο πιο δύσκολη είναι η σχεδίαση και η συντήρησή του. Στην περίπτωσή μας, θεωρούμε ότι το μέγεθος ισοδυναμεί με τον αριθμό των κόμβων ενός συστήματος. Συνεπώς, δεδομένου ενός συστήματος *S*, το οποίο απεικονίζεται μέσω του γράφου `S=(E,R)`, το μέγεθος του *S* δίνεται από τον τύπο:

$$Size(S) = cardinality(E) = |E|.$$

Σε αυτό το σημείο θα πρέπει να υπενθυμίσουμε ότι, το σύστημά μας αποτελείται από το σύστημα προτύπου και κάποιες ερωτήσεις που τίθενται σε αυτό. Η μέτρηση του μεγέθους μιας μονάδας σχέσης είναι προφανής, δηλαδή ισούται με το πλήθος των κόμβων που ανήκουν σε κάθε μονάδα. Ανάλογα η μέτρηση όλων των στοιχείων του συστήματος προτύπου αντιστοιχεί στην καταμέτρηση όλων των κόμβων που το απαρτίζουν. Για την μέτρηση του μεγέθους όλου του συστήματος είναι απαραίτητη η μέτρηση και του μεγέθους κάθε ερώτησης που τίθεται στο σύστημα προτύπου, γεγονός που προϋποθέτει τη λεπτομερή γραφική αναπαράστασή τους. Κατά το λεπτομερή σχεδιασμό μιας ερώτησης, ως κόμβοι παρουσιάζονται τα γνωρίσματα των σχέσεων που εμφανίζονται στις γραμμές `SELECT` και `WHERE` της ερώτησης, καθώς και κάθε πράξη (`JOIN`, `SUM`, `AVG`, `GROUP BY`, κ.τ.λ.) που εφαρμόζεται πάνω σ' αυτά. Εφόσον εδώ παραλείπεται η αναλυτική σχηματική περιγραφή μιας ερώτησης, δίδεται ο παρακάτω τύπος υπολογισμού του μεγέθους μιας ερώτησης:

$$Size(Q) = \#\,input\ \ attributes\ + \#\,output\ \ attributes\ +\ \ \#\ joins + \#\gamma =$$
$$\#\,attributes\ \ in\ WHERE\ \ clause + \#\,attributes\ \ in\ SELECT\ \ clause + \#\ joins + \#\gamma,$$

όπου $\#_{\gamma}$ είναι το πλήθος των υπολογιστικών συναρτήσεων (`SUM`, `AVG`, συναρτήσεις της μορφής `y=12*x`, κ.τ.λ.), που εμφανίζονται στη γραμμή των `SELECT`, `GROUP BY` και `HAVING` μιας SQL ερώτησης.

Η συνάρτηση του μεγέθους ικανοποιεί τις παρακάτω ιδιότητες όπως αυτές παρουσιάζονται στο [1]:

- *Μη αρνητικότητα*. Ισχύει προφανώς, διότι ο αριθμός των κόμβων πάντα είναι μη αρνητικός (στη χειρότερη περίπτωση $|E| = 0$). Επομένως, $Size(S) \geq 0$.
- *Μηδενική τιμή*. Το μέγεθος ενός γράφου `S` είναι ίσο με το μηδέν όταν το σύνολο των κόμβων `E` είναι κενό. Δηλαδή, $E=\varnothing \Rightarrow Size(S) = |E| = 0$.
- *Προσθετικότητα μονάδας*. Το μέγεθος ενός συστήματος `S=(E,R)`, ισούται με το άθροισμα των μεγεθών των δύο μονάδων του, $m1(E_{m1},R_{m1})$και $m2(E_{m2},R_{m2})$, εάν κάθε κόμβος του `S` να ανήκει είτε στο `m1` είτε στο `m2`. Προφανώς, ο αριθμός των κόμβων ενός γράφου ισούται με το άθροισμα των κόμβων των επιμέρους υπογράφων του, οπότε ισχύει: $E = E1 \cup E2 \Rightarrow |E| = |E1| + |E2| \Rightarrow Size(S) = Size(m1)+ Size(m2)$.

Συνδεσιμότητα (Coupling). Η συνδεσιμότητα, όπως παρουσιάζεται [1], δηλώνει το κατά πόσο οι κόμβοι μιας μονάδας συσχετίζονται με κόμβους άλλων μονάδων του συστήματος, δηλαδή κατά πόσο υπάρχει εξάρτηση μεταξύ των μονάδων του συστήματος. Διαισθητικά, με τη μέτρηση του coupling αποτιμάται το κατά πόσο είναι δύσκολη η διατήρηση (maintenance) ενός συστήματος Βάσης Δεδομένων. Απόδειξη, λοιπόν, ενός καλού σχεδιασμού αποτελεί το μικρό ποσοστό συνδεσιμότητας. Η μετρική της συνδεσιμότητας, έχει νόημα να υπολογιστεί για τις μονάδες: i) σχέσης, ii) ερώτησης, iii) όψης, και iv) διαδικασίας, ως προς το συνολικό σύστημα. Σε αυτή την εργασία, μελετάμε τη συνδεσιμότητα μεταξύ των μονάδων σχέσης του συστήματος προτύπου, και τη συνδεσιμότητα των μονάδων ερωτήσεων/όψεων/διαδικασιών σε σχέση με τις μονάδες σχέσης του συστήματος προτύπου. Πιο απλά, υπολογίζουμε το κατά πόσο ένας πίνακας εξαρτάται από τους υπόλοιπους πίνακες του προτύπου (ύπαρξη περιορισμών αναφορικής ακεραιότητας), και σε τι βαθμό εξαρτάται κάθε

ερώτηση, που τίθεται στο πρότυπο, από τους πίνακες του προτύπου (πίνακες που μετέχουν στην ερώτηση). Συνεπώς, στη περίπτωση, όπου ένας πίνακας του προτύπου υποστεί μια αλλαγή (εισαγωγή/διαγραφή/ανανέωση πλειάδας ή γνωρίσματος) ,η τιμή της συνδεσιμότητας κάθε μονάδας, παρουσιάζει το μέγιστο πλήθος των αλλαγών, που θα πρέπει να πραγματοποιηθούν στη μονάδα για να είναι σε συνεπή μορφή.

Για τον υπολογισμό της συνδεσιμότητας των μονάδων του συστήματος, λαμβάνουμε υπόψη τις εισερχόμενες ακμές μιας μονάδας. Για τη περίπτωση των μονάδων του συστήματος, οι εισερχόμενες ακμές απεικονίζουν τους περιορισμούς αναφορικής ακεραιότητας μεταξύ των σχέσεων. Ανάλογα, στη περίπτωση των ερωτήσεων, οι εσωτερικές ακμές μιας ερώτησης καθορίζει το βαθμό εξάρτησης της ερώτησης από τα πεδία των σχέσεων, που εμφανίζονται σε αυτή. Ο τυπικός ορισμός της συνδεσιμότητας μιας μονάδας, *m*, είναι ο εξής :

$$Coupling\ (m) = |\ \text{Incoming}\ (m)\ |\ .$$

Ο παραπάνω τύπος μπορεί εύκολα να εφαρμοστεί στην περίπτωση μονάδων σχέσεων, εφόσον όλες οι εισερχόμενες ακμές της μονάδας μπορούν να μετρηθούν μέσω του γράφου. Για την περίπτωση μίας ερώτησης όμως, θα πρέπει να διευκρινιστεί ότι η συνδεσιμότητα της εξαρτάται από το πλήθος των γνωρισμάτων που εμφανίζονται στις γραμμές `SELECT`, και `WHERE` της ερώτησης. Στο σχήμα 5.5 παρουσιάζεται ο υπολογισμός της μετρικής coupling μιας ερώτησης.

e.g. Q :

SELECT $A1, ..., Ak, B1, ..., Bm$
FROM A, B
WHERE $A_1 = B_1$ *AND* $B_1 = A_k$ *AND* $A_k = B_m$
GROUP BY A_k

$Coupling\ (Q) = k + m + 4,$ *όπου* k *είναι πλήθος των γνωρισμάτων της σχέσης* A,
m *το πλήθος των γνωρισμάτων της* B
και 4 *είναι τα διαφορετικά γνωρίσματα ,*
που εμφανίζονται στη γραμμή WHERE *της ερώτησης* Q.

Σχήμα 5.2 Υπολογισμός της συνδεσιμότητας μιας ερώτησης.

Η συνδεσιμότητα ικανοποιεί τις παρακάτω ιδιότητες όπως αυτές παρουσιάζονται στο [1]:

- *Μη αρνητικότητα.* Δεδομένου ότι υπολογίζεται το πλήθος των εισερχόμενων ακμών μιας μονάδας, η συνδεσιμότητα, προφανώς, δεν μπορεί να λάβει

αρνητικές τιμές. Οπότε Coupling(m)≥0. Κατ' επέκταση, ισχύει: Coupling(PS)≥0, όπου PS το σύστημα προτύπου, το οποίο περιλαμβάνει τις μονάδες των σχέσεων και συνδέεται με τις μονάδες των ερωτήσεων που τίθενται σε αυτό.

- *Μηδενική τιμή*. Προφανώς, όταν το σύνολο των εισερχόμενων ακμών μιας μονάδας είναι κενό, δηλαδή όταν η μονάδα δεν συνδέεται με καμία άλλη μονάδα του γράφου, η συνδεσιμότητά της είναι ίση με το μηδέν. Δηλαδή, Incoming(m)=∅ ⇒ |Incoming(m)|= 0 ⇒ Coupling(m)=0.
- *Μονοτονικότητα*. Προσθέτοντας μια εισερχόμενη ακμή σε μια μονάδα, η συνδεσιμότητα της μονάδας δεν μειώνεται, γεγονός που ισχύει εξ' ορισμού.
- *Ένωση συνδεδεμένων μονάδων*. Κατά τη ένωση δύο μονάδων, m1, m2, οι οποίες συνδέονται μεταξύ τους με εξωτερικές ακμές, η συνδεσιμότητα της μονάδας που προκύπτει m′ είναι μικρότερη ή ίση από το άθροισμα των συνδεσιμοτήτων των δύο μονάδων, δηλαδή ισχύει: Coupling(m′)≤Coupling(m1)+Coupling(m2). Η ιδιότητα ισχύει προφανώς, εφόσον η οποιαδήποτε εισερχόμενη ακμή που υπήρχε αρχικά μεταξύ των δύο μονάδων, μετά την ένωσή τους θεωρείται πλέον εσωτερική ακμή της νέας μονάδας m′.
- *Ένωση ασύνδετων μονάδων*. Σύμφωνα με αυτή την ιδιότητα, η συνδεσιμότητα μιας μονάδας m′, η οποία προκύπτει από την ένωση δύο μονάδων m1, m2 που δεν συνδέονται μεταξύ τους με εξωτερικές ακμές, ισοδυναμεί με το άθροισμα των συνδεσιμοτήτων των δύο μονάδων. Προφανώς, εφόσον, οι δύο μονάδες είναι ασύνδετες μεταξύ τους, το coupling της κάθε μίας αφορά εξωτερικές ακμές προς άλλες μονάδες του συστήματος, οι οποίες διατηρούνται μετά την ένωση των μονάδων. Οπότε ισχύει:

  Coupling(m′)=Coupling(m1)+Coupling(m2).

Συνεκτικότητα (Cohesion). Η συνεκτικότητα, όπως ορίζεται στο [1], δηλώνει το ποσοστό συνοχής των στοιχείων μιας μονάδας, δηλαδή το κατά πόσο τα στοιχεία που ανήκουν σε μια μονάδα επιτελούν μια συγκεκριμένη «εργασία». Ο σκοπός της εν λόγω μετρικής, δηλαδή, είναι να πριμοδοτήσει μονάδες οι οποίες δεν ενσωματώνουν διάφορες λειτουργίες, αλλά αντιθέτως, επικεντρώνονται στην εξυπηρέτηση ενός

στόχου. Η συνεκτικότητα αποτελεί μια μετρική που δεν μπορεί να οριστεί γενικά σε ένα σύστημα, παρά μόνο στα υποσυστήματά του, δηλαδή στις μονάδες, στα συστήματα μονάδων και στο σύστημα προτύπου. Ζητούμενο, σε κάθε περίπτωση, είναι το cohesion κάθε μονάδας να είναι μεγάλο, διότι τότε (α) το σύστημα μπορεί να συντηρηθεί με μεγαλύτερη ευκολία και (β) σε περίπτωση εξέλιξης των απαιτήσεων, υπάρχει μία μονάδα που πρέπει να εντοπισθεί και να συντηρηθεί. Παραδείγματα μονάδων με μέγιστη συνεκτικότητα είναι: i) μια σχέση, η οποία εξασφαλίζει την 3NF, και ii) μια ερώτηση ή όψη, η οποία απλά διαβάζει έναν πίνακα (`SELECT * FROM R`). Στο χώρο της τεχνολογίας λογισμικού, η ιδεατή συνεκτικότητα προκύπτει όταν ο γράφος μιας μονάδας είναι σε μορφή κλίκας, όπου όλοι οι κόμβοι συνδέονται με όλους τους άλλους. Αντιθέτως, στη δική μας μοντελοποίηση, στην οποία οι ακμές του κόμβου υποκρύπτουν μια σχέση εξάρτησης, η συνεκτικότητα πρέπει να αποτιμηθεί στη βάση κόμβων με λειτουργικότητα και όχι στη βάση ακμών. Το cohesion μιας σχέσης, θα πρέπει να μειώνεται όταν εμφανίζονται μεταξύ των στοιχείων του συναρτησιακές εξαρτήσεις, *fd*, που παραβιάζουν την 3NF. Ο τύπος υπολογισμού της συνεκτικότητας μιας σχέσης είναι ο εξής:

$$Cohesion(relation) = \frac{1}{1 + \#\ fd}. \qquad (1)$$

Στο παραπάνω τύπο δεν λαμβάνουμε υπόψη τις συναρτησιακές εξαρτήσεις των πεδίων μιας σχέσης από το πρωτεύον κλειδί της. Οι εξαρτήσεις των πεδίων από το πρωτεύον κλειδί αποτελούν «καλές εξαρτήσεις», ενώ σκοπός του υπολογισμού της συνεκτικότητας, για μας, είναι η εύρεση των «κακών εξαρτήσεων», αυτών που παραβιάζουν την 3NF.

Αντίστοιχα, η συνεκτικότητα μιας ερώτησης μειώνεται ανάλογα με το πλήθος, θ, των πράξεων (=, ≤, κ.τ.λ.), που εμφανίζονται στη γραμμή `WHERE` και το πλήθος, γ, των υπολογιστικών συναρτήσεων (`SUM`, `AVG`, συναρτήσεις της μορφής `y=12*x`, κ.τ.λ.), που εμφανίζονται στη γραμμή των `SELECT`, `GROUP BY` και `HAVING` μιας SQL ερώτησης. Ο τύπος υπολογισμού της συνεκτικότητας μιας ερώτησης, όψης, διαδικασίας είναι:

$$Cohesion\,(query\,/\,view\,/\,procedure\,) = \frac{1}{1 + \#\theta\ + \#\gamma}. \qquad (2)$$

Η συνεκτικότητα ενός συστήματος μονάδων MS ή του συστήματος προτύπου PS δίνεται από τον τύπο:

$$Cohesion\,(MS \mid PS) = avg\,(Cohesion\,(m_i)), \quad για\ όλα\ τα\ m_i\ που\ ανήκουν\ στο\ MS\ ή\ στο\ PS\ .$$

Σύμφωνα με τους [1], μια μονάδα του συστήματος παρουσιάζει μέγιστη συνεκτικότητα όταν κάθε κόμβος συνδέεται με όλους τους κόμβους της μονάδας. Αυτή η θεώρηση δεν μπορεί να ισχύσει στις Βάσεις Δεδομένων για τους εξής λόγους:

i. Δεν είναι δυνατόν τα πεδία μιας σχέσης να συνδέονται άμεσα μεταξύ τους με ακμές.
ii. Εάν οι κόμβοι μιας μονάδας συνδέονται μεταξύ τους μέσω κόμβων εξαρτήσεως, τότε οι εξαρτήσεις αυτές είναι «κακές εξαρτήσεις», που παραβιάζουν την 3NF.

Με βάση τους παραπάνω λόγους, δεν είναι δυνατή η απόδειξη της ιδιότητας της *μονοτονικότητας*, κατά την οποία θεωρείται ότι η εισαγωγή μιας νέας ακμής ανάμεσα σε δύο κόμβους μιας μονάδας, δεν μπορεί να οδηγήσει σε μείωση της συνεκτικότητας της μονάδας. Για να γίνει πιο κατανοητό το πρόβλημα της απόδειξης της συγκεκριμένης ιδιότητας, ας υποθέσουμε ότι εισάγεται μια νέα ακμή στους κόμβους μιας μονάδας σχέσης. Εάν η ακμή δηλώνει εξάρτηση ενός πεδίο της σχέσης από ένα πεδίο της σχέσης, πλην του πρωτεύοντος κλειδιού, τότε η συνεκτικότητα της μονάδας μειώνεται, διότι η συγκεκριμένη μονάδα επιτελεί δύο «εργασίες».

Παρόμοια, για τη περίπτωση των Βάσεων Δεδομένων δεν έχει νόημα η ιδιότητα της μηδενικής τιμής της συνεκτικότητας μιας μονάδας. Μηδενική συνεκτικότητα για μια μονάδα σχέσης, προϋποθέτει μη ύπαρξη πεδίων σε μια σχέση. Αντίστοιχα, μηδενική συνεκτικότητα για μια μονάδα ερώτησης/όψης/διαδικασίας, υποδηλώνει ότι δεν υπάρχει κανένα γνώρισμα στο `SELECT clause` και καμία σχέση στο `FROM clause` της ερώτησης, πράγμα αδύνατο.

Οι ιδιότητες που ισχύουν για τη συνεκτικότητα μιας μονάδας (πίνακας, ερώτηση, όψη, διαδικασίας, κ.τ.λ) είναι οι εξής:

- *Μη αρνητικότητα και κανονικοποίηση*.
    - Συνεκτικότητα μιας σχέσης: Είναι προφανές ότι το cohesion, όπως προκύπτει από τον τύπο *(1)* δεν μπορεί να πάρει αρνητική τιμή. Επιπλέον, είναι κανονικοποιημένο στο διάστημα (0,1].

- Συνεκτικότητα μιας ερώτησης: Παρόμοια, το cohesion μιας ερώτησης που δίνεται από τον τύπο *(2)*, είναι μη αρνητικός αριθμός. Το cohesion, και σε αυτή την περίπτωση, παίρνει τιμές από το διάστημα (0,1].

- *Μέγιστη τιμή*.
  - Συνεκτικότητα μιας σχέσης: Ο τύπος *(1)* λαμβάνει τη μέγιστη τιμή `1`, όταν δεν υπάρχουν συναρτησιακές εξαρτήσεις μεταξύ των γνωρισμάτων της σχέσης. Δηλαδή ισχύει: `# f = 0` ⇒ `Cohesion(R)=1.`
  - Συνεκτικότητα μιας ερώτησης: Ο τύπος *(2)*, λαμβάνει τη μέγιστη τιμή `1`, όταν η ερώτηση δεν εκτελεί καμία λειτουργία πλην της ανάγνωσης ενός πίνακα. Δηλαδή ισχύει, $\#\theta + \#\gamma = 0 \Rightarrow$ `Cohesion(Q)=1.`
- *Μηδενική τιμή*. Θεωρητικά το cohesion τείνει στο μηδέν όσο αυξάνει ο αριθμός των συναρτησιακών εξαρτήσεων μεταξύ των γνωρισμάτων μιας σχέσης, ή ο αριθμός των πράξεων μιας ερώτησης.

Μέγεθος στοιχείων (Data_Volume). Η μετρική αυτή, πρωτίστως, αποτιμά το μέγεθος του αποθηκευτικού χώρου που θα πρέπει να δεσμευτεί για την κατασκευή μιας σχέσης ή μιας υλοποιήσιμης όψης, καθώς επίσης και τον συνολικό χώρο που καταλαμβάνει κάθε πίνακας με τις εγγραφές του. Ο υπολογισμός του μεγέθους των στοιχείων ενός πίνακα, αποτελεί έναν γενικό τρόπο αποτίμησης του χώρου του δίσκου που καταλαμβάνεται. Προφανώς, μια καλή σχεδίαση μιας Βάσης Δεδομένων, προσπαθεί να ελαχιστοποιήσει τον απαιτούμενο αποθηκευτικό χώρο, με την αποφυγή επανάληψης πληροφορίας. Κατά τον υπολογισμό του μεγέθους των στοιχείων, θα πρέπει να λάβουμε υπόψη το μέγεθος κάθε γνωρίσματος (`sizeof`) μιας σχέσης, καθώς και το πλήθος των γραμμών που αποθηκεύονται (`#rows`) στον αντίστοιχο πίνακα. Το `sizeof` θεωρούμε ότι απεικονίζει τον τύπο δήλωσης κάθε γνωρίσματος (π.χ., integer, varchar, κ.τ.λ.). Ο ορισμός, λοιπόν, του μεγέθους των στοιχείων ενός πίνακα δίνεται με την βοήθεια του τύπου:

$$Data\_Volume(R \mid MV) = \#rows_{R|MV} * (\sum_{i=1}^{n} sizeof(A_i)),$$

όπου `i=1,..n` και A_i το `i`-οστό γνώρισμα του πίνακα `R` ή της materialized view `MV`.

Η μετρική αυτή δεν δίδεται στο [1], οπότε οι ιδιότητες που μπορούν να ισχύσουν εδώ είναι οι εξής:

- *Μη αρνητικότητα*. Προφανώς το μέγεθος των στοιχείων ενός πίνακα δεν μπορεί να είναι αρνητικός αριθμός.
- *Μηδενική τιμή*. Προφανώς, το μέγεθος των στοιχείων ενός πίνακα είναι ίσο με το μηδέν όταν δεν υπάρχει κανένα πεδίο στον πίνακα, ή όταν δεν έχουν εισαχθεί πλειάδες.
- *Μονοτονικότητα*. Στην περίπτωση της ένωσης δύο πινάκων, όπου το αποτέλεσμα είναι ένας πίνακας που περιλαμβάνει όλα τα γνωρίσματα των πινάκων και κατ' επέκταση όλες τις εγγραφές τους, το μέγεθος των στοιχείων του τελικού πίνακα ισοδυναμεί με το άθροισμα των στοιχείων των επιμέρους πινάκων. Δηλαδή ισχύει: $R = R1 \cup R2 \Rightarrow size(R) = size(R1) + size(R2)$.

5.3. Μετρικές ποιότητας των προτεινόμενων προτύπων

Σε αυτή την ενότητα παρουσιάζονται οι μετρικές ποιότητας των σχεδιαστικών προτύπων. Για κάθε πρόβλημα, μελετούνται όλες οι προτεινόμενες λύσεις. Στη συγκεκριμένη εργασία θα ασχοληθούμε με δύο τύπους ερωτήσεων, οι οποίοι αφορούν:

1. στην ανάκτηση όλης της πληροφορίας, δηλαδή ζητάμε όλες τις πλειάδες που είναι αποθηκευμένες σε όλους τους πίνακες του σχήματος μιας βάσης, και
2. στην ανάκτηση μιας συγκεκριμένης εγγραφής, που είναι καταχωρημένη στη βάση.

Ο λόγος για τον οποίο μελετάμε μόνο αυτές τις δύο περιπτώσεις ερωτήσεων είναι ότι, αποτελούν τις δύο πιο συχνές ερωτήσεις που τίθενται προς μια βάση δεδομένων.

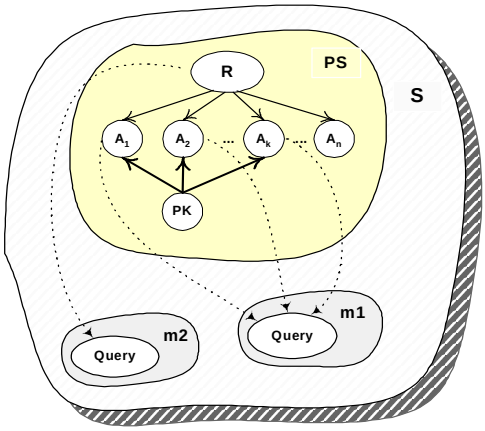
Σε περίπτωση δημιουργίας μιας ερώτησης, όψης ή διαδικασίας, περιγράφονται τα γενικά βήματα της κατασκευής, έτσι ώστε να είναι πιο κατανοητός ο τρόπος υπολογισμού των μετρικών. Υπενθυμίζεται, όμως, ότι, για λόγους απλοποίησης, οι ακμές ερώτησης εφαρμόζονται μεταξύ του κόμβου της ερώτησης και του κόμβου της σχέσης, υποδηλώνοντας τη συσχέτιση του κόμβου της ερώτησης με τα γνωρίσματα της σχέσης.

5.3.1. Τεχνητό Κλειδί (Artificial key)

Σημαντικά προβλήματα, κατά τη σχεδίαση μιας βάσης, εμφανίζονται όταν το υποψήφιο κλειδί: i) είναι συνδυασμός γνωρισμάτων, ii) είναι τύπου συμβολοσειράς ή iii) έχει περιορισμένο χρόνο ζωής. Για την αντιμετώπιση των παραπάνω προβλημάτων προτείνονται οι εξής εναλλακτικές σχεδίασης:

1. επιλογή του συνδυασμού γνωρισμάτων ως πρωτεύον κλειδί.
2. δημιουργία τεχνητού κλειδιού, το οποίο ενημερώνεται αυτόματα από το σύστημα, και επιπλέον εφαρμογή περιορισμού μοναδικότητας στα γνωρίσματα του υποψήφιου κλειδιού.
3. δημιουργία τεχνητού κλειδιού, το οποίο ενημερώνεται μέσω μιας διαδικασίας. Οι μέγιστες τιμές κάθε τεχνητού κλειδιού αποθηκεύονται σε έναν πίνακα καταχώρησης, στον οποίον πραγματοποιείται η αναζήτηση και ανανέωση της εκάστοτε τιμής. Παρόμοια με τη $2^{η}$ περίπτωση θα πρέπει να εφαρμοστεί περιορισμός μοναδικότητας στα γνωρίσματα του υποψήφιου κλειδιού.

Θεωρούμε τη γενική περίπτωση της σχέσης `R(A1, …, An)`, όπου το υποψήφιο κλειδί είναι ο συνδυασμός των γνωρισμάτων `(A1, …, Ak)`. Στο Σχήμα 5.3 αναπαρίσταται και δίδονται οι μετρικές ποιότητας της περίπτωσης όπου επιλέγεται ως πρωτεύον κλειδί, ο συνδυασμός των γνωρισμάτων. Στο Σχήμα 5.4 δίνεται η αναπαράσταση και οι μετρικές για την περίπτωση, όπου δημιουργείται ένα τεχνητό κλειδί `(A_ID)`, το οποίο ανανεώνεται αυτόματα από το σύστημα. Τέλος στο Σχήμα 5.5 μελετάται η περίπτωση δημιουργίας τεχνητού κλειδιού, το οποίο ενημερώνεται μέσω μιας διαδικασίας. Οι μέγιστες τιμές των τεχνητών κλειδιών όλων των πινάκων της βάσης αποθηκεύονται στο πίνακα `AT_KEYS`.



$Size(PS) = n + 2.$

$Coupling(PS) = 0.$

$Cohesion(PS) = 1.$

$Data\_Volume(PS) = \#rows * (\sum_{i=1}^{n} sizeof(A_i)).$

Query $m1$: *Select a specific tuple by defining attributes of PK*

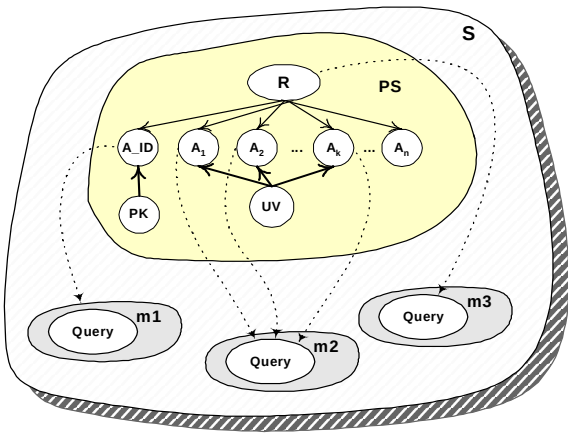
$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R \\ WHERE\ A_1 = term\ AND\ A_2 = term\ AND ... A_k = term \end{array}\right\} \Rightarrow \begin{cases} Size(m1) = n + 2 * k + 1. \\ Coupling(m1) = n + k. \\ Cohesion(m1) = 1/k + 1. \end{cases}$$

Query $m2$: *Select all tuples*

$$\left.\begin{array}{l} SELECT\ * \\ FROM\ R \end{array}\right\} \Rightarrow \begin{cases} Size(m2) = n + k. \\ Coupling(m2) = n. \\ Cohesion(m2) = 1. \end{cases}$$

$Size(S) = (n+2) + (n+2*k+1) + (n+k) = 3*n + 2*k + 3 = 3*n + 3*k + 3.$

Σχήμα 5.3 Αναπαράσταση και μετρικές της σχεδιαστικής λύσης, όπου θέτει ως πρωτεύον κλειδί το συνδυασμό γνωρισμάτων.



$Size(PS) = n + 4.$

$Coupling(PS) = 0.$

$Cohesion(PS) = 1.$

$Data\_Volume(PS) = \#rows * (sizeof(A\_ID) + \sum_{i=1}^{n} sizeof(A_i)).$

Query $m1$: *Select a specific tuple by defining* A_ID

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R \\ WHERE\ A\_ID = term \end{array}\right\} \Rightarrow \left\{\begin{array}{l} Size(m1) = n + 2*1 + 1 = n + 3. \\ Coupling(m1) = n + 1. \\ Cohesion(m1) = 1/2. \end{array}\right.$$

Query $m2$: *Select a specific tuple by defining attributes with Unique Values*

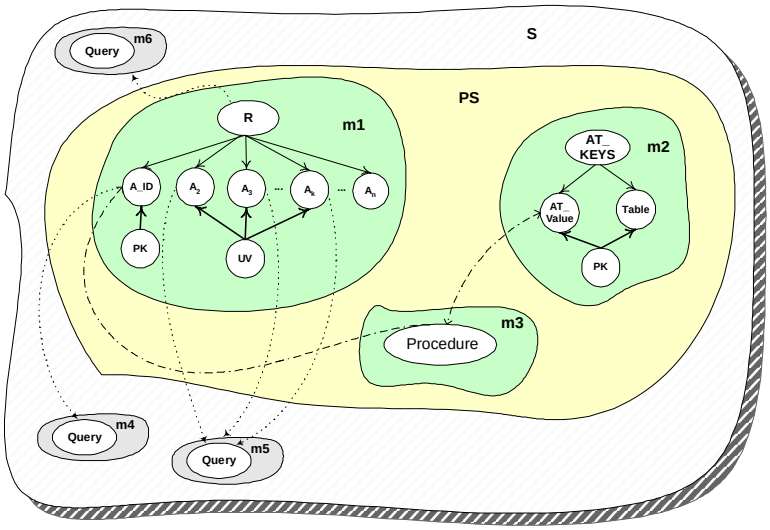
$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R \\ WHERE\ A_1 = term\ AND\ A_2 = term\ AND ... A_k = term \end{array}\right\} \Rightarrow \left\{\begin{array}{l} Size(m2) = n + 2*k + 1. \\ Coupling(m2) = n + k. \\ Cohesion(m2) = 1/k + 1 \end{array}\right.$$

Query $m3$: *Select all tuples*

$$\left.\begin{array}{l} SELECT\ * \\ FROM\ R \end{array}\right\} \Rightarrow \left\{\begin{array}{l} Size(m3) = n. \\ Coupling(m3) = n. \\ Cohesion(m3) = 1. \end{array}\right.$$

$Size(S) = (n + 4) + (n + 3) + (n + 2*k + 1) + n = 4*n + 2*k + 8.$

Σχήμα 5.4 Αναπαράσταση και μετρικές της σχεδιαστικής λύσης με εισαγωγή τεχνητού κλειδιού (A_ID).



$Coupling(m1) = Coupling(m2) = 1.$

$Cohesion(m1) = Cohesion(m2) = 1.$

$\Pr ocedure\ m3 : Insert\ new\ tuple\ in\ R$

$$\left.\begin{array}{c} Step1 : Find\ \max(AT\_key\_value)\ for\ relation\ R\ in\ table\ AT\_KEYS \\ Step2 : Set\ AT\_key\_value := \max(AT\_key\_value) + 1; \\ Udpate\ AT\_Key\_value\ of\ AT\_KEYS\ table \\ Step3 : Insert\ new\ tuple \end{array}\right\} \Rightarrow \left\{\begin{array}{c} Size(m3) = (1+1+1) + (1+1+1) + n = n + 6. \\ Coupling(m3) = n + 2. \\ Cohesion(m3) = \frac{1}{3}. \end{array}\right.$$

$$Size(PS) = size(m1) + size(m2) + size(\Pr ocedure) = (n+6) + (4+1) + (6+n) = 2*n+17.$$

$$Coupling(PS) = 0.$$

$$Cohesion(PS) = \frac{7}{9}.$$

$$Data\_Volume(PS) = \#rows_R * (sizeof(A\_ID) + \sum_{i=1}^{n} sizeof(A_i)) + \#rows_{AT\_KEYES} * (sizeof(AT\_Value) + sizeof(Table)).$$

$Query\ m4 : Select\ a\ specific\ tuple\ by\ defining\ A\_ID$

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R \\ WHERE\ A\_ID = term \end{array}\right\} \Rightarrow \left\{\begin{array}{c} Size(m4) = n + 2*1 + 1 = n + 3. \\ Coupling(m4) = n + 1. \\ Cohesion(m4) = \frac{1}{2}. \end{array}\right.$$

$Query\ m5 : Select\ a\ specific\ tuple\ by\ defining\ attributes\ with\ Unique\ Values$

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R \\ WHERE\ A_1 = term\ AND\ A_2 = term\ AND ... A_k = term \end{array}\right\} \Rightarrow \left\{\begin{array}{c} Size(m2) = n + 2*k + 1. \\ Coupling(m2) = n + k. \\ Cohesion(m2) = \frac{1}{k+1}. \end{array}\right.$$

$Query\ m6 : Select\ all\ tuples$

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R \end{array}\right\} \Rightarrow \left\{\begin{array}{c} Size(m3) = n. \\ Coupling(m3) = n. \\ Cohesion(m3) = 1. \end{array}\right.$$

$$Size(S) = (2*n+17) + (n+3) + (n+2*k+1) + n = 5*n + 2*k + 21.$$

Σχήμα 5.5 Αναπαράσταση και μετρικές της σχεδιαστικής λύσης με τεχνητό κλειδί και ενός βοηθητικού πίνακα καταχώρησης μέγιστων τιμών.

5.3.2. Κανονικές Μορφές (Normal Forms)

Οι κανονικές μορφές, εν γένει, έχουν ως σκοπό την αποφυγή του φαινόμενου του πλεονασμού τιμών και του φαινόμενου ασυνέπειας τιμών. Με αυτόν τον τρόπο διασφαλίζεται ένας καλός και αποδοτικός σχεδιασμός μιας Βάσης Δεδομένων.

- **1NF**

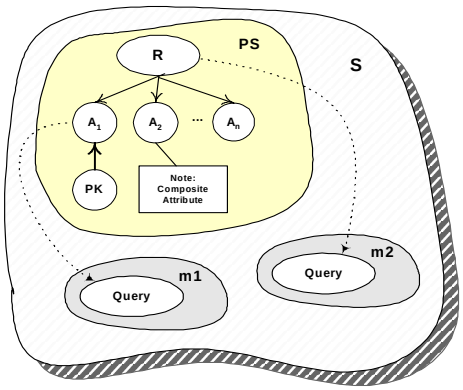
Η πρώτη κανονική μορφή «επιβάλλει» όλα τα γνωρίσματα να είναι ατομικά, απομακρύνοντας τα πλειότιμα και τα σύνθετα γνωρίσματα. Για τη περίπτωση του σύνθετου γνωρίσματος παρουσιάζονται δύο εναλλακτικές σχεδιάσεις:

1. διατήρηση του σύνθετου γνωρίσματος παραβιάζοντας τη 1NF.
2. δημιουργία ξεχωριστών πεδίων για το σύνθετο γνώρισμα.

Για τη περίπτωση πλειότιμου γνωρίσματος δίδονται οι εξής εναλλακτικές σχεδιάσεις:

1. διατήρηση του πλειότιμου γνωρίσματος παραβιάζοντας τη 1NF.
2. δημιουργία ξεχωριστών πεδίων για το πλειότιμο γνώρισμα.
3. δημιουργία ξεχωριστού πίνακα, ο οποίος περιλαμβάνει το πλειότιμο γνώρισμα.

Θεωρούμε τη γενική περίπτωση της σχέσης `R(A1,A2, …,An)`, με πρωτεύον κλειδί το πεδίο `A1`. Στο σχήμα 5.6 επιλέγεται η παραβίαση της 1NF λόγω του σύνθετου γνωρίσματος `A2`. Στο Σχήμα 5.7 δημιουργούνται k ατομικά πεδία, στα οποία απλοποιείται το σύνθετο γνώρισμα. Στο Σχήμα 5.8 επιλέγεται η παραβίαση της 1NF, λόγω ύπαρξης πλειότιμου γνωρίσματος `A2`. Η εναλλακτική σχεδίαση, όπου κατασκευάζονται k (όσο και το μέγιστο πλήθος των τιμών του πλειότιμου γνωρίσματος) ξεχωριστά πεδία, παρουσιάζεται στο Σχήμα 5.9. Στο Σχήμα 5.10 παρουσιάζεται η σχεδίαση της κανονικοποίησης, όπου η σχέση `R` μετατρέπεται σε `R1(A1,A3,…,An)`, ενώ παράλληλα δημιουργείται μια νέα σχέση η `R2 (A1,A2)`, η οποία έχει πρωτεύον κλειδί το συνδυασμό των πεδίων `(A1, A2)`. Μία παραλλαγή της τελευταίας λύσης, παρουσιάζεται στο Σχήμα 5.11, όπου αντί για το συνδυασμό `(A1,A2)`, επιλέγεται η δημιουργία τεχνητού κλειδιού.



$Size(PS) = n + 2.$

$Coupling(PS) = 0.$

$Cohesion(PS) = 1.$

$Data\_Volume(PS) = \#rows_R * (\sum_{i=1}^{n} sizeof(A_i)).$

Query $m1$: *Select a specific tuple*

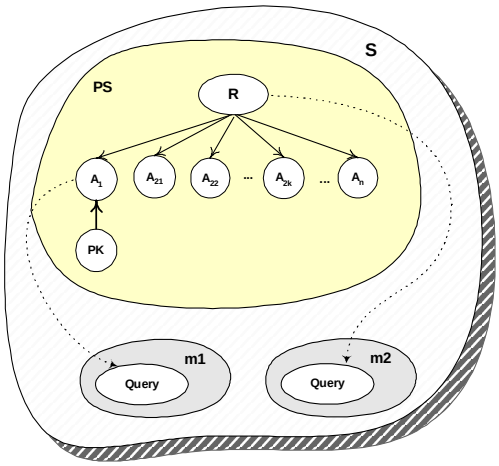
$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R \\ WHERE\ A_1 = term \end{array}\right\} \Rightarrow \begin{cases} Size(m1) = n + 2*1 + 1 = n + 3. \\ Coupling(m1) = n + 1. \\ Cohesion(m1) = 1/2. \end{cases}$$

Query $m2$: *Select all tuples*

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R \end{array}\right\} \Rightarrow \begin{cases} Size(m2) = n. \\ Coupling(m2) = n \\ Cohesion(m2) = 1. \end{cases}$$

$Size(S) = (n + 2) + (n + 3) + n = 3 * n + 5.$

Σχήμα 5.6 Αναπαράσταση και μετρικές της σχεδιαστικής λύσης, όπου παραβιάζεται η 1NF λόγω σύνθετου γνωρίσματος.



$Size(PS) = n + 2 + k.$

$Coupling(PS) = 0.$

$Cohesion(PS) = 1.$

$Data\_Volume(PS) = \#rows_R * (sizeof(A_1) + \sum_{i=1}^{k} sizeof(A_{2i}) + \sum_{j=3}^{n} sizeof(A_j)) = \#rows_R * (\sum_{i=1}^{n} sizeof(A_i)).$

Query $m1$: *Select a specific tuple*

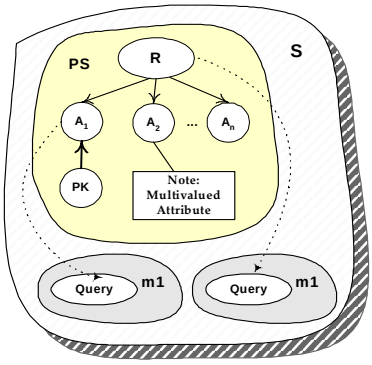
$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R \\ WHERE\ A_1 = term \end{array}\right\} \Rightarrow \left\{\begin{array}{c} Size(m1) = n + k + 2 * 1 + 1 = n + k + 3. \\ Coupling(m1) = n + k + 1. \\ Cohesion(m1) = 1/2. \end{array}\right.$$

Query $m2$: *Select all tuples*

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R \end{array}\right\} \Rightarrow \left\{\begin{array}{c} Size(m2) = n + k. \\ Coupling(m2) = n + k. \\ Cohesion(m2) = 1. \end{array}\right.$$

$Size(S) = (n + 2 + k) + (n + k + 3) + (n + k) = 3 * n + 3 * k + 5.$

Σχήμα 5.7 Αναπαράσταση και μετρικές της σχεδιαστικής λύσης, όπου ικανοποιείται η 1NF με την κατασκευή k πεδίων για τις k τιμές του σύνθετου γνωρίσματος.



$Size(PS) = n + 2.$

$Coupling(PS) = 0.$

$Cohesion(PS) = 1.$

$Data\_Volume(PS) = \#rows_R * (\sum_{i=1}^{n} sizeof(A_i))$

Query $m1$: *Select a specific tuple*

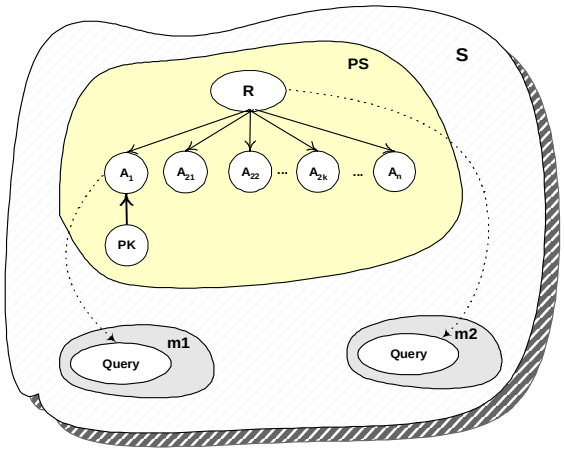
$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R \\ WHERE\ A_1 = term \end{array}\right\} \Rightarrow \begin{cases} Size(m1) = n + 2*1 + 1 = n + 3. \\ Coupling(m1) = n + 1. \\ Cohesion(m1) = 1/2. \end{cases}$$

Query $m2$: *Select all tuples*

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R \end{array}\right\} \Rightarrow \begin{cases} Size(m2) = n. \\ Coupling(m2) = n. \\ Cohesion(m2) = 1. \end{cases}$$

$Size(S) = (n + 2) + (n + 3) + n = 3 * n + 5.$

Σχήμα 5.8 Αναπαράσταση και μετρικές της σχεδιαστικής λύσης, όπου παραβιάζεται η 1NF λόγω πλειότιμου γνωρίσματος.



$Size(PS) = n + k + 2.$

$Coupling\ (PS) = 0.$

$Cohesion\ (PS) = 1.$

$Data\_Volume\ (PS) = \#rows_R * (sizeof\ (A_1) + \sum_{i=1}^{k} sizeof\ (A_{2i}) + \sum_{j=3}^{n} sizeof\ (A_j)).$

Query $m1$: *Select a specific tuple*

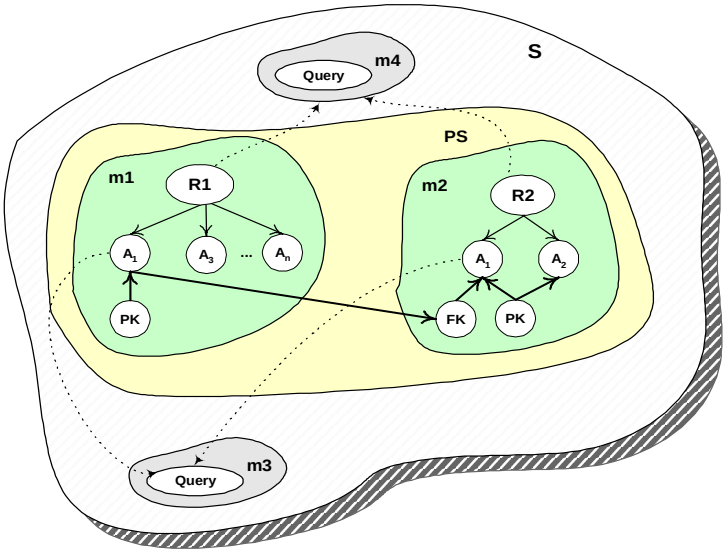
$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R \\ WHERE\ A_1 = term \end{array}\right\} \Rightarrow \left\{\begin{array}{c} Size(m1) = n + k + 2 * 1 + 1 = n + k + 3. \\ Coupling\ (m1) = n + k + 1. \\ Cohesion\ (m1) = 1/2. \end{array}\right.$$

Query $m2$: *Select all tuples*

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R \end{array}\right\} \Rightarrow \left\{\begin{array}{c} Size(m1) = n + k. \\ Coupling\ (m2) = n + k. \\ Cohesion\ (m2) = 1. \end{array}\right.$$

$Size(S) = (n + k + 2) + (n + k + 3) + (n + k) = 3 * n + 3 * k + 5.$

Σχήμα 5.9 Αναπαράσταση και μετρικές της σχεδιαστικής λύσης, όπου δημιουργούνται *k* πεδία για *k* τιμές του πλειότιμου γνωρίσματος ώστε να ισχύει η 1NF.



$Coupling(m1) = 0$
$Coupling(m2) = 1.$
$Cohesion(m1) = Cohesion(m2) = 1.$

$Size(PS) = ((n-1)+2)+5 = n+6.$
$Coupling(PS) = 0.$
$Cohesion(PS) = 1.$
$Data\_Volume(PS) = \#rows_{R1} * (sizeof(A_1) + \sum_{j=3}^{n} sizeof(A_j)) + \#rows_{R2} * (sizeof(A_1) + sizeof(A_2)),$
$where\ (\#rows_{R2} \approx average\ number\ of\ values\ for\ each\ A_1 = k).$

Query $m3$: *Select a specific tuple*

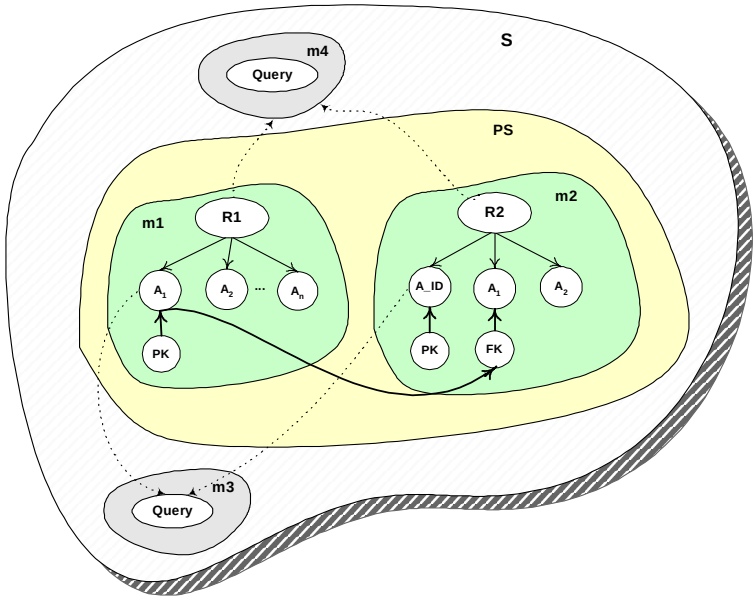
$$\left.\begin{array}{c} SELECT\ * \\ FROM\ \ R1, R2 \\ WHERE\ \ R1.A_1 = term\ AND\ R1.A_1 = R2.A_1 \end{array}\right\} \Rightarrow \left\{\begin{array}{c} Size(m3) = ((n-1)+1)+2*2+1 = n+5. \\ Coupling(m3) = (n-1)+1+2 = n+2. \\ Cohesion(m3) = 1/3. \end{array}\right.$$

Query $m4$: *Select all tuples*

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ \ R1, R2 \\ WHERE\ \ R1.A_1 = R2.A_1 \end{array}\right\} \Rightarrow \left\{\begin{array}{c} Size(m4) = ((n-1)+1)+2*1+1 = n+3. \\ Coupling(m4) = (n-1)+1+2 = n+2. \\ Cohesion(m4) = 1/2. \end{array}\right.$$

$Size(S) = (n+6)+(n+5)+(n+3) = 3*n+14.$

Σχήμα 5.10 Αναπαράσταση και μετρικές της σχεδιαστικής λύσης, όπου εξασφαλίζεται η 1NF με την κατασκευή ενός νέου πίνακα, ο οποίος έχει ως πρωτεύον κλειδί τον συνδυασμό γνωρισμάτων .



$Coupling(m1) = 0$
$Coupling(m2) = 1.$
$Cohesion(m1) = Cohesion(m2) = 1.$

$Size(PS) = ((n-1)+2)+6 = n+7.$
$Coupling(PS) = 0.$
$Cohesion(PS) = 1.$
$Data\_Volume(PS) = \#rows_{R1} * (sizeof(A_1) + \sum_{j=3}^{n} sizeof(A_j)) +$
$\#rows_{R2} * (sizeof(A\_ID) + sizeof(A_1) + sizeof(A_2)),$
where $(\#rows_{R2} \approx$ *average number of values for each* $A_1 = k)$.

Query $m3$: *Select a specific tuple*

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ \ R1, R2 \\ WHERE\ \ \ R1.A_1 = term\ AND\ R1.A_1 = R2.A_1 \end{array}\right\} \Rightarrow \left\{\begin{array}{c} Size(m3) = ((n-1)+2)+(2*2+1) = n+6. \\ Coupling(m3) = (n-1)+1+2 = n+2. \\ Cohesion(m3) = 1/3. \end{array}\right.$$

Query $m4$: *Select all tuples*

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ \ R1, R2 \\ WHERE\ \ \ R1.A_1 = R2.A_1 \end{array}\right\} \Rightarrow \left\{\begin{array}{c} Size(m4) = ((n-1)+1)+(2*1+1) = n+3. \\ Coupling(m4) = (n-1)+1+2. \\ Cohesion(m4) = 1/2. \end{array}\right.$$

$Size(S) = (n+6)+(n+6)+(n+3) = 3*n+15.$

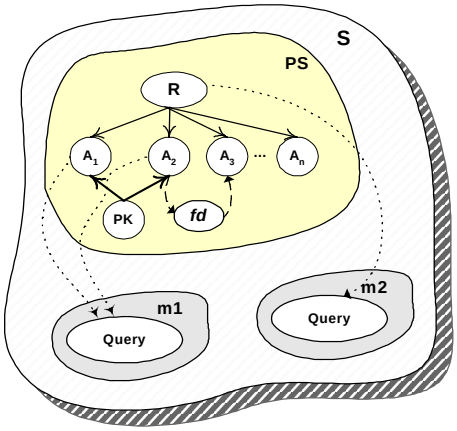
Σχήμα 5.11 Αναπαράσταση και μετρικές της σχεδιαστικής λύσης, όπου εξασφαλίζεται η 1NF με την κατασκευή ενός νέου πίνακα, ο οποίος έχει ως πρωτεύον κλειδί ένα τεχνητό κλειδί.

- **2NF**

Η δεύτερη κανονική μορφή «απαγορεύει» την ύπαρξη μερικών εξαρτήσεων γνωρισμάτων από μέρος του πρωτεύοντος κλειδιού, με σκοπό την αποφυγή των φαινόμενων πλεονασμού και ασυνέπειας των τιμών. Οι εναλλακτικές σχεδιαστικές λύσεις είναι οι εξής:

1. διατήρηση των μερικών εξαρτήσεων παραβιάζοντας τη 2NF.
2. δημιουργία ξεχωριστής σχέσης για κάθε μερική εξάρτηση.

Θεωρούμε τη γενική περίπτωση της σχέσης R(A1,A2,A3, …,An), με πρωτεύον κλειδί το συνδυασμό των πεδίων (A1,A2). Η μερική εξάρτηση, η οποία παραβιάζει τη 2NF, είναι A2→A3. Στο σχήμα 5.15 παρουσιάζεται η σχεδίαση, η οποία παραβιάζει τη 2NF. Στο Σχήμα 5.16 παρουσιάζεται η σχεδίαση της κανονικοποίησης, όπου η σχέση R μετατρέπεται σε R1(A1,A3,…,An), ενώ παράλληλα δημιουργείται μια νέα σχέση η R2 (A2,A3), η οποία έχει πρωτεύον κλειδί το συνδυασμό των πεδίων (A2, A3).



$Size(PS) = n + 3.$

$Coupling(PS) = 0.$

$Cohesion(PS) = 1/2.$

$Data\_Volume(PS) = \#rows_R * (\sum_{i=1}^{n} sizeof(A_i)).$

Query $m1$: *Select a specific tuple*

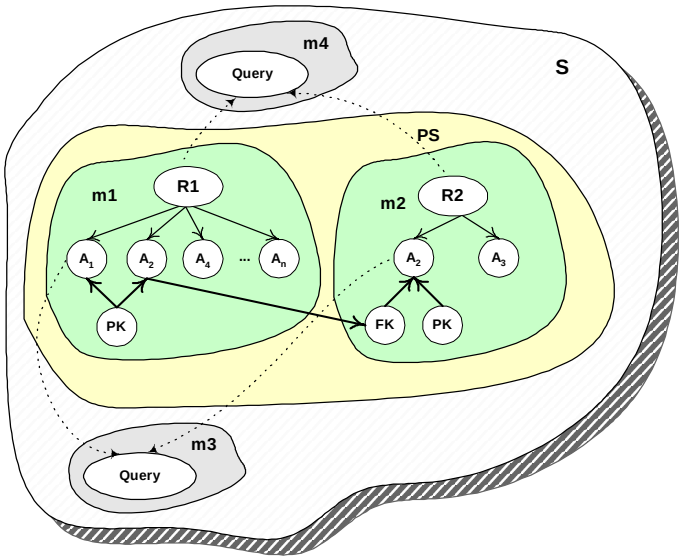
$$\left.\begin{array}{c} SELECT\ * \\ FROM\ \ R \\ WHERE\ \ A_1 = term\ AND\ A_2 = term \end{array}\right\} \Rightarrow \left\{\begin{array}{c} Size(m1) = n + 2 * 2 + 1 = n + 5. \\ Coupling(m1) = n + 2. \\ Cohesion(m1) = 1/3. \end{array}\right.$$

Query $m2$: *Select all tuples*

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ \ R \end{array}\right\} \Rightarrow \left\{\begin{array}{c} Size(m2) = n. \\ Coupling(m2) = n. \\ Cohesion(m2) = 1. \end{array}\right.$$

$Size(S) = (n + 3) + (n + 5) + n = 3 * n + 8.$

Σχήμα 5.12 Αναπαράσταση και μετρικές της σχεδιαστικής λύσης, όπου παραβιάζεται η 2NF.



$Coupling(m1) = 0$

$Coupling(m2) = 1.$

$Cohesion(m1) = Cohesion(m2) = 1.$

$Size(PS) = (n+1) + 4 = n + 5.$

$Coupling(PS) = 0.$

$Cohesion(PS) = 1.$

$$Data\_Volume(MS) = \#rows_{R1} * (sizeof(A_1) + sizeof(A_2) + \sum_{i=4}^{n} sizeof(A_i)) + \#rows_{R2} * (sizeof(A_2) + sizeof(A_3)).$$

Query $m3$: *Select a specific tuple*

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ \ R1, R2 \\ WHERE\ \ R1.A_1 = term\ AND\ R1.A_2 = term\ AND\ R1.A_2 = R2.A_2 \end{array}\right\} \Rightarrow \left\{\begin{array}{l} Size(m3) = n + 2*3 + 1 = n + 7. \\ Coupling(m3) = n + 3. \\ Cohesion(m3) = 1/4. \end{array}\right.$$

Query $m4$: *Select all tuples*

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ \ R1, R2 \\ WHERE\ \ R1.A_2 = R2.A_2 \end{array}\right\} \Rightarrow \left\{\begin{array}{l} Size(m4) = n + 2*1 + 1 = n + 3. \\ Coupling(m4) = n + 2. \\ Cohesion(m4) = 1/2. \end{array}\right.$$

$Size(S) = (n+5) + (n+7) + (n+3) = 3*n + 15.$

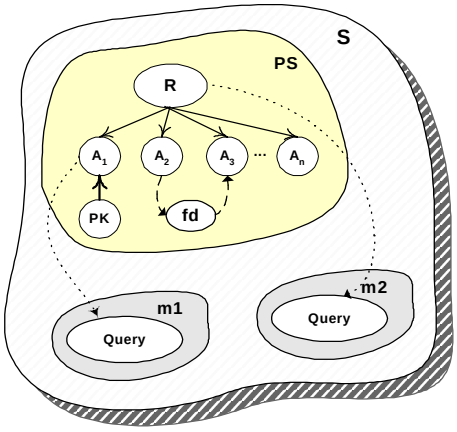
Σχήμα 5.13 Αναπαράσταση και μετρικές της σχεδιαστικής λύσης, όπου εφαρμόζεται η 2NF.

- **3NF**

Η τρίτη κανονική μορφή προϋποθέτει την ύπαρξη της 2NF και επιπλέον «απαγορεύει» την ύπαρξη μεταβατικών εξαρτήσεων, κατά τις οποίες γνωρίσματα που δεν ανήκουν στο πρωτεύον κλειδί προσδιορίζουν άλλα γνωρίσματα. Οι εναλλακτικές σχεδιαστικές λύσεις είναι οι εξής:

1. διατήρηση των μεταβατικών εξαρτήσεων παραβιάζοντας τη 3NF.
2. δημιουργία ξεχωριστής σχέσης για κάθε μεταβατική εξάρτηση.

Θεωρούμε τη γενική περίπτωση της σχέσης R(A1,A2,A3,A4 …,An), με πρωτεύον κλειδί το πεδίο A1. Η εξάρτηση, η οποία παραβιάζει τη 3NF, είναι A2→A3. Στο σχήμα 5.14 παρουσιάζεται η σχεδίαση, η οποία παραβιάζει τη 3NF. Στο Σχήμα 5.15 παρουσιάζεται η σχεδίαση της κανονικοποίησης, όπου η σχέση R μετατρέπεται σε R1(A1,A2,A4…,An), ενώ παράλληλα δημιουργείται μια νέα σχέση η R2 (A2,A3), η οποία έχει πρωτεύον κλειδί το συνδυασμό των πεδίων (A2, A3).



$Size(PS) = n + 3.$

$Coupling\ (PS) = 0.$

$Cohesion\ (PS) = \frac{1}{2}.$

$Data\_Volume(PS) = \#rows_R * (\sum_{i=1}^{n} sizeof\ (A_i)).$

Query $m1$: *Select a specific tuple*

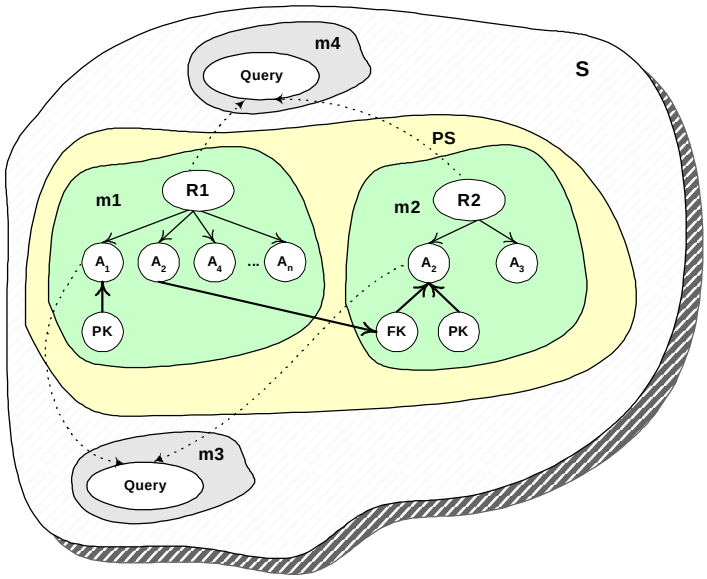
$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R \\ WHERE\ A_1 = term \end{array}\right\} \Rightarrow \left\{\begin{array}{c} Size(m1) = n + 2*1 + 1 = n + 3. \\ Coupling\ (m1) = n + 1. \\ Cohesion\ (m1) = \frac{1}{2}. \end{array}\right.$$

Query $m2$: *Select all tuples*

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R \end{array}\right\} \Rightarrow \left\{\begin{array}{l} Size(m2) = n. \\ Coupling\ (m2) = n. \\ Cohesion\ (m2) = 1. \end{array}\right.$$

$Size(S) = (n + 3) + (n + 3) + n = 3 * n + 6.$

Σχήμα 5.14 Αναπαράσταση και μετρικές της σχεδιαστικής λύσης, όπου παραβιάζεται η 3NF.



$Coupling\ (m1) = 0$
$Coupling\ (m2) = 1.$
$Cohesion\ (m1) = Cohesion\ (m2) = 1.$

$Size(PS) = ((n-1)+2)+5 = n+6.$
$Coupling\ (PS) = 0.$
$Cohesion\ (PS) = 1.$
$Data\_Volume\ (PS) = \#rows_{R1} * (sizeof\ (A_1) + sizeof\ (A_2) + \sum_{i=4}^{n} sizeof\ (A_i)) +$
$\#rows_{R2} * (sizeof\ (A_2) + sizeof\ (A_3)).$

Query $m3$: *Select a specific tuple*

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R1, R2 \\ WHERE\ R1.A_1 = term\ AND\ R1.A_2 = R2.A_2 \end{array}\right\} \Rightarrow \left\{\begin{array}{c} Size(m3) = n + 2*2+1 = n+5. \\ Coupling\ (m3) = n+2. \\ Cohesion\ (m3) = 1/3. \end{array}\right.$$

Query $m4$: *Select all tuples*

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R1, R2 \\ WHERE\ R1.A_2 = R2.A_2 \end{array}\right\} \Rightarrow \left\{\begin{array}{c} Size(m4) = n + 2*1+1 = n+3. \\ Size(m4) = n + 2*1+1 = n+3. \\ Cohesion\ (m4) = 1/2. \end{array}\right.$$

$Size(S) = (n+6)+(n+5)+(n+3) = 3*n+14.$

Σχήμα 5.15 Αναπαράσταση και μετρικές της σχεδιαστικής λύσης, όπου εφαρμόζεται η 3NF.

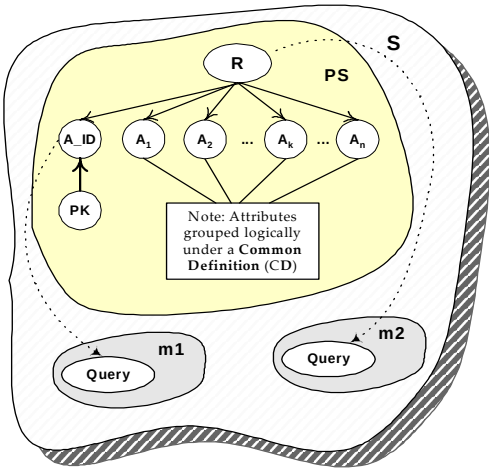
5.3.3. Αξονική Περιστροφή (Pivot Case)

Πολλές φορές κατά τη σχεδίαση μιας βάσης υπάρχουν γνωρίσματα, τα οποία μπορούν να ομαδοποιηθούν σε μια κοινή κατηγορία. Ένα καλό παράδειγμα αυτής της περίπτωσης αποτελεί η καταγραφή των οικονομικών (Μισθός, Επίδομα, Φόρος, Καθαρό Εισόδημα) ενός εργαζομένου μιας εταιρίας. Οι δύο εναλλακτικές σχεδιαστικές λύσεις είναι οι εξής:

1. Flat design, όπου παρατίθενται όλα τα πεδία.
2. Encoded design, όπου δημιουργούνται δύο σχέσεις. Η μία σχέση (master relation) καταχωρεί πληροφορία με τη μορφή κωδικών και τιμών, ενώ η δεύτερη (lookup relation) περιλαμβάνει τους κωδικούς και τις περιγραφές του κάθε ομαδοποιημένου γνωρίσματος.

Η μετάβαση από το encoded στο flat design ονομάζεται pivot.

Θεωρούμε τη γενική περίπτωση της σχέσης `R(A_ID,A1,A2,A3,A4 …,An)`, με πρωτεύον κλειδί το πεδίο `A_ID`. Τα γνωρίσματα `A1,A2,A3,A4 …,An` μπορούν να ομαδοποιηθούν υπό μια κοινή έννοια (CD-Common Definition). Στο Σχήμα 5.16 παρουσιάζεται η flat σχεδίαση, ενώ στο Σχήμα 5.17 μελετάται η encoded σχεδίαση.



$Size(PS) = n + 3.$

$Coupling(PS) = 0.$

$Cohesion(PS) = 1.$

$Data\_Volume(PS) = \#rows_R * (\sum_{i=1}^{n} sizeof(A_i)).$

Query $m1$: *Select a specific tuple*

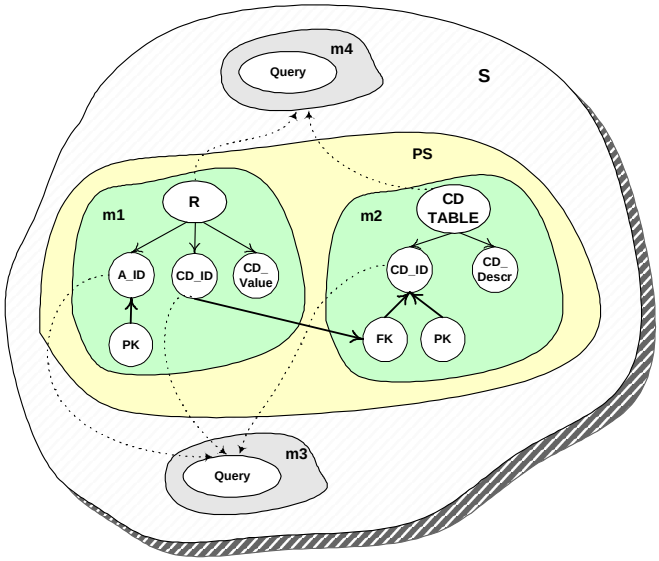
$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R \\ WHERE\ \ A\_ID = term \end{array}\right\} \Rightarrow \begin{cases} Size(m1) = n + 1 + 2 * 1 + 1 = n + 4. \\ Coupling(m1) = n + 2. \\ Cohesion(m1) = 1/2. \end{cases}$$

Query $m2$: *Select all tuples*

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R \end{array}\right\} \Rightarrow \begin{cases} Size(m2) = n + 1. \\ Coupling(m2) = n + 1. \\ Cohesion(m2) = 1. \end{cases}$$

$Size(S) = (n + 3) + (n + 4) + n + 1 = 3 * n + 8.$

Σχήμα 5.16 Αναπαράσταση και μετρικές με χρήση flat design.



$Coupling(m1) = 0$
$Coupling(m2) = 1.$
$Cohesion(m1) = Cohesion(m2) = 1.$

$Size(PS) = 10.$
$Coupling(PS) = 0.$
$Cohesion(PS) = 1.$
$Data\_Volume(MS) = \#rows_R * (\sum_{i=i}^{3} sizeof(A_i)) + \#rows_{CD\_TABLE} * (\sum_{j=1}^{2} sizeof(A_j)).$

Query $m3$: *Select a specific tuple*

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ \ R, CD\_TABLE \\ WHERE\ \ R.A\_ID = term\ AND\ R.CD\_ID\ = CD\_TABLE.A_1 \end{array}\right\} \Rightarrow \left\{\begin{array}{l} Size(m3) = 4 + 2 * 2 + 2 = 10. \\ Coupling(m3) = 4 + 3 = 7. \\ Cohesion(m3) = 1/3. \end{array}\right.$$

Query $m4$: *Select all tuples*

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ \ R, CD\_TABLE \\ WHERE\ \ R.CD\_ID\ AND\ CD\_TABLE.A_1 \end{array}\right\} \Rightarrow \left\{\begin{array}{l} Size(m4) = 4 + 2 * 1 + 1 = 7. \\ Coupling(m4) = 4 + 2 = 6. \\ Cohesion(m4) = 1/2. \end{array}\right.$$

$Size(S) = 10 + 9 + 7 = 26.$

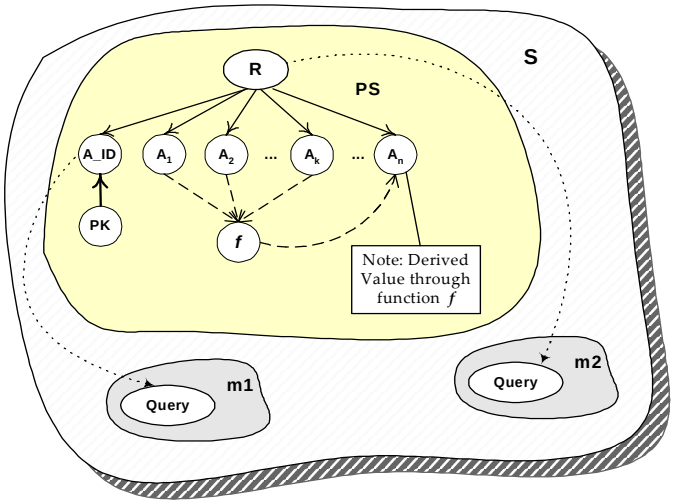
Σχήμα 5.17 Αναπαράσταση και μετρικές με χρήση encoded design.

5.3.4. Παραγόμενα Γνωρίσματα (Derived Attributes)

Το πρόβλημα, στο οποίο αντιστοιχίζεται το συγκεκριμένο πρότυπο, αφορά στα πεδία τα οποία μπορούν να παραχθούν μέσω μιας συνάρτησης υπολογισμού, από άλλα πεδία μιας σχέσης. Οι εναλλακτικές σχεδιαστικές λύσεις είναι οι εξής:

1. δημιουργία ενός πεδίου, που αντιστοιχίζεται στο παραγόμενο γνώρισμα, και του οποίου η τιμή υπολογίζεται με την βοήθεια ενός trigger, μέσω μιας συνάρτησης υπολογισμού.
2. δημιουργία μιας υλοποιήσιμης όψης (materialized view), η οποία περιλαμβάνει το παραγόμενο γνώρισμα.
3. δημιουργία μιας εικονικής όψης (virtual view), η οποία παράγει κάθε φορά τις τιμές του παραγόμενου γνωρίσματος.

Θεωρούμε τη γενική περίπτωση της σχέσης `R(A_ID,A1,A2,A3,A4 …,An)`, με πρωτεύον κλειδί το πεδίο `A_ID`. Το γνώρισμα `An` παράγεται μέσω μιας συνάρτησης υπολογισμού, η οποία εφαρμόζεται στα γνωρίσματα `A1,A2,A3,…,Ak`. Στο Σχήμα 5.18 παρουσιάζεται η σχεδίαση, κατά την οποία το παραγόμενο γνώρισμα αποτελεί μέρος της σχέσης και η τιμή του υπολογίζεται κάθε φορά μέσω της συνάρτησης *f*. Το Σχήμα 5.19 παρουσιάζει τη περίπτωση κατασκευής μιας materialized view, ενώ το Σχήμα 5.20 τη περίπτωση δημιουργίας μιας virtual view.



$Size(PS) = n + 4.$

$Coupling(PS) = 0.$

$Cohesion(PS) = \frac{1}{2}.$

$Data\_Volume(PS) = \#rows_R * (\sum_{i=1}^{n-1} sizeof(A_i) + sizeof(A_n)).$

Query $m1$: *Select a specific tuple*

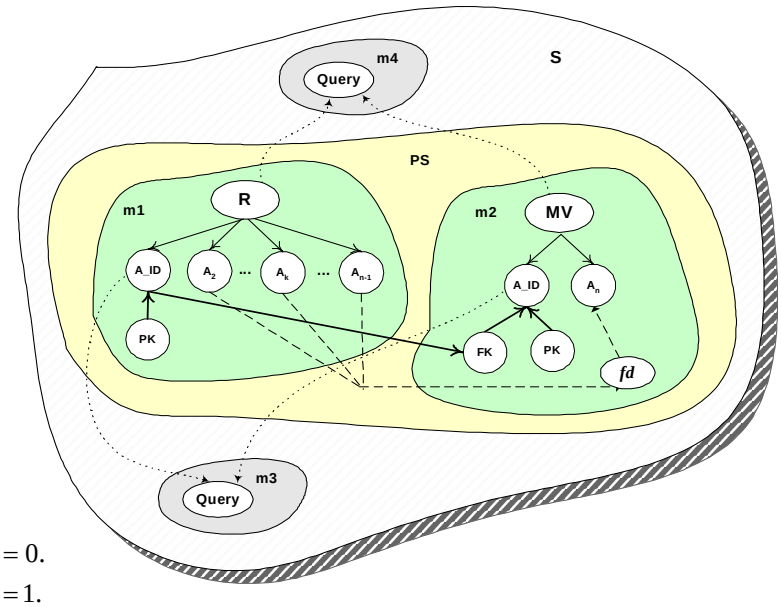
$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R \\ WHERE\ \ A\_ID = term \end{array}\right\} \Rightarrow \begin{cases} Size(m1) = n + 1 + 2 * 1 + 1 = n + 4. \\ Coupling(m1) = n + 2. \\ Cohesion(m1) = \frac{1}{2}. \end{cases}$$

Query $m2$: *Select all tuples*

$$\left.\begin{array}{l} SELECT\ * \\ FROM\ R \end{array}\right\} \Rightarrow \begin{cases} Size(m2) = n + 1. \\ Coupling(m2) = n + 1. \\ Cohesion(m2) = 1. \end{cases}$$

$Size(S) = (n + 4) + (n + 4) + (n + 1) = 3 * n + 9.$

Σχήμα 5.18 Αναπαράσταση και μετρικές της σχεδιαστικής λύσης, όπου ο πίνακας περιλαμβάνει το παραγόμενο γνώρισμα.



$Coupling(R) = 0.$

$Cohesion(R) = 1.$

Materialized View (MV) : *When Calculates derived attribute* A_n

$$\left.\begin{array}{c} SELECT\ *,\ A_n = f(A_1,..,A_k) \\ FROM\ R \end{array}\right\} \Rightarrow \begin{cases} Size(MV) = n+1. \\ Coupling(MV) = n+2. \\ Cohesion(MV) = 1/2. \end{cases}$$

$$Materialized\ View\ as\ a\ table: \begin{cases} Size(MV) = 6. \\ Coupling = 2. \\ Cohesion = 1. \end{cases}$$

$Size(PS) = (n-1)+3+5 = n+7.$

$Coupling(PS) = 0.$

$Cohesion(PS) = 3/4.$

$$\left.\begin{array}{c} Data\_Volume(PS) = \#rows_R * (\sum_{i=1}^{n-1} sizeof(A_i)) + \#rows_{MV} * (\sum_{j=1}^{2} sizeof(A_j)) \\ but\ \#rows_R = \#rows_{MV} \end{array}\right\} \Rightarrow$$

$$\Rightarrow Data\_Volume(PS) = \#rows_R * (2 * sizeof(A_1) + \sum_{i=2}^{n} sizeof(A_i)).$$

Query $m3$: *Select a specific tuple*

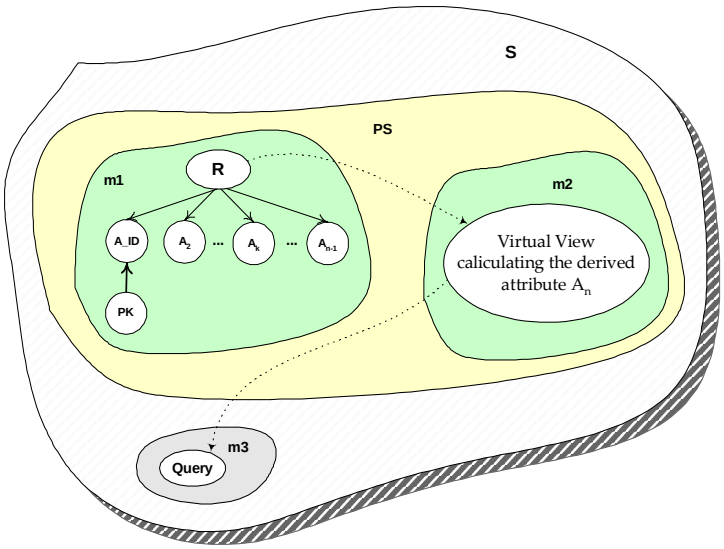
$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R, MV \\ WHERE\ \ R.A\_ID = term\ AND\ R.A_1 = MV.A_1 \end{array}\right\} \Rightarrow \begin{cases} Size(m3) = (n+1)+2*2+1 = n+6. \\ Coupling(m3) = (n+1)+3 = n+4. \\ Cohesion(m3) = 1/3. \end{cases}$$

Query $m4$: *Select all tuples*

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R, CD\_TABLE \\ WHERE\ \ R.A_1 = CD\_TABLE.A_1 \end{array}\right\} \Rightarrow \begin{cases} Size(m4) = (n+1)+2*1+1 = n+4. \\ Coupling(m4) = (n+1)+2 = n+3. \\ Cohesion(m4) = 1/2. \end{cases}$$

$Size(S) = (n+7)+(n+6)+(n+4) = 3*n+17.$

Σχήμα 5.19 Αναπαράσταση και μετρικές της σχεδιαστικής λύσης, όπου κατασκευάζεται μια Materialized View (*MV*), όπου καταχωρείται το παραγόμενο γνώρισμα.



$Coupling(R) = 1.$
$Cohesion(R) = 1.$

View $m2$: *Select all tuples*

$$\left.\begin{array}{c} SELECT\ *, A_n = f(A_1,..,A_k) \\ FROM\ R \end{array}\right\} \Rightarrow \begin{cases} Size(m2) = n+2. \\ Coupling(m2) = n+1. \\ Cohesion(m2) = 1/2. \end{cases}$$

$Size(PS) = (n+1).$
$Coupling(PS) = 0.$
$Cohesion(PS) = 1.$
$Data\_Volume(PS) = \#rows_R * (\sum_{i=1}^{n-1} sizeof(A_i)).$

Query $m3$: *Select a specific tuple*

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ View \\ WHERE\ \ A\_ID = term \end{array}\right\} \Rightarrow \begin{cases} Size(m3) = n+1+2*1+1 = n+4. \\ Coupling(m3) = n+2. \\ Cohesion(m3) = 1/2. \end{cases}$$

$Size(S) = (n+1) + (n+2) + (n+4) = 3*n+7.$

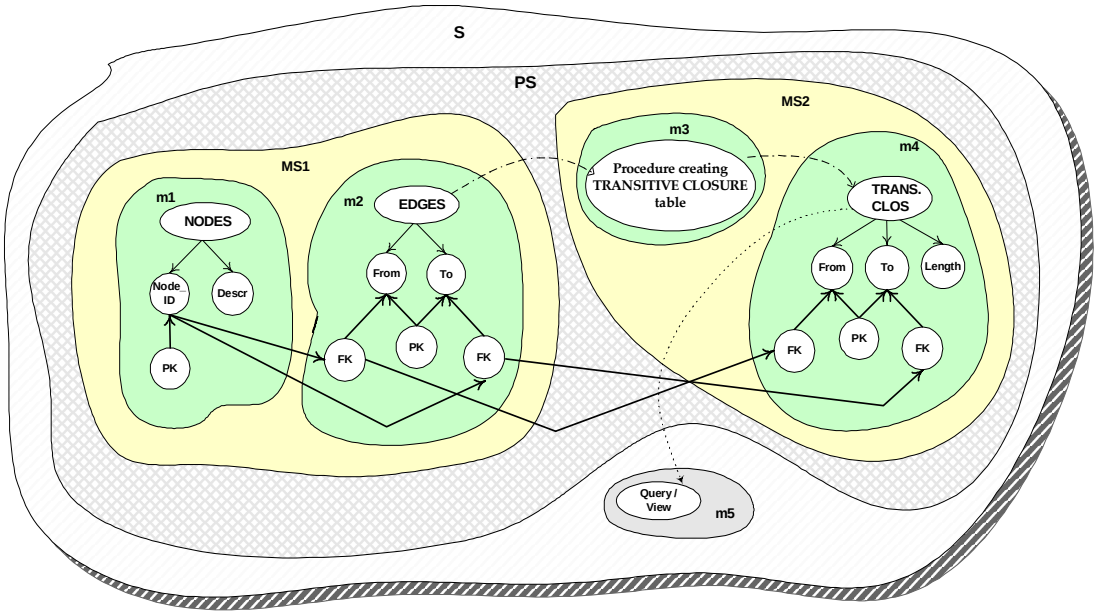
Σχήμα 5.20 Αναπαράσταση και μετρικές της σχεδιαστικής λύσης, όπου κατασκευάζεται μια Virtual View (*VV*), όπου υπολογίζεται το παραγόμενο γνώρισμα.

5.3.5. Γράφοι, Αναδρομική Κλειστότητα, Συνάθροιση-Σύνθεση

Σε αυτή την ενότητα παρουσιάζεται το πρότυπο το οποίο περιγράφει την περίπτωση ενός γράφου, και κατά συνέπεια τις περιπτώσεις συνάθροιση-σύνθεση. Η εύρεση όλων των δυνατών μονοπατιών, εφόσον δεν υπάρχει κύκλος στο γράφο, αποτελεί μια πολύ σημαντική λειτουργία, η οποία και μελετάται εκτενώς εδώ. Η πρώτη

σχεδιαστική λύση περιλαμβάνει τη κατασκευή μιας σχέσης καταγραφή όλων των μονοπατιών μέσω μιας διαδικασία υπολογισμού της αναδρομικής κλειστότητας του γράφου. Στη δεύτερη λύση εφαρμόζεται η διαδικασία εύρεσης μονοπατιών, η οποία μπορεί να περιλαμβάνει cursors, χωρίς να καταχωρεί τα αποτελέσματα.

Θεωρούμε τη γενική περίπτωση όπου η σχέση `NODES(Node_ID,Descr)`, με πρωτεύον κλειδί το πεδίο `Node_ID`, καταχωρεί τους κόμβους τους γράφου, ενώ η σχέση `EDGES(From, To)`, με πρωτεύον κλειδί το συνδυασμό `(From, To)`, καταχωρεί τις ακμές του γράφου. Στο Σχήμα 5.21 παρουσιάζεται η σχεδίαση, κατά την οποία κατασκευάζεται μια νέα σχέση `TRANS.CLOS(From,To,Length)`, με πρωτεύον κλειδί το συνδυασμό των πεδίων `(From,To)`, στην οποία αποθηκεύονται όλα τα δυνατά μονοπάτια του γράφου και το μήκος τους. Ο υπολογισμός όλων των δυνατών μονοπατιών πραγματοποιείται μέσω της μεθόδου υπολογισμού της αναδρομικής κλειστότητας, που παρουσιάστηκε στο κεφάλαιο 4. Στο Σχήμα 5.23 δίδεται η δεύτερη σχεδιαστική λύση, η οποία υπολογίζει, χωρίς να καταχωρεί, τα μονοπάτια του γράφου.



Σχήμα 5.21 Αναπαράσταση της σχεδιαστικής λύσης, όπου κατασκευάζεται πίνακας αναδρομικής κλειστότητας με την μέθοδο του κεφαλαίου 4 ενότητα 4.5.

$$\left.\begin{array}{l} Size\,(m1) = 4. \\ Coupling\ \ (m1) = 0 \\ Cohesion\ \ (m1) = 1. \\ Size\,(m2) = 6. \\ Coupling\ \ (m2) = 2. \\ Cohesion\ \ (m2) = 1. \end{array}\right\} \Rightarrow \begin{cases} Size\,(MS1) = Size\,(m1) + Size\,(m2) = 10. \\ Coupling\ \ (MS1) = 0. \\ Cohesion\ \ (MS1) = 1 \\ Data\ \_Volume\ (MS1) = \#\,rows_{\ NODES} * (sizeof\ \ (ID) + sizeof\ \ (Descr)) + \\ \qquad\qquad \#\,rows_{\ EGDES} * (sizeof\ \ (From) + sizeof\ \ (To)). \end{cases}$$

Pr *ocedure* $m3$:

$$Step1.Copy\ \ table\ \ EDGES\ \ to\ TRANS\ .CLOS\ .\ table \Rightarrow \begin{cases} Size\,(Step1) = 3 + 3 = 6. \\ Coupling\ \ (Step1) = 3. \\ Cohesion\ \ (Step1) = 1. \end{cases}$$

$$Step\,2.Find\ \ paths\ \ with\ \ length\ +1\ and\ \ insert\ \ in\ temporary\ \ table \Rightarrow \begin{cases} Size\,(Step\,2) = 9. \\ Coupling\ \ (Step\,2) = 7. \\ Cohesion\ \ (Step\,2) = 1/3. \end{cases}$$

$$Step\,3.Copy\ \ temporary\ \ table\ to\ TRANS\ .CLOS\ .\ table \Rightarrow \begin{cases} Size\,(Step\,3) = 3. \\ Coupling\ \ (Step\,3) = 3 \\ Cohesion\ \ (Step\,3) = 1. \end{cases}$$

$$Step1\ and\ \ Step\,2\ and\ \ Step\,3 \Rightarrow \begin{cases} Size\,(m3) = 18. \\ Coupling\ \ (m3) = 13. \\ Cohesion\ \ (m3) = 7/9. \end{cases}$$

$Size\,(m4) = 7.$

$Coupling\ \ (m4) = 2 + 1 = 3.$

$Cohesion\ \ (m4) = 1.$

$Data\ \_Volume\ (m4) = \#\,rows_{\ TRANSITIVE\ \_CLOSURE\ \_TABLE} * (sizeof\ \ (From) + sizeof\ \ (To) + sizeof\ \ (Length)).$

$Size\,(MS\,2) = 25.$

$Coupling\ \ (MS\,2) = 15.$

$$Cohesion\ \ (MS\,2) = \frac{\frac{7}{9} + 1}{2} = \frac{8}{9}.$$

Query $m4$: *Searching graph for...*

$$\left.\begin{array}{ll} i) & the\ root \\ ii) & leaves \end{array}\right\} \Rightarrow \begin{cases} Size\,(m4) = 4. \\ Coupling\ \ (m4) = 2. \\ Cohesion\ \ (m4) = 1/2. \end{cases}$$

iii) *a path* :

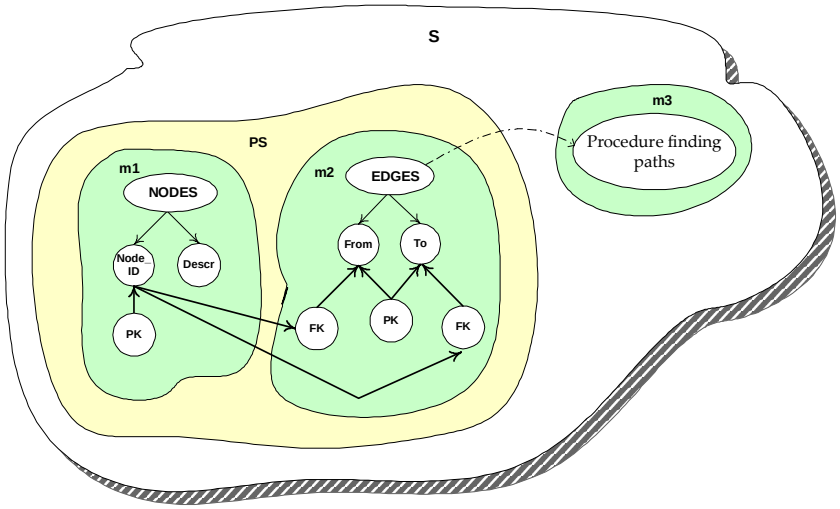
$$\left.\begin{array}{c} SELECT\ \ * \\ FROM\ \ TRANS\ .CLOS \\ WHERE\ \ From\ = term\ \ AND\ \ To\ = term \end{array}\right\} \Rightarrow \begin{cases} Size\,(m4) = 8. \\ Coupling\ \ (m4) = 5. \\ Cohesion\ \ (m4) = 1/3 \end{cases}$$

iv) *all paths* :

$$\left.\begin{array}{c} SELECT\ \ * \\ FROM\ \ TRANS\ .CLOS \end{array}\right\} \Rightarrow \begin{cases} Size\,(m4) = 3. \\ Coupling\ \ (m4) = 3. \\ Cohesion\ \ (m4) = 1. \end{cases}$$

$Size\,(S) = 10 + 25 + 4 + 8 + 3 = 50.$

Σχήμα 5.22 Μετρικές της σχεδιαστικής λύσης, όπου κατασκευάζεται πίνακας αναδρομικής κλειστότητας με την μέθοδο του κεφαλαίου 4 ενότητα 4.5.



$Size(PS) = 8$

$Coupling(m1) = Coupling(m2) = 2.$

$Cohesion(m1) = 0$

$Cohesion(m2) = 1.$

$Coupling(PS) = 0.$

$Cohesion(PS) = 1.$

$Data\_Volume(PS) = \#rows_{NODES} * (sizeof(ID) + sizeof(Descr)) + \#rows_{EGDES} * (sizeof(From) + sizeof(To)).$

$\Pr ocedure\ m3\ is\ the\ same\ as\ previous\ MS2,$

given a node From = term finds all the nodes that it reaches through a path.

$Size(S) = 8 + size(\Pr ocedure) = 8 + 18 = 26.$

Σχήμα 5.23 Αναπαράσταση και μετρικές της σχεδιαστικής λύσης, όπου για την εύρεση των μονοπατιών ενός γράφου εκτελείται μια διαδικασία.

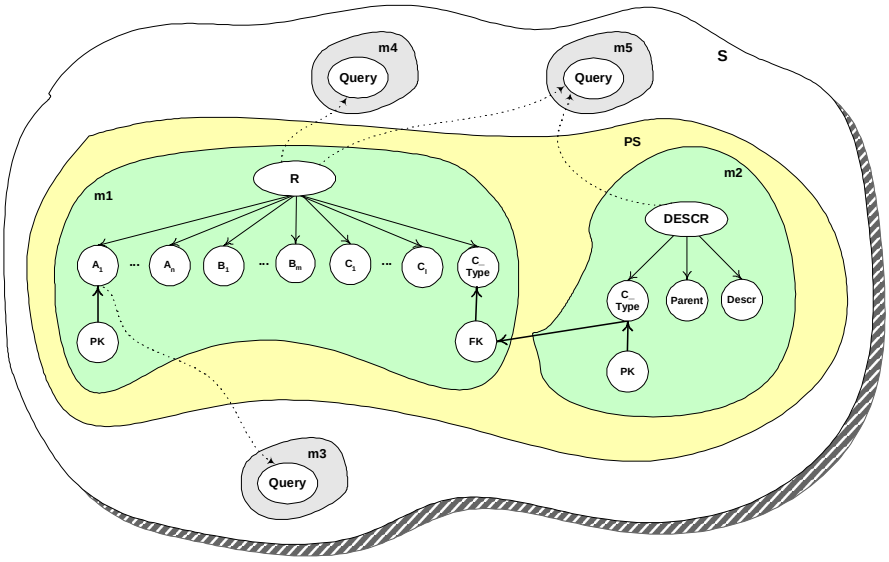
5.3.6. Γενίκευση-Εξειδίκευση (IsA)

Η γενίκευση είναι μια διαδικασία κατά την οποία διάφορες σχέσεις (υποκλάσεις) γενικεύονται σε μια, πιο αφηρημένη, σχέση (υπερκλάση). Η υπερκλάση περιλαμβάνει τις ομοιότητες των υποκλάσεων. Για το πρότυπο της γενίκευσης προτείνονται πέντε εναλλακτικές σχεδιάσεις:

1. δημιουργία μιας σχέσης που περιλαμβάνει τα γνωρίσματα όλων των κλάσεων.
2. δημιουργία σχέσεων για κάθε κλάση. Η σχέση της υπερκλάσης αποτελεί πίνακα αναζήτησης.

3. δημιουργία ξεχωριστών σχέσεων για κάθε κλάση, χωρίς περιορισμούς αναφορικής ακεραιότητας.
4. δημιουργία ξεχωριστών σχέσεων για κάθε κλάση, με περιορισμούς αναφορικής ακεραιότητας σε μια σχέση, η οποία αποτελεί πίνακα αναζήτησης.
5. δημιουργία ξεχωριστών σχέσεων για κάθε κλάση, με περιορισμούς αναφορικής ακεραιότητας στη σχέση της υπερκλάσης. Επιπλέον δημιουργείται μία σχέση, η οποία βοηθά στην αναζήτηση.

Θεωρούμε τη γενική περίπτωση όπου η σχέση A(A1,…,An) αποτελεί την υπερκλαση των σχέσεων B(B1,…,Bm) και C(C1,…,Cl). Στο Σχήμα 5.24 παρουσιάζεται η πρώτη σχεδιαστική λύση, όπου κατασκευάζεται η σχέση R(A1,…,An,B1,…,Bm,C1,…,Cl, Class_Type), με πρωτεύον κλειδί το A1. Η σχέση DESCR (Class_Type,Parent, Descr) καταγράφει την ιεραρχία των σχέσεων. Στο Σχήμα 5.25 παρουσιάζεται η δεύτερη σχεδιαστική λύση, όπου δημιουργούνται οι σχέσεις A(A1,…,An,Class_Type), B(A1,B1,…,Bm)και C(A1,C1,…,Cl). Στη σχέση A καταχωρούνται όλες οι εμφανίσεις των πρωτευόντων κλειδιών, όλων των σχέσεων. Η εύρεση της σχέσης στην οποία αντιστοιχεί κάθε τιμή του A1, πραγματοποιείται με αναζήτηση στον πίνακα DESCR. Στο Σχήμα 5.27 δίνεται η τρίτη σχεδιαστική λύση, όπου κατασκευάζονται ξεχωριστοί πίνακες A(A1,…,An), B(A1,…,An,B1,…,Bm) και C(A1,…,An,C1,…,Cl) για κάθε σχέση. Οι σχέσεις των υποκλάσεων περιλαμβάνουν τα πεδία τους καθώς και τα πεδία της υπερκλάσης. Η τέταρτη σχεδιαστική λύση, Σχήμα 5.28, δημιουργεί ξεχωριστές σχέσεις για κάθε κλάση, περιλαμβάνοντας τα πεδία της κάθε κλάσης και τα πεδία της υπερκλάσης, A(A1,…,An), B(A1,…,An,B1,…,Bm)και C(A1,…,An,C1,…,Cl). Η διαφορά, σε αυτή τη περίπτωση είναι, η ύπαρξη περιορισμών αναφορικής ακεραιότητας, από τα πρωτεύοντα κλειδιά κάθε σχέσης, στο πεδίο A1 της σχέσης ID_LOOKUP(A1,Table), στη οποία καταχωρούνται οι όλες οι τιμές των πρωτευόντων κλειδιών καθώς και η αντίστοιχη σχέση, όπου εμφανίζονται. Τέλος, η πέμπτη σχεδιαστική λύση, η οποία δημιουργεί τις σχέσεις A(A1,…,An,Class_Type), B(A1,…,An,B1,…,Bm) και C(A1,…,An,C1,…,Cl)παρουσιάζεται στο Σχήμα 5.29.



$$\left.\begin{array}{l} Size(m1) = n + m + l + 3. \\ Coupling(m1) = 1. \\ Cohesion(m1) = 1. \\ Size(m2) = 6. \\ Coupling(m2) = 0. \\ Cohesion(m2) = 1. \end{array}\right\} \Rightarrow \begin{cases} Size(PS) = n + m + l + 9. \\ Coupling(PS) = 0. \\ Cohesion(PS) = 1. \end{cases}$$

$$Data\_Volume(PS) = \#rows_R * (\sum_{i=1}^{n} sizeof(A_i) + \sum_{j=1}^{m} sizeof(B_j) + \sum_{k=1}^{l} sizeof(C_k) + sizeof(C\_Type)) + \#rows_{DESCR} * (sizeof(C\_Type) + sizeof(Parent) + sizeof(Descr)).$$

Query $m3$: *Select a tuple by defining ID*

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R \\ WHERE\ A_1 = term \end{array}\right\} \Rightarrow \begin{cases} Size(m3) = (n + m + l + 1) + 2 + 1 = n + m + l + 4. \\ Coupling(m3) = n + m + l + 1 + 2 = n + m + l + 3. \\ Cohesion(m3) = 1/2. \end{cases}$$

Query $m4$: *Select all tuples*

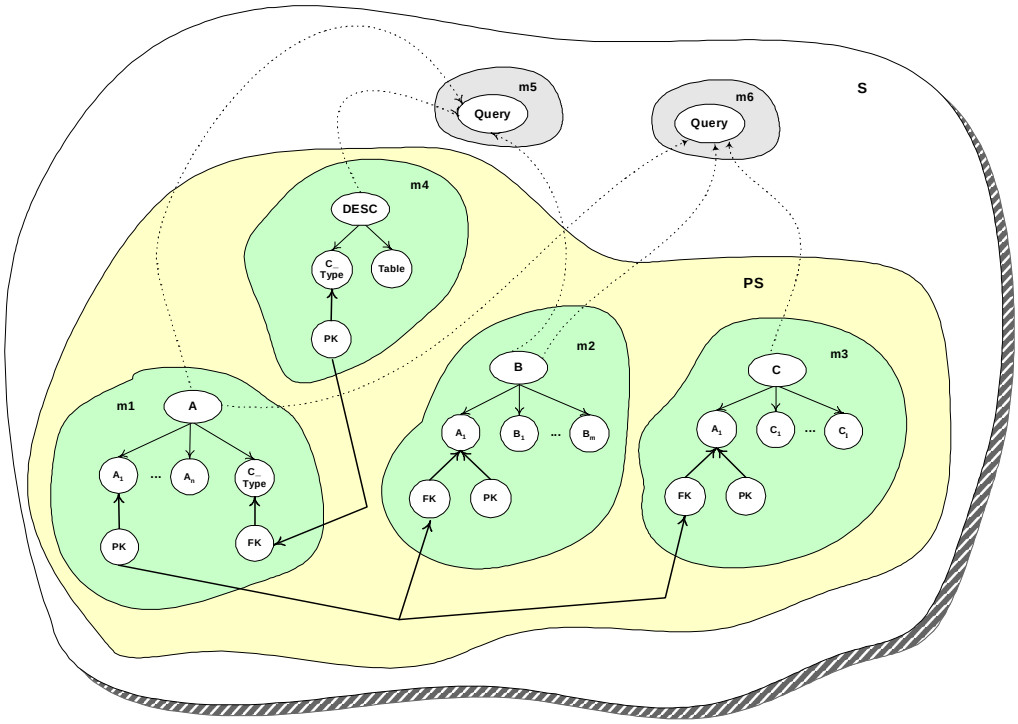
$$\left.\begin{array}{l} SELECT\ * \\ FROM\ R \end{array}\right\} \Rightarrow \begin{cases} Size(m4) = n + m + l + 1. \\ Coupling(m4) = n + m + l + 1. \\ Cohesion(m4) = 1. \end{cases}$$

Query $m5$: *Select tuples of a specific class*

$$\left.\begin{array}{c} SELECT\ * \\ FROM\ R, DESCR \\ WHERE\ DESCR.C\_Type = term\ AND\ DESCR.C\_Type = R.C\_Type \end{array}\right\} \Rightarrow \begin{cases} Size(m5) = n + m + l + 9. \\ Coupling(m5) = n + m + l + 6. \\ Cohesion(m5) = 1/3 \end{cases}$$

$$Size(S) = n + m + l + 9 + n + m + l + 4 + n + m + l + 1 + n + m + l + 9 = 4 * n + 4 * m + 4 * l + 23.$$

Σχήμα 5.24 Αναπαράσταση και μετρικές της $1^{ης}$ σχεδιαστικής λύσης, όπου ένας πίνακας περιλαμβάνει τα γνωρίσματα όλων των κλάσεων (A, B, C) .



Σχήμα 5.25 Αναπαράσταση της $2^{ης}$ σχεδιαστικής λύσης, όπου κατασκευάζεται ένας πίνακας για την υπερκλάση, `A`, βοηθητικός για την αναζήτηση, και ένας πίνακας για κάθε υποκλάση.

$$\left.\begin{array}{l} Size(m1) = n + 4. \\ Coupling(m1) = 1. \\ Cohesion(m1) = 1. \\ Size(m2) = m + 4. \\ Coupling(m2) = 1. \\ Cohesion(m2) = 1. \\ Size(m3) = l + 4. \\ Coupling(m3) = 1. \\ Cohesion(m3) = 1. \\ Size(m4) = 4. \\ Coupling(m4) = 0. \\ Cohesion(m4) = 1. \end{array}\right\} \Rightarrow \left\{\begin{array}{l} Size(PS) = n + m + l + 16. \\ Coupling(PS) = 0. \\ Cohesion(PS) = 1. \\ Data\_Volume(PS) = \#rows_A * (\sum_{i=1}^{n} sizeof(A_i) + sizeof(Class\_Type)) + \\ \qquad \#rows_B * (sizeof(A_1) + \sum_{j=1}^{m} sizeof(B_j)) + \\ \qquad \#rows_C * (sizeof(A_1) + \sum_{k=1}^{l} sizeof(C_k)) + \\ \qquad \#rows_{DESCR} * (sizeof(Class\_Type) + sizeof(Table)). \end{array}\right.$$

Query $m5$: *Select specific tuple that we don't know its class.*

$$\left.\begin{array}{l} SELECT\ \ DESCR.Table\ INTO : table \\ From\ \ A, DESCR \\ WHERE\ \ A.A_1 = term\ AND\ \ A.C\_Type = DESCR.C\_Type \end{array}\right\} Q_1$$

$$\left.\begin{array}{l} string\ \ x = 'SELECT\ *\ FROM\ \ A' \\ string\ \ y = 'WHERE\ \ A.A_1 = ' \\ execute(concatenate(x, : table, z, : table, '.A_1')) \end{array}\right\} Q_2$$

$$Size(m5) = Size(Q_1) + Size(Q_2) = 7 + ((Size(: table) - 3) + 3) = Size(: table) + 1.$$

$$Coupling(m5) = 4 + Size(: table) - 3 + 2 = Size(: table) + 3.$$

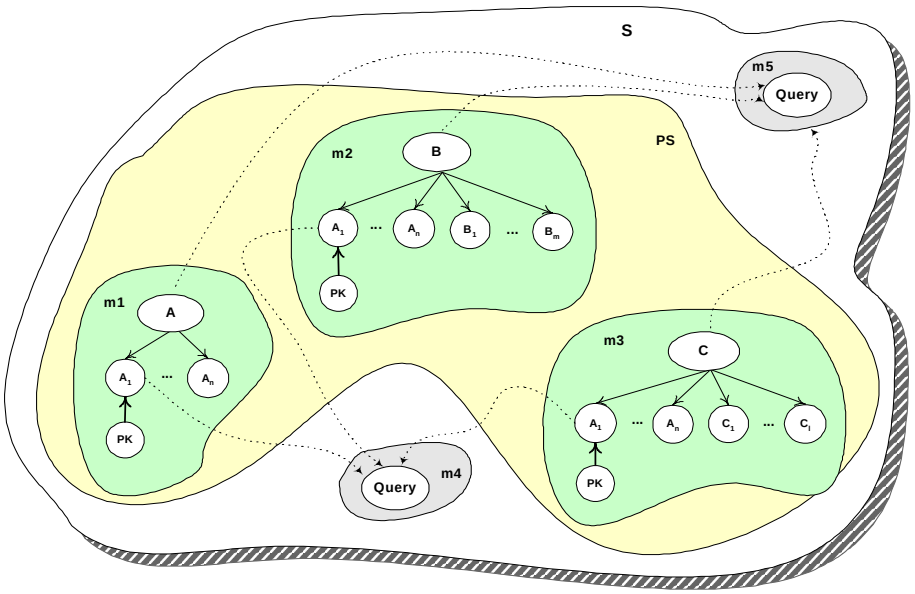
$$Cohesion(m5) = \frac{Cohesion(Q_1) + Cohesion(Q_2)}{2} = \frac{\frac{1}{3} + \frac{1}{2}}{2} = \frac{5}{12}.$$

Query $m6$: *Select all tuples*

$$\left.\begin{array}{l} SELECT\ \ A.A_1, ..., A.A_n,\ B.B_1, ..., B.B_m, ..., C.C_1, ..., C.C_l \\ FROM\ (A\ \ LEFT\ OUTER\ JOIN\ \ B\ \ ON\ \ A.A_1 = B.A_1) \\ \qquad LEFT\ OUTER\ JOIN\ \ C\ ON\ A.A_1 = C.A_1 \end{array}\right\} \Rightarrow \left\{\begin{array}{l} Size(m6) = n + m + l + 6. \\ Coupling(m6) = n + m + l + 3. \\ Cohesion(m6) = 1/3. \end{array}\right.$$

$$Size(S) = 2 * (n + m + l) + Size(: table) + 23.$$

Σχήμα 5.26 Μετρικές της $2^{ης}$ σχεδιαστικής λύσης, όπου κατασκευάζεται ένας πίνακας για την υπερκλάση, A, βοηθητικός για την αναζήτηση, και ένας πίνακας για κάθε υποκλάση.



$$Size(PS) = (2+n)+(m+n+2)+(l+n+2) = 3*n+m+l+6.$$

$$Coupling(m1) = Cuupling(m2) = Coupling(m3) = 0.$$

$$Cohesion(m1) = Cohesion(m2) = Cohesion(m3) = 1.$$

$$Data\_Volume(PS) = \#rows_A * (\sum_{i=1}^{n} sizeof(A_i)) +$$

$$\#rows_B * (\sum_{i=1}^{n} sizeof(A_i) + \sum_{j=1}^{m} sizeof(B_j)) + \#rows_C * (\sum_{i=1}^{n} sizeof(A_i) + \sum_{k=1}^{l} sizeof(C_k)).$$

Query $m4$: *Select specific tuple*

$$\left.\begin{array}{l} SELECT\ * \\ FROM\ A \\ WHERE\ A.A_1 = term \end{array}\right\} \Rightarrow Q_1$$

$$\left.\begin{array}{l} SELECT\ * \\ FROM\ B \\ WHERE\ B.A_1 = term \end{array}\right\} \Rightarrow Q_2$$

$$\left.\begin{array}{l} SELECT\ * \\ FROM\ C \\ WHERE\ C.A_1 = term \end{array}\right\} \Rightarrow Q_3$$

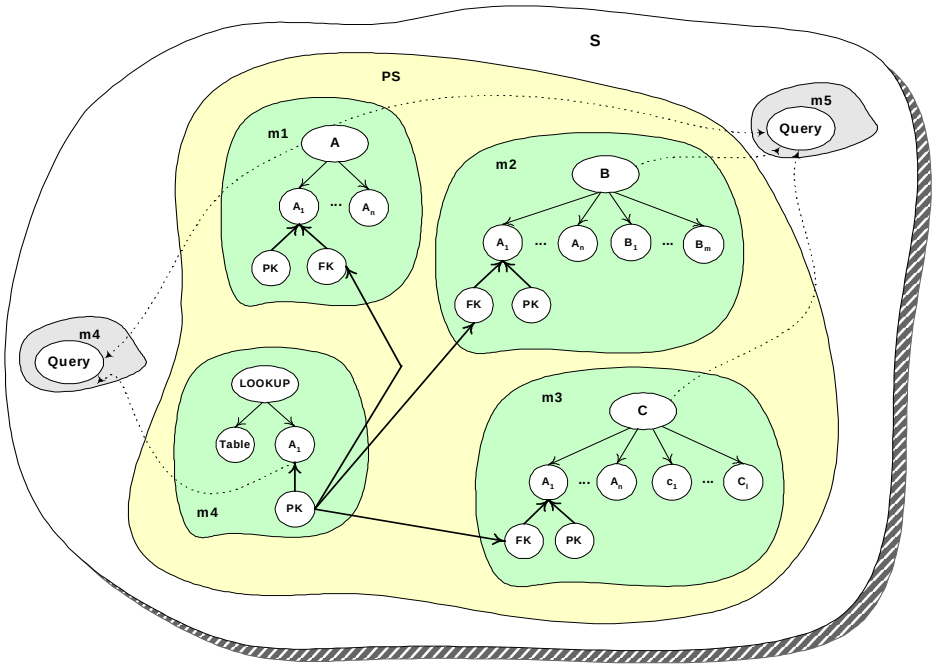
$$\Rightarrow \begin{cases} Size(m4) = n+(n+m)+(n+l)+3*3 = 3*n+m+l+9. \\ Coupling(m4) = n+(n+m)+(n+l)+3 = 3*n+m+l+3. \\ Cohesion(m4) = \dfrac{\frac{1}{2}+\frac{1}{2}+\frac{1}{2}}{3} = \dfrac{1}{2}. \end{cases}$$

Query $m5$: *Select all tuples*

$$\left.\begin{array}{l} (\ (\ (SELECT\ A_1, ..., A_n, NULL, ..., NULL, NULL, ..., NULL \\ \quad FROM\ A) \qquad\qquad UNION \\ \quad (SELECT\ A_1, ..., A_n, B_1, ..., B_m, NULL, ..., NULL \\ \quad FROM\ B)\ ) \qquad\qquad UNION \\ \quad (SELECT\ A_1, ..., A_n, NULL, ..., NULL, C_1, ..., C_l \\ \quad FROM\ C)\ ) \end{array}\right\} \Rightarrow \begin{cases} Size(m5) = 3*n+m+l. \\ Coupling(m5) = 3*n+m+l. \\ Cohesion(m5) = 1. \end{cases}$$

$$Size(S) = 9*n+3*(m+l)+15.$$

Σχήμα 5.27 Αναπαράσταση και μετρικές της $3^{ης}$ σχεδιαστικής λύσης, όπου κατασκευάζονται ξεχωριστοί πίνακες για κάθε κλάση χωρίς περιορισμούς αναφορικής ακεραιότητας.



$Size(PS) = (n+3) + (n+m+3) + (n+l+3) + 4 = 3*n+m+l+13.$

$Coupling(m1) = Cuupling(m2) = Coupling(m3) = 1.$

$Coupling(m4) = 0$

$Cohesion(m1) = Cohesion(m2) = Cohesion(m3) = Cohesion(m4) = 1.$

$$Data\_Volume(PS) = \#rows_A * (\sum_{i=1}^{n} sizeof(A_i)) +$$
$$\#rows_B * (\sum_{i=1}^{n} sizeof(A_i) + \sum_{j=1}^{m} sizeof(B_j)) + \#rows_C * (\sum_{i=1}^{n} sizeof(A_i) + \sum_{k=1}^{l} sizeof(C_k)) +$$
$$(\#rows_A + \#rows_B + \#rows_C) * (sizeof(A_i) + sizeof(Table)).$$

Query $m5$: *Select specific tuple that we don't know its class*.

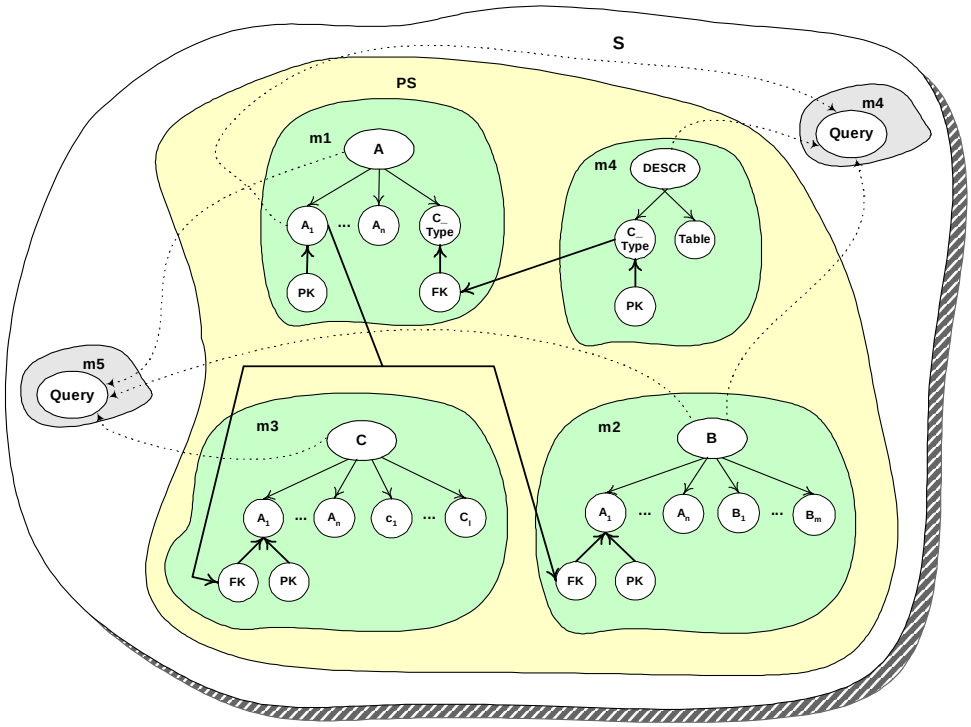
$$\left.\begin{array}{l} \left.\begin{array}{l} SELECT\ \ DESCR.Table\ INTO : table \\ From\ \ DESCR \\ WHERE\ \ A.A_1 = term \end{array}\right\} Q_1 \\ \left.\begin{array}{l} string\ \ x = 'SELECT\ \ *\ FROM\ \ ' \\ string\ \ y = 'WHERE\ ' \end{array}\right\} Q_2 \\ execute(concatenate(x, : table, z, : table, '.A_1 = ', , term)) \end{array}\right\} \Rightarrow \begin{cases} Size(m5) = 4 + ((size(: table) - 3) + 3) = size(: table) + 4. \\ Coupling(m5) = 2 + (Size(: table) - 3) + 1 = Size(: table). \\ Cohesion(m5) = \dfrac{\frac{1}{2} + \frac{1}{2}}{2} = 1/2. \end{cases}$$

Query $m6$: *Select all tuples*

$$\left.\begin{array}{l} (\quad (\quad (SELECT\ \ A_1, ..., A_n, \ NULL, ..., NULL, \ NULL, ..., \ NULL \\ \qquad FROM\ \ A) \qquad\qquad\qquad UNION \\ \qquad (SELECT\ \ A_1, ..., A_n, \ B_1, ..., B_m, \ NULL, ..., \ NULL \\ \qquad FROM\ \ B) \quad) \qquad\qquad UNION \\ \qquad (SELECT\ \ A_1, ..., A_n, \ NULL, ..., \ NULL, C_1, ..., \ C_l \\ \qquad FROM\ \ C) \quad) \end{array}\right\} \Rightarrow \begin{cases} Size(m6) = 3*n+m+l. \\ Coupling(m6) = 3*n+m+l. \\ Cohesion(m6) = 1. \end{cases}$$

$Size(S) = 6*n + 2*(m+l) + Size(: table) + 17.$

Σχήμα 5.28 Αναπαράσταση και μετρικές της 4ης σχεδιαστικής λύσης, όπου κατασκευάζονται ξεχωριστοί πίνακες για κάθε κλάση με περιορισμούς αναφορικής ακεραιότητας, και ένας βοηθητικός πίνακας αναζήτησης.



$Size(PS) = (4+n)+(n+m+3)+(n+l+3)+4 = 3*n+m+l+14.$

$Coupling(m4) = 0.$

$Coupling(m1) = Cuupling(m2) = Coupling(m3) = 1.$

$Cohesion(m1) = Cohesion(m2) = Cohesion(m3) = Cohesion(m4) = 1.$

$$Data\_Volume(PS) = \#rows_A * (\sum_{i=1}^{n} sizeof(A_i) + sizeof(Class\_Type)) +$$
$$\#rows_B * (\sum_{i=1}^{n} sizeof(A_i) + \sum_{j=1}^{m} sizeof(B_j)) + \#rows_C * (\sum_{i=1}^{n} sizeof(A_i) + \sum_{k=1}^{l} sizeof(C_k)) +$$
$$\#rows_{DESCR} * (sizeof(Class\_Type) + sizeof(Table)).$$

Query $m5$: *Select specific tuple that we don't know its class*.

$$\left.\begin{array}{l} \left.\begin{array}{c} SELECT\ \ DESCR.Table\ \ INTO : table \\ From\ \ DESCR \\ WHERE\ \ A.A_1 = term \end{array}\right\} Q_1 \\ \left.\begin{array}{c} string\ \ x = 'SELECT\ *\ FROM\ \ ' \\ string\ \ y = 'WHERE\ ' \end{array}\right\} Q_2 \\ execute(concatenate(x, : table, z, : table, '.A_1 = ', , term)) \end{array}\right\} \Rightarrow \begin{cases} Size(m5) = 4 + ((size(: table) - 3) + 3) = size(: table) + 4. \\ Coupling(m5) = 2 + (Size(: table) - 3) + 1 = Size(: table). \\ Cohesion(m5) = \dfrac{\frac{1}{2} + \frac{1}{2}}{2} = 1/2. \end{cases}$$

Query $m6$: *Select all tuples*

$$\left.\begin{array}{l} SELECT\ A.A_1, ..., A.A_n,\ B.B_1, ..., B.B_m, ..., C.C_1, ..., C.C_l \\ FROM\ (A\ LEFT\ OUTER\ JOIN\ B\ ON\ A.A_1 = B.A_1) \\ \qquad LEFT\ OUTER\ JOIN\ C\ \ ON\ A.A_1 = C.A_1 \end{array}\right\} \Rightarrow \begin{cases} Size(m6) = n + m + l + 6. \\ Coupling(m6) = n + m + l + 3. \\ Cohesion(m6) = 1/3. \end{cases}$$

$Size(S) = 4*n + 2*(m+l) + Size(: table) + 24.$

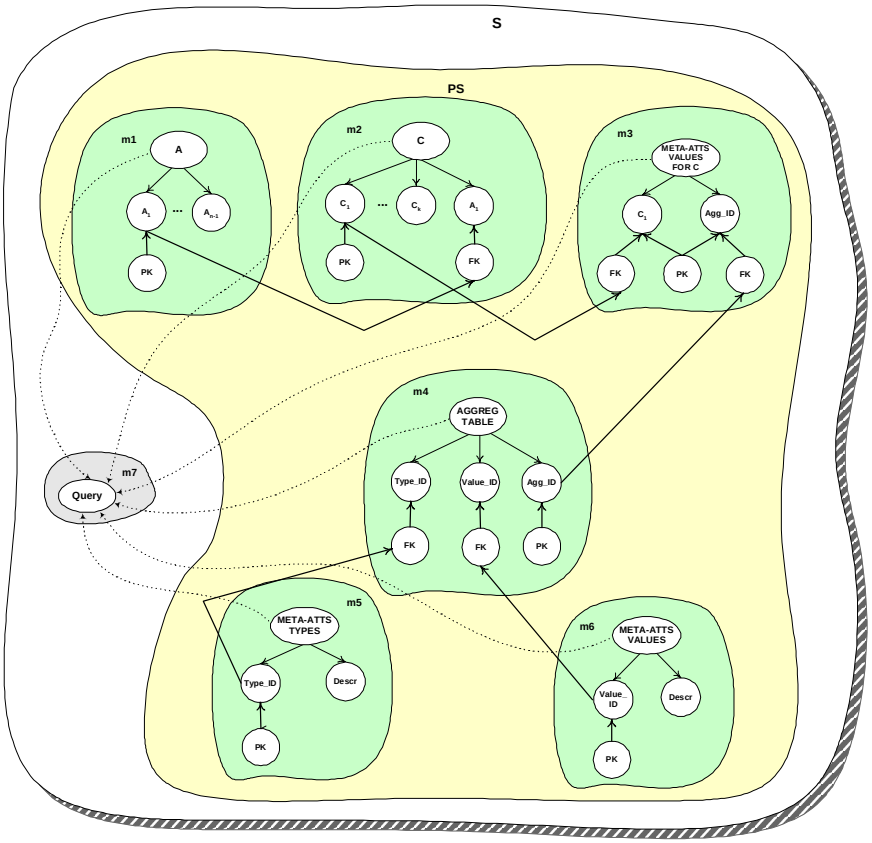
Σχήμα 5.29 Αναπαράσταση και μετρικές της $5^{ης}$ σχεδιαστικής λύσης, όπου κατασκευάζονται ξεχωριστοί πίνακες για κάθε κλάση με περιορισμούς αναφορικής ακεραιότητας στον πίνακα της υπερκλάσης, και ένας βοηθητικός πίνακας αναζήτησης.

5.3.7. Materialization

Το πρότυπο του Materialization, απεικονίζει τη συσχέτιση μεταξύ μιας αφηρημένης (abstract) σχέσης και μιας πιο συγκεκριμένης (concrete). Όπως αναφέρθηκε και στο κεφάλαιο 4, το πρόβλημα που αντιμετωπίζεται κατά την σχεδίαση, αφορά στα μεταγνωρίσματα της Abstract σχέσης. Ας θεωρήσουμε τη γενική περίπτωση των σχέσεων A (A1,…, An) και C(C1,…,Ck), οι οποίες αποτελούν τις Abstract και Concrete σχέσεις αντίστοιχα. Το πεδίο An, αποτελεί μετα-γνώρισμα της σχέσης A. Η πρώτη σχεδιαστική λύση, Σχήμα 5.30, απομακρύνει από τη σχέση A το μετα-γνώρισμα An, ενώ επιπλέον κατασκευάζονται οι σχέσεις:

- META-ATTS_TYPES, η οποία καταχωρεί τις περιγραφές των τύπων των μεταγνωρισμάτων.
- META-ATTS_VALUES, η οποία καταχωρεί τις περιγραφές των τιμών των μεταγνωρίσματος.
- AGGREG.TABLE, όπου καταχωρείται ο συνδυασμός των τύπων με τις αντίστοιχες τιμές μπορεί να λάβει κάθε μετα-γνώρισμα.
- META-ATTS_VALUES_FOR_C, όπου καταχωρούνται, για κάθε στιγμιότυπο της Concrete κλάσης, ο τύπος και η τιμή του μεταγνωρίσματος που αντιστοιχεί στο στιγμιότυπο.

Στο Σχήμα 5.32 παρουσιάζεται η δεύτερη εναλλακτική σχεδίαση, σύμφωνα με την οποία δημιουργείται η σχέση A(A1,…, An1,…Anm), με πρωτεύον κλειδί το A1 και παρουσιάζοντας όλες τις τιμές των διαφορετικών τύπων των μεταγνωρισμάτων με τη μορφή ξεχωριστών πεδίων (An1,…,Anm), και η C(C1,…Ck,A1), με πρωτεύον κλειδί το C1.



Σχήμα 5.30 Αναπαράσταση της σχεδιαστικής λύσης, όπου κατασκευάζονται ξεχωριστοί πίνακες των μεταγνωρισμάτων (TYPES_TABLE και VALUES_TABLE) της `Abstract` κλάσης (A) και ένας συναθροιστικός βοηθητικός πίνακας (AGG_TABLE).

$Size(PS) = (n+1) + (k+4) + 6 + 7 + 4 + 4 = n + k + 26.$

$Coupling(m1) = Coupling(m5) = Coupling(m6) = 0.$

$Coupling(m2) = Coupling(m3) = 1.$

$Coupling(m4) = 2.$

$Cohesion(m1) = Cohesion(m2) = Cohesion(m3) = Cohesion(m4) = Cohesion(m5) = Cohesion(m6) = 1.$

$$Data\_Volume(PS) = \#rows_A * (\sum_{i=1}^{n-1} sizeof(A_i)) + \#rows_C * (sizeof(A_1) + \sum_{j=1}^{k} sizeof(C_j)) +$$
$$\#rows_{META-ATTS\ FOR\ C} * (sizeof(C_1) + sizeof(Agg\_ID)) +$$
$$\#rows_{AGGR\_TABLE} * (sizeof(Agg\_ID) + sizeof(Type\_ID) + sizeof(Value\_ID)) +$$
$$\#rows_{META-ATTS\ TYPES} * (sizeof(Type\_ID) + sizeof(Descr)) +$$
$$\#rows_{META-ATTS\ VALUES} * (sizeof(Value\_ID) + sizeof(Descr)).$$

$Query\ m7: Select...$

i) a specific tuple

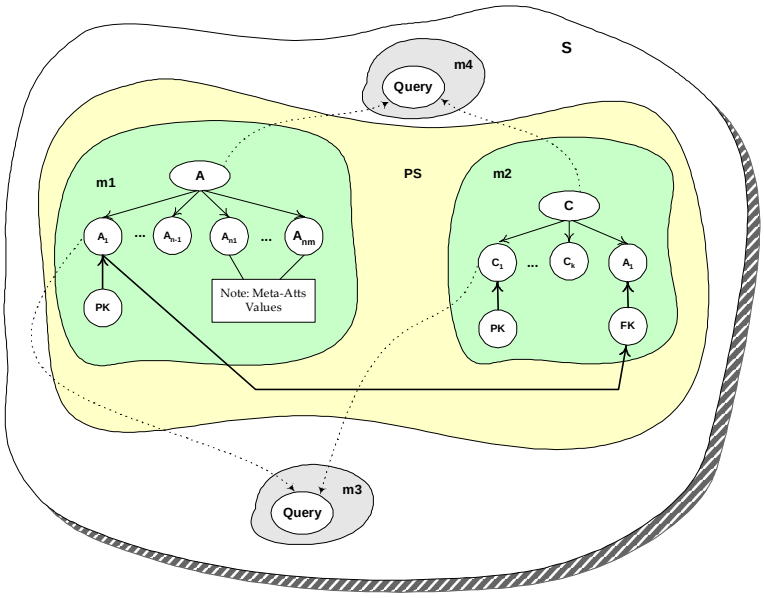
$$\left.\begin{array}{l} SELECT\ A1,...,An-1,C1,...,Ck,\ T.Descr,V.Descr \\ FROM\ A,C,META-ATTS\_FOR\_C\ AS\ M,\ AGGR\_TABLE\ AS\ A, \\ \qquad META-ATTS\_TYPES\ AS\ T,\ META-ATTS\_VALUES\ AS\ V \\ WHERE\ A.A1 = term\ AND\ A.A1 = C.A1\ AND\ C.C1 = M.C1\ AND \\ \qquad C.Agg\_ID = A.Agg\_ID\ AND\ A.Type = T.Type\ AND\ A.Value = V.Value. \end{array}\right\} \Rightarrow \begin{cases} Size(m7) = n + k + 19. \\ Coupling(m7) = n + k + 11. \\ Cohesion(m7) = 1/7. \end{cases}$$

i) all tuples

$$\left.\begin{array}{l} SELECT\ A1,...,An-1,C1,...,Ck,\ T.Descr,V.Descr \\ FROMA,C,\ META-ATTS\_FOR\_C\ AS\ M,\ AGGR\_TABLE\ AS\ A, \\ \qquad META-ATTS\_TYPES\ AS\ T,\ META-ATTS\_VALUES\ AS\ V \\ WHERE\ \ A.A1 = C.A1\ AND\ C.C1 = M.C1\ AND\ C.Agg\_ID = A.Agg\_ID \\ \qquad AND\ A.Type = T.Type\ AND\ A.Value = V.Value. \end{array}\right\} \Rightarrow \begin{cases} Size(m7) = n + k + 16. \\ Coupling(m7) = n + k + 11. \\ Cohesion(m7) = 1/6. \end{cases}$$

$Size(S) = 3*(n+k) + 61.$

Σχήμα 5.31 Μετρικές της σχεδιαστικής λύσης, όπου κατασκευάζονται ξεχωριστοί πίνακες των μεταγνωρισμάτων (`TYPES_TABLE` και `VALUES_TABLE`) της `Abstract` κλάσης (`A`) και ένας συναθροιστικός βοηθητικός πίνακας (`AGG_TABLE`).



$Size(PS) = (n-1+2+m)+(k+4) = n+k+m+5.$

$Coupling(m1) = 0.$

$Cupling(m2) = 1.$

$Cohesion(m1) = Cohesion(m2) = 1.$

$Data\_Volume(MS) = \#rows_A * (\sum_{i=1}^{n-1} sizeof(A_i) + \sum_{i=1}^{m} A_{nj}) + \#rows_C * (sizeof(A_1) + \sum_{i=1}^{k} sizeof(C_i)).$

Query $m3$: *Select a specific tuple*

$$\left.\begin{array}{l} SELECT * \\ FROM\ A, C \\ WHERE\ C_1 = term\ AND\ C.A_1 = A.A_1 \end{array}\right\} \Rightarrow \begin{cases} Size(m3) = n-1+m+k+6 = n+m+k+5. \\ Coupling(m3) = (n+m+k-1)+3 = n+m+k+2. \\ Cohesion(m3) = 1/3. \end{cases}$$

Query $m4$: *Select all tuples*

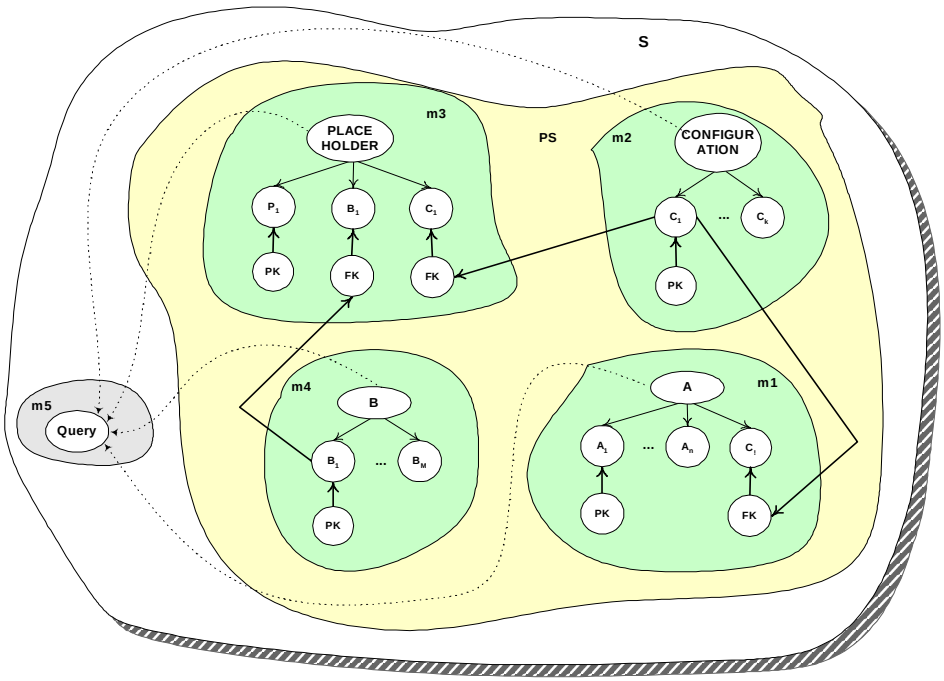
$$\left.\begin{array}{l} SELECT * \\ FROM\ A, C \\ WHERE\ C.A_1 = A.A_1 \end{array}\right\} \Rightarrow \begin{cases} Size(m3) = n-1+m+k+3 = n+m+k+2. \\ Coupling(m3) = (n+m+k-1)+2 = n+m+k+1. \\ Cohesion(m3) = 1/2. \end{cases}$$

$Size(S) = 3*(n+m+k)+12.$

Σχήμα 5.32 Αναπαράσταση και μετρικές της σχεδιαστικής λύσης, όπου κατασκευάζονται ξεχωριστά πεδία ($A_{n1}, \ldots, A_{nm}$) για κάθε τιμή των μεταγνωρισμάτων στον πίνακα της `Abstract` κλάσης, `A`.

5.3.8. Configuration

Το πρότυπο του Configuration απεικονίζει το φαινόμενο της διαμέρισης ενός συνόλου στοιχείων με βάση κάποιους κανόνες. Το πρόβλημα που παρουσιάζεται σε αυτή τη περίπτωση είναι η μεταβολή των ιδιοτήτων, τις οποίες έχουν τα στοιχεία. Παράδειγμα της χρήσης του προτύπου Configuration αποτελεί η καταγραφή του οδηγού σπουδών μιας σχολής, όπου τα μαθήματα που μπορεί/πρέπει να πάρει ένας φοιτητής για τη λήψη του πτυχίου του, μεταβάλλονται με το χρόνο. Ας θεωρήσουμε τη γενική περίπτωση, όπου η `A(A1,…An)`, με πρωτεύον κλειδί το πεδίο `A1`, αποτελεί τη σχέση της οποίας τα στοιχεία (στιγμιότυπα) διαμερίζονται με βάση κάποιους κανόνες. Οι κανόνες στηρίζονται στις καταχωρήσεις του πίνακα `B(B1,…,Bm)`. Μέσω της σχέσης `CONFIGURATION(C1,…,Ck)`, με πρωτεύον κλειδί το `C1`, απεικονίζονται όλες οι διαμερίσεις του συνόλου, ενώ η σχέση `PLACEHOLDER(P1,B1,C1)`, έχει πρωτεύον κλειδί το `P1`, και συσχετίζει κάθε κανόνα με τα αντίστοιχα στιγμιότυπα της σχέσης `B`.



$Size(PS) = (n+4) + (k+2) + 7 + (m+2) = n + k + m + 15.$

$Coupling(m1) = 1.$

$Coupling(m3) = 2.$

$Coupling(m2) = Coupling(m4) = 0.$

$Cohesion(m1) = Cohesion(m2) = Coupling(m3) = Coupling(m4) = 1.$

$$Data\_Volume(PS) = \#rows_A * (\sum_{i=1}^{n} sizeof(A_i) + sizeof(C_1)) + \#rows_{CONFIGURATION} * (\sum_{i=1}^{k} sizeof(C_i)) + \#rows_{PLACEHOLDER} * (sizeof(P_1) + sizeof(C_1) + sizeof(B_1)) + \#rows_B * (\sum_{i=1}^{m} sizeof(B_i)).$$

$Query\,m5 : Select...$

$i)\,a\ specific\ tuple\ (e.g., courses\ of\ a\ student)$

$$\left.\begin{array}{l} SELECT\ A_1,...,A_n, B_1,...,B_m \\ FROM\ A, B, PLACEHOLDER \\ WHERE\ A_1 = term\ AND\ A.C_1 = PLACEHOLDER.C_1 \\ \qquad AND\ PLACEHOLDER.B_1 = B.B_1 \end{array}\right\} \Rightarrow \begin{cases} Size(m5) = n + m + 9. \\ Coupling(m5) = n + m + 5. \\ Coehsion(m5) = 1/4. \end{cases}$$

$ii)\,all\ tuples$

$$\left.\begin{array}{l} SELECT\ A_1,...,A_n, B_1,...,B_m \\ FROM\ A, B, PLACEHOLDER, CONFIGURATION \\ WHERE\ A.C_1 = CONFIGURATION.C_1\ AND \\ \qquad CONFIGURATION.C_1 = PLACEHOLDER.C_1 \\ \qquad AND\ PLACEHOLDER.B_1 = B.B_1 \end{array}\right\} \Rightarrow \begin{cases} Size(m5) = n + m + 9. \\ Coupling(m5) = n + m + 5. \\ Coehsion(m5) = 1/4. \end{cases}$$

$Size(S) = 3 * (n + m) + k + 33.$

Σχήμα 5.33 Αναπαράσταση και μετρικές της σχεδιαστικής λύσης του προτύπου Configuration.

5.4. Συμπεράσματα μετρικών

Σε αυτή την ενότητα παραθέτουμε τα συμπεράσματα στα οποία μπορούμε να οδηγηθούμε, συγκρίνοντας τις αποτιμήσεις των μετρικών ποιότητας των προτεινόμενων σχεδιαστικών λύσεων, για κάθε πρότυπο.

5.4.1. Τεχνητό Κλειδί

Η σχεδίαση, η οποία θέτει ως πρωτεύον κλειδί το συνδυασμό γνωρισμάτων, παρουσιάζει τη μικρότερη τιμή ως προς το χώρο που απαιτείται για την αποθήκευση της πληροφορίας, με αμέσως επόμενη τη δεύτερη σχεδίαση, κατά την οποία εισάγεται τεχνητό κλειδί που ενημερώνεται αυτόματα από το σύστημα. Σε ότι αφορά στις ερωτήσεις που τίθενται προς τη βάση, η δεύτερη σχεδίαση, και η τρίτη σχεδίαση, όπου η ενημέρωση της τιμής του τεχνητού κλειδιού πραγματοποιείται μέσω μιας διαδικασίας, παρουσιάζουν μικρότερο κόστος συντήρησης (μετρική Coupling). Συνεπώς, μπορούμε να συμπεράνουμε ότι, η δημιουργία τεχνητού κλειδιού που ενημερώνεται αυτόματα, αποτελεί μια καλή σχεδίαση.

5.4.2. Κανονικές Μορφές

5.4.2.1. 1NF

Για την περίπτωση τόσο των σύνθετων όσο και των πλειότιμων γνωρισμάτων, αποτελέσματα με μικρότερο κόστος συντήρησης και απαιτούμενο αποθηκευτικό χώρο δίνει η σχεδίαση, η οποία παραβιάζει την 1NF. Αυτό το «μη αναμενόμενο» αποτέλεσμα, τονίζει την αναγκαιότητα ορισμού νέων μετρικών, μέσω των οποίων μπορεί να αποτιμηθεί η ορθότητα μιας σχεδίασης.

5.4.2.2. 2NF – 3NF

Τα αποτελέσματα των μετρικών της 2NF είναι ίδια με αυτά της 3NF, γι’ αυτό το λόγο και τα μελετάμε ως μια περίπτωση. Παρατηρούμε, λοιπόν, ότι, μικρό αποθηκευτικό χώρο, αλλά και μικρό κόστος συντήρησης από πλευράς των ερωτήσεων που τίθενται στη βάση, παρουσιάζουν οι λύσεις, οι οποίες παραβιάζουν τις κανονικές μορφές.

Αντίθετα, η παραβίαση των κανονικών μορφών, οδηγεί σε σχήματα βάσης με μικρή συνεκτικότητα (δηλαδή, μικρό κόστος συντήρησης της βάσης).

5.4.3. Αξονική Περιστροφή

Μέσα από τη μελέτη των μετρικών, για τη περίπτωση του pivot, συμπεραίνουμε ότι καλύτερη σχεδίαση αποτελεί το encoded design, διότι:

i. παρουσιάζει το μικρότερο κόστος συντήρησης τόσο από πλευράς βάσης, όσο και από πλευράς των ερωτήσεων που τίθενται σε αυτή.
ii. ο αποθηκευτικός χώρος που απαιτείται για την καταχώρηση της πληροφορίας, είναι πολύ μικρότερος από αυτόν του flat.

5.4.4. Παραγόμενα Γνωρίσματα

Μέσα από τη μελέτη των μετρικών παρατηρούμε ότι, η δημιουργία εικονικής όψης απαιτεί το μικρότερο χώρο αποθήκευσης, αλλά παρουσιάζει το μεγαλύτερο κόστος συντήρησης από πλευράς ερωτήσεων. Η κατασκευή ενός πίνακα που περιλαμβάνει το παραγόμενο γνώρισμα παρουσιάζει την αμέσως μικρότερη απαίτηση σε χώρο, και το μικρότερο κόστος συντήρησης από πλευράς ερωτήσεων. Τέλος, η δημιουργία υλοποιήσιμης όψης παρουσιάζει μεγάλες τιμές, τόσο ως προς το κόστος συντήρησης, όσο και ως προς τον αποθηκευτικό χώρο που δεσμεύει.

5.4.5. Γενίκευση

Ανάμεσα στις πέντε προτεινόμενες σχεδιαστικές λύσεις για τη περίπτωση του IsA, , αποτελεί την καλύτερη, με βάση τις μετρικές σχεδίαση. Μέσα από τη μελέτη των μετρικών παρατηρήσαμε ότι η κατασκευή ενός πίνακα, ο οποίος περιλαμβάνει όλα τα πεδία των κλάσεων, απαιτεί μικρότερο αποθηκευτικό χώρο. Οι λύσεις με το μικρότερο κόστος συντήρησης σε επίπεδο ερωτήσεων είναι: i) η δεύτερη σχεδιαστική λύση, όπου κατασκευάζεται ένας πίνακας για την υπερκλάση βοηθητικός για την αναζήτηση και ένας πίνακας για κάθε υποκλάση, ii) η τέταρτη λύση, όπου κατασκευάζονται ξεχωριστοί πίνακες για κάθε κλάση με περιορισμούς αναφορικής ακεραιότητας και ένας βοηθητικός πίνακας αναζήτησης, και iii) η πέμπτη

σχεδιαστική λύση, όπου κατασκευάζονται ξεχωριστοί πίνακες για κάθε υποκλάση με περιορισμούς αναφορικής ακεραιότητας στον πίνακα της υπερκλάσης και ένας βοηθητικός πίνακας αναζήτησης. Τέλος η τρίτη λύση, όπου κατασκευάζονται ξεχωριστοί πίνακες για κάθε κλάση χωρίς περιορισμούς αναφορικής ακεραιότητας, παρουσιάζει το μικρότερο κόστος συντήρησης από πλευράς βάσης

5.4.6. Materialization

Για τη περίπτωση του materialization η σχεδίαση που δημιουργεί ξεχωριστούς πίνακες για τα μεταγνωρίσματα της Abstract κλάσης, δεσμεύει το μικρότερο χώρο για την αποθήκευση της πληροφορίας. Επιπλέον, παρουσιάζει το μικρότερο κόστος συντήρησης από τη πλευρά των ερωτήσεων. Αντίθετα, η κατασκευή ενός πίνακα που περιλαμβάνει όλα τα γνωρίσματα της Abstract κλάσης εμφανίζεται να έχει το μικρότερο κόστος συντήρησης, όσον αφορά στο σχήμα της βάσης.

ΚΕΦΑΛΑΙΟ 6. ΣΥΜΠΕΡΑΣΜΑΤΑ

Στην εργασία αυτή, προτείναμε ένα συντεταγμένο τρόπο παρουσίασης λύσεων για προβλήματα σχεδίασης βάσεων δεδομένων. Συγκεκριμένα, στηριχθήκαμε στην τεχνική των σχεδιαστικών προτύπων για αντικειμενοστρεφές λογισμικό, την οποία και επεκτείναμε για προβλήματα σε βάσεις δεδομένων. Αρχικά προτείναμε ένα γενικό τρόπο ορισμού σχεδιαστικών προτύπων για σχεσιακές βάσεις δεδομένων. Στη συνέχεια, ασχοληθήκαμε με εννιά προβλήματα που εμφανίζονται συχνά κατά τη σχεδίαση βάσεων δεδομένων, και προτείναμε λύσεις με τη μορφή σχεδιαστικών προτύπων. Για κάθε σχεδιαστικό πρότυπο, δόθηκε το σύνολο των σχεδιαστικών λύσεων, μέσα από το οποίο μπορεί να επιλέξει ένας σχεδιαστής. Επιπλέον για κάθε λύση, μελετήθηκε η συμπεριφορά της βάσης, που σχεδιάζεται με βάση τη λύση αυτή, σε επίπεδο στιγμιότυπου (εγγραφών) και σε επίπεδο σχήματος (πεδίων) όλων των σχετικών πινάκων. Με αυτόν τον τρόπο, ο σχεδιαστής είναι σε θέση να επιλέξει την καλύτερη, γι' αυτόν, σχεδίαση, γνωρίζοντας εκ των προτέρων τα πλεονεκτήματα και τα μειονεκτήματά της. Τέλος, μέσα από την μελέτη μετρικών, οι οποίες παρουσιάζονται στο [1], πραγματοποιήθηκε μια γενική αποτίμηση του μεγέθους, της απόδοσης και του κόστους διατήρησης, της εκάστοτε λύσης. Με την καταγραφή και την εκμάθηση των σχεδιαστικών προτύπων, εκτός του ότι παρέχεται ένας κοινός κώδικας επικοινωνίας μεταξύ των σχεδιαστών, ο σχεδιαστής απαλλάσσεται από το «βάρος» μιας σχεδίασης «από το μηδέν», της οποίας δεν γνωρίζει την απόδοση εξ αρχής.

Η μελλοντική εργασία προσανατολίζεται στις παρακάτω κατευθύνσεις:

- Ανακάλυψη νέων μετρικών. Μελετώντας τις κανονικές μορφές, παρατηρήσαμε ότι δεν υπάρχουν μετρικές, οι οποίες να δείχνουν γιατί η παραβίαση της πρώτης κανονικής μορφής δεν αποτελεί καλή σχεδίαση. Η

εύρεση, λοιπόν, μετρικών, που να λαμβάνουν υπόψη την συνέπεια και την πληρότητα των τιμών μιας βάσης, είναι αναγκαία.

- Μελέτη και απόδειξη μετρικών, όπως, για παράδειγμα το μήκος (Length), που έχουν ήδη προταθεί στη σχετική βιβλιογραφία.
- Παρουσίαση επιπλέον σχεδιαστικών προτύπων. Εμπλουτισμός του συνόλου των σχεδιαστικών λύσεων του κάθε προτύπου.
- Μελέτη του κατά πόσο τα σχεδιαστικά πρότυπα, που προτείνονται εδώ, είναι χρήσιμα και κατανοητά από το σύνολο των σχεδιαστών. Δηλαδή, σε συνεργασία με μια ομάδα σχεδιαστών και μέσα από μια σειρά πειραμάτων, να αποδειχθεί κατά πόσο η εκμάθηση των συγκεκριμένων προτύπων «βοηθά» κατά τη σχεδίαση μιας Βάσης Δεδομένων.

ΑΝΑΦΟΡΕΣ

[1] L.C. Briand, S. Morasca and V.R. Basili. "*Property-Based Software Engineering Measurement*". IEEE Transactions on Software Engineering, Vol. 22, pp. 68-86, January 1996.

[2] M. Dahchour. "*Formalizing Materialization Using a Metaclass Approach*". CAISE 1998: pp. 401-421, 1998.

[3] M. Dahchour, M. Kolp, A. Pirrote and E. Zimanyi. "*Generic Relationships in Information Modelling*", Journal of Data Semantics, Volume 4, 2005.

[4] J. Dong. "*Adding pattern related information in structural and behavioral diagrams*". Information & Software Technology 46(5): pp.93-300, 2004.

[5] J. Evermann. "*The Association Construct in Conceptual Modelling - An Analysis Using the Bunge Ontological Model*". CAISE 2005: pp. 33-47, 2005.

[6] R. B. France, Dae-Kyoo Kim, S. Ghosh, E. Song. "*A UML-Based Pattern Specification Technique*". IEEE Trans. Software Eng. 30(3): pp. 193-206, 2004.

[7] J. A. Garda, J. A. Casal, R. G. Vazquez, J. Pazos, S. R. Yanez, A. Silva. "*A methodological framework for generic conceptualisation: problem-sensitivity in software engineering*". Information & Software Technology 46(10): pp. 635-649, 2004.

[8] E.Gamma, R. Helm, R. Johnson and J. Vlissides. "*Design Patterns Elements of Reusable Object-Oriented Software*". Professional Comptuting Series. Addison

Wedley, Reading, 1995.

[9] D. Gornik. "*UML Data Modelling Profile* ", IBM, 2002.

[10] S. Greenspan, J. Mylopoulos and A. Borgida. "*Capturing more world knowledge in the requirements specification*". In Proc. 6th International Conference on Software Engineering (ICSE '82), pp. 225-234, Tokyo, Japan, 1982.

[12] M. Kolp. "*Semantics Relationships*", Lecture Semantics Relationships, University of Toronto, 2001. In collaboration with A. Pirotte and M. Danhchour, University of Louvain.

[13] D. Lorentz. "*Oracle Database SQL Reference, 10gRelease 2(10.2)*". Oracle, 2005.

[14] J. K. H. Mak, C. Sze-Tsan Choy, D. P. K. Lun. "*Precise Modeling of Design Patterns in UML*". In Proc. 6th International Conference on Software Engineering (ICSE '04), pp: 252-261, Edinburgh, United Kingdom, 23-28 May 2004.

[15] J. Melton, A. Simon. "*SQL:1999: understanding relational language components*". Morgan Kaufmann Publishers Inc., 2001

[16] S. Lujan-Mora, P. Vassiliadis, J. Trujillo. "*Data Mapping Diagrams For Data Warehouse Design With UML*". In Proc. 23rd International Conference on Conceptual Modeling (ER 2004), pp. 191-204, Shanghai, China, 8-12 November 2004.

[17] J. Mylopoulos. "*Conceptual Modelling and Telos*", in Loucopoulos, P. and Zicari, R. (eds.) Conceptual Modelling, Databases and CASE: An Integrated View of Information Systems Development, McGraw Hill, 1992.

[18] J. Mylopoulos. "*Information Modelling in the Time of the Revolution*", Information Systems 23(3-4), pp. 127-156, June 1998.

[19] J. Mylopoulos. "*Ontologies*", Lecture Conceptual Modelling, University of Toronto, 2001.

[20] L. Prechelt, B. Unger. "*A Series of Controlled Experiments on Design Patterns: Methodology and Results*". Proc. Softwaretechnik '98 GI Conference, pp. 53-60, Paderborn, September 7-9, 1998.

[21] L. Prechelt, B. Unger, W. F. Tichy, P. Brössler, L. G. Votta. " *A Controlled Experiment in Maintenance Comparing Design Patterns to Simpler Solutions*". IEEE Trans. Software Eng. 27(12): pp. 1134-1144, 2001.

[22] Graeme C. Simsion, Graham C. Witt. "*Data Modeling Essentials, 2nd Edition: A Comprehensive Guide to Data Analysis, Design, and Innovation*". Professional Comptuting Series. Coriolis Group Books, 2001.

[23] T. Taibi, D. Chek Ling Ngo. "*Formal specification of design pattern combination using BPSL*". Information & Software Technology 45(3): pp. 157-170, 2003.

[24] A. Tropashko and D. K. Burleson. "*SQL Design Patterns: Expert Guide to SQL Programming*". Rampant Techpress. IT In-Focus, April 2007.

[25] Y. Wand, R. Weber. "An *Ontological Model of an Information System*". IEEE Trans. Software Eng. 16(11): pp. 1282-1292, 1990.

ΣΥΝΤΟΜΟ ΒΙΟΓΡΑΦΙΚΟ

Η Ευγενία Σταθοπούλου γεννήθηκε το 1979 στην Αθήνα. Το 1998 εισήχθη στο Τμήμα Πληροφορικής του Πανεπιστημίου Ιωαννίνων, αποφοιτώντας το 2003 . Κατά τη χρονική περίοδο 2003-2004 εργάστηκε στην εταιρία IST, συμμετέχοντας στην ομάδα OTENET για την αναπτυξη λογισμικού του Athens2004. Το 2004 ξεκίνησε τις μεταπτυχιακές σπουδές της στο Τμήμα Πληροφορικής, απ' όπου και αποφοίτησε το 2007.