Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Μηχανικών Πληροφορικής

Πτυχιακή εργασία

Τίτλος: Συστήματα Συλλογισμού σε Χωροχρονικές βάσεις δεδομένων.

Κωνσταντίνος Σόγιας 3563
Αλέξανδρος Παπαδάκης 3674
Γρηγόρης Καπαράκης 3497

Επιβλέπων εκπαιδευτικός : Παπαδάκης Νικόλαος

Ευχαριστίες

Σε αυτό το σημείο θα θέλαμε να ευχαριστήσουμε τον καθηγητή μας κύριο Παπαδάκη Νικόλαο όπου χωρίς την καθοδήγηση και τις συμβουλές του δεν θα είχαμε φτάσει ως εδώ. Επίσης θα θέλαμε να ευχαριστήσουμε τις οικογένειες μας για την υποστήριξη που δείξανε κατά το πέρας αυτής της πτυχιακής εργασίας, αλλά και για την υποστήριξη που δεχτήκαμε καθόλη την διάρκεια των σπουδών μας.

Abstract

A database is not constant, and its data is constantly changing. Some actions push the data to change the situation and this results in integrity being violated. But a basis for being properly structured and functioning must follow to all the conditions of integrity. The results of an operation may have a direct or indirect impact on the data. These acts are called fluents. For example, we have a base that records the data of employees of a company. If an employee makes a misdemeanor, he / she does not get a salary and as indirect he / she cannot get a promotion. But if an employee is bad, he will not get a raise, but that's not mean he has been misdemeanor. As we can see from the previous example, some objects are affected by the actions we perform, while some remain unaffected. This situation called as a framework problem and finalized by McCanthy and Hayes in 1969, Lifshitz, 1990, Thielscher, 1997; Reiter, 1980, 1991, Fikes and Nilsson, 1971; McCain and Turner, 1995, Denecker and Ternovska, 2007[1][2][3][4][5][6]. A data base is important to maintain its integrity. So, a base is considered structured and functionally correct when it maintains all integrity constraints as you perform an act. What you often find is after acts that modify or import data, cause the database to violate the integrity conditions that have been defined. Therefore, this problem forces us to handle our situations more carefully so that these violations do not occur. Due to the above-mentioned problems, we use the fluent where we adjust them in such a way that each operation has an effect only on the next state. Maintaining fluents is therefore very important. Many times, designers and developers cannot be aware of all the changes that occur, but it is necessary to take them into account. One way is to do it manually, that is, for every act that happens to be aware of and to keep the resulting transactions. This may be possible in a small system, but in a rather complicated way it is particularly difficult as an action we can do in t_0 has a different base effect if t_1 occurs. This has resulted in a database that supports all the limitations that have been put in place and an application that makes it easy to run this database. The data base has four tables and five triggers. Triggers support the integrity constraints as well as the design of the tables, based on the correct functioning of the limitations that have been set. Finally, with the application implemented in the JAVA programming language, you are checking the database and making possible actions from the user to the database.

Σύνοψη

Μία βάση δεδομένων δεν είναι σταθερή και συνεχώς τα δεδομένα της μεταβάλλονται. Κάποιες πράξεις ωθούν τα δεδομένα να αλλάζουν κατάσταση και αυτό έχει σαν αποτέλεσμα οι συνθήκες ακεραιότητας να παραβιάζονται. Όμως μια βάση για να είναι σωστά δομημένη και να λειτουργεί πρέπει να τηρεί όλες τις συνθήκες ακεραιότητας. Τα αποτελέσματα μιας πράξης μπορεί να έχουν άμεσο ή έμμεσο αντίκτυπο στα δεδομένα. Οι πράξεις αυτές λέγονται fluents. Παραδείγματος χάριν Έχουμε μια βάση που καταγραφεί τα δεδομένα των υπάλληλων μιας εταιρίας. Εάν ένας υπάλληλος κάνει κάποιο πλημμέλημα έχει σαν αποτέλεσμα να μην παίρνει μισθό και σαν έμμεσο να μην μπορεί να πάρει προαγωγή. Όμως εάν κάποιος υπάλληλος είναι κακός δεν θα πάρει αύξηση αλλά δεν συνεπάγετε ότι έχει κάνει και πλημμέλημα. Όπως λοιπόν παρατηρούμε από το προηγούμενο παράδειγμα, κάποια αντικείμενα επηρεάζονται από τις πράξεις που εκτελούμε, ενώ κάποια αλλά παραμένουν ανεπηρέαστα. Αυτό χαρακτηρίζετε σαν frame problem και οριστικέ από τον McCanthy and Hayes το 1969, Lifshitz, 1990, Thielscher, 1997, Reiter, 1980, 1991, Fikes and Nilsson, 1971, McCain and Turner, 1995, Denecker and Ternovska, 2007[1][2][3][4][5][6]. Όπως αναφέραμε σε προηγουμένη παράγραφο, μια βάση είναι σημαντικό να τηρεί τις συνθήκες ακεραιότητας της. Έτσι μια βάση θεωρείτε δομημένη και λειτουργικά σωστή όταν τηρεί όλους τους περιορισμούς ακεραιότητας καθώς εκτελείτε μια πράξη. Αυτό που εντοπίζετε συχνά, είναι μετά από πράξεις που τροποποιούν ή εισάγουν δεδομένα, να αναγκάζουν την βάση να παραβιάζει τις συνθήκες ακεραιότητας που έχουν οριστεί. Συνεπώς αυτό το πρόβλημα μας αναγκάζει να χειριζόμαστε τις καταστάσεις της βάσεις με περισσότερη προσοχή ώστε να μην γίνονται αυτοί οι παραβιασμοί. Λόγο των προβλημάτων που αναφέρονται παραπάνω γίνεται χρήση των fluent όπου τα προσαρμόζουμε με τέτοιο τρόπο έτσι ώστε κάθε πράξη να έχει επίδραση μόνο στην επόμενη κατάσταση. Η διατήρηση των fluents λοιπόν είναι πολύ σημαντική. Πολλές φορές οι σχεδιαστές και οι προγραμματιστές δεν μπορούν να γνωρίζουν όλες τις αλλαγές που προκύπτουν, είναι απαραίτητο όμως να τα λάβουμε υπόψιν μας. Ένας τρόπος είναι να το κάνουμε χειροκίνητα δηλαδή για κάθε πράξη που συμβαίνει να γνωρίζουμε και να τηρούμε τις συναλλαγές που προκύπτουν. Κάτι τέτοιο μπορεί να είναι εφικτό σε ένα μικρό σύστημα, αλλά σε ένα αρκετά περίπλοκο είναι ιδιαίτερα δύσκολο καθώς μπορεί μια πράξη που κάνουμε σε t0 να έχει διαφορετική επίδραση στην βάση εάν συμβεί t1. Έτσι υλοποιήθηκε μια βάση δεδομένων οπού υποστηρίζει όλους τους περιορισμούς που έχουν τεθεί και μια εφαρμογή που καθιστά εύκολη την λειτουργία της αυτής της βάσης δεδομένων. Η βάση έχει τέσσερις πίνακες και πέντε triggers. Με την βοήθεια των triggers υποστηρίζονται οι περιορισμοί ακεραιότητας αλλά και η σχεδίαση των πινάκων έγινε με γνώμονα την ορθή λειτουργία των περιορισμών που έχουν τεθεί. Τέλος με την εφαρμογή που υλοποιήθηκε στην γλώσσα προγραμματισμού JAVA ελέγχετε η βάση δεδομένων και γίνονται εφικτές οι ενέργειες από τον χρήστη προς την βάση δεδομένων.

Πίνακας περιεχομένων

Contents

Λίστα Εικόνων

1. Εισαγωγή

Η πτυχιακή εργασία (ΠΕ) αποτελεί ένα από τα σημαντικά στάδια των προπτυχιακών σπουδών. Προσομοιάζει το πραγματικό εργασιακό περιβάλλον στο οποίο θα δραστηριοποιηθούν οι απόφοιτοι και περιλαμβάνει σημαντική συμμετοχή του καθηγητή σε επίπεδο καθοδήγησης. Εναπόκειται στον σπουδαστή να εκμεταλλευτεί στο έπακρο την ευκαιρία αυτή και να αποκομίσει τα μέγιστα δυνατά οφέλη σε σχέση με την ακαδημαϊκή και επαγγελματική του κατάρτιση. Ο παρών οδηγός δημιουργήθηκε, ώστε να παρέχει καθοδήγηση στην εκπόνηση και συγγραφή των ΠΕ.

Μέχρι πρότινος, ο χώρος και ο χρόνος στις βάσεις δεδομένων ήταν αντικείμενα τα οποία τα εξέταζαν ξεχωριστά. Στη συνέχεια, δημιουργήθηκε η ανάγκη για έρευνα πάνω στην αμοιβαία επίδραση και στην συνεργασία των δυο αυτών πεδίων για την κάλυψη διαφόρων αναγκών που αντιμετωπίζουμε καθημερινά στην σύγχρονη κοινωνία. Αυτές οι ανάγκες απαιτούν τη διαχείριση χωροχρονικών εννοιών.

Η πλειοψηφία των βάσεων δεδομένων στις μέρες μας χρησιμοποιεί μεγάλο όγκο δεδομένων με χρονική πληροφορία. Παραδείγματος χάριν, εφαρμογές που απευθύνονται σε τράπεζες, λογιστές, εμπορικά καταστήματα, super market κ.α., όλες αυτές οι εφαρμογές έχουν σαν κύριο στοιχείο στη βάση δεδομένων τον χρόνο. Οι χωρικές βάσεις δεδομένων χρησιμοποιούνται σε εφαρμογές οι οποίες αναφέρονται κυρίως σε συστήματα γεωγραφικών πληροφοριών. Τα συστήματα τα οποία χρησιμοποιούν γεωγραφικές πληροφορίες είναι ιδιαίτερα αναγκαία στις μέρες μας όπως και εκείνα των χρονικών πληροφοριακών συστημάτων. Τέτοια συστήματα είναι κυρίως εκείνα τα οποία έχουν ως κύρια αρμοδιότητα την χαρτογράφηση περιοχών, δικτύων (οδικών, τηλεφωνικών, υπολογιστικών) όπως και σε άλλες πολλές εφαρμογές. Η ένωση αυτών των δύο εννοιών δημιούργησε τις χωροχρονικές βάσεις δεδομένων, οι οποίες δίνουν την δυνατότητα να χρησιμοποιούνται διάφορες καταστάσεις των χωρικών βάσεων δεδομένων σε συνάρτηση με τον χρόνο. Δηλαδή υπάρχει ένα εργαλείο για την προσπέλαση και την δυνατότητα χωρικών ερωτημάτων στην βάση δεδομένων σε καταστάσεις του παρόντος καθώς και του παρελθόντος.

1.1 Περίληψη

Σκοπός αυτής της πτυχιακής εργασίας είναι η δημιουργία και η εκτέλεση μιας χωροχρονικής βάσης δεδομένων, στην οποία προσομοιάζεται η διαχείριση και ο έλεγχος των εργαζομένων μιας πολυεθνικής εταιρίας. Μέσα από την μελέτη αυτή, δημιουργήθηκαν ανάγκες για την σωστή διαχείριση και καταχώρηση δεδομένων καθώς και πληροφοριών που αφορούν τους υπαλλήλους της εταιρίας.

Πιο συγκεκριμένα, για την επίτευξη του αρχικού στόχου χρησιμοποιήθηκε η ORACLE DATABASE, όπου έγινε διαχείριση της βάσης μας με το SQL DEVELOPER. Ο αλγόριθμος και το γραφικό περιβάλλον δημιουργήθηκαν στην γλώσσα προγραμματισμού JAVA.

Το πρόγραμμα που αναπτύχθηκε, δίνει την δυνατότητα σε ένα διαχειριστή να έχει εικόνα όλων τον εργαζομένων μιας επιχείρησης και να γνωρίζει ανα πάσα στιγμή πληροφορίες για οποιοδήποτε εργαζόμενο. Έχει δοθεί μεγάλη βαρύτητα στις χωρικές βάσεις δεδομένων, έτσι ώστε, όταν ο διαχειριστής σημειώσει κάποια ενέργεια, η βάση να ενεργήσει σωστά λαμβάνοντας υπόψιν της τα αξιώματα που της έχουν τεθεί. Επίσης, έχει δοθεί αντίστοιχη βαρύτητα στις χρονικές βάσεις δεδομένων, έτσι ώστε να καταγράφονται με Timestamps οι καταστάσεις κάθε εργαζομένου.

1.2 Κίνητρο για την Διεξαγωγή της Εργασίας – Στόχοι

Μια εταιρία όταν έχει μεγάλο όγκο υπαλλήλων και πόσο μάλιστα μια πολυεθνική εταιρία, που μπορεί να έχει καταστήματα σε διάφορα μέρη του κόσμου, είναι πολύ δύσκολο να έχει εικόνα των υπαλλήλων της. Έτσι, δημιουργήθηκε η ανάγκη να αναπτυχθεί ένα εργαλείο, ακολουθώντας χωροχρονική διαμόρφωση σε μια βάση δεδομένων για τον έλεγχο κατάστασης των εργαζομένων. Στόχος είναι να αποδωθεί και να υλοποιηθεί σε επίπεδο εφαρμογής, ένα απλοϊκό χωρο-χρονικό μοντέλο βάσεις δεδομένων.

2. Βάσεις δεδομένων

Καθημερινά συναντάμε διάφορα προβλήματα όπου πρέπει να τα αντιμετωπίσουμε και να βρούμε μια λύση. Επίσης καθημερινά βρισκόμαστε αντιμέτωποι από το να πάρουμε κάποιες αποφάσεις είτε είναι εύκολες είτε είναι δύσκολες και χρειάζεται να το μελετήσουμε για να καταλήξουμε στην καταλληλότερη. Ένα παράδειγμα που μπορούμε να δώσουμε είναι το εξής: οι περισσότεροι έχουν βρεθεί στην θέση να στέκονται σε μια βιτρίνα με ρούχα και να βλέπουν διάφορες τιμές από μπλούζες, παντελόνια, παπούτσια και διάφορα άλλα είδη. Κάθε ένα από αυτά έχει μια τιμή πχ. μπλούζα 40 ευρώ , παπούτσια 100 ευρώ, παντελόνι 30 ευρώ, καπέλο 20 ευρώ. Γνωρίζοντας εμείς ότι διαθέτουμε για παράδειγμα 150 ευρώ αρχίζουμε και κάνουμε διάφορους υπολογισμούς για να καταλήξουμε στο τι μπορούμε να πάρουμε με τα χρήματα που διαθέτουμε, δηλαδή να καταλήξουμε στην καλύτερη δυνατή απόφαση. Όλα τα παραπάνω είναι δεδομένα. Οι ετικέτες με τις τιμές, τα χρήματα που διαθέτουμε όλα είναι δεδομένα τα οποία πρέπει να τα επεξεργαστούμε για να αντλήσουμε χρήσιμες και αξιοποιήσιμες πληροφορίες για την τελική μας απόφαση. Τα δεδομένα λοιπόν μπορούν να εκφραστούν από ένα πλήθος αριθμών, λέξεων, ποσοτήτων, ιδεών και λειτουργιών. Το επόμενο στάδιο είναι να χρησιμοποιήσουμε όλα τα δεδομένα που αντλήσαμε να τα επεξεργαστούμε για να συλλέξουμε πληροφορίες οι οποίες θα μας οδηγήσουν να φτάσουμε στην απόφαση. Στην περίπτωση μας με διάφορες πράξεις θα καταλήξουμε στην ιδανικότερη αγορά με τα χρήματα που διαθέτουμε έπειτα από πληροφορίες που θα εξάγουμε από πράξεις που θα κάνουμε επεξεργάζοντας τα δεδομένα μας.

2.1 Τι είναι βάσεις δεδομένων

Με τον όρο Βάση Δεδομένων (database) εννοούμε ένα σύνολο από πληροφορίες οι οποίες είναι οργανωμένες με τέτοιο τρόπο ώσπου να έχουμε την δυνατότητα να τις διαχειριστούμε να τις ενημερώνουμε έχοντας εύκολη πρόσβαση σε αυτές[7]. Οι βάσεις δεδομένων είναι ¨χτισμένες¨ για να απλουστεύουν την αποθήκευση δεδομένων, την ανάκτηση τους, καθώς και την επεξεργασία και την διαγραφή τους σε συνεργασία με διάφορες άλλες λειτουργίες επεξεργασίας δεδομένων[8].

Στα αρχαία χρόνια όπου δεν είχε αναπτυχθεί η τεχνολογία επομένως δεν υπήρχαν υπολογιστές, τα δεδομένα τα αποθήκευαν σε ογκώδη αποθετήρια δεδομένων τα οποία ονομάζουμε βιβλία. Έπειτα με την πάροδο των χρόνων η βελτίωση της τεχνολογίας καθώς και η επέκταση της γνώσης δημιούργησε την ανάγκη να μεταφερθούν όλες οι κοινότητες βιβλίων στις πρώτες πραγματικές βιβλιοθήκες δηλαδή σε μια ¨βάση δεδομένων¨. Ο σκοπός της βιβλιοθήκης ήταν να διασφαλιστεί ότι τα δεδομένα θα μπορούν αποθηκευτούν και να ανακτηθούν εύκολα και αποτελεσματικά. Από το 1960 έως σήμερα, από τότε που άρχισε να εξελίξετέ ο υπολογιστής, έχουμε διάφορα μοντέλα βάσεων δεδομένων που χρησιμοποιούταν. Κάθε εάν από αυτά είχε πλεονεκτήματα καθώς και μειονεκτήματα[9]. Το 1960-1980 χρησιμοποιήθηκε το μοντέλο Flat Files. Το 1970-1990 έχουμε το μοντέλο Hierarchical database όπου περιείχε δεδομένα ιεραρχικά διευθετημένα[10]. Μπορούμε να το

παρομοιάσουμε σαν οικογενειακό δέντρο όπου υπάρχει σχέση γονέα και παιδιού και συνεπώς κάθε γονέας μπορεί να έχει πολλά παιδιά ενώ ένα παιδί μπορεί να έχει ένα γονέα κτλ. Το 1970-1990 έχουμε και το μοντέλο Network Database το οποίο εφευρέτης του είναι ο Charles Bachmann[10]. Το μοντέλο βάσης δεδομένων δικτύου ήταν μια εξέλιξη του μοντέλου ιεραρχικής βάσης δεδομένων. Σχεδιάστηκε με σκοπό να επιλύσει κάποια προβλήματα του Hierarchical database και πιο συγκεκριμένα την έλλειψη ευελιξίας. Αυτό το μοντέλο επιτρέπει σε κάθε παιδί να έχει πολλούς γονείς κάτι που δεν επιτρεπόταν στο ιεραρχικό μοντέλο. Επομένως αντιπροσωπεύει πιο πολύπλοκες σχέσεις δεδομένων. Από το 1980 εώς σήμερα χρησιμοποιούμαι το μοντέλο Relational database (σχεσιακό μοντέλο βάσης δεδομένων). Το παραπάνω μοντέλο το οποίο προτάθηκε από τον E. F. Codd άφησε πίσω του τα προηγούμενα μοντέλα και έκανε ένα μεγάλο βήμα μπροστά στην ισχυροποίηση των βάσεων δεδομένων. Το σχεσιακό μοντέλο δίνει την δυνατότητα στις οντότητες να συσχετίζονται μεταξύ τους μόνο μέσω ενός κοινού χαρακτηριστικού. Στους πίνακες υπάρχουν πρωτεύοντα κλειδιά τα οποία προσδιορίζουν τις πληροφορίες του πίνακα. Στη συνέχεια η σχέση μεταξύ πινάκων μπορεί να οριστεί μέσω της χρήσης ξένων κλειδιών ενός πεδίου σε έναν πίνακα που συνδέεται με το πρωτεύον κλειδί άλλου πίνακα. Με αυτόν τον τρόπο το σχεσιακό μοντέλο βάσης δεδομένων μας δίνει μια εξαιρετική δυνατότητα, την αποθήκευση άπειρων πληροφοριών χρησιμοποιώντας μικρούς πίνακες. Η προσπέλαση των δεδομένων είναι εξαιρετικά αποτελεσματική έτσι ο χρήστης απλά με την υποβολή ενός ερωτήματος στην βάση του επιστρέφει τις ζητούμενες πληροφορίες[11]. Οι σχεσιακές βάσεις δεδομένων δημιουργούνται χρησιμοποιώντας μια γλώσσα υπολογιστή την SQL (Structured Query Language) η οποία είναι εύκολα αναγνώσιμη από τον άνθρωπο. Από το 1990 έως σήμερα υπάρχει και το μοντέλο Object-oriented database (βάση δεδομένων με βάση τα αντικείμενα). Στο παραπάνω σύστημα βάσης δεδομένων τα δεδομένα ή οι πληροφορίες παρουσιάζονται με μορφή αντικειμένων, όπως και σε κάθε αντικειμενοστραφής γλώσσα προγραμματισμού. Η διαφορά με την σχεσιακή βάση δεδομένων είναι ότι η αντικειμενοστραφής βάση δεδομένων λειτουργεί στο πλαίσιο πραγματικών γλωσσών δεδομένων όπως Java ή C++.

2.2 Συστήματα Διαχείρισης Βάσεων Δεδομένων(ΣΔΒΔ)

Για να δημιουργήσουμε και να διαχειριστούμε μια βάση δεδομένων χρειαζόμαστε ένα λογισμικό συστήματος δηλαδή ένα σύστημα διαχείρισης βάσεων δεδομένων (DBMS)[12]. Το ΣΔΒΔ δίνει την δυνατότητα στον χρήστη καθώς και στους προγραμματιστές να δημιουργούν να τροποποιούν, να ανακτούν, να διαγράφουν, να ενημερώνουν καθώς και να διαχειρίζονται τα δεδομένα. Το ΣΔΒΔ είναι πολύ χρήσιμο διότι είναι το μέσο για την διεπαφή μεταξύ της βάσης δεδομένων και των τελικών χρηστών ή των εφαρμογών επιτυγχάνοντας ότι τα δεδομένα μας είναι οργανωμένα και έχουν εύκολη προσπέλαση. Το μεγαλύτερο μέρος των συστημάτων διαχείρισης βάσεων δεδομένων χρησιμοποιούν την δομημένη γλώσσα

αναζήτησης SQL[13]. Τα περισσότερα ΣΔΒΔ για να διευκολύνουν τους χρήστες παρέχουν ένα γραφικό περιβάλλον στο οποίο μπορείς εύκολα να διαχειριστείς και να επεξεργαστείς τα δεδομένα όμως στο παρασκήνιο αυτές τις διεργασίες εκτελούνται μέσω της SQL. Σήμερα υπάρχουν πολλά εμπορικά ΣΔΒΔ τα οποία είναι ανοικτού κώδικα. Στην πραγματικότητα για να επιλέξουμε την καταλληλότερη ΣΔΒΔ είναι μια πολύπλοκη εργασία. Τα ΣΔΒΔ υψηλού επιπέδου που βρίσκονται στην κορυφή της αγοράς στις μέρες μας είναι η Oracle, Microsoft SQL Server , IBM DB2 τα οποία είναι από τις πιο αξιόπιστες επιλογές για μεγάλα συστήματα δεδομένων[14]. Για μικρές εταιρίες ή για οικιακή χρήση τα ΣΔΒΔ που χρησιμοποιούνται πιο συχνά είναι η Microsoft Access όπως και το FileMakerPro.

2.3 Χρονικές Βάσεις Δεδομένων (Temporal Databases)

Στις βάσεις δεδομένων όπως και στην πραγματικότητα ένα πολύ σημαντικό στοιχείο είναι ο χρόνος διότι συνδέεται με σχεδόν τα πάντα που μας απασχολούν. Κατά την διάρκεια χρόνου τα δεδομένα μας και συνεπώς οι πληροφορίες μας συχνά μεταβάλλονται και αυτό δημιουργεί διάφορα προβλήματα στην βάση δεδομένων μας. Πρέπει όμως να υπάρχει πλήρη υποστήριξη της χρονικά μεταβαλλόμενης φύσης των πραγμάτων που αναπαριστούν, για αυτό το λόγο δημιουργήθηκαν οι χρονικές βάσεις δεδομένων (Temporal Databases)[15]. Σε συγκεκριμένα χρονικά σημεία καταγράφονται συμβάντα στα οποία τα αντικείμενα και οι σχέσεις μεταξύ των αντικειμένων, μεταβάλλονται στο πέρασμα του χρόνου. Η μοντελοποίηση της χρονικής διάστασης στον πραγματικό κόσμο είναι μια ικανότητα πολύ σημαντική και πλήρως απαραίτητη για πολλές εφαρμογές στην πληροφορική. Σε διάφορους κλάδους συναντάμε βάσεις δεδομένων που είναι σε συνάρτηση με τον χρόνο, όπως στην λογιστική, στην οικονομετρία, σε συστήματα γεωγραφικών πληροφοριών, σε έλεγχο συστημάτων κρατήσεων, σε τράπεζες, ακόμα και σε κλάδους με επιστημονική ανάλυση δεδομένων και σε άλλες πολλές περιπτώσεις[15]. Οι συμβατικές βάσεις δεδομένων δεν είναι χρήσιμες σε τέτοιες περιπτώσεις διότι απευθύνονται σε μια μόνο κατάσταση της επιχείρησης σε μια χρονική στιγμή. Το πρόβλημα που δημιουργείτε με τις συμβατικές βάσεις δεδομένων είναι ότι ενώ τα περιεχόμενα της βάσης συνεχίζουν τα μεταβάλλονται καθώς προστίθενται καινούριες πληροφορίες, αυτές οι μεταβολές αναγνωρίζονται ως τροποποιήσεις στην υπάρχουσα κατάσταση, με αποτέλεσμα τα παλαιότερα δεδομένα να διαγράφονται από την βάση δεδομένων. Τα περιεχόμενα που περιέχει μια βάση δεδομένων σε τρέχων χρόνο θεωρούνται ως στιγμιότυπα. Σε μια βάση δεδομένων σε συνάρτηση με το χρόνο (temporal database) έχουμε πλήρη υποστήριξη της διατήρησης των χρονικά μεταβαλλόμενων δεδομένων και ειδικευμένων ερωτημάτων που σχετίζονται με το παρελθόν, το παρόν και το μέλλον των δεδομένων αυτών, κάτι το οποίο είναι αδύνατο να συμβεί στις συμβατικές βάσεις δεδομένων[15].

2.4 Χωρικές Βάσεις Δεδομένων (Spatial Databases)

Μια άλλη εξίσου πολύ σημαντική έννοια που συνδέεται με τις βάσεις δεδομένων εκτός του χρόνου είναι και ο χώρος. Για αυτό το λόγο πέρα από τις χρονικές βάσεις δεδομένων (Temporal Databases), έχουμε και τις χωρικές βάσεις δεδομένων (Spatial Databases), οι οποίες είναι βάσεις δεδομένων σχεδιασμένες για την αποθήκευση και την πρόσβαση σε χωρικά δεδομένα ή δεδομένα που ορίζουν τον γεωμετρικό χώρο[16][17]. Τέτοια δεδομένα συνδέονται κυρίως με γεωγραφικές τοποθεσίες. Η αποθήκευση των δεδομένων σε μια χωρική βάση δεδομένων έχουν την μορφή των συντεταγμένων, σημείων, γραμμών, πολυγώνων και τοπολογιών[17]. Μια πιο δύσκολη κατάσταση που μπορεί να χειριστεί μια χωρική βάση δεδομένων είναι να χειριστεί δεδομένα τα οποία είναι πιο σύνθετα, δηλαδή τρισδιάστατα αντικείμενα, τοπολογική κάλυψη και γραμμικά δίκτυα[16].

2.5 Χωρο-Χρονικές Βάσεις Δεδομένων (SpatioTemporal Databases)

Μέχρι τώρα έχουμε δει την συνάρτηση των βάσεων δεδομένων με τον χρόνο δηλαδή τις χρονικές βάσεις δεδομένων (Temporal Databases) και την συνάρτηση των βάσεων δεδομένων με τον χώρο δηλαδή τις χωρικές βάσεις δεδομένων (Spatial Databases). Η ανάγκη όμως να υπάρξει μια βάση δεδομένων σε συνάρτηση του χρόνου και του χώρου δημιούργησε τις χωρο-χρονικές βάσεις δεδομένων (Spatiotemporal Databases)[15]. Οι χωρο-χρονικές βάσεις δεδομένων διαχειρίζονται χωρικά δεδομένα στα οποία τα γεωμετρικά χαρακτηριστικά μεταβάλλονται δυναμικά. Οι χωρο-χρονικές βάσεις δεδομένων έχουν πολλές σημαντικές εφαρμογές, τις χρησιμοποιούμε σε Συστήματα Γεωγραφικών πληροφοριών, σε Συστήματα Εντοπισμού Θέσης (GPS), σε Συστήματα Παρακολούθησης κυκλοφορίας ακόμα και σε Συστήματα Περιβαλλοντικών Πληροφοριών[18]. Επομένως είναι αντιληπτό ότι τα συστήματα χωρο-χρονικών βάσεων δεδομένων είναι πολύ ενεργά στον τομέα των βάσεων δεδομένων[16].

3. Μέθοδοι Υλοποίησης

3.1 Γλώσσα Προγραμματισμού JAVA

Η java είναι μια γλώσσα που δημιουργήθηκε από τον James Gosling το 1991, όπου εργαζόταν στην εταιρία Sun Microsystems και φτιάχτηκε από την ανάγκη για ένα καινούργιο εργαλείο για την ανάπτυξη λογισμικού σε μικροσυσκευές. Η αρχική ονομασία της Java ήταν oak και ήταν εμπνευσμένο από μία βελανιδιά που βρισκόταν έξω από το γραφείο του James, όπου περνούσε τον περισσότερο χρόνο του. Αργότερα για λόγους πνευματικών δικαιωμάτων πήρε την παρούσα ονομασία εμπνευσμένη από τον Java coffee όπου ήταν και ο αγαπημένος του καφές. Τα εργαλεία εκείνης της εποχής ήταν η C και η C++ και ύστερα από δόκιμες κατέληξαν στο συμπέρασμα ότι οι υπάρχουσες γλώσσες δεν μπορούσαν να καλύψουν τις ανάγκες εκείνης της δουλειάς. Έτσι όπως ήταν λογικό η java φτιάχτηκε στα πρότυπα και στην σύνταξη της C/C++, έτσι ώστε να φαίνεται γνώριμο και εύκολο στους προγραμματιστές της εποχής.

Ή πρώτη έκδοση της Java εμφανίστηκε από την Sun Microsystems το 1996 και είχε υποσχεθεί ότι θα μπορούσε να τρέξει σε οποιαδήποτε πλατφόρμα χωρίς επιπλέον κόστος, πράγμα που την έκανε αρκετά δημοφιλή και εύκολη στην χρήση της. Για να γίνει αυτό και να είναι κατανοητό από κάθε σύστημα ανεξαρτήτου επεξεργαστή και λειτουργικού συστήματος , δημιουργήθηκε μια εικονική μηχανή που η δουλειά της ήταν να δημιουργεί μια σειρά από αρχεία .class, όπου ήταν η μορφή που παίρνει ο πηγαίος κώδικας της java όταν μεταγλωττιστεί. Έτσι όταν πρόκειται να εκτελεστεί σε ένα μηχάνημα, το java virtual machine που είναι εγκατεστημένο σε αυτό, θα διαβάσει τα αρχεία .class και θα κάνει την μετάφραση σε γλωσσά μηχανής που υποστηρίζει ο επεξεργαστής και το λειτουργικό σύστημα.

Παρόλο που η εικονική μηχανή προσφέρει όλα αυτά τα πλεονεκτήματα, η Java αρχικά ήταν πιο αργή σε σχέση με άλλες γλώσσες όπως η δημοφιλής για την εποχή C++. Μετρήσεις που έγιναν απέδειξαν ότι η C++ είναι αρκετές φορές γρηγορότερη από την Java. Έτσι γίνονται συνεχές βελτιώσεις και προσπάθειες από τη Oracle για τη βελτιστοποίηση της εικονικής μηχανής. Επιπλέον με την καθιέρωση των μεταγλωττιστών JIT (Just In Time), οι οποίοι μετατρέπουν τον κώδικα byte απευθείας σε γλώσσα μηχανής, η διαφορά ταχύτητας από τη C++ έχει μικρύνει κατά πολύ.

Το πλεονέκτημα της γλώσσας είναι η αντικειμενοστρέφια που προσφέρει και βασίζεται στην χρήση αντικειμένων. Τα αντικείμενα είναι πεδία πληροφορίας, μέθοδος προβολής και επεξεργασίας και ανήκουν σε κλάσεις, οι οποίες δηλώνονται με έναν τύπο και ένα μετατροπέα ορατότητας. Υπάρχουν τέσσερεις μετατροπείς ορατότητας, public, protect, private, package-private. Τα public είναι ορατά σε όλες τις κλάσεις της εφαρμογής. Τα protected είναι ορατά από κλάσεις του ίδιου πακέτου

ή εάν είναι extends στην κλάση και σε εκτός πακέτου. Τα private είναι ορατά μόνο από την ίδια κλάση και τέλος τα package-private είναι ορατά από κλάσεις του ίδιου πακέτου. Ακολουθεί παράδειγμα αντικειμένου για την παρακάτω κλάση.

3.2 Γλώσσα προγραμματισμού SQL

Η γλώσσα προγραμματισμού SQL αναπτύχθηκε για πρώτη φορά στη δεκαετία του 1970 από τους ερευνητές της IBM Raymond Boyce και Donald Chamberlin. Η γλώσσα προγραμματισμού, γνωστή στη συνέχεια ως SEQUEL, δημιουργήθηκε μετά από τη δημοσίευση του εγγράφου Edgar Frank Todd[19]. "Ένα σχεσιακό μοντέλο δεδομένων για μεγάλες κοινές τράπεζες δεδομένων", το 1970[19]. Στην εργασία του, ο Todd πρότεινε να εκπροσωπούνται όλα τα δεδομένα σε μία βάση δεδομένων με τη μορφή σχέσεων. Βασίστηκε σε αυτή τη θεωρία ότι ο Boyce και ο Chamberlin πρότειναν την SQL. Στο βιβλίο "Oracle Quick Guides (Cornelio Books 2013)", ο συγγραφέας Malcolm Coxall γράφει ότι η αρχική έκδοση SQL σχεδιάστηκε για να χειραγωγήσει και να ανακτήσει τα δεδομένα που αποθηκεύονται στα αρχικά συστήματα διαχείρισης σχεσιακών βάσεων της IBM, γνωστά ως "System R." Μόνο μερικά χρόνια αργότερα, η γλώσσα SQL διατέθηκε δημοσίως. Το 1979, μια εταιρεία που ονομάστηκε Relational Software, η οποία αργότερα έγινε Oracle, κυκλοφόρησε εμπορικά τη δική της έκδοση της γλώσσας SQL που ονομάζεται Oracle V2. Έκτοτε, το Αμερικανικό Εθνικό Ινστιτούτο Προτύπων (ANSI) και ο Διεθνής Οργανισμός Τυποποίησης έκριναν ότι η γλώσσα SQL αποτελεί την τυπική γλώσσα στην επικοινωνία σχεσιακής βάσης δεδομένων. Ενώ μεγάλοι προμηθευτές SQL τροποποιούν τη γλώσσα στις επιθυμίες τους, οι περισσότεροι βασίζουν τα προγράμματα SQL τους από την έκδοση που έχει εγκριθεί από το ANSI[20].

Η SQL, η οποία αντιπροσωπεύει τη δομημένη γλώσσα ερωτημάτων, είναι μια γλώσσα προγραμματισμού που χρησιμοποιείται για να επικοινωνεί και να χειρίζεται βάσεις δεδομένων. Προκειμένου να αξιοποιηθούν τα περισσότερα από τα δεδομένων που συλλέγουν, πολλές επιχειρήσεις πρέπει να επενδύσουν στην SQL. Τα προγράμματα SQL δημιουργούνται από επιχειρήσεις και άλλους οργανισμούς ως τρόπος πρόσβασης και χειρισμού των πληροφοριών και των δεδομένων που αποθηκεύονται στις βάσεις δεδομένων τους, καθώς και για τη δημιουργία και την τροποποίηση νέων πινάκων. Για να κατανοηθεί πλήρως η SQL, είναι σημαντικό να είναι γνωστό πρώτα τι ακριβώς είναι μια βάση δεδομένων. Σύμφωνα με τη Microsoft, μια βάση δεδομένων είναι ένα εργαλείο συλλογής και οργάνωσης πληροφοριών[19]. Οι βάσεις δεδομένων μπορούν να αποθηκεύουν πληροφορίες για ανθρώπους, προϊόντα, παραγγελίες ή οτιδήποτε άλλο. Πολλές βάσεις δεδομένων αρχίζουν σε ένα πρόγραμμα επεξεργασίας κειμένου ή υπολογιστικό φύλλο, αλλά καθώς μεγαλώνουν, πολλές επιχειρήσεις θα είναι χρήσιμες να τις μεταφέρουν σε μια βάση δεδομένων που δημιουργείται από ένα σύστημα διαχείρισης βάσεων δεδομένων. Για τον έλεγχο των πληροφοριών σε αυτές τις βάσεις δεδομένων χρησιμοποιείται SQL, η οποία επιτρέπει

στους χρήστες να ανακτούν τα συγκεκριμένα δεδομένα που αναζητούν όταν το χρειάζονται. Ενώ είναι μια απλή γλώσσα προγραμματισμού, η SQL είναι επίσης πολύ ισχυρή. Μια βάση δεδομένων λέει ότι η SQL μπορεί να εισάγει δεδομένα σε πίνακες βάσεων δεδομένων, να τροποποιεί δεδομένα σε υπάρχοντες πίνακες βάσεων δεδομένων και να διαγράφει δεδομένα από πίνακες βάσεων δεδομένων SQL. Επιπλέον, η SQL μπορεί να τροποποιήσει την ίδια τη δομή της βάσης δεδομένων δημιουργώντας, τροποποιώντας και διαγράφοντας πίνακες και άλλα αντικείμενα βάσης δεδομένων[20].

4. Κύριο μέρος Πτυχιακής Εργασίας

Το κεφάλαιο αυτό, πραγματεύεται το κυριότερο μέρος της πτυχιακής εργασίας, ενώ επισημαίνει αφενός την ανάλυση του προβλήματος και αφετέρου την υλοποίηση και επίλυση του θέματος. Το μοντέλο το οποίο δουλεύτηκε είναι η διαχείριση μιας βάσης δεδομένων για πολυεθνική εταιρία με μεγάλο πλήθος εργαζομένων. Σκοπός είναι, η εύκολη διαχείριση της βάσης από το διαχειριστή αλλά και η διατήρηση όλων των σχέσεων μεταξύ των πινάκων, καθώς και των αξιωμάτων που τέθηκαν για τη σωστή λειτουργία του συστήματος.

4.1 Ανάλυση Προβλήματος

Ένα από τα προβλήματα που παρουσιάστηκαν, είναι αυτό του Ramification Problem. Επομένως, για να δοθεί λύση, έπρεπε με κάποιο τρόπο να υπάρξει χειρισμός των έμμεσων αλλά και των άμεσων αλλαγών μιας πράξης, έτσι ώστε να διατηρηθεί η ακεραιότητα των περιορισμών που έχουν τεθεί. Παρακάτω θα περιγραφεί το Ramification Problem με ένα πολύ απλό παράδειγμα. Έστω ότι υπάρχει ένας κινητήρας, ο οποίος για να λειτουργήσει χρειάζεται δύο διακόπτες (switch1,switch2). Οι διακόπτες μπορούν να έχουν δύο καταστάσεις. Την κατάσταση ανοιχτού κυκλώματος(0) και την κατάσταση κλειστού κυκλώματος(1). Επίσης και ο κινητήρας (engine) έχει ακριβώς τις ίδιες καταστάσεις. Για να βρεθεί όμως, ο κινητήρας σε κατάσταση κλειστού κυκλώματος θα πρέπει να βρίσκονται και οι δύο διακόπτες στην κατάσταση 1. Έτσι λοιπόν, αν ο switch1 βρίσκεται στην κατάσταση 0 και ο switch2 βρίσκεται στην κατάσταση 1 ,τότε ο engine οφείλει να βρίσκεται στην κατάσταση 0. Αν αλλάξει η κατάσταση του switch1 σε 1 ,τότε πρέπει να τηρηθεί ο περιορισμός ακεραιότητας που έχουμε θέσει και ο engine να βρεθεί σε κατάσταση 1. Επομένως αντιλαμβανόμαστε, ότι το Ramification Problem αναφέρεται στις έμμεσες επιδράσεις μια ενέργειας υπό την παρουσία περιορισμών.

Το πρόβλημα το προσεγγίστηκε με την βοήθεια της διαδοχικής εκτέλεσης, προσέγγιση η οποία επεκτείνει τη λύση που είχαν προτείνει ο McCain και ο Turner. Κατά αυτόν τον τρόπο, ακολουθήθηκαν συγκεκριμένα αξιώματα τα οποία περιγράφουν το άμεσο αποτέλεσμα μιας ενέργειας. Παρακάτω θα υπογραμμιστούν τα αξιώματα τα οποία χρησιμοποιήθηκαν για την επίλυση του στόχου.

Αξιώματα:

occur(misdemeanor(p), t) ⊃ illegal(p, 5m) (1)

occur(take_pardon(p), t) ⊃ ¬ illegal(p, ∞) (2)

occur(bad_grade(p), t) ⊃ ¬ good_employee(p, ∞) (3)

occur(good_grade(p), t) ⊃ good_employee(p, ∞) (4)

illegal(p, t_1) ⊃ suspended(p,t_1) (5)

illegal(p, t_1) ⊃ ¬ take_promotion (p,t_1) (6)

suspended(p, t_1) ⊃ ¬ take_salary (p,t_1) (7)

¬ good_employee (p, t_1) ⊃ ¬ take_promotion (p,t_1) (8)

¬ suspended (p, t_1) ∧ good_employee (p,t_2) ⊃ takebonus(p, min(t_1, t_2) (9)

¬ good_employee (p, t_1) ⊃ ¬ take_bonus (p, t_1) (10)

¬ suspended (p, t_1) ⊃ take_salary (p,t_1) (11)

Το πρώτο αξίωμα εκφράζει το αποτέλεσμα της πράξης misdemeanor σε ένα υπάλληλο. Αν λοιπόν αποδειχθεί ότι ένας υπάλληλος έχει διαπράξει κάποιο πλημμέλημα σε χρόνο t, σημαίνει ότι αμέσως τίθεται illegal για τους επόμενους 5 μήνες. Ο μόνος τρόπος για να αλλάξει αυτή η κατάσταση, εκτός του να περάσει ο χρόνος που έχει οριστεί, είναι να ενεργοποιηθεί η κατάσταση take_pardon (βλ. αξίωμα (2)) σε t<5 μηνών και με αυτό τον τρόπο παύει να είναι illegal για t = ∞ μέχρις ότου να ενεργοποιηθεί μια κατάσταση όπου θα ανατρέξει αυτόν τον χρόνο.

Στην συνέχεια επισημαίνεται στο αξίωμα (5), ότι από την στιγμή όπου ο υπάλληλος τεθεί illegal μέσω του αξιώματος (1) τίθεται αυτόματα ως suspended και έχει ως αποτέλεσμα το να μην μπορεί να πάρει μισθό, πράγμα που αποδεικνύεται μέσω του αξιώματος (7). Γίνεται αντιληπτό, ότι οι περιορισμοί ακεραιότητας πρέπει να ακολουθούνται πιστά για να μην δημιουργούνται λογικά λάθη. Συνεπώς, όσο μεγαλύτερος είναι ο όγκος των αναγκαίων δεδομένων και περιορισμών, τόσο δυσκολότερο γίνεται το εγχείρημα αυτό.

Όπως αναφέρθηκε παραπάνω , υπάρχουν 11 αξιώματα που πρέπει να λάβουμε υπόψιν μας. Άρα κάθε ενέργεια που γίνεται στη βάση, πρέπει να συμμορφώνεται στους κανόνες που έχουν τεθεί για τη σωστή λειτουργία της. Άξια λόγου, αποτελεί η σχέση (1) occur(misdemeanor(p), t) ⊃ illegal(p, 5m) ,όπου μόλις κάποιος διαχειριστής θέσει ως misdemeanor κάποιον υπάλληλο, αυτόματα τίθεται και ως illegal για πέντε μήνες. Παρατηρείται όμως, στη σχέση (5) illegal(p, t_1) ⊃ suspended(p,t_1) πως, όταν κάποιος υπάλληλος είναι illegal θεωρείται και αυτόματα suspended, ενώ δεν μπορεί να πάρει προαγωγή , πράγμα που τονίζεται στη σχέση (6). Στην σχέση (7) παρατηρούμε ότι, εάν κάποιος υπάλληλος θεωρείται suspended δεν μπορεί να πάρει μισθό. Αυτό δημιουργεί διάφορα προβλήματα διότι με μια αλλαγή αρχίζει ένα ντόμινο από άλλων αλλαγών, οι οποίες θα πρέπει πάντα να συμμορφώνονται στα αξιώματα που έχουν ορισθεί.

4.2 Σχεδιασμός Υλοποίησης

Σκοπός λοιπόν, είναι η υλοποίηση μιας βάσης δεδομένων όπου θα επιτρέπει τις παραπάνω λειτουργίες και ταυτόχρονα θα λαμβάνει και υπόψιν τους περιορισμούς ακεραιότητας που έχουν τεθεί. Θα γίνει προσπάθεια δημιουργίας λοιπόν, μιας βάσης, η οποία θα αποτελείται από τους απαραίτητους πίνακες ενώ θα υπάρχουν και κάποιοι triggers, όπου θα παίρνουν τον ρόλο των αυτόματων αλλαγών που θα πραγματοποιούνται λαμβάνοντας υπόψιν πάντα τους περιορισμούς ακεραιότητας. Επομένως, θα χρειαστεί σίγουρα η δημιουργία ενός πίνακα για τους εργαζόμενους στον οποίο θα υπάρχουν τα δημογραφικά τους στοιχεία. Ταυτόχρονα, θα χρειαστεί ένας δεύτερος, όπου θα συνδέεται με τον πρώτο και στον οποίο, θα υπάρχουν οι ενέργειες που θα μπορεί να πραγματοποίησει ο διαχειριστής στο εκάστοτε εργαζόμενο. Τέλος θα υπάρξει ανάγκη ενός τρίτου, στον οποίο βάσει των ενεργειών θα επηρεάζονται τα ανάλογα πεδία, λαμβάνοντας υπόψιν τα αξιώματα που έχουμε ορίσει και ενός τέταρτου, που θα λειτουργεί ως ιστορικό για τον κάθε υπάλληλο και για να γνωρίζει ο διαχειριστής τις ενέργειες που έχουν πραγματοποιηθεί για κάθε εργαζόμενο. Στην συνέχεια, πρέπει να δημιουργηθούν οι triggers που θα διατηρούν τους περιορισμούς, και θα γίνονται οι κατάλληλες αλλαγές που χρειάζονται ανάλογα την ενέργεια του διαχειριστή. Είναι αναγκαίος ένας trigger για κάθε υπάλληλο που εισέρχεται για πρώτη φορά στο σύστημα και στον οποίο θα δημιουργεί με το id του πεδία στους πίνακες με τα αξιώματα και με τις ενέργειες. Είναι αυτονόητο, πως όταν συμβαίνει αυτό θα αρχικοποιείται σαν να μην έχει δεχθεί καμία ενέργεια από τον διαχειριστή. Κάποιος άλλος trigger θα λαμβάνει τον ρόλο της ενέργεια misdemeanor, δηλαδή θα τίθεται ως suspended ενώ ο υπάλληλος δεν θα μπορεί να πάρει προαγωγή, δεν θα μπορεί να πάρει μισθό και ότι άλλο απαιτούν οι περιορισμοί ακεραιότητας. Θα χρειαστεί επίσης ένας trigger για την περίπτωση όπου ο διαχειριστής συγχωρήσει τον υπάλληλο, άλλος ένας σε περίπτωση επιβράβευσης και τέλος ένας τρίτος σε περίπτωση που ο διαχειριστής επιλέξει την ενέργεια BAD_GRADED. Κάθε trigger θα ενημερώνει και τον πίνακα με το ιστορικό του κάθε υπαλλήλου.

4.3 Υλοποίηση

Έτσι δημιουργήθηκε μια βάση δεδομένων, η οποία απαρτίζεται από τέσσερις πίνακες, τον πίνακα EMPLOYEE, τον FLUENTS, τον ACTION και τον HISTORY και πέντε triggers, τον EMPLOYEE_TRIGGER, τον MISDEMEANOR_TRIGGER, τον BAD_GRADE_TRIGGER, τον GOOD_GRADE_TRIGGER και τέλος τον TAKE_PARDON_TRIGGER.

1. Ο πίνακας EMPLOYEE όπου έχει τρία πεδία. Στο πρώτο πεδίο, όπου είναι και το κύριο κλειδί του πίνακα, είναι το ID κάθε εργαζόμενου, το πεδίο NAME και το πεδίο LASTNAME. Το ID είναι πεδίο τύπου number, ενώ το NAME και το LASTNAME είναι τύπου varchar. Παρακάτω φαίνεται η δήλωση του πίνακα.

```
CREATE TABLE "C##"."EMPLOYEE"
 (  "ID" NUMBER,
"NAME" VARCHAR2(20 BYTE),
"LASTNAME" VARCHAR2(20 BYTE)
 ) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS" ;
```

Εικόνα 1 Employee

2. Ο πίνακας FLUENTS απαρτίζεται από οκτώ πεδία. Στο πρώτο πεδίο, το οποίο είναι και το κύριο κλειδί του πίνακα συναντάμε το ID_ACTION ενώ είναι και foreign key για τον πίνακα ACTION, όπως θα υπογραμμισθεί παρακάτω. Ως δεύτερο πεδίο λαμβάνεται το GOOD_EMPLOYEE, ως τρίτο το BAD_EMPLOYEE, ως τέταρτο το ILLEGAL, ως πέμπτο το TAKE_SALARY, ως έκτο το TAKE_BONUS,ως έβδομο το TAKE_PROMOTION και τέλος το SUSPENDED. Όσον αφορά τα πεδία και τον τύπο τους ισχύει: Το ID_ACTION, GOOD_EMPLOYEE, BAD_EMPLOYEE, TAKE_SALARY, TAKE_BONUS, TAKE_PROMOTION είναι τύπου number, ενώ τα ILLEGAL, SUSPENDED είναι τύπου timestamp.Παρακάτω αποφαίνεται η δήλωση του πίνακα.

```
CREATE TABLE "C##"."FLUENTS"
 (  "ID_ACTION" NUMBER,
"GOOD_EMPLOYEE" NUMBER DEFAULT 0,
"BAD_EMPLOYEE" NUMBER DEFAULT 0,
"ILLEGAL" TIMESTAMP (6) DEFAULT NULL,
"TAKE_SALARY" NUMBER DEFAULT 1,
"TAKE_BONUS" NUMBER DEFAULT 0,
"TAKE_PROMOTION" NUMBER DEFAULT 0,
"SUSPENDED" TIMESTAMP (6) DEFAULT NULL
 ) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS" ;
```

Εικόνα 2 Fluents

3. Ο πίνακας ACTION απαρτίζεται από πέντε πεδία. Στο πρώτο πεδίο συναντάται το κύριο κλειδί, το οποίο είναι και foreign key για τον πίνακα EMPLOYEE με ονομασία ID_EMPLOYEE, στο δεύτερο πεδίο το MISDEMEANOR, στο τρίτο το TAKE_PARDON, στο τέταρτο το GOOD_GRADE και στο τελευταίο το BAD_GRADE. Το κλειδί του πίνακα είναι τύπου number ,ενώ τα υπόλοιπα τέσσερα είναι timestamp. Παρακάτω φαίνεται η δήλωση του πίνακα.

```
CREATE TABLE "C##"."ACTION"
 (  "ID_EMPLOYEE" NUMBER,
"MISDEMEANOR" TIMESTAMP (6) DEFAULT NULL,
"TAKE_PARDON" TIMESTAMP (6) DEFAULT NULL,
"GOOD_GRADE" TIMESTAMP (6) DEFAULT NULL,
"BAD_GRADE" TIMESTAMP (6) DEFAULT NULL
 ) SEGMENT CREATION IMMEDIATE
PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
TABLESPACE "USERS" ;
```

Εικόνα 3 Action

4. Ο πίνακας HISTORY απαρτίζεται από τέσσερα πεδία. Στο πρώτο πεδίο, όπου είναι και το κύριο κλειδί του πίνακα έχουμε το ID_HISTORY το οποίο είναι και foreign key του πίνακα FLUENTS, στο δεύτερο έχουμε το START_TIMESTAMP, στο τρίτο έχουμε το END_TIMESTAMP και τέλος έχουμε το REASON. Το ID_HISTORY είναι τύπου number ενώ το START_TIMESTAMP και το END_TIMESTAMP είναι τύπου timestamp. Τέλος το REASON είναι τύπου varchar. Παρακάτω υπογραμμίζεται η δήλωση του πίνακα.

```
CREATE TABLE "C##"."HISTORY"
 (  "ID_HISTORY" NUMBER,
"START_TIMESTAMP" TIMESTAMP (6),
"END_TIMESTAMP" TIMESTAMP (6),
"REASON" VARCHAR2(50 BYTE)
 ) SEGMENT CREATION IMMEDIATE
 PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
NOCOMPRESS LOGGING
 STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
 BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
 TABLESPACE "USERS" ;
```

Εικόνα 4 History

Άξιο λόγου αποτελεί το γεγονός, ότι έχει δημιουργηθεί ακόμα ένας πίνακας με όνομα ACTION_TEST και ο οποίος έχει τον ρόλο του διαμεσολαβητή για την σωστή λειτουργία των triggers που έχουν δημιουργηθεί. Αυτός ο πίνακας βοηθάει στο να πραγματοποιηθούν δυο ή περισσότερες εγγραφές συγχρόνως στον αρχικό πίνακα ACTION. Τα πεδία που έχουν τεθεί ως number είναι, είτε κλειδιά για τους πίνακες, είτε λειτουργούν ως μεταβλητές τύπου bit για να γίνει αντιληπτό λόγου χάριν, εάν κάποιος υπάλληλος παίρνει μισθό ή όχι. Τα πεδία που έχουν οριστεί ως timestamps χρησιμοποιούνται για να επισημανθεί η διάρκεια του χρόνου κατά την οποία έχει εκτελεστεί κάποια ενέργεια ή το πόσο θα διαρκέσει κάποια ενέργεια μέχρι να λήξει κάποιο περιστατικό. Οι τέσσερις αυτοί πίνακες αρκούν για να φιλοξενούν όλα τα απαραίτητα δεδομένα και για να είναι λειτουργική η εφαρμογή, ενώ δεν αρκούν για να διατηρήσουμε τους περιορισμούς ακεραιότητας. Επομένως για να διαφυλαχθεί η ορθή λειτουργία αυτών των σχέσεων χρησιμοποιήθηκαν οι triggers. Όπως αναφέρθηκε παραπάνω υλοποιήθηκαν πέντε triggers.

1. Ο EMPLOYEE_TRIGGER ο οποίος την στιγμή που δημιουργείται ένας νέος υπάλληλος ενεργοποιείται και δημιουργεί εγγραφές στους πίνακες ACTION, ACTION_TEST και FLUENTS. Έτσι, κάθε υπάλληλος αρχικοποιείται με συγκεκριμένες τιμές στα fluent και στην πορεία αυτή, μπορεί ο διαχειριστής να επεξεργαστεί αυτή την πληροφορία κατά βούληση του.

```
  CREATE OR REPLACE EDITIONABLE TRIGGER "C##"."EMPLOYEE_TRIGGER"
AFTER INSERT ON EMPLOYEE
REFERENCING NEW AS NEW OLD AS OLD
FOR EACH ROW
BEGIN
INSERT INTO ACTION_TEST
(ID_EMPLOYEE)

VALUES
(:NEW.ID);

INSERT INTO FLUENTS
(ID_ACTION)

VALUES
(:NEW.ID);

INSERT INTO ACTION
(ID_EMPLOYEE)

VALUES
(:NEW.ID);

END;
/
ALTER TRIGGER "C##"."EMPLOYEE_TRIGGER" ENABLE;
```

Εικόνα 5 Employee Trigger

2. Ο MISSDEMEANOR_TRIGGER ο οποίος κατά τη στιγμή που γίνεται κάποιο update στον πίνακα MISSDEMEANOR ενεργοποιείται και κάνει update στα δεδομένα του πίνακα FLUENTS για τον υπάλληλο που έχει επιλεχθεί μέσω του id που έχουν οι πίνακες κοινό. Βάσει τον αξιωμάτων που έχουν οριστεί, ο υπάλληλος τίθεται ως suspended σταματάει να παίρνει μισθό, ενώ,δεν μπορεί να πάρει προαγωγή και ενημερώνεται ο πίνακας history για την αλλαγή που πραγματοποιήσαμε για τον συγκεκριμένο υπάλληλο.

```
  CREATE OR REPLACE EDITIONABLE TRIGGER "C##"."MISDEMEANOR_TRIGGER"
BEFORE UPDATE OF MISDEMEANOR ON "ACTION_TEST"
REFERENCING NEW AS NEW OLD AS OLD

FOR EACH ROW

BEGIN
IF :OLD.MISDEMEANOR IS NOT NULL THEN

UPDATE FLUENTS SET ILLEGAL = '05-MAY-17 00.00.00.000000000'  WHERE ID_ACTION = :OLD.ID_EMPLOYEE ;

UPDATE FLUENTS SET SUSPENDED = '05-MAY-17 00.00.00.000000000'  WHERE ID_ACTION = :OLD.ID_EMPLOYEE;
UPDATE FLUENTS SET TAKE_PROMOTION = 0  WHERE ID_ACTION = :OLD.ID_EMPLOYEE ;
UPDATE FLUENTS SET TAKE_SALARY = 0  WHERE ID_ACTION = :OLD.ID_EMPLOYEE ;
UPDATE ACTION SET TAKE_PARDON = '' WHERE ID_EMPLOYEE= :OLD.ID_EMPLOYEE ;
UPDATE ACTION SET MISDEMEANOR = :NEW.MISDEMEANOR  WHERE ID_EMPLOYEE = :OLD.ID_EMPLOYEE ;

INSERT INTO HISTORY (ID_HISTORY, START_TIMESTAMP, END_TIMESTAMP, REASON)
SELECT :OLD.ID_EMPLOYEE, :NEW.MISDEMEANOR, FLUENTS.ILLEGAL , 'MISDEMEANOR' FROM FLUENTS WHERE ID_ACTION = :OLD.ID_EMPLOYEE  ;
END IF;
END;
```

Εικόνα 6 Misdemeanor Trigger

3. Ο GOOD_GRADE_TRIGGER ενεργοποιείται όταν ο διαχειριστής επιλέξει την ενέργεια επιβράβευσης ενός υπαλλήλου. Με την εκτέλεση της ενέργειας good grade αυτόματα μπαίνει σε λειτουργία ο trigger αυτός και το πεδίο take bonus που βρίσκεται στον πίνακα fluents γίνετε ίσο με 1, αυτό σημαίνει, ότι ο υπάλληλος πλέον παίρνει κάποιο επιπλέον εισόδημα. Το πεδίο bad employee γίνεται ίσο με 0, ενώ με τον τρόπο αυτό, εάν ο υπάλληλος για κάποιο λόγο "ήταν" bad employee παύει να είναι και τέλος ενημερώνεται ο πίνακας history για την ενέργεια αυτή.

```
  CREATE OR REPLACE EDITIONABLE TRIGGER "C##"."GOOD_GRADE_TRIGGER"
BEFORE UPDATE OF GOOD_GRADE ON "ACTION_TEST"
REFERENCING NEW AS NEW OLD AS OLD
FOR EACH ROW
BEGIN
UPDATE FLUENTS SET GOOD_EMPLOYEE = 1  WHERE ID_ACTION = :OLD.ID_EMPLOYEE ;
UPDATE FLUENTS SET TAKE_BONUS = 1  WHERE ID_ACTION = :OLD.ID_EMPLOYEE;
UPDATE FLUENTS SET TAKE_PROMOTION = 1  WHERE ID_ACTION = :OLD.ID_EMPLOYEE ;
UPDATE FLUENTS SET BAD_EMPLOYEE = 0 WHERE ID_ACTION = :OLD.ID_EMPLOYEE ;

UPDATE ACTION SET BAD_GRADE = '' WHERE ID_EMPLOYEE= :OLD.ID_EMPLOYEE  ;
UPDATE ACTION SET GOOD_GRADE = :NEW.GOOD_GRADE  WHERE ID_EMPLOYEE= :OLD.ID_EMPLOYEE  ;

INSERT INTO HISTORY (ID_HISTORY, START_TIMESTAMP, END_TIMESTAMP, REASON)
SELECT :OLD.ID_EMPLOYEE, :NEW.GOOD_GRADE, '' , 'GOOD_GRADE' FROM FLUENTS WHERE ID_ACTION = :OLD.ID_EMPLOYEE  ;
```

Εικόνα 7 Good Grade Trigger

4. Ο BAD_GRADE_TRIGGER τίθεται σε λειτουργία την στιγμή που γίνεται update στον πίνακα με τα actions. Εκείνη την στιγμή, ενεργοποιείται ο trigger αυτός και αρχικά θέτει την τιμή του πεδίου good_employee ίση με 0. Καταλαβαίνουμε λοιπόν, ότι δεν μπορεί ένας υπάλληλος να έχει συγχρόνως good_ employee και bad_employee τιμή ίση με 1. Εάν έπαιρνε κάποιο bonus

ο υπάλληλος σταματάει να έχει αυτό το προνόμιο και ενημερώνεται ο πίνακας history για την ανάλογη αλλαγή.

```
  CREATE OR REPLACE EDITIONABLE TRIGGER "C##"."BAD_GRADE_TRIGGER"
BEFORE UPDATE OF BAD_GRADE ON "ACTION_TEST"
REFERENCING NEW AS NEW OLD AS OLD
FOR EACH ROW
BEGIN
UPDATE FLUENTS SET GOOD_EMPLOYEE = 0  WHERE ID_ACTION = :OLD.ID_EMPLOYEE ;
UPDATE FLUENTS SET TAKE_BONUS = 0  WHERE ID_ACTION = :OLD.ID_EMPLOYEE;
UPDATE FLUENTS SET TAKE_PROMOTION = 0  WHERE ID_ACTION = :OLD.ID_EMPLOYEE ;
UPDATE FLUENTS SET BAD_EMPLOYEE = 1 WHERE ID_ACTION = :OLD.ID_EMPLOYEE ;

UPDATE ACTION SET GOOD_GRADE = '' WHERE ID_EMPLOYEE= :OLD.ID_EMPLOYEE  ;
UPDATE ACTION SET BAD_GRADE = :NEW.BAD_GRADE  WHERE ID_EMPLOYEE= :OLD.ID_EMPLOYEE  ;

INSERT INTO HISTORY (ID_HISTORY, START_TIMESTAMP, END_TIMESTAMP, REASON)
SELECT :OLD.ID_EMPLOYEE, :NEW.BAD_GRADE, '' , 'BAD_GRADE' FROM FLUENTS WHERE ID_ACTION = :OLD.ID_EMPLOYEE  ;
```

Εικόνα 8 Bad Grade Trigger

5. Τέλος ο TAKE_PARDON_TRIGGER ενεργοποιείται την στιγμή όπου ο διαχειριστής θα επιλέξει την αντίστοιχη ενέργεια για κάποιον υπάλληλο. Αυτόματα, ο υπάλληλος παύει να είναι illegal, suspended και το πεδίο TAKE_SALARY στον πίνακα με τα fluents γίνεται ίσο με 1, δηλαδή μπορεί ο υπάλληλος πλέον να παίρνει μισθό.

```
  CREATE OR REPLACE EDITIONABLE TRIGGER "C##"."TAKE_PARDON_TRIGGER"
BEFORE UPDATE OF TAKE_PARDON ON "ACTION_TEST"

REFERENCING NEW AS NEW OLD AS OLD

FOR EACH ROW
BEGIN
IF :NEW.TAKE_PARDON IS NOT NULL THEN
UPDATE FLUENTS SET ILLEGAL = ''  WHERE ID_ACTION = :OLD.ID_EMPLOYEE ;
UPDATE FLUENTS SET SUSPENDED = ''  WHERE ID_ACTION = :OLD.ID_EMPLOYEE;
UPDATE FLUENTS SET TAKE_SALARY = 1  WHERE ID_ACTION = :OLD.ID_EMPLOYEE ;

UPDATE ACTION SET MISDEMEANOR = '' WHERE ID_EMPLOYEE= :OLD.ID_EMPLOYEE ;
UPDATE ACTION SET TAKE_PARDON = :NEW.TAKE_PARDON  WHERE ID_EMPLOYEE= :OLD.ID_EMPLOYEE  ;

INSERT INTO HISTORY (ID_HISTORY, START_TIMESTAMP, END_TIMESTAMP, REASON)
SELECT :OLD.ID_EMPLOYEE, :NEW.TAKE_PARDON, '' , 'TAKE_PARDON' FROM FLUENTS WHERE ID_ACTION = :OLD.ID_EMPLOYEE  ;
END IF;
END;
```

Εικόνα 9 Take Pardon Trigger

Αφού υλοποιήθηκαν οι πίνακες και οι κατάλληλοι triggers για την ορθή λειτουργία της βάσης, πάντα με γνώμονα την τήρηση των περιορισμών ακεραιότητας, έπρεπε να δημιουργηθεί και κάποιο πρόγραμμα που να παρέχει στον χρήστη κάποια διεπαφή για

την εύκολη χρήση των λειτουργιών που έχουν πραγματοποιηθεί. Επιλέχθηκε, όπως αναφέρθηκε και στο κεφάλαιο 1, η γλώσσα προγραμματισμού Java. Επιλέχθηκε αυτή η γλώσσα και όχι κάποια άλλη όπως Php, JavaScript διότι σκοπός ήταν να δημιουργηθεί μια desktop εφαρμογή που θα λειτουργεί χωρίς δίκτυο και δεν θα χρειάζεται κάποιος φυλλομετρητής για την λειτουργία της. Για την ανάπτυξη του κώδικα χρησιμοποιήθηκε το προγραμματιστικό περιβάλλον NetBeans. Το τελικό project ονομάστηκε Ptyxiakh V_1. Το project αυτό, περιέχει ως source package την main κλάση του προγράμματος, μια κλάση με όνομα GUI όπου ευθύνεται για την διεπαφή που χειρίζεται ο χρήστης, δηλαδή το γραφικό περιβάλλον, την κλάση Button_Action και τέλος μια κλάση με όνομα SQL όπου λαμβάνει τον ρόλο επικοινωνίας του προγράμματος με την βάση δεδομένων. Όσον αφορά το γραφικό περιβάλλον έγινε επιλογή της βιβλιοθήκης Swing της java, καθώς αρχικός στόχος ήταν ένα απλό περιβάλλον με της απαραίτητες μόνο λειτουργίες. Δόθηκε περισσότερη έμφανση στην σχεδίαση και υλοποίησης της βάσης δεδομένων και παράλληλα της επικοινωνίας του συστήματος με αυτήν. Στην κλάση GUI λοιπόν βρήσκεται ο κώδικας που εκτελείται πρώτος μετά την main φυσικά. Αυτός ο κώδικας δίνει τρείς επιλογές στο χρήστη.

1. Να εισαχθεί καινούριος υπάλληλος στο σύστημα.
2. Να κάνει κάποια ενέργεια προς κάποιον υπάλληλο.
3. Να παρακολουθήσει το ιστορικό των εργαζομένων.

Ο κώδικας εμφανίζεται παρακάτω.

```java
package ptuxiaki.v1;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

import java.awt.*;

public class GUI {

    static JTextField[] textField = new JTextField[10];

    public void Intro() {

        JFrame IntroFrame = new JFrame("Intro");
        GridLayout experimentLayout = new GridLayout(0, 2);
        JPanel IntroPanel_Create = new JPanel();
        IntroPanel_Create.setLayout(experimentLayout);

        IntroFrame.add(IntroPanel_Create);
        JLabel CreateUserText = new JLabel("Add User: ");
        JButton CreateUserButton = new JButton("Submit");
        CreateUserButton.addActionListener(new Buttons_Action.Action_CreateUser());

        JLabel Update_Text = new JLabel("Add Suspend: ");
        JButton Update_Susp_Button = new JButton("Submit");
        Update_Susp_Button.addActionListener(new Buttons_Action.Action_UpdateSusp());

        JLabel History_Text = new JLabel("History: ");
        JButton History_Button = new JButton("Check");
        History_Button.addActionListener(new Buttons_Action.Action_History());

        //Add Panel
        IntroPanel_Create.add(CreateUserText);
        IntroPanel_Create.add(CreateUserButton);
        IntroPanel_Create.add(Update_Text);
        IntroPanel_Create.add(Update_Susp_Button);
        IntroPanel_Create.add(History_Text);
        IntroPanel_Create.add(History_Button);

        IntroFrame.setVisible(true);
        IntroFrame.setSize(200, 300);
        IntroFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        IntroFrame.setLocationRelativeTo(null);
        IntroFrame.pack();

    }

}
```

Εικόνα 10 Java κώδικας GUI

Ανάλογα την επιλογή του χρήστη, καλείται και μια κλάση από την κλάση Buttons_Action. Σε περίπτωση που ο χρήστης επιλέξει να εισαχθεί καινούριος υπάλληλος καλείται η κλάση CreateUser(). Εάν επιλέξει να κάνει κάποια ενέργεια προς κάποιον υπάλληλο καλείται η κλάση UpdateSusp(), τέλος εάν επιλέξει να δει το ιστορικό καλείται η Action_History(). Ο κώδικας για αυτές τις λειτουργίες επισημαίνεται παρακάτω.

- Action_CreateUser

```
static class Action_CreateUser implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {

        JFrame Add_User_Frame = new JFrame("Add User");
        JPanel Add_User_Panel = new JPanel(new SpringLayout());

        String[] labels = {"Name: ", "SurName: ", "Address: ", "Tel: ", "E-mail: "};
        int numPairs = labels.length;

        for (int i = 0; i < numPairs; i++) {
            JLabel l = new JLabel(labels[i], JLabel.TRAILING);
            Add_User_Panel.add(l);
            textField[i] = new JTextField(10);
            l.setLabelFor(textField[i]);
            Add_User_Panel.add(textField[i]);
        }
        SpringUtilities.makeCompactGrid(Add_User_Panel,
                numPairs, 2, //rows, cols
                6, 6, //initX, initY
                6, 6);//xPad, yPad*/

        JPanel p = new JPanel();
        Add_User_Frame.add(p, BorderLayout.PAGE_END);
        JButton Create_User_Submit = new JButton("Submit");
        p.add(Create_User_Submit);
        Create_User_Submit.addActionListener(new AddUser());

        Add_User_Frame.setVisible(true);
        Add_User_Frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        Add_User_Frame.add(Add_User_Panel);
        Add_User_Frame.setLocationRelativeTo(null);
        Add_User_Frame.pack();
    }

}
```

Εικόνα 11 Java κωδικας Create User

- Action_UpdateSusp

```
static class Action_UpdateSusp implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
        JFrame Add_Susp_Frame = new JFrame("Update Suspend");
        JPanel Add_Susp_Panel = new JPanel(new SpringLayout());

        JComboBox jc = new JComboBox();
        Connection con = null;
        Statement st = null;
        ResultSet rs = null;

        try {
            con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl", "c##", "3563");
            st = con.createStatement();
            String s = "SELECT ID FROM EMPLOYEE";
            rs = st.executeQuery(s);
            while (rs.next()) {
                jc.addItem(rs.getString(1));
            }

        } catch (SQLException ex) {
            Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null, ex);
        } finally {
            try {
                st.close();
                rs.close();
                con.close();
            } catch (SQLException ex) {
                Logger.getLogger(GUI.class.getName()).log(Level.SEVERE, null, ex);
            }

        }

        jc.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent event) {
                if (event.getStateChange() == ItemEvent.SELECTED) {
                    var = event.getItem().toString();
                    System.out.println(var);
                }
            }
        });

        String[] labels = {"Misdemeanor: ", "Take Pardon: ", "Good Grade: ", "Bad Grade: "};
        int numPairs = labels.length;
        for (int i = 0; i < numPairs; i++) {
            JLabel l = new JLabel(labels[i], JLabel.TRAILING);
            Add_Susp_Panel.add(l);
            textField[i] = new JTextField(10);
            l.setLabelFor(textField[i]);
            Add_Susp_Panel.add(textField[i]);

        }
```

Εικόνα 12 Java κώδικας Update Suspend

```
        }
        SpringUtilities.makeCompactGrid(Add_Susp_Panel,
                numPairs, 2, //rows, cols
                6, 6, //initX, initY
                6, 6);//xPad, yPad

        JPanel p = new JPanel();
        JPanel panel_jc = new JPanel();
        Add_Susp_Frame.add(p, BorderLayout.PAGE_END);
        Add_Susp_Frame.add(panel_jc, BorderLayout.PAGE_START);
        JButton Update_Susp_Submit = new JButton("Submit");
        p.add(Update_Susp_Submit);
        Update_Susp_Submit.addActionListener(new UpdateSusp());

        JButton allUsers = new JButton("Available Users");
        panel_jc.add(allUsers);
        allUsers.addActionListener(new AllUsersTable());

        JLabel ID = new JLabel("ID: ");
        panel_jc.add(ID);
        panel_jc.add(jc);

        Add_Susp_Frame.setVisible(true);
        Add_Susp_Frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        Add_Susp_Frame.add(Add_Susp_Panel);
        Add_Susp_Frame.setLocationRelativeTo(null);
        Add_Susp_Frame.pack();
    }
}
static class UpdateSusp implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
        SQL Insert_Susp = new SQL();

        try {

            Insert_Susp.Poiniupdate(var, textField[0].getText(), textField[1].getText(), textField[2].getText(), textField[3].getText());
        } catch (SQLException ex) {
            Logger.getLogger(Buttons_Action.class.getName()).log(Level.SEVERE, null, ex);
        } catch (ParseException ex) {
            Logger.getLogger(Buttons_Action.class.getName()).log(Level.SEVERE, null, ex);
        }

    }
}
```

Εικόνα 13 Java κώδικας Update Suspend

- Action_History

```
static class Action_History implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {

        String url = "jdbc:oracle:thin:@localhost:1521:orcl";
        String username = "C##";
        String password = "3563";
        Statement stmt = null;

        Connection connection = null;
        try {

            connection = DriverManager.getConnection(url, username, password);
            stmt = connection.createStatement();
            String sql;
            sql = "SELECT * FROM HISTORY  ORDER BY START_TIMESTAMP ASC";

            ResultSet rs = stmt.executeQuery(sql);
            JTable table = new JTable(buildTableModel(rs));

            rs.close();
            stmt.close();
            connection.close();
            JOptionPane.showMessageDialog(null, new JScrollPane(table));

        } catch (SQLException ex) {
            Logger.getLogger(Buttons_Action.class.getName()).log(Level.SEVERE, null, ex);
        }
        if (connection != null) {
        } else {
            System.out.println("Failed to make connection!");
        }

    }

}
```

Εικόνα 14 Java κώδικας Action History

```
public static DefaultTableModel buildTableModel(ResultSet rs)
        throws SQLException {

    ResultSetMetaData metaData = rs.getMetaData();

    // names of columns
    Vector<String> columnNames = new Vector<String>();
    int columnCount = metaData.getColumnCount();
    for (int column = 1; column <= columnCount; column++) {
        columnNames.add(metaData.getColumnName(column));
    }

    // data of the table
    Vector<Vector<Object>> data = new Vector<Vector<Object>>();
    while (rs.next()) {
        Vector<Object> vector = new Vector<Object>();
        for (int columnIndex = 1; columnIndex <= columnCount; columnIndex++) {
            vector.add(rs.getObject(columnIndex));
        }
        data.add(vector);
    }

    return new DefaultTableModel(data, columnNames);
}
```

Εικόνα 15 Πίνακας Εγγραφών

Η κλάση SQL καλείται κάθε φορά που χρειάζεται να επικοινωνήσει η εφαρμογή με την βάσει δεδομένων. Καλείται σε περίπτωση εκχώρησης στο σύστημα κάποιου νέου υπάλληλου και σε περίπτωση καταχώρισης κάποιας ενέργειας από το διαχειριστή προς κάποιο υπάλληλο. Ο κώδικας παρατίθεται παρακάτω.

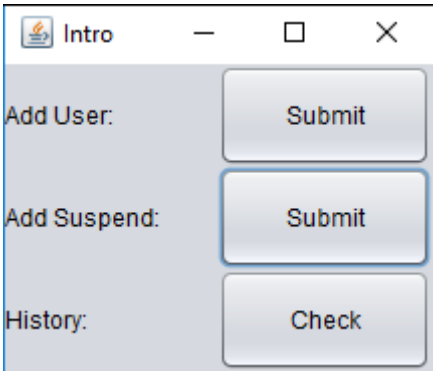
```java
package ptuxiaki.v1;

import java.sql.*;
import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import javax.swing.JOptionPane;

public class SQL {

    public void AddUser(String x, String y) {
        try {
            String url = "jdbc:oracle:thin:@localhost:1521:orcl";
            Connection conn = DriverManager.getConnection(url, "C##", "3563");
            Statement st = conn.createStatement();
            st.executeUpdate("INSERT INTO EMPLOYEE VALUES ('','" + x + "','" + y + "')");

            conn.close();
        } catch (Exception e) {
            System.err.println("Got an exception! ");
            System.err.println(e.getMessage());
        }

    }

    public void Poiniupdate(String id, String misdemeanor, String takePardon, String goodGrade, String badGrade) throws SQLException, ParseException {

        try {
            String url = "jdbc:oracle:thin:@localhost:1521:orcl";
            Connection conn = DriverManager.getConnection(url, "C##", "3563");
            Statement st = conn.createStatement();
            java.util.Date date;
            java.util.Date date1;
            date1 = new java.util.Date();
            DateFormat format = new SimpleDateFormat("dd/MM/yyyy");
            if (!"".equals(misdemeanor)) {
                String x = misdemeanor;

                date = format.parse(x);



            } else if (!"".equals(takePardon)) {
                String x = takePardon;

                date = format.parse(x);

            } else if (!"".equals(goodGrade)) {
                String x = goodGrade;

                date = format.parse(x);

            }
              else {
```

Εικόνα 16 Java κώδικας update στους πίνακες

```java
                String x = badGrade;

                date = format.parse(x);



            }

            if (date.after(date1) || date.equals(date1)) {
                if (takePardon.equals("") && goodGrade.equals("") && badGrade.equals("")) {
                    st.executeUpdate("UPDATE ACTION_TEST SET MISDEMEANOR = TO_TIMESTAMP('" + misdemeanor + "','DD-MM-YYYY HH24:MI:SS.FF') WHERE ID_EMPLOYEE= " + id + " ");
                } else if (misdemeanor.equals("") && goodGrade.equals("") && badGrade.equals("")) {

                    st.executeUpdate("UPDATE ACTION_TEST SET TAKE_PARDON = TO_TIMESTAMP('" + takePardon + "','DD-MM-YYYY HH24:MI:SS.FF') WHERE ID_EMPLOYEE= " + id + " ");

                } else if (misdemeanor.equals("") && takePardon.equals("") && badGrade.equals("")) {

                    st.executeUpdate("UPDATE ACTION_TEST SET GOOD_GRADE = TO_TIMESTAMP('" + goodGrade + "','DD-MM-YYYY HH24:MI:SS.FF') WHERE ID_EMPLOYEE= " + id + " ");

                } else if (misdemeanor.equals("") && takePardon.equals("") && goodGrade.equals("")) {

                    st.executeUpdate("UPDATE ACTION_TEST SET BAD_GRADE = TO_TIMESTAMP('" + badGrade + "','DD-MM-YYYY HH24:MI:SS.FF') WHERE ID_EMPLOYEE= " + id + " ");

                }

            } else {
                JOptionPane.showMessageDialog(null, "TRY A VALID VALUE");
                System.out.println(date);
                System.out.println(date1);
            }
            conn.close();
        } catch (Exception e) {
            System.out.println(misdemeanor);
            System.out.println(goodGrade);
            System.out.println(badGrade);
            System.err.println("Got an exception! ");
            System.err.println(e.getMessage());
        }

    }

    void Poiniupdate(String text, String text0, String text1, String text2, String text3, String text4) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }

    void Poini(String text, String text0, String text1) {
        throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.
    }
}
```

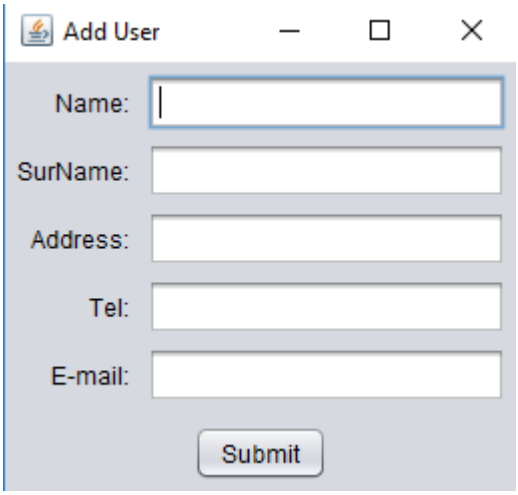
Εικόνα 17 Java κώδικας update στους πίνακες

Από την μεριά του χρήστη, εκτελώντας την εφαρμογή εμφανίζεται ένα frame για να επιλέξει μια από τις τρεις επιλογές που του δίδονται.



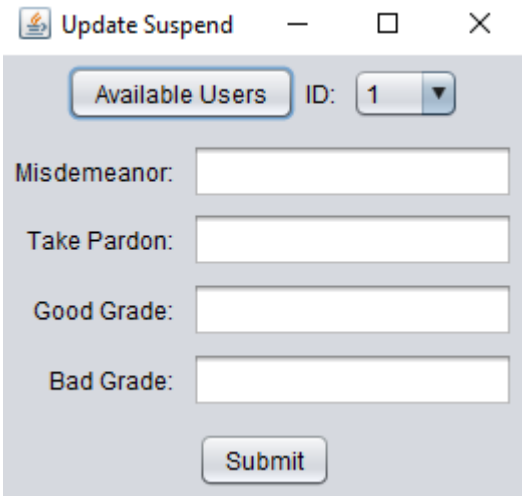
Εικόνα 18 Main Frame

Στην πρώτη επιλογή, δηλαδή στην επιλογή Add User ,εμφανίζεται το frame για την εγγραφή νέου χρήστη. Ο διαχειριστής συμπληρώνει μια φόρμα με τα δημογραφικά στοιχεία του υπαλλήλου και αποκτάει ένα μοναδικό id ως υπάλληλος.



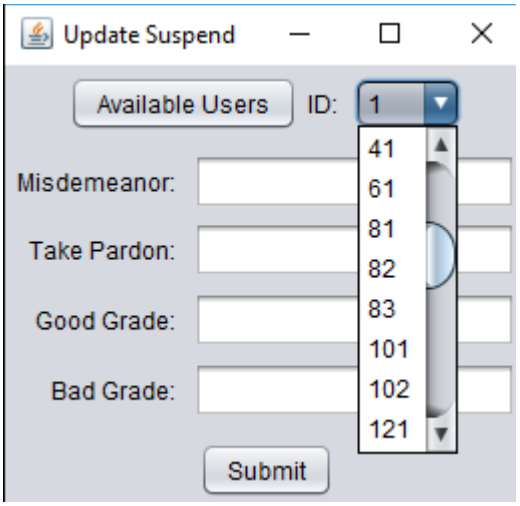
Εικόνα 19 Add User

Εάν επιλέξει την επιλογή Add Suspend, εμφανίζεται frame με τρείς δυνατότητες. Ο χρήστης έχει την δυνατότητα να δει τους διαθέσιμους υπαλλήλους, πράγμα που είναι εφικτό μέσω drop down menu, επιλέγοντας ανάμεσα σε όλα τα διαθέσιμα id κάποιον υπάλληλο ενώ τέλος, ο χρήστης είναι δυνατό να ορίσει ημερομηνία για την ενέργεια που θα προβεί προς κάποιο υπάλληλο.



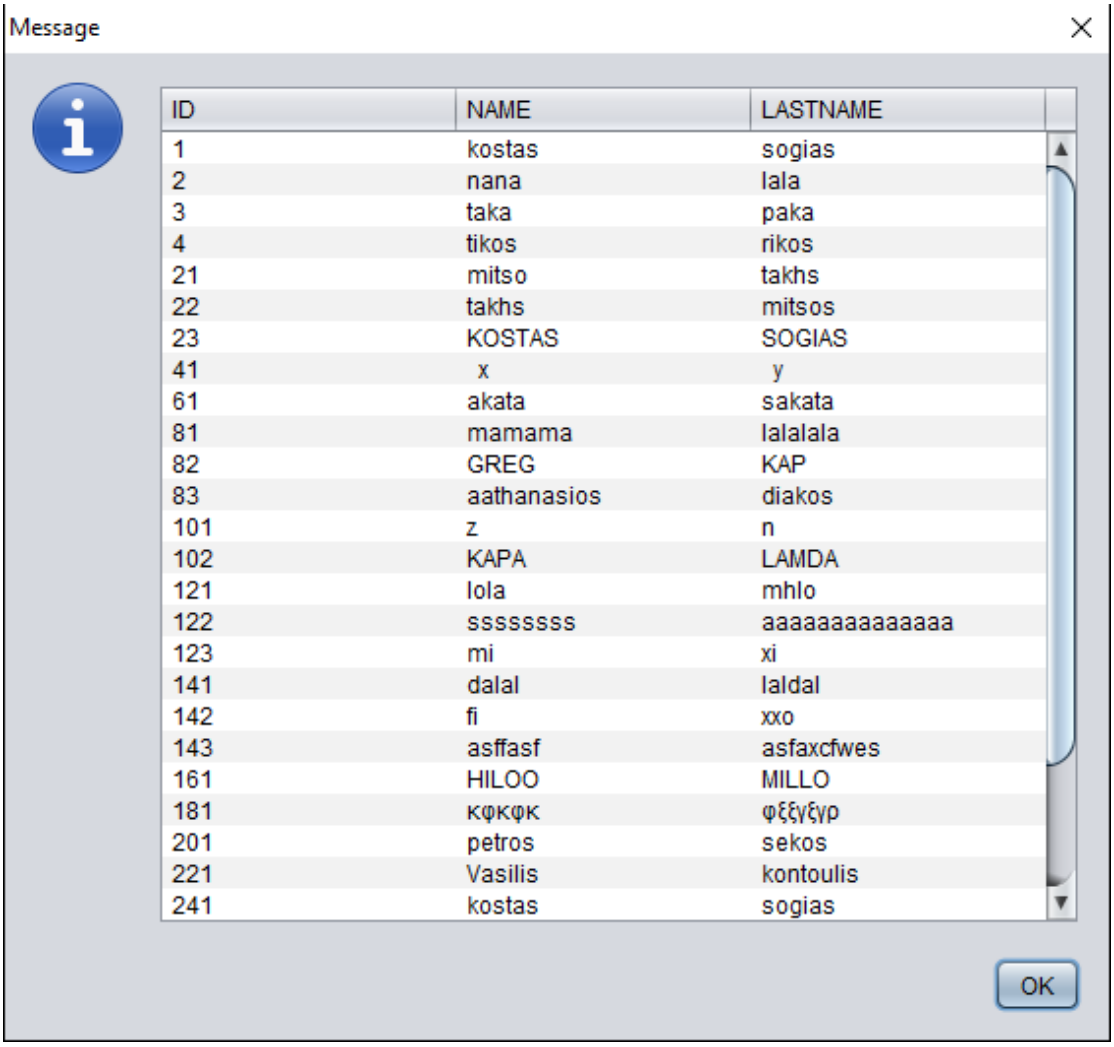
Εικόνα 20 Update Suspend

Στο drop down menu εμφανίζονται τα id με τον παρακάτω τρόπο.



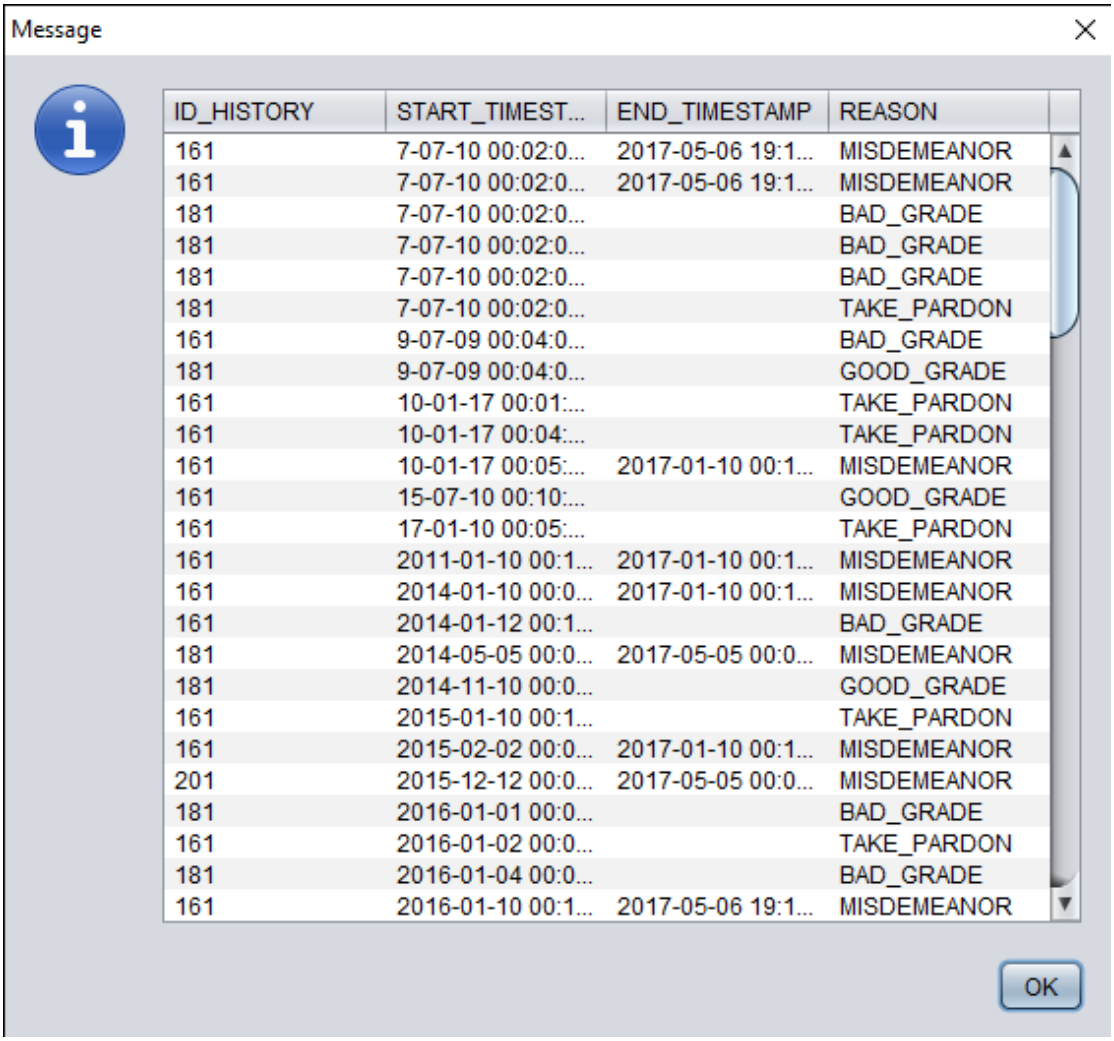
Εικόνα 21 Update Suspend Drop down menu

Εάν πατηθεί το κουμπί Availavle Users, εμφανίζεται ένα παράθυρο το οποίο περιέχει όλους τους υπαλλήλους ταξινομημένους κατά αύξοντα αριθμο με κριτήριο το id τους,δηλαδή από τον νεότερο υπάλληλο προς τον παλαιότερο.



Εικόνα 22 Πίνακας Διαθέσιμων Χρηστών

Στην περίπτωση τις τρίτης επιλογής, δηλαδή της επιλογής History εμφανίζεται όλο το ιστορικό των ενεργειών προς τους υπαλλήλους με χαρακτηριστικό το id τους, την αρχή του γεγονότος, την λήξη του αλλά και την αιτία.



Εικόνα 23 Πίνακας History

5. Συμπεράσματα

Στην προσπάθεια υλοποίησης αλλά και σχεδίασης αυτής της πτυχιακής εργασίας, επιτεύχθηκε η καλύτερη κατανόηση μιας βάσεις πιο σύνθετης. Δηλαδή μια βάση οπού η λειτουργία της δεν είναι μονάχα η αποθήκευση χρήσιμων δεδομένων αλλά και η αλληλεπίδραση τον πινάκων μεταξύ τους. Αντιμετωπίστηκε δυσκολία στην τήρηση των περιορισμών ακεραιότητας που είχαν τεθεί εξ αρχής. Αυτό οδήγησε στην κατανόηση καινούριων τεχνολογιών αλλά και την έρευνα σε τεχνολογίες που ήδη ήταν γνωστές για την ομάδα υλοποίησης. Οι χωροχρονικές βάσεις δεδομένων έχουν πληθώρα εφαρμογών, μια από τις σημαντικές εφαρμογές που συναντάτε αλλά και θα συναντηθεί ακόμα περισσότερο στο μέλλον είναι στα μοντέλα τεχνίτης νοημοσύνης. Η τεχνητή νοημοσύνη είναι ένας κλάδος της επιστήμης ή μάλλον προηγμένης επιστήμης που ασχολείται με το πώς μπορεί να εφαρμοστεί η νοημοσύνη. Αλλά μετά την εφαρμογή της επόμενης σημαντικής πτυχής που πρέπει να αντιμετωπιστεί είναι το πώς θα αποθηκευτούν τα δεδομένα που θα αποθηκευτούν, για αυτό χρειαζόμαστε βάσεις δεδομένων. Η βάση δεδομένων είναι βασικά μια ομάδα δεδομένων που αποθηκεύει τα δεδομένα, τόσο σε διαδοχική όσο και σε μη διαδοχική μορφή. Τα δεδομένα είναι αναπόσπαστο κομμάτι της τεχνητής νοημοσύνης. Μία από τις προκλήσεις με τα μοντέλα κατάρτισης και τα μοντέλα βαθιάς μάθησης είναι ο τεράστιος όγκος δεδομένων και η ισχύς επεξεργασίας που χρειάζεστε για να εκπαιδεύσετε ένα νευρωνικό δίκτυο, για παράδειγμα, σε περίπλοκη αναγνώριση προτύπων σε τομείς όπως η ταξινόμηση εικόνων ή η επεξεργασία φυσικής γλώσσας . Ως εκ τούτου, οι βάσεις δεδομένων τεχνητής νοημοσύνης αρχίζουν να αναδύονται στην αγορά ως ένας τρόπος βελτιστοποίησης της διαδικασίας εκμάθησης και κατάρτισης της τεχνητής νοημοσύνης για τις επιχειρήσεις. αναμένονται στο μέλλον ακόμα περισσότερες δημοσιεύσεις γύρο από το συγκεκριμένο τομέα καθώς το μοντέλο αυτό μπορεί να συνεισφέρει πολλά στης τεχνολογίες του μέλλοντος.

5.1 Συμπεράσματα

Με το πέρας της πτυχιακής εργασίας κατανοήθηκε εις βάθος η σχεδιάσει μιας βάσεις δεδομένων οπού δεν έχει ως μοναδικό σκοπό την αποθήκευση δεδομένων, αλλά και την τηρήσει κάποιον περιορισμών ακεραιότητας. Επίσης κατά την υλοποίηση συναντήθηκαν προβλήματα που δεν είχαν λυθεί σε παλαιότερες εργασίες ή project από την ομάδα αυτής της πτυχιακής εργασίας, με αποτέλεσμα να ασχοληθεί με θέματα που δεν είχε ασχοληθεί στο παρελθόν.

Βιβλιογραφία

[1] J. McCarthy and P. J. Hayes, “Some philosophical problems from the standpoint of artificial intelligence,” *Mach. Intell.*, vol. 4, no. 463–502, pp. 463–502, 1969.

[2] V. Lifschitz, “Frames in the space of situations,” *Artif. Intell.*, vol. 46, no. 3, pp. 365–376, 1990.

[3] M. Thielscher, “Ramification and causality,” *Artif. Intell.*, vol. 89, no. 1–2, pp. 317–364, 1997.

[4] M. Denecker and E. Ternovska, “Inductive situation calculus,” *Artif. Intell.*, vol. 171, no. 5–6, pp. 332–360, 2007.

[5] R. E. Fikes and N. J. Nilsson, “Strips: A new approach to the application of theorem proving to problem solving,” *Artif. Intell.*, vol. 2, no. 3–4, pp. 189–208, 1971.

[6] N. McCain and H. Turner, “A Causal Theory of Ramifications and Qualifications,” *Proc. Fourteenth Int. Jt. Conf. Artif. Intell.*, pp. 1978–1984, 1995.

[7] A. Leake and A. Hughes, “database (DB).” [Online]. Available: https://searchsqlserver.techtarget.com/definition/database.

[8] Π. Μαστρογαλια, “ΤΙ ΣΗΜΑΙΝΕΙ ΒΑΣΗ ΔΕΔΟΜΕΝΩΝ;,” 2004.

[9] T. E. of E. Britannica, “Database computer science,” 2018. [Online]. Available: https://www.britannica.com/technology/database.

[10] “A Timeline of Database History.” [Online]. Available: https://www.quickbase.com/articles/timeline-of-database-history.

[11] D. R. Brackenridge, “United States Patent,” vol. 2, no. 12, 1992.

[12] Craig S. Mullins and S. Christiansen, “database management system (DBMS).” [Online]. Available: https://searchsqlserver.techtarget.com/definition/database-management-system.

[13] Essays, “The Evolution Of Database Management System,” *23rd March, 2015*, 2015. [Online]. Available: https://www.ukessays.com/essays/information-technology/the-evolution-of-database-management-system-information-technology-essay.php.

[14] M. Chapple, “What Is a Database Management System (DBMS)?,” *July 03, 2018*, 2018. [Online]. Available: https://www.lifewire.com/database-management-system-1019609.

[15] Θ. Τζουραμάνης, “Μέθοδοι προσπέλασης και επεξεργασίας ερωτήσεων σε χρονικές και χωροχρονικές βάσεις δεδομένων,” 2002. [Online]. Available:

http://thesis.ekt.gr/thesisBookReader/id/20183#page/1/mode/2up.

[16] S. Farooq, “SPATIAL DATABASES Concept, Design and Management.” [Online]. Available: http://www.geol-amu.org/notes/m14b-4-4.htm.

[17] Techopedia, “Spatial Database.” [Online]. Available: https://www.techopedia.com/definition/17287/spatial-database.

[18] A. Neumann, *Encyclopedia of GIS*. 2017.

[19] J. Biscobing, “relational database.” [Online]. Available: https://searchdatamanagement.techtarget.com/definition/relational-database.

[20] C. Brooks, “What is SQL?,” *January 21, 2014*, 2014. [Online]. Available: https://www.businessnewsdaily.com/5804-what-is-sql.html.