Τεχνολογία Βάσεων Δεδομένων SirenBase

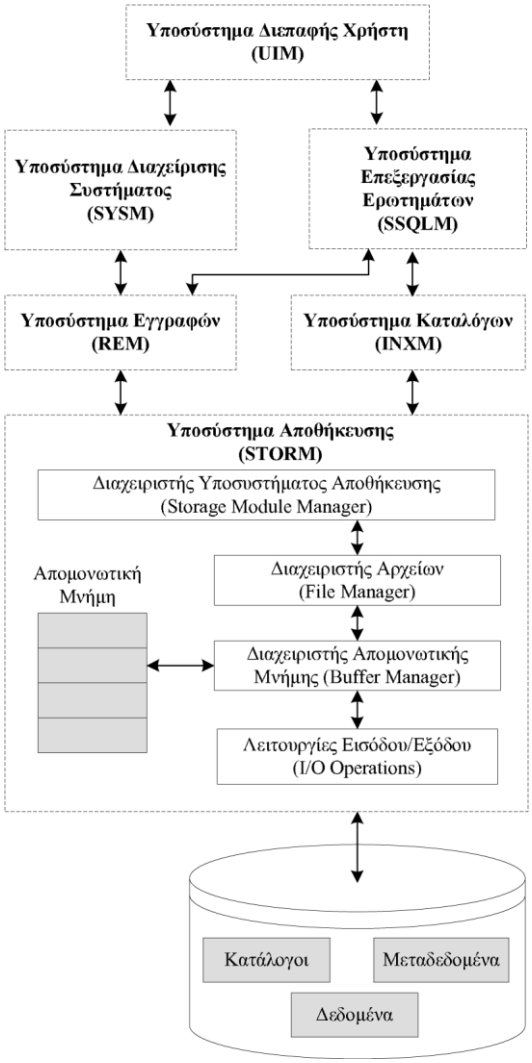
Έγγραφο τεκμηρίωσης του κώδικα

Γιαννουλούδης Στέργιος 1877
Γκανιάς Ευριπίδης 1866
Ντερελής Θανάσης 1687

Περιεχόμενα

Εισαγωγή

Στο παρακάτω σχήμα φαίνεται το διάγραμμα κλάσεων του προγράμματος. Η ομάδα δεν ασχολήθηκε με το Υποσύστημα Αποθήκευσης, γι αυτό δεν θα αναλυθεί. Στη συνέχεια θα αναλυθούν τα υπόλοιπα τμήματα του προγράμματος, ξεκινώντας από τα υποσυστήματα χαμηλότερου επιπέδου.



Υποσύστημα Εγγραφών (REM)

Το Υποσύστημα Εγγραφών είναι υπεύθυνο για τη διαχείριση των αρχείων σε επίπεδο εγγραφών. Αυτό πραγματοποιείται με τη βοήθεια των παρακάτω κλάσεων οι οποίες χρησιμοποιούν το τμήμα STORM για να πραγματοποιήσουν τις απαραίτητες λειτουργίες.

Διαχειριστής Αρχείων Εγγραφών (REM_RecordFileManager)

Η Κλάση είναι υπεύθυνη για τη διαχείριση των αρχείων εγγραφών, δηλαδή τη δημιουργία, καταστροφή, άνοιγμα και κλείσιμο αυτών.

Περιέχει μια private μεταβλητή για την αποθήκευση ενός δείκτη του Storage Manager. Οι συναρτήσεις της κλάσης περιγράφονται παρακάτω.

Δημιουργία Αρχείου Εγγραφών (CreateRecordFile)

Δίνουμε αρχικές τιμές σε μία δομή rfi τύπου RecordFileInfo (Δομή που αποθηκεύει πληροφορία για το αρχείο).

Δημιουργούμε και ανοίγουμε ένα καινούργιο αρχείο με τη βοήθεια του StorageManager και παίρνουμε ένα χειριστή fh για το αρχείο. Από τον fh δεσμεύουμε μία νέα σελίδα καλώντας τη συνάρτηση ReservePage, η οποία μας δίνει έναν χειριστή σελίδας ph. Μέσω του χειριστή σελίδας, καλώντας τις κατάλληλες συναρτήσεις παίρνουμε το pageID και ένα δείκτη στα δεδομένα που είναι αποθηκευμένα στη σελίδα.

Αποθηκεύουμε το rfi στη σελίδα αυτή, σημειώνουμε τη σελίδα ως “dirty” και την γράφουμε στον δίσκο. Τέλος την κάνουμε “unpin” ώστε να φύγει από την κύρια μνήμη αν χρειαστεί και κλείνουμε το αρχείο.

Καταστροφή Αρχείου Εγγραφών (DestroyRecordFile)

Καλούμε την αντίστοιχη συνάρτηση από τον Storage Manager.

Άνοιγμα Αρχείου Εγγραφών (OpenRecordFile)

Αν υπάρχει ανοιχτός χειριστής αρχείου τον κλείνουμε. Ανοίγουμε το αρχείο με τη βοήθεια του Storage Manager και παίρνουμε τον χειριστή αρχείου. Από τον τελευταίο καλούμε την GetFirstPage μέσω της οποίας παίρνουμε ένα χειριστή σελίδας ph (η πρώτη σελίδα περιέχει πληροφορία για το αρχείο) που θα μας δώσει τα δεδομένα, τα οποία θα αποθηκεύσουμε σε μορφή RecordFileInfo. Τέλος, μετατρέπουμε την τιμή της isOpen σε true δηλώνοντας ότι το αρχείο εγγραφών είναι ανοιχτό.

Κλείσιμο Αρχείου Εγγραφών (CloseRecordFile)

Αν το αρχείο δεν είναι ανοιχτό επιστρέφουμε σφάλμα. Παίρνουμε ένα χειριστή σελίδας για την πρώτη σελίδα του αρχείου. Από αυτόν παίρνουμε ένα δείκτη στα δεδομένα μέσα στα οποία αντιγράφουμε τις πληροφορίες του αρχείου. Έπειτα, σημειώνουμε τη σελίδα ως “dirty” και την γράφουμε στον δίσκο. Τέλος την κάνουμε “unpin” ώστε να φύγει από την κύρια μνήμη αν χρειαστεί. Μέσω του STOPM καλούμε την CloseFile για τον χειριστή αρχείου που έχουμε στο private και μεταβάλουμε την τιμή της isOpen σε false.

Χειριστής Αρχείων Εγγραφών (REM_RecordFileHandle)

Η κλάση είναι υπεύθυνη για το χειρισμό των αρχείων εγγραφών, δηλαδή την ανάγνωση, την εισαγωγή, τη διαγραφή και την ενημέρωση των εγγραφών μέσα στο αρχείο. Για την καλύτερη μεταχείριση των πληροφοριών για τις σελίδες και τα αρχεία έχουν δημιουργηθεί δύο δομές:

- **RecordFileInfo**
  Κρατάει πληροφορίες για το αρχείο εγγραφών (πλήθος των records στο αρχείο, μέγεθος του κάθε record, πλήθος των εγγραφών σε κάθε σελίδα)

- **RecordPageInfo**
  Κρατάει πληροφορίες για τη σελίδα εγγραφών (πλήθος των εγγραφών, πλήθος δεσμευμένων εγγραφών, πίνακας που δείχνει την κατάσταση της κάθε θέσης - ελεύθερη ή όχι) .

  Σημείωση: Το μέγεθος του πίνακα sStatus ορίζεται από προεπιλογή στις 1200 θέσεις, πράγμα που σημαίνει ότι η σελίδα θα χωράει μέχρι και 1200 εγγραφές.

Επίσης έχει δημιουργηθεί η παρακάτω private συνάρτηση για τη διευκόλυνση του εντοπισμού και της διόρθωσης λαθών στον κώδικα σε σχέση με τις παραπάνω δομές.

- **Δημιουργία Νέας Δομής Πληροφοριών Σελίδας (InitRecordPageInfo)**

  Δημιουργεί, αρχικοποιεί μία δομή RecordPageInfo και την επιστρέφει.

Η κλάση περιέχει τέσσερα πεδία: ένα που δείχνει αν το αρχείο είναι ανοιχτό, ένα δείκτη σε χειριστή αρχείων του STORM, ένα RecordfileInfo που περιέχει πληροφορίες για το (ανοιχτό) αρχείο και τον κωδικό της σελίδας που βρίσκονται οι πληροφορίες του αρχείου. Τέλος, οι παρακάτω συναρτήσεις υλοποιούν τις απαραίτητες λειτουργίες της κλάσης.

Διάβασμα Εγγραφής (ReadRecord)

Αν το αρχείο δεν είναι ανοιχτό επιστρέφεται σφάλμα. Μέσω του rid που παίρνουμε ως όρισμα παίρνουμε τις τιμές των pageID και slot της εγγραφής που θέλουμε να διαβάσουμε. Μέσω του χειριστή αρχείου του STORM που έχουμε, παίρνουμε την πληροφορία της σελίδας που είναι αποθηκευμένη στην πρώτη εγγραφή της σελίδας.

Στη συνέχεια, ελέγχουμε αν υπάρχει εγγραφή στο slot το οποίο μας ενδιαφέρει. Αν υπάρχει, θέτουμε στο χειριστή εγγραφών του ορίσματος τον κωδικό της εγγραφής και δεσμεύουμε χώρο για τα δεδομένα (την πληροφορία για το μέγεθος της εγγραφής του αρχείου την πήραμε από την πληροφορία του αρχείου «fInfo»). Σε διαφορετική περίπτωση επιστρέφει σφάλμα που ενημερώνει τον χρήστη ότι η εγγραφή που ζήτησε δεν υπάρχει.

Στο χώρο που μόλις δεσμεύσαμε, αντιγράφουμε τα δεδομένα που θέλουμε από την κατάλληλη θέση των δεδομένων της σελίδας.

Εισαγωγή Εγγραφής (InsertRecord)

Αν το αρχείο δεν είναι ανοιχτό επιστρέφεται σφάλμα. Η μεταβλητή currentPageID κρατάει πληροφορία για τη σελίδα που βρισκόμαστε την τρέχουσα στιγμή. Αρχικοποιούμε αυτή τη μεταβλητή με τον κωδικό σελίδας της πρώτης σελίδας του αρχείου.

Σαρώνουμε όλες τις σελίδες του αρχείου και παίρνουμε την πληροφορία τους. Για την κάθε σελίδα, ελέγχουμε αν υπάρχει ελεύθερο slot για αποθήκευση. Αν βρεθεί στις ήδη υπάρχουσες σελίδες, μέσω του πίνακα που μας δείχνει ποιες σελίδες είναι ελεύθερες και ποιες όχι, βρίσκουμε το άδειο slot της σελίδας και το δεσμεύουμε για τα νέα δεδομένα. Ενημερώνουμε το RecordID που μας έδωσε ο χρήστης, καθώς και την πληροφορία της σελίδας, σύμφωνα με τις αλλαγές που πρόκειται να γίνουν. Σε κάθε σελίδα που ψάχνουμε, αν η σελίδα είναι γεμάτη, την κάνουμε unpin ώστε - αν χρειαστεί - να φύγει από την κύρια μνήμη, αφού δεν θα τη χρειαστούμε στη συνέχεια.

Εάν από την σάρωση των σελίδων που προηγήθηκε δεν έχουμε καταφέρει να βρούμε ελεύθερο χώρο σε κάποια σελίδα, πρέπει να δεσμεύσουμε μία νέα σελίδα. Αυτό θα γίνει μέσω του χειριστή αχρείων, ο οποίος θα δεσμεύσει μια σελίδα και θα με δώσει τον χειριστή της. Αρχικοποιούμε την πληροφορία της νέας σελίδας με τις κατάλληλες τιμές, μέσω του ph παίρνουμε ένα δείκτη στα δεδομένα της νέας σελίδας. Ενημερώνουμε το RecordID που μας έδωσε ο χρήστης, καθώς και την πληροφορία που μόλις ορίσαμε, σύμφωνα με τις αλλαγές που πρόκειται να γίνουν και σημειώνουμε τη νέα σελίδα ως dirty, ώστε οι αλλαγές να γραφούν στο δίσκο.

Τέλος, παίρνουμε το χειριστή της σελίδας από τον χειριστή αρχείου, δίνοντας τον κωδικό της σελίδας. Από αυτόν, παίρνουμε ένα δείκτη στα δεδομένα της σελίδας μέσα στα οποία γράφουμε, στην κατάλληλη θέση, τα δεδομένα της νέας εγγραφής. Σημειώνουμε τη σελίδα ως dirty και την γράφουμε στον δίσκο.

Διαγραφή Εγγραφής (DeleteRecord)

Από το όρισμα RecordID που μας δίνεται, παίρνουμε την πληροφορία της σελίδας που βρίσκεται η εγγραφή και τροποποιούμε τα δεδομένα. Πιο συγκεκριμένα, μειώνουμε τον αριθμό των δεσμευμένων εγγραφών της σελίδας και σημειώνουμε το slot της εγγραφής που θέλουμε να διαγράψουμε ως ελεύθερο. Τέλος, ενημερώνουμε τα δεδομένα της σελίδας αντιγράφοντας μέσα σε αυτά την παραπάνω δομή.

Ενημέρωση Εγγραφής (UpdateRecord)

Από το όρισμα της συνάρτησης παίρνουμε το RecordID της σελίδας και από αυτό, τα pageID και slot. Μέσω του χειριστή αρχείων παίρνουμε ένα χειριστή της σελίδας στην οποία βρίσκονται τα παλιά δεδομένα του ορίσματος. Αντικαθιστούμε τα παλιά δεδομένα της εγγραφής με τα νέα, σημειώνουμε τη σελίδα ως dirty και την γράφουμε στον δίσκο. Τέλος, την κάνουμε unpin ώστε να φύγει από την κύρια μνήμη αν χρειαστεί.

Εγγραφή Σελίδων στο Δίσκο (FlushPages)

Καλούμε, μέσω του χειριστή αρχείων, την αντίστοιχη συνάρτηση του τμήματος STORM.

Σάρωση Αρχείων Εγγραφών (REM_RecordFileScan)

Η κλάση είναι υπεύθυνη για τη σάρωση των εγγραφών, δηλαδή το άνοιγμα μια σάρωσης, το κλείσιμο αυτής και την επιστροφή της επόμενης εγγραφής που πληρεί κάποιες προϋποθέσεις. Ανοίγοντας μια σάρωση θέτουμε τα απαραίτητα πεδία-προϋποθέσεις, τα οποία φιλτράρουν τις εγγραφές που θα επιστραφούν.

Στην κλάση περιέχονται οκτώ πεδία: μια λογική μεταβλητή που δηλώνει αν η σάρωση είναι ανοιχτή, ένας δείκτης σε χειριστή αρχείων εγγραφών, ο τύπος, το μήκος και η απόσταση του πεδίου από την αρχή της εγγραφής, ο τελεστής σύγκρισης, μια αναφορά στην τιμή σύγκρισης

και ο κωδικός της τρέχουσας εγγραφής. Οι συναρτήσεις που υλοποιούν τις απαραίτητες λειτουργίες της κλάσεις, περιγράφονται παρακάτω.

Άνοιγμα Σάρωσης Εγγραφών (OpenRecordScan)

Αν η σάρωση είναι ήδη ανοιχτή επιστρέφουμε σφάλμα εκτέλεσης. Αρχικοποιούμε όλα τα πεδία της κλάσης με τις τιμές των αντίστοιχων παραμέτρων της συνάρτησης και σημειώνουμε ότι η σάρωση είναι ανοιχτή.

Μέσω του χειριστή αρχείων εγγραφών παίρνουμε τον κωδικό της πρώτης σελίδας και ξεκινάμε ένα βρόγχο, σαρώνοντας όλες τις σελίδες του αρχείου, παίρνοντας κάθε φορά έναν χειριστή για τη εκάστοτε σελίδα. Μέσω του χειριστή παίρνουμε τα την πληροφορία και τα δεδομένα της σελίδας. Διατρέχουμε τα slots μέχρι να βρούμε το πρώτο που περιέχει κάποια εγγραφή και κρατάμε τα στοιχεία της εγγραφής στη μεταβλητή cRecord, δηλαδή στην τρέχουσα εγγραφή.

Επιστροφή Επόμενης Εγγραφής (GetNextRecord)

Αν δεν είναι ανοιχτή η scan επιστρέφουμε σφάλμα. Σε αντίθετη περίπτωση, παίρνουμε ένα χειριστή εγγραφής για την τρέχουσα εγγραφή καθώς και τον κωδικό σελίδας και το slot του. Στη συνέχεια, αρχικοποιούμε τέσσερεις μεταβλητές: δύο ζευγάρια ακέραιο-συμβολοσειρά, ένα για το αντίστοιχο πεδίο της τρέχουσας εγγραφής και ένα για την τιμή σύγκρισης για κάθε τύπο, γιατί δεν ξέρουμε ακόμα τον τύπο τις τιμής σύγκρισης.

Δημιουργούμε ένα βρόγχο για κάθε σελίδα του αρχείου, ξεκινώντας από τη σελίδα που έχουμε κρατημένη ότι σταματήσαμε την τελευταία φορά. Παίρνουμε την πληροφορία της σελίδας και δημιουργούμε ένα δεύτερο βρόγχο για κάθε slot της σελίδας που βρισκόμαστε (αν η δεν έχουμε αλλάξει σελίδα ξεκινάμε από το slot που σταματήσαμε).

Αν η εγγραφή δεν είναι κενή στη θέση που βρισκόμαστε, ελέγχουμε αν η σάρωση βρήκε αποτελέσματα σύμφωνα με τον τελεστή σύγκρισης. Αν βρήκε, σταματάει τους δύο βρόγχους επανάληψης και επιστρέφει τον χειριστή της εγγραφής που βρήκε και θέτει την επόμενη εγγραφή ως την εγγραφή που σταματήσαμε. Αν δεν έχει βρει καμιά εγγραφή που να πληρεί τις προϋποθέσεις, επιστρέφει αντίστοιχο σφάλμα.

Κλείσιμο Σάρωσης Εγγραφών (CloseRecordScan)

Ελέγχουμε αν είναι ανοιχτή η scan και αν δεν είναι, επιστρέφουμε σφάλμα. Θέτουμε ουδέτερες τιμές στα πεδία τις κλάσης και ενημερώνουμε το κατάλληλο πεδίο ότι έκλεισε η scan.

Χειριστής Εγγραφών (REM_RecordHandle)

Η κλάση είναι υπεύθυνη για το χειρισμό των εγγραφών. Αποτελείται από δύο πεδία τον κωδικό και τα δεδομένα της εγγραφής. Οι βασικές λειτουργίες της είναι η επιστροφή του κωδικού και των δεδομένων της εγγραφής.

Επιστροφή των Δεδομένων της Εγγραφής (GetData)

Επιστρέφει μέσω του ορίσματος τα δεδομένα του αντικειμένου.

Επιστροφή του Κωδικού της Εγγραφής (GetRecordID)

Επιστρέφει μέσω του ορίσματος το recordID του αντικειμένου.

Κωδικός Εγγραφής (REM_RecordID)

Η κλάση αναπαριστά τον κωδικό της εγγραφής. Αποτελείται από κωδικό της σελίδας στην οποία βρίσκεται η εγγραφή, καθώς και το slot στο οποίο βρίσκεται. Το slot συμβολίζει τη θέση της εγγραφής μέσα στη σελίδα. Επίσης, περιέχει πληροφορία για το αν έχει οριστεί ο κωδικός της σελίδας και το slot της εγγραφής.

Επιστροφή του Κωδικού Σελίδας της Εγγραφής (GetPageID)

Αν η σελίδα έχει οριστεί, επιστρέφει μέσω του ορίσματος της συνάρτησης το pageID της σελίδας και η συνάρτηση επιστρέφει OK. Αλλιώς επιστρέφει σφάλμα.

Επιστροφή του Slot της Εγγραφής (GetSlot)

Αν η θέση (slot) έχει οριστεί, επιστρέφει μέσω του ορίσματος της συνάρτησης το slot της εγγραφής και η συνάρτηση επιστρέφει OK. Αλλιώς επιστρέφει σφάλμα.

Ανάθεση Κωδικού Σελίδας για την Εγγραφή (SetPageID)

Ελέγχει αν η pageID του ορίσματος είναι διάφορη του μηδέν, αφού την πρώτη σελίδα την κρατάμε για εσωτερικές πληροφορίες του αρχείου. Αν είναι, ενημερώνει τη σελίδα με το νέο pageID και επιστρέφει OK. Αλλιώς, επιστρέφει σφάλμα.

Ανάθεση Slot για την Εγγραφή (SetSlot)

Ελέγχει αν το slot του ορίσματος είναι διάφορο του μηδέν αφού το πρώτο slot το κρατάμε για εσωτερικές πληροφορίες της σελίδας. Αν είναι, ενημερώνει τη θέση με το νέο slot και επιστρέφει OK. Αλλιώς, επιστρέφει σφάλμα.

Χρήσιμες Δομές (REM_Types)

Αυτό το Header αρχείο κρατάει δύο χρήσιμες δομές, χωρίς συναρτήσεις.

t_compOp

Αναπαριστά τους τελεστές σύγκρισης: ίσο, μικρότερο, μεγαλύτερο, διάφορο, μικρότερο-ίσο, μεγαλύτερο-ίσο και τον κενό τελεστή.

t_attrType

Αναπαριστά των τύπο των δεδομένων: ακέραιος ή συμβολοσειρά.

Υποσύστημα Καταλόγων (INXM)

Το τμήμα IXNM είναι υπεύθυνο για την δημιουργία και τον χειρισμό καταλογών της Βάσης Δεδομένων. Ο κατάλογος είναι τύπου B+ tree και χτίζεται πάνω σε ένα πεδίο κάποιου πίνακα της βάσης. Στα φύλλα του δέντρου αποθηκεύονται ζεύγη της μορφής (key, value), όπου key είναι το record id μιας εγγραφής και value είναι η τιμή του πεδίου για το οποίο χτίζεται το δέντρο, για την συγκεκριμένη εγγραφή. Κάθε κόμβος του δέντρου είναι μία σελίδα του STORM και τα φύλλα του δέντρου αποτελούν μία συνδεδεμένη λίστα. Στην αρχή του αρχείου του καταλόγου διατηρείται ένας file header με πληροφορίες για όλο το καταλόγο, ενώ στην αρχή κάθε σελίδας/κόμβου διατηρείται ένας node header με πληροφορίες για τον συγκεκριμένο κόμβο.

Οι κλάσεις που απαρτίζουν το INXM είναι:

- INXM_IndexManager.h
- INXM_IndexHandle.h
- INXM_IndexScan.h

Κλάση INXM_IndexManager.h

Η κλάση αυτή είναι υπεύθυνη για την δημιουργία, την καταστροφή, το άνοιγμα και το κλείσιμο ενός αρχείου καταλόγου. Η κλάση πραγματοποιεί κλήσεις στο τμήμα STORM που βρίσκεται ένα επίπεδο κάτω από το τμήμα INXM. Οι συναρτήσεις της κλάσης περιγράφονται αμέσως παρακάτω.

Δημιουργία καταλόγου (CreateIndex)

Η μέθοδος αυτή δημιουργεί τον κατάλογο. Δέχεται ως όρισμα το όνομα και τον αριθμό του καταλόγου καθώς και τον τύπο και το μέγεθος του χαρακτηριστικού με βάση το οποίο χτίζεται ο κατάλογος. Δεσμεύει μία σελίδα από τον STORM για αποθήκευση και αρχικοποίηση του file header.

Καταστροφή καταλόγου (DestroyIndex)

Η μέθοδος αυτή δέχεται ως όρισμα το όνομα και τον αριθμό ενός καταλόγου και έπειτα προχωρά στην καταστροφή του. Η καταστροφή γίνεται καλώντας την αντίστοιχη μέθοδο του STORM.

Άνοιγμα καταλόγου (OpenIndex)

Η μέθοδος αυτή είναι υπεύθυνη για το άνοιγμα ενός αρχείου καταλόγου, τη φόρτωση δηλαδή του αρχείου από τον δίσκο. Αρχικά, με χρήση του STORM παίρνουμε έναν χειριστή αρχείου και με χρήση αυτού παίρνουμε έναν χειριστή σελίδας για την πρώτη σελίδα του αρχείου. Αυτή η σελίδα περιέχει τον file header δηλαδή όλες τις πληροφορίες σχετικά με τον κατάλογο.

Κλείσιμο καταλόγου (CloseIndex)

Η μέθοδος αυτή είναι υπεύθυνη για το κλείσιμο ενός αρχείου καταλόγου. Πριν γίνει το κλείσιμου του αρχείου θα πρέπει να έχουμε γράψει στο δίσκο την πρώτη σελίδα του αρχείου με ότι αλλαγές έχουν γίνει σε αυτή. Δηλαδή, ενημερώνουμε τον file header. Έπειτα καλείται η αντιστοίχη μέθοδος του STORM που πραγματοποιεί το κλείσιμο.

Κλάση INXM_IndexHandle.h

Η κλάση αυτή αποτελεί τον χειριστή του καταλόγου. Εκτελεί τη λειτουργία της εισαγωγής εγγραφών στο δέντρο που χειρίζεται. Για την διαχείρηση τον κόμβων, των φύλλων και των εγγραφών έχουν δημιουργηθεί οι παρακάτω δομές:

IndexFileHeader

Κρατάει πληροφορίες για ολοκλήρο το αρχείο του καταλογού. Το πλήθος τον κόμβων του δέντρου, το πλήθος τον κόμβων που είναι φύλλα, το id της σελίδας που βρίσκεται η ρίζα του δέντρου, ο τύπος του χαρακτηριστικού με βάση το οποίο χτίστηκε το δέντρο και το μήκος αυτού του χαρακτηριστικού.

NodeHeader

Κρατάει πληροφορίες σχετικά τα περιεχόμενα ενός κόμβου. Το πλήθος των αντικειμένων που είναι ανα πάσα στιγμή αποθηκευμένα στον κόμβο, μια Boolean μεταβλητή που δείχνει αν ο τρέχον κόμβος είναι φύλλο ή ενδιάμεσος κόμβος και το id των σελίδων/κόμβων που είναι αντίστοιχα πατέρας, δεξής αδερφός, αριστερός αδερφός του κόμβου. Σε περίπτωση που ο κόμβος είναι ρίζα ο δείκτης προς τον πατέρα έχει τιμή -1, σε περίπτωση που ο κόμβος είναι ενδιάμεσος οι δείκτες προς τα αδέρφια έχουν τιμή -1.

InterItem

Αποτελεί τη δόμη για ένα αντικέιμενο που αποθήκευεται σε ενδιάμεσο κόμβο. Περιέχει το κλειδί και δύο δείκτες που δείχνουν αντίστοιχα στο αρίστερο και το δεξί παιδί του κόμβου.

LeafItem

Αποτελεί τη δομή για ένα αντικείμενο που αποθηκεύεται σε ένα φύλλο. Περιέχει το record id της εγγραφής και την τιμή για το αντίστοιχο χαρακτηριστικό, για το οποίο έχει χτιστεί το δέντρο.

Εισαγωγή καταχώρησης (InsertEntry)

Η μέθοδος αυτή δέχεται ως όρισμα ένα record id και τα αντίστοιχα δεδομένα για το χαρακτηριστικό για το οποίο έχει χτιστεί το δέντρο και είναι υπεύθυνη για την εισαγωγή αυτής της καταχώρησης στο δέντρο.

Αρχικά, γίνεται έλεγχος για το αν έχει χτιστεί δέντρο. Σε περίπτωση που δεν έχει χτιστεί καλείται η μέθοδος makeRoot() (περιγράφεται παρακάτω). Αν υπάρχει ήδη δέντρο καλείται η μέθοδος searchRightPage() για την εύρεση της σωστής σελίδας και θέσης που θα πρέπει να μπει η νέα εγγραφή. Έπειτα ελέγχεται αν η σελίδα που βρέθηκε έχει αρκετό χώρο για την νέα εγγραφή. Σε περίπτωση που έχει χώρο καλείται η insert() που αναλαμβάνει την εγγραφή στη σελίδα. Σε περίπτωση που η σελίδα δεν έχει διαθέσιμο χώρο καλείται η μέθοδος splitLeafNode() η οποία είναι υπεύθυνη για τον διαχωρισμό του κόμβου και τη δημιουργία νέας σελίδας. Στη συνέχεια, εφόσον δημιουργήθηκε χώρος καλείται η insert().

Δημιουργία ρίζας (makeRoot)

Η μέθοδος αυτή είναι υπεύθυνη για την δημιουργία της ρίζας του δέντρου. Δέχεται τα ίδια δεδομένα με την insertEntry() για μία μόνο εγγραφή. Στη συνέχεια δεσμεύει μία σελίδα από τον STORM (είναι η επόμενη από την αρχική όπου βρίσκεται ο file header) και δημιουργεί τον node header και αρχικοποιεί κατάλληλα τις τιμές του. Τέλος, εγγράφει στη σελίδα το αντικείμενο και μαζί με τον header τα γράφει ξανά στο δίσκο.

Εύρεση σωστής σελίδας (searchRightPage)

Η μέθοδος αυτή είναι υπεύθυνη για την εύρεση μιας σελίδας (το id της), η οποία περιέχει μια συγκεκριμένη τιμή. Η τιμή αυτή είναι όρισμα της μεθόδου. Η μέθοδος ξεκινώντας από τη ρίζα ακολουθεί τους κατάλληλους δείκτες και περνώντας από τους ενδιάμεσους κόμβους φτάνει στον κόμβο φύλλο που περιέχει και τη τιμή. Η μέθοδος αυτή υλοποιείται αναδρομικά.

Διαχωρισμός φύλλου (splitLeafNode)

Η μέθοδος αυτή είναι υπεύθυνη για τον διαχωρισμό ενός κόμβου φύλλο σε δύο κόμβους. Αρχικά, δημιουργεί έναν νέο κόμβο φύλλο (με δέσμευση σελίδας) και αρχικοποιεί τις τιμές του node header. Μετακινεί τα μισά αντικείμενα από τον παλιό-γεμάτο κόμβο στον νέο και ρυθμίζει κατάλληλα τους δείκτες των κόμβων. Τα μισά αντικείμενα που μετακινούνται είναι αυτά που ανήκουν στο δεύτερο μισό του φύλλου, δηλαδή είναι τα μεγαλύτερα αντικείμενα.

Στη συνέχεια, σε περίπτωση που ο κόμβος που διασπάστηκε έχει πατέρα, δηλαδή δεν είναι ρίζα, θα πρέπει να αντιγραφεί στον πατέρα η μικρότερη τιμή που είναι αποθηκευμένη στο νέο φύλλο. Γίνεται έλεγχος για το αν ο πατέρας έχει χώρο. Αν έχει χώρο γίνεται η αντιγραφεί ενώ σε περίπτωση που δεν έχει χώρο καλείται η μέθοδος splitInterNode.

Στη περίπτωση που ο προς διάσπαση κόμβος/φύλλο είναι ρίζα τότε δημιουργείται απευθείας ένα ενδιάμεσος κόμβος και αντιγράφεται εκεί η μικρότερη τιμή του νέου φύλλο.

Διαχωρισμός ενδιάμεσου κόμβου (splitInterNode)

Η μέθοδος αυτή είναι υπεύθυνη για τον διαχωρισμό ενός ενδιάμεσου κόμβου. Η λογική που ακολουθείτε είναι αντίστοιχη με της μεθόδου splitLeafNode. Δημιουργείται ένας νέος κόμβος/σελίδα και αντιγράφονται τα μισά αντικείμενα του γεμάτου κόμβου σε αυτόν. Ρυθμίζονται ανάλογα και οι δείκτες. Εφόσον, έγινε διάσπαση ενδιάμεσου κόμβου θα πρέπει να αλλάξει ανάλογα και ο κόμβος πατέρας. Λαμβάνονται υπόψη οι ίδιες περιπτώσεις με την splitInterNode.

Εισαγωγή (insert)

Η μέθοδος αυτή είναι υπεύθυνη για την εισαγωγή ενός αντικειμένου τύπου leafItem μέσα σε μια σελίδα. Αρχικά, βρίσκεται η σωστή θέση που θα πρέπει να μπει το αντικείμενο, μετακινούνται οι υπόλοιπες τιμές του φύλλου κατά μία θέση και στη συνέχεια εγγράφεται το αντικείμενο. Τέλος, ενημερώνεται ο node header του φύλλου κατάλληλα για τις αλλαγές.

keyCompare()

Η μέθοδος αυτή συγκρίνει δύο κλειδιά μεταξύ τους. Τα κλειδιά μπορεί να είναι είτε τύπου int, είτε τύπου string. Ανάλογα με το αποτέλεσμα της σύγκρισης επιστρέφει και την ανάλογη τιμή.

getInterItemSize()

Η μέθοδος αυτή επιστρέφει το μέγεθος μιας δομής τύπου InterItem.

getLeafItemSize()

Η μέθοδος αυτή επιστρέφει το μέγεθος μιας δομής τύπου LeafItem.

getInterItemData()

Η μέθοδος αυτή δέχεται ως όρισμα μια δομή τύπου InterItem και επιστρέφει μία μεταβλητή/ πίνακα τύπου char* με τα δεδομένα του συγκεκριμένου item.

getLeafItemData()

Η μέθοδος αυτή κάνει την αντίστοιχη δουλειά με την παράπανω μέθοδο αλλά για δομή τύπου LeafItem.

getInterItemStruct()

Η μέθοδος αυτή δέχεται ως όρισμα έναν δείκτη σε δεδομένα μιας δομής τύπου InterItem και επιστρέφει μία δομή τύπου InterItem. Είναι η αντίστροφη διαδικασία της getInterItemData().

getLeafItemStruct()

Η μέθοδος αυτή κάνει την αντίστοιχη δουλειά με την παραπάνω μέθοδο αλλά για δομή τύπου LeafItem. Είναι η αντίστροφη διαδικασία της getLeafItemData().

checkInterNodeForSpace()

Η μέθοδος αυτή ελέγχει αν ένας ενδιάμεσος κόμβος του δέντρου έχει χώρο για την εισαγωγή ενός νέου αντικειμένου τύπου InterItem. Επιστρέφει μια τιμή true ή false.

checkLeafNodeForSpace()

Η μέθοδος αυτή ελέγχει αν ένας κόμβος που είναι και φύλλο του δέντρου έχει χώρο για την εισαγωγή μιας νέας εγγραφής. Επιστρέφει μια τιμή true ή false.

Κλάση INXM_IndexScan.h

Η κλάση αυτή επιτρέπει την προσπέλαση και την ανάκτηση των εγγραφών που είναι αποθηκευμένες στο δέντρο και ικανοποιούν κάποια συνθήκη.

Άνοιγμα καταλόγου για προσπέλαση (OpenIndexScan)

Η μέθοδος αυτή είναι υπεύθυνη για το άνοιγμα ενός αρχείου καταλόγου με σκόπο την προσπέλαση και ανάκτηση δεδομένων/εγγραφών. Δέχεται ως όρισμα έναν χειριστή αρχείου καταλόγου, τον τελεστή και την τιμή (value) με βάση τα οποία θα γίνει κάποια σύγκριση. Αρχικά, με χρήση της μεθόδου searchRightPage που έχει οριστεί στη κλάση INXM_IndexHandle, βρίσκει τη σελίδα/φύλλο και τη θέση/slot που είναι αποθηκευμένη η τιμή value. Αν δεν υπάρχει η τιμή value τότε βρίσκει την αμέσως μεγαλύτερη τιμή. Δημιουργεί ένα REM_RecordID και το αρχικοποιεί με τα στοιχεία της παραπάνω τιμής.

Κλείσιμο καταλόγου για προσπέλαση (CloseIndexScan)

Η μέθοδος αυτή είναι υπεύθυνη για το κλείσιμο του καταλόγου που έχει ανοιξει για προσπέλαση. Τερματίζεται δηλαδή η προσπέλαση του καταλόγου.

Επιστροφή επόμενης καταχώρησης (GetNextEntry)

Η μέθοδος αυτή είναι υπεύθυνη για την εύρεση και την επιστροφή ενός record id, το οποίο ικανοποιεί τη συνθήκη, που έχει οριστεί κατα το ανοίγμα του καταλόγου για προσπέλαση. Πριν από την εκτέλεση αυτής της μεθόδου έχει προηγηθεί η κλήση της OpenIndexScan(), επομένως γνωρίζουμε το page id και το slot της τιμής value ή της αμέσως μεγαλύτερης τιμής. Αυτή η πληροφορία αποθηκεύεται στη μεταβλητή cRecord. Έχοντας ως αφετηρία αυτό το slot και ανάλογα με τον τελεστή που έχει ορισθεί επιστρέφουμε την τιμή που είναι είτε αμέσως μεγαλύτερη (δεξιά) του value, είτε είναι αμέσως μικρότερη (αριστερά) του value. Κάθε φορά ενημερώνουμε το cRecord έτσι ώστε στην επόμενη εκτέλεση η μέθοδος να επιστρέψει την επόμενη τιμή που ικανοποιεί τη συνθήκη. Σε περίπτωση που εξαντλήσουμε όλες τις εγγραφές της σελίδας/φύλλου τότε προχωράμε στο αμέσως επόμενο φύλλο (δεξί ή αριστερό), αφού τα φύλλα του δέντρου αποτελούν συνδεδεμένη λίστα.

Υποσύστημα Διαχείρισης Συστήματος (SYSM)

Το υποσύστημα διαχείρισης συστήματος είναι υπεύθυνο για τη διαχείριση των Βάσεων Δεδομένων σε επίπεδο συστήματος. Χωρίζεται σε διαχειριστή συστήματος και διαχειριστή μεταδεδομένων.

Διαχειριστής Συστήματος (SYSM_SystemManager)

Είναι υπεύθυνος για τη διαχείριση του συστήματος, δηλαδή τη δημιουργία, καταστροφή, άνοιγμα και κλείσιμο των Βάσεων Δεδομένων. Περιέχει τέσσερα πεδία: τη διαδρομή για το φάκελο με τις Βάσεις Δεδομένων, ένα διαχειριστή αρχείων εγγραφών, το όνομα της Βάσης Δεδομένων που είναι ανοιχτή και μια μεταβλητή που μας δείχνει αν υπάρχει κάποια ανοιχτή Βάση Δεδομένων. Παρακάτω περιγράφονται οι βασικές λειτουργίες της κλάσης.

Δημιουργία Βάσης Δεδομένων (CreateDatabase)

Δημιουργούμε τον φάκελο στον οποίο θα φιλοξενηθούν όλα τα αρχεία της βάσης. Στη συνέχεια, ελέγχουμε αν υπάρχει βάση με το ίδιο όνομα ή αν δεν βρέθηκε ο φάκελος και επιστρέφουμε το αντίστοιχο σφάλμα. Τέλος, δημιουργούμε τα μεταδεδομένα του συστήματος χρησιμοποιώντας το διαχειριστή αρχείων εγγραφών.

Καταστροφή Βάσης Δεδομένων (DropDatabase)

Για να διαγράψουμε το φάκελο της βάσης πρέπει να είναι άδειος. Για το λόγο αυτό, δημιουργούμε ένα πρότυπο αρχείων, για τη σάρωση όλων των αρχείων του φακέλου της βάσης. Παίρνουμε τον χειριστή του πρώτου αρχείου που συναντάμε στο φάκελο και αν η τιμή που παίρνει είναι δεκτή, ξεκινάμε ένα βρόγχο επανάληψης για να διαγράψουμε τα αρχεία του φακέλου ένα προς ένα.

Στο βρόγχο, για κάθε αρχείο που βρίσκουμε, εκτός από τα “.” και “..”, παίρνουμε το όνομα του αρχείου από το χειριστή του και φτιάχνουμε το μονοπάτι μέχρι το αρχείο αυτό. Στη συνέχεια, καταστρέφουμε το αρχείο και προχωράμε στο επόμενο, μέχρι να σβηστούν όλα τα αρχεία του φακέλου. Τέλος, διαγράφουμε το φάκελο και επιστρέφουμε σφάλμα σε περίπτωση προβλήματος διαγραφής του.

Άνοιγμα Βάσης Δεδομένων (OpenDatabase)

Αν κάποια βάση είναι ήδη ανοιχτή επιστρέφουμε σφάλμα. Επιχειρούμε να δημιουργή-σουμε το φάκελο της βάσης, για να ελέγξουμε αν υπάρχει ήδη.

Δηλώνουμε ότι η βάση είναι ανοιχτή θέτοντας το όνομά της στο κατάλληλο πεδίο της κλάσης και ενημερώνουμε το σύστημα ότι κάποια βάση είναι ανοιχτή.

Κλείσιμο Βάσης Δεδομένων (CloseDatabase)

Αν δεν υπάρχει ανοιχτή βάση επιστρέφουμε σφάλμα. Σε διαφορετική περίπτωση, κλείνου-με τη βάση, δηλαδή σβήνουμε το όνομα της ανοιχτής βάσης και ενημερώνουμε το σύστημα ότι καμία βάση δεν είναι ανοιχτή.

Διαχειριστής Μεταδεδομένων (SYSM_MetaManager)

Είναι υπεύθυνος για τη διαχείριση των μεταδεδομένων, δηλαδή την εισαγωγή, διαγραφή και ενημέρωση των αυτών. Τα μεταδεδομένα χωρίζονται σε μεταδεδομένα πίνακα και πεδίου. Έχουν δημιουργηθεί δύο δομές για την ευκολότερη διαχείριση των μεταδεδομένων:

- **Μεταδεδομένα Πίνακα (RelMet)**

  Κρατάει πληροφορία για το όνομα του πίνακα, το μέγεθος των εγγραφών που περιέχει ο πίνακας, το πλήθος των εγγραφών που χωράνε σε μια σελίδα του αρχείου του πίνακα και το πλήθος των καταλόγων που έχουν δημιουργηθεί πάνω σε πεδία του πίνακα.

- **Μεταδεδομένα Πεδίου (AttrMet)**

  Κρατάει πληροφορία για το όνομα του πίνακα που βρίσκεται το πεδίο, το όνομα του πεδίου, τη μετατόπισή του μέσα στην εγγραφή, τον τύπο του, το μήκος του και τον αριθμό καταλόγου του (αν δεν υπάρχει παίρνει άκυρη τιμή).

Η κλάση περιέχει δύο πεδία: ένα διαχειριστή εγγραφών και ένα διαχειριστή συστήματος. Οι βασικές λειτουργίες της αναλύονται παρακάτω.

Εισαγωγή Μεταδεδομένων Πίνακα (InsertRelationMetadata)

Αν δεν υπάρχει ανοιχτή βάση ή αν ο πίνακας υπάρχει ήδη στο αρχείο των μεταδεδομένων, επιστρέφει σφάλμα.

Δημιουργούμε το μονοπάτι για τα μεταδεδομένα των πινάκων της βάσης και ανοίγουμε το αρχείο με τη βοήθεια του διαχειριστή αρχείων εγγραφών, παίρνοντας το χειριστή του αρχείου. Δημιουργούμε την νέα εγγραφή με τα δεδομένα που παίρνουμε ως ορίσματα, εισάγοντάς τα ένα προς ένα στην κατάλληλη θέση σύμφωνα με την κατάλληλη μετατόπιση του καθενός, την εισάγουμε στο αρχείο των μεταδεδομένων πίνακα με τη βοήθεια του χειριστή αρχείων και κλείνουμε το αρχείο.

Αν η εγγραφή υπήρχε, επιστρέφουμε κατάλληλο μήνυμα σφάλματος.

Εισαγωγή Μεταδεδομένων Πεδίου (InsertAttributeMetadata)

Αν δεν υπάρχει ανοιχτή βάση ή αν το πεδίο του πίνακα υπάρχει ήδη στο αρχείο των μεταδεδομένων, επιστρέφει σφάλμα.

Δημιουργούμε το μονοπάτι για τα μεταδεδομένα των πεδίων της βάσης και ανοίγουμε το αρχείο με τη βοήθεια του διαχειριστή αρχείων εγγραφών, παίρνοντας το χειριστή του αρχείου. Δημιουργούμε την νέα εγγραφή με τα δεδομένα που παίρνουμε ως ορίσματα, εισάγοντάς τα ένα προς ένα στην κατάλληλη θέση σύμφωνα με την κατάλληλη μετατόπιση του καθενός, την εισάγουμε στο αρχείο των μεταδεδομένων πεδίου με τη βοήθεια του χειριστή αρχείων και κλείνουμε το αρχείο.

Αν η εγγραφή υπήρχε, επιστρέφουμε κατάλληλο μήνυμα σφάλματος.

Διαγραφή Μεταδεδομένων Πίνακα (DeleteRelationMetadata)

Αν δεν υπάρχει ανοιχτή βάση επιστρέφουμε σφάλμα. Δημιουργούμε το μονοπάτι για τα μεταδεδομένα πίνακα, ανοίγουμε το αρχείο στο οποίο οδηγεί το μονοπάτι με τη βοήθεια του διαχειριστή αρχείων εγγραφών και παίρνουμε το χειριστή αρχείων εγγραφών.

Ανοίγουμε μια καινούρια σάρωση, με ορίσματα που παίρνουμε από τη συνάρτηση, για να πάρουμε την εγγραφή που μας ενδιαφέρει. Αν αυτή αποτύχει, σημαίνει ότι το αρχείο δεν έχει εγγραφές και γι αυτό κλείνουμε την σάρωση. Αλλιώς παίρνουμε την εγγραφή που μας ενδιαφέρει και μέσω του χειριστή εγγραφών που αυτή επιστρέφει, διαγράφουμε την εγγραφή. Στη συνέχεια κλείνουμε τη σάρωση και το αρχείο.

Με την ίδια διαδικασία ανοίγουμε το αρχείο μεταδεδομένων πεδίου και διαγράφουμε τις εγγραφές που αναπαριστούν πεδία του πίνακα που μόλις διαγράψαμε.

Διαγραφή Μεταδεδομένων Πεδίου (DeleteAttributeMetadata)

Αν δεν υπάρχει ανοιχτή βάση επιστρέφουμε σφάλμα. Δημιουργούμε το μονοπάτι για τα μεταδεδομένα πεδίου, ανοίγουμε το αρχείο στο οποίο οδηγεί το μονοπάτι με τη βοήθεια του διαχειριστή αρχείων εγγραφών και παίρνουμε το χειριστή αρχείων εγγραφών.

Ανοίγουμε μια καινούρια σάρωση, με ορίσματα που παίρνουμε από τη συνάρτηση, για να πάρουμε την εγγραφή που μας ενδιαφέρει. Αν αυτή αποτύχει, σημαίνει ότι το αρχείο δεν έχει εγγραφές και γι αυτό κλείνουμε την σάρωση.

Σαρώνουμε όλες της εγγραφές του αρχείου ψάχνοντας το όνομα του πεδίου και για κάθε μία από αυτές πραγματοποιούμε μία νέα σάρωση όλων των εγγραφών ,του ίδιου αρχείου, (με τον ίδιο τρόπο) ψάχνοντας όμως τώρα το όνομα του πίνακα στον οποίο βρίσκεται το πεδίο.

Όταν βρούμε την εγγραφή με τα κοινά στοιχεία pageID και slot (δηλαδή την συγκεκριμένη ιδιότητα ενός συγκεκριμένου πίνακα) τότε την διαγράφουμε από το αρχείο.

Ενημέρωση Μεταδεδομένων Πίνακα (UpdateRelationMetadata)

Αν δεν υπάρχει ανοιχτή βάση επιστρέφουμε σφάλμα. . Δημιουργούμε το μονοπάτι για τα μεταδεδομένα πίνακα, ανοίγουμε το αρχείο στο οποίο οδηγεί το μονοπάτι με τη βοήθεια του διαχειριστή αρχείων εγγραφών και παίρνουμε το χειριστή αρχείων εγγραφών. Αν αυτό αποτύχει σημαίνει ότι το αρχείο είναι άδειο και γι αυτό το κλείνουμε.

Αλλιώς σαρώνουμε το αρχείο και παίρνουμε ένα χειριστή για την εγγραφή που μας ενδιαφέρει. Από αυτόν παίρνουμε τα δεδομένα της εγγραφής και τα αντικαθιστούμε με τις νέες τιμές. Ενημερώνουμε την εγγραφή με τη βοήθεια του χειριστή αρχείων εγγραφών.

Τέλος, ελέγχουμε αν έχει αλλάξει το όνομα της ιδιότητας και με όμοια διαδικασία ενημερώνουμε και το αρχείο μεταδεδομένων πεδίου για την αλλαγή του ονόματος του πίνακα σε όλα του τα πεδία.

Ενημέρωση Μεταδεδομένων Πεδίου (UpdateAttributeMetadata)

Αν δεν υπάρχει ανοιχτή βάση επιστρέφουμε σφάλμα. Δημιουργούμε το μονοπάτι για τα μεταδεδομένα πεδίου, ανοίγουμε το αρχείο στο οποίο οδηγεί το μονοπάτι με τη βοήθεια του διαχειριστή αρχείων εγγραφών και παίρνουμε το χειριστή αρχείων εγγραφών.

Ανοίγουμε μια καινούρια σάρωση, με ορίσματα που παίρνουμε από τη συνάρτηση, για να πάρουμε την εγγραφή που μας ενδιαφέρει. Αν αυτή αποτύχει, σημαίνει ότι το αρχείο δεν έχει εγγραφές και γι αυτό κλείνουμε την σάρωση.

Αν το αρχείο δεν είναι άδειο, τότε σαρώνουμε όλες της εγγραφές του, ψάχνοντας το όνομα της σχέσης και για κάθε μία από αυτές πραγματοποιούμε μία νέα σάρωση όλων των εγγραφών, του ίδιου αρχείου, (με τον ίδιο τρόπο) ψάχνοντας όμως τώρα το όνομα του πίνακα στον οποίο βρίσκεται.

Όταν βρούμε την εγγραφή με τα κοινά στοιχεία pageID και slot (δηλαδή την συγκεκριμένη ιδιότητα ενός συγκεκριμένου πίνακα) τότε παίρνουμε τα δεδομένα της και στη θέση τους βάζουμε τις νέες τιμές. Τέλος, ενημερώνουμε την εγγραφή με τη βοήθεια του χειριστή αρχείων εγγραφών.

Επιστροφή Δομής Μεταδοδεμένων Πίνακα (GetRelationMetadata)

Αν δεν υπάρχει ανοιχτή βάση επιστρέφουμε σφάλμα. Δημιουργούμε το μονοπάτι για τα μεταδεδομένα πίνακα, ανοίγουμε το αρχείο στο οποίο οδηγεί το μονοπάτι με τη βοήθεια του διαχειριστή αρχείων εγγραφών και παίρνουμε το χειριστή αρχείων εγγραφών.

Ανοίγουμε μια καινούρια σάρωση, με ορίσματα που παίρνουμε από τη συνάρτηση, για να πάρουμε την εγγραφή που μας ενδιαφέρει. Αν αυτή αποτύχει, σημαίνει ότι το αρχείο δεν έχει εγγραφές και γι αυτό κλείνουμε την σάρωση. Αλλιώς παίρνουμε την εγγραφή που μας ενδιαφέρει και μέσω του χειριστή εγγραφών που αυτή επιστρέφει, παίρνουμε τα δεδομένα της εγγραφής.

Τέλος, ενημερώνουμε τα πεδία της δομής μεταδεδομένων πίνακα με τα στοιχεία που είναι αποθηκευμένα στα δεδομένα της εγγραφής.

Επιστροφή Δομής Μεταδεδομένων Πεδίου (GetAttributeMetadata)

Αν δεν υπάρχει ανοιχτή βάση επιστρέφουμε σφάλμα. Δημιουργούμε το μονοπάτι για τα μεταδεδομένα πεδίου, ανοίγουμε το αρχείο στο οποίο οδηγεί το μονοπάτι με τη βοήθεια του διαχειριστή αρχείων εγγραφών και παίρνουμε το χειριστή αρχείων εγγραφών.

Ανοίγουμε μια καινούρια σάρωση, με ορίσματα που παίρνουμε από τη συνάρτηση, για να πάρουμε την εγγραφή που μας ενδιαφέρει. Αν αυτή αποτύχει, σημαίνει ότι το αρχείο δεν έχει εγγραφές και γι αυτό κλείνουμε την σάρωση. Αλλιώς παίρνουμε την εγγραφή που μας ενδιαφέρει (πρέπει να έχει και το ίδιο όνομα πίνακα) και μέσω του χειριστή εγγραφών που αυτή επιστρέφει, παίρνουμε τα δεδομένα της εγγραφής.

Τέλος, ενημερώνουμε τα πεδία της δομής μεταδεδομένων πεδίου με τα στοιχεία που είναι αποθηκευμένα στα δεδομένα της εγγραφής.

Χρήσιμες Σταθερές (SYSM_Parameters)

Σε αυτό το Header αρχείο αποθηκεύουμε σταθερές που έχουν σχέση με τα αρχεία των μεταδεδομένων ώστε οποιαδήποτε αλλαγή σε σχέση με τα μεγέθη και της μετατοπίσεις αυτών να γίνεται μόνο σε αυτό το αρχείο. Παρακάτω αναλύεται η κάθε σταθερά και φαίνεται ο σκοπός και η χρησιμότητά της.

REL_NAME_SIZE

Καθορίζει το μέγεθος σε byte του ονόματος ενός πίνακα.

ATTR_NAME_SIZE

Καθορίζει το μέγεθος σε byte του ονόματος ενός πεδίου.

REL_SIZE

Καθορίζει το μέγεθος σε byte των εγγραφών που αποθηκεύονται στο αρχείο μεταδεδομέ-νων πίνακα.

ATTR_SIZE

Καθορίζει το μέγεθος σε byte των εγγραφών που αποθηκεύονται στο αρχείο μεταδεδομέ-νων πεδίου.

REL_NAME_OFFSET

Η μετατόπιση του πεδίου «όνομα» σε μια εγγραφή του αρχείου μεταδεδομένων πίνακα.

REL_REC_SIZE_OFFSET

Η μετατόπιση του πεδίου «μέγεθος εγγραφής» σε μια εγγραφή του αρχείου μεταδεδομέ-νων πίνακα.

REL_ATTR_NUM_OFFSET

Η μετατόπιση του πεδίου «πλήθος πεδίων» σε μια εγγραφή του αρχείου μεταδεδομένων πίνακα.

REL_INX_NUM_OFFSET

Η μετατόπιση του πεδίου «πλήθος καταλόγων» σε μια εγγραφή του αρχείου μεταδεδομέ-νων πίνακα.

ATTR_REL_NAME_OFFSET

Η μετατόπιση του πεδίου «όνομα πίνακα» σε μια εγγραφή του αρχείου μεταδεδομένων πεδίου.

ATTR_NAME_OFFSET

Η μετατόπιση του πεδίου «όνομα πεδίου» σε μια εγγραφή του αρχείου μεταδεδομένων πεδίου.

ATTR_OFFSET_OFFSET

Η μετατόπιση του πεδίου «μετατόπιση πεδίου» σε μια εγγραφή του αρχείου μεταδεδομέ-νων πεδίου.

ATTR_TYPE_OFFSET

Η μετατόπιση του πεδίου «τύπος πεδίου» σε μια εγγραφή του αρχείου μεταδεδομένων πεδίου.

ATTR_LENGTH_OFFSET

Η μετατόπιση του πεδίου «μήκος πεδίου» σε μια εγγραφή του αρχείου μεταδεδομένων πεδίου.

ATTR_INX_NO_OFFSET

Η μετατόπιση του πεδίου «αριθμός καταλόγου» σε μια εγγραφή του αρχείου μεταδεδομέ-νων πεδίου.

Υποσύστημα Διαχείρησης Εντολών (SSQLM)

Το υποσύστημα είναι υπεύθυνο για την εκτέλεση των εντολών που έχει δώσει ο χρήστης. Χωρίζεται σε δύο μέρη, στην υπογλώσσα DDL και στην υπογλώσσα DML.

Διαχειριστής Υπογλώσσας DDL (SSQLM_DDL_Manager)

Η κλάση είναι υπεύθυνη για την εκτέλεση των εντολών που έχουν να κάνουν με την κατασκευή και καταστροφή πινάκων και καταλόγων.

Στο private τμήμα, περιέχει τις εξής μεταβλητές: path τύπου string, στην οποία αποθηκεύεται το όνομα του καταλόγου του συστήματος στον οποίο είναι αποθηκευμένοι οι πίνακες της βάσης. Ένα δείκτη σε έναν RecordFileManager. Ένα δείκτη σε ένα MetaManager. openedDataBase τύπου string, στην οποία αποθηκεύεται το όνομα της εκάστοτε ανοιχτής βάσης δεδομένων.

Επίσης υπάρχει η συνάρτηση checkAttrType, σκοπός της οποίας είναι παίρνοντας σαν char τον τύπο δεδομένων του attribute που θέλει να εισάγει ο χρήστης όταν δημιουργεί τον πίνακα, να επιστρέψει τον τύπο δεδομένων σε μορφή t_attrType, και το μέγεθος αυτού.

Ορισμός ανοιχτής Βάσης (SetOpenedDatabase)

Η συνάρτηση παίρνει ως όρισμα ένα όνομα βάσης δεδομένων και το προσθέτει στη μεταβλητή path.

Έλεγχος τύπου δεδομένων πεδίου (checkAttrType)

Ο σκοπός της συνάρτησης αναφέρεται παραπάνω. Η συνάρτηση παίρνει ως όρισμα τύπου char τον τύπο δεδομένων του attribute που θέλει να εισάγει ο χρήστης όταν δημιουργεί ένα νέο πίνακα. Μετατρέπει αυτό το όρισμα σε string στη μεταβλητή attrTypeAsString. Αν το μήκος της attrTypeAsString είναι μικρότερο ή ίσο του 3 τότε σημαίνει οτι είναι INT. Δίνουμε στα ορίσματα επιστροφής τις κατάλληλες τιμές. Αλλιώς το attribute είναι τύπου STRING. Επεξεργαζόμαστε την attrTypeAsString για να πάρουμε το μέγεθος του attribute και δίνουμε τιμές στα ορίσματα επιστροφής.

Δημιουργία πίνακα (Create Table)

Σκοπός της συνάρτησης είναι η δημιουργία ενός πίνακα στη βάση δεδομένων.

Παίρνει σαν ορίσματα το όνομα του πίνακα και τη λίστα των πεδίων που αυτός θα έχει.

Μετατρέπει τη λίστα σε string στη μεταβλητή attrList. Με επεξεργασία της λίστας εξάγουμε για κάθε πεδίο το όνομά του και τον τύπο του. Ενημερώνουμε τους πίνακες μεταδεδομένων πεδίων και πινάκων και τέλος δημιουργούμε ένα νέο αρχείο.

Καταστροφή Πίνακα (Drop Database)

Παίρνει σαν όρισμα το όνομα του πίνακα που θέλουμε να διαγράψουμε.

Σαρώνει το αρχείο μεταδεδομένων και σβήνει τις εγγραφές που σχετίζονται με τον πίνακα, καθώς και τους καταλόγους που αντιστοιχούν στα πεδία του. Τέλος σβήνει το αρχείο εγγραφών του πίνακα.

Δημιουργία Καταλόγου (Create Index)

Παίρνει σαν ορίσματα το όνομα του πίνακα και του πεδίου για το οποίο θα δημιουργήσει κατάλογο.

Παίρνει τα μεταδεδομένα του πίνακα και του πεδίου, δημιουργεί τον κατάλογο και ενημερώνει τα μεταδεδομένα.

Καταστροφή Καταλόγου (Drop Index)

Παίρνει σαν ορίσματα το όνομα του πίνακα και του πεδίου για το οποίο θα δημιουργήσει κατάλογο.

Παίρνει τα μεταδεδομένα του πίνακα και του πεδίου, διαγράφει τον κατάλογο και ενημερώνει τα μεταδεδομένα.

Διαχειριστής Υπογλώσσας DML

Η κλάση είναι υπεύθυνη για την εκτέλεση των εντολών που αφορούν τον χειρισμό των δεδομένων.

Στο private τμήμα του έχει τις εξής μεταβλητές: Ένα δείκτη σε έναν RecordFileManager. Ένα δείκτη σε ένα MetaManager. openedDataBase τύπου string, στην οποία αποθηκεύεται το όνομα της εκάστοτε ανοιχτής βάσης δεδομένων. path τύπου string, στην οποία αποθηκεύεται το όνομα του καταλόγου του συστήματος στον οποίο είναι αποθηκευμένοι οι πίνακες της βάσης.

Στη συνέχεια παρατίθενται και αναλύονται οι συναρτήσεις που βρίσκονται στο private και στο public κομμάτι.

Ορισμός ανοιχτής Βάσης (SetOpenedDatabase)

Η συνάρτηση παίρνει ως όρισμα ένα όνομα βάσης δεδομένων και το προσθέτει στη μεταβλητή path.

Πρώτο σταδιο επικύρωσης ερωτήματος (queryValidationStageOne)

Σκοπός της συνάρτησης είναι να ελέγξει αν τα πεδία που αναφέρονται στο SELECT τμήμα του ερωτήματος ανήκουν στους πίνακες που αναφέρονται στο FROM τμήμα του ερωτήματος.

Παίρνει σαν ορίσματα τα προαναφερθέντα τμήματα και τα μετατρέπει σε string. Ξεκινάει με τα ορίσματα του SELECT, ελέγχει την μορφή με την οποία δίνονται. Αν είναι της μορφής tableName.attributeName, τότε κρατά το όνομά του και τον πίνακα στον οποίο ανήκει και διατρέχει τους πίνακες του τμήματος FROM για να δει αν υπάρχει ταύτιση. Μόλις συμβεί η πρώτη ταύτιση, ορίζει την μεταβλητή select_from ίση με true και σταματά τη σύγκριση. Αν η select_from δεν έχει γίνει true, τότε επιστρέφεται μήνυμα λάθους.

Αν το πεδίο δεν είναι στην παραπάνω μορφή, τότε με κλήση άλλης συνάρτησης ανακτούμε τον πίνακα στον οποίο αυτό ανήκει και μέσα σε αυτή γίνεται και ο παραπάνω έλεγχος. Τέλος επιστρέφει όλα τα ορίσματα της SELECT στη μορφή tableName.attributeName.

Επικύρωση Συνθήκης WHERE(ValidateWhereSubconditions)

Σκοπός της συνάρτησης είναι να ελέγξει αν τα πεδία στις υποσυνθήκες του τμήματος WHERE ανήκουν στους πίνακες του τμήματος FROM ενός query. Παίρνει σαν ορίσματα τις υποσυνθήκες του τμήματος WHERE και τους πίνακες του τμήματος FROM και τα μετατρέπει σε string.

Για κάθε υποσυνθήκη, καλεί τις κατάλληλες συναρτήσεις και αναγνωρίζει τον τελεστή σύγκρισής της καθώς και το όρισμα στα δεξιά και τα αριστερά του τελεστή. Για το αριστερό όρισμα καλεί την συνάρτηση queryValidationStageOne() για να ελέγξει αν αυτό υπάρχει στους πίνακες του FROM. Για το δεξιό όρισμα, πρώτα ελέγχουμε οτι δεν είναι τιμή (αριθμός ή συμβολοσειρά) και στη συνέχεια καλούμε την ίδια συνάρτηση με πριν. Τέλος, επιστρέφουμε τις υποσυνθήκες, έχοντας αλλάξει τη μορφή των πεδίων που συμμετέχουν σε αυτές σε tableName.attributeName.

Αναγνώριση Τελεστή Σύγκρισης (reconCompOp)

Σκοπός της συνάρτησης είναι να αναγνωρίσει τον τελεστή σύγκρισης μίας υποσυνθήκης του ερωτήματος WHERE και να επιστρέψει τον τελεστή σε τύπο char και t_compOp.

Παίρνει την υποσυνθήκη και αφού κάνει κατάλληλους ελέγχους, αναγνωρίζει τον τελεστή και δίνει κατάλληλες τιμές στα ορίσματα επιστροφής.

Ανάλυση υποσυνθήκης (breakSubCond)

Σκοπός της συνάρτησης είναι να επιστρέψει το αριστερό και το δεξί όρισμα μιας υποσυνθήκης. Αυτό επιτυγχάνεται με τη μετατροπή του ορίσματος σε string και την κατάτμησή του. Να σημειωθεί ότι τα ορίσματα που παίρνε η συνάρτηση κατά την κλήση της είναι η υποσυνθήκη αλλά και ο τελεστής σύγκρισης.

Έλεγχος τύπου δεδομένων πεδίου (checkAttrType)

Σκοπος είναι η αναγνώριση του τύπου δεδομένων του πεδίου. Η συνάρτηση σαρώνει το αρχείο attr.met και μόλις βρει το πεδίο αποθηκεύει τον πίνακα στον οποίο ανήκει και τύπο δεδομένων του. Ελέγχει αν το πεδίο που ψάχνουμε ανήκει όντος στον πίνακα που πήρε ως όρισμα και επιστρέφει τον τύπο του.

Σάρωση πινάκων τμήματος FROM(scanFromTables)

Σαρώνουμε το attr.met για το πεδίο που πήραμε ως όρισμα και από εκεί παίρνουμε τον πίνακα στον οποίο ανήκει. Μετατρέπουμε το τμήμα του FROM σε string και αφαιρούμε τα κενά. Στη συνέχεια σαρώνουμε όλους τους πίνακες του FROM και ελέγχουμε αν ο πίνακας του πεδίου (που βρήκαμε παραπάνω) υπάρχει και στο FROM. Αν από τη σάρωση στο attr.met πάρουμε πάνω από έναν πίνακα για το συγκεκριμένο πεδίο και υπάρξει και δεύτερη ταύτιση με πίνακα του FROM τότε επιστρέφουμε μήνυμα λάθους. Σε περίπτωση που το όρισμα είναι τιμή και άρα δεν υπάρχει στο attr.met τότε σαν πίνακας επιστρέφεται το αλφαριθμητικό “NO_TABLE”.

Δημιουργία πίνακα TEMP (createTempTable)

Σκοπος της συνάρτησης είναι να δημιουργήσει έναν πίνακα ο οποίος θα αποτελείται από όλους τους συνδυασμούς εγγραφών των πινάκων που βρίσκονται στο FROM. Αρχικά φτιάχνουμε τη δομή του πίνακα που θα δημιουργήσουμε. Δίνουμε αρχικές τιμές στα μεταδεδομένα του. Στη συνέχεια τροποποιούμε τα μεταδεδομένα του νέου πίνακα και των πεδίων του. Ενημερώνουμε το αρχείο rel.met και δημιουργούμε το αρχείο του νέου πίνακα.

Στη συνέχεια διατρέχουμε κάθε πίνακα του FROM ώστε να πάρουμε όλους τους συνδυασμούς πλειάδων με τις οποίες γεμίζουμε τον νέο πίνακα.

Εκτέλεση Υποσυνθήκης (executeSubCondition)

Παίρνουμε τα μεταδεδομένα των πεδίων που μας ενδιαφέρουν. Ελέγχουμε αρχικά για το αριστερό πεδίο, αν υπάρχει κατάλογος και τον ανοίγουμε, αλλιώς ξεκινάμε μία scan. Απο τον κατάλογο, ή τον πίνακα select παίρνουμε μία μία τις τιμές των πεδίων κάθε εγγραφής (σαν string αλλά και σαν int). Στη συνέχεια, παίρνουμε περιπτώσεις για τον τελεστή σύγκρισης της συνθήκης και ελέγχουμε τον τύπο των ορισμάτων. Αν για τις τιμές των ορισμάτων ισχύει ο αντίθετος τελεστής απο αυτόν της συνθήκης, διαγράφουμε την εγγραφή απο τον πίνακα. Ουσιαστικά σβήνουμε απο τον πίνακα select τις εγγραφές που δεν ικανοποιούν τη συνθήκη.

Εκτέλεση υποσυνθήκης με τιμή(executeSubConditionValue)

Παίρνουμε τα μεταδεδομένα του πεδίου του αριστερού τμήματος της συνθήκης. Ανοίγουμε τον πίνακα select.temp. Ελέγχουμε αν υπάρχει κατάλογος για το πεδίο και αν ναι, τον ανοίγουμε. Ορίζουμε τον αντίθετο τελεστή από αυτόν της υποσυνθήκης. Σαρώνουμε τον πίνακα (ή τον κατάλογο αν υπάρχει) και παίρνουμε μία μία τις πλειάδες που έχουν αντίθετη τιμή (από αυτή της υποσυνθήκης) και τις διαγράφουμε από τον πίνακα select.temp.

Διαγραφή προσωρινού πίνακα (DropTempTable)

Πρώτα διαγράφουμε τα μεταδεδομένα που αφορούν τον πίνακα και στη συνέχεια τον ίδιο τον πίνακα select.temp.

Εμφάνιση Πίνακα(ShowTable)

Διαβάζουμε τα μεταδεδομένα του πίνακα που δώσαμε ως όρισμα στη συνάρτηση. Στη συνέχεια σαρώνουμε το attr.met για να βρούμε όλα τα πεδία του πίνακα και παίρνουμε το όνομα του κάθε πεδίου, ελέγχουμε αν αυτό υπάρχουν στο τμήμα SELECT του ερωτήματος και τα εμφανίζουμε. Ξεκινάμε μία νέα σάρωση στον πίνακα που πήραμε ως όρισμα και παίρνουμε όλες τις εγγραφές του. Για κάθε μία παίρνουμε τα δεδομένα της και ξεκινάμε μία νέα σάρωση στον attr.met για να βρούμε το όνομα του πεδίου που μας ενδιαφέρει αλλά και το offset του ώστε να τυπώσουμε την τιμή του πεδίου.

Επιλογή Εγγραφής (SelectRecord)

Εκτέλεση ενός ερωτήματος επιλογής. Αρχικά καλώντας τις αντίστοιχες συναρτήσεις ελέγχουμε την ορθότητα του ερωτήματος. Έπειτα δημιουργούμε έναν προσωρινό πίνακα ο οποίος προκύπτει από το καρτεσιανό γινόμενο των πινάκων του FROM. Μετά για κάθε υποσυνθήκη αναγνωρίζουμε τον τελεστή σύγκρισής της και τους όρους που θα συγκρίνουμε. Επεξεργαζόμαστε τους όρους, ελέγχουμε αν συγκρίνουμε πεδίο με τιμή ή πεδίο με πεδίο και καλούμε την συνάρτηση εκτέλεσης του ερωτήματος. Τέλος καλούμε τη συνάρτηση εμφάνισης του αποτελέσματος και διαγράφουμε τον προσωρινό πίνακα.

Εισαγωγή Εγγραφής(InsertRecord)

Παίρνουμε τα μεταδεδομένα του πίνακα στον οποίο πρόκειται να εισάγουμε τη νέα εγγραφή. Δεσμεύουμε χώρο για τη νέα εγγραφή. Ανοίγουμε το αρχείο attr.met και το σαρώνουμε. Για κάθε πεδίο που μας ενδιαφέρει, κρατάμε το όνομα του και παίρνουμε τα μεταδεδομένα του πεδίου με σκοπό να αναγνωρίσουμε τον τύπο δεδομένων του. Επεξεργαζόμαστε το ερώτημα για να αποσπάσουμε τις τιμές των πεδίων που θέλουμε να εισάγουμε. Αποθηκεύουμε τη νέα τιμή στο κατάλληλο offset της νέας εγγραφής (μεταβλητή rData). Δημιουργούμε σταδιακά τη νέα εγγραφή σαρώνοντας όλα τα πεδία του πίνακα και αποθηκεύοντας τις νέες τιμές τους σε

αυτή. Κλείνουμε τα αρχεία που ανοίξαμε προηγουμένως, ανοίγουμε τον πίνακα που μας ενδιαφέρει και εισάγουμε σ' αυτόν την νέα εγγραφή με τη βοήθεια του τμήματος REM.

Ανανέωση Εγγραφής (UpdateRecord)

Σημείωση: Η συνάρτηση υποστηρίζει συνθήκη μόνο σε ένα πεδίο. Όχι πολλαπλές υποσυνθήκες.

Αναγνωρίζουμε τον τελεστή της συνθήκης και τα ορίσματά της. Παίρνουμε τα μεταδεδομένα του πεδίου της υποσυνθήκης και του πεδίου το οποίο θα ενημερώσουμε. Αναγνωρίζουμε αν η τιμή της συνθήκης είναι αριθμός ή χαρακτήρες. Σαρώνουμε τον πίνακα και στις εγγραφές που ικανοποιούν τη συνθήκη, εισάγουμε τα νέα δεδομένα στο σωστό πεδίο με την βοήθεια της αντίστοιχης συνάρτησης του REM.

Διαγραφή εγγραφής (DeleteRecord)

Αναγνωρίζουμε τον τελεστή της συνθήκης και τα ορίσματά της. Παίρνουμε τα μεταδεδομένα του πίνακα στον οποίο θα κάνουμε τις αλλαγές και αναγνωρίζουμε αν η τιμή της συνθήκης είναι αριθμός ή χαρακτήρες. Σαρώνουμε τον πίνακα και διαγράφουμε τις εγγραφές που ικανοποιούν τη συνθήκη. Τέλος κάνουμε τις απαραίτητες τροποποιήσεις στα αρχεία των μεταδεδομένων.

User Interface Module (UIM)

Το τμήμα αυτό αποτελεί την διεπαφή της εφαρμογής με τον χρήστη. Είναι υπεύθυνο για την διαχείριση της εισόδου του χρήστη.

Κλάση UIM_UserInterfaceModule.h

Η κλάση περιέχει τις συναρτήσεις επεξεργασίας και εκτέλεσης των εντολών.

Constructor κλάσης (UIM_UserInterface)

Η μέθοδος αυτή αποτελεί τον constructor της κλάσης και είναι υπεύθυνη για την δημιουργία και αρχικοποίηση των αντικειμένων που αντιπροσωπεύουν τα υπόλοιπα modules του συστήματος.

Μέθοδος startUI()

Η μέθοδος αυτή είναι η πρώτη μέθοδος που καλείται. Αρχικά αναμένει την είσοδο από τον χρήστη. Κάθε εντολή πρέπει να τελειώνει με ερωτηματικό. Αν η εντολή που έδωσε ο χρήστης σαν είσοδο δεν τελειώνει με ερωτηματικό, απλός αλλάζει γραμμή και περιμένει την υπόλοιπη είσοδο. Μόλις πάρει την εντολή την επεξεργάζεται και την τροποποιεί κατάλληλα. Η εφαρμογή τερματίζει όταν ο χρήστης δώσει την εντολή “quit;”.

Για την εκτέλεση script ο χρήστης πρέπει να δώσει την εντολή “run script *dir*” όπου dir η απόλυτη διαδρομή του script στο σύστημα. Προσοχή! Το μονοπάτι (dir) πρέπει να περιληφθεί σε διπλά εισαγωγικά.

Οι υπόλοιπες εντολές ακολουθούν τους κανόνες της γλώσσας SQL.

Μέθοδος runScript( )

Η μέθοδος είναι υπεύθυνη για την εκτέλεση κάποιου script που δίνεται από τον χρήστη. Διαβάζει την είσοδο μέχρι να βρει ερωτηματικό ή τη λέξη “go” οπότε και καλεί την αντίστοιχη συνάρτηση για να εκτελέσει το ερώτημα.

Μέθοδος ExecuteQuery()

Η μέθοδος αυτή επεξεργάζεται την είσοδο, ελέγχει το είδος/την κατηγορία του query που έχει δοθεί από τον χρήστη και καλεί την αντίστοιχη μέθοδο.

Μέθοδος Execute_SYSM_Command()

Η μέθοδος αυτή είναι υπεύθυνη για την επεξεργασία των queries που αφορούν τη διαχείριση των βάσεων δεδομένων. Αφού αναγνωρίσει την εντολή, επεξεργάζεται την είσοδο και καλεί την αντίστοιχη συνάρτηση.

Μέθοδος Execute_SSQLM_DDL_Command()

Η μέθοδος αυτή είναι υπεύθυνη για την επεξεργασία των queries που αφορούν την δημιουργία και καταστροφή πινάκων και καταλόγων της βάσης. Αφού αναγνωρίσει την εντολή, επεξεργάζεται την είσοδο και καλεί την αντίστοιχη συνάρτηση.

Μέθοδος Execute_SSQLM_DML_Command()

Η μέθοδος αυτή είναι υπεύθυνη για την επεξεργασία των queries που αφορούν τον χειρισμό των δεδομένων στους πίνακες της ενεργής βάσης δεδομένων. Ανάλογα με την εντολή που έχει δώσει ο χρήστης, επεξεργάζεται κατάλληλα την είσοδο και καλεί την αντίστοιχη συνάρτηση.

Να σημειωθεί ότι στην εισαγωγή δεδομένων σε κάποιο πίνακα (insert into), οι συμβολοσειρές πρέπει να περικλείονται σε μονά ή διπλά εισαγωγικά!

Μέθοδος fixQuery()

Η μέθοδος αυτή είναι υπεύθυνη για τον έλεγχο και την κατάλληλη μορφοποίηση της εισόδου που δίνεται από τον χρήστη. Για την ακρίβεια, αν διαβαστούν πολλαπλά κενά μετατρέπονται σε ένα. Κεφαλαία γράμματα μετατρέπονται σε μικρά. Αν διαβαστεί η γραμματοσειρά ‘string’ ή ‘int’ την μετατρέπει σε κεφαλαία γράμματα. Αν διαβαστεί tab τότε επίσης μετατρέπεται σε ένα κενό και τέλος αν διαβαστεί χαρακτήρας εισαγωγικού (‘) τότε ψάχνει τον αντίστοιχο χαρακτήρα που κλείνει τα εισαγωγικά. Σκοπός της είναι να φέρει την είσοδο σε τέτοια μορφή ώστε στη συνέχεια να γίνει σωστά η επεξεργασία των στοιχείων του ερωτήματος του χρήστη.

Μέθοδοι changeEQ(), changeNE(), changeLT(), changeGT(), changeOther()

Οι μέθοδοι αυτές είναι υπεύθυνες για τον έλεγχο τον υποστηριζόμενων τελεστών και της έκφρασης δεξιά και αριστερά του τελεστή. Συγκεκριμένα, πριν η μετά από τον τελεστή έχει παραλειφθεί κάποιο κενό τότε το προσθέτει. Η δομή δηλαδή που ακολουθείται είναι: ‘έκφραση’(κενό)τελεστής(κενό)’έκφραση’. Αυτό γίνεται για να υπάρχει μια κοινή οργάνωση κατά την ανάγνωση όλων των τελεστών. Διπλοί τελεστές όπως !=, ==, <> θεωρούνται ως ένας.

Επεκτάσεις

Επέκταση στο σύστημα τύπων της γλώσσας

Η πρώτη είναι να υποστηρίζονται από το σύστημα οι τύποι δεδομένων FLOAT και DOUBLE για τη δημιουργία πεδίων πινάκων. Για την υλοποίηση της δυνατότητας έγιναν οι απαραίτητες προσθήκες στο αρχείο REM_Types.h και τροποποιήσεις στις παρακάτω συναρτήσεις:

- checkAttrType της κλάσης DDL_Manager
- GetNextRecord της κλάσης REM_RcordFileScan
- Insert, Update, Delete, Select record της κλάσης SSQLM_DML_Manager

και αφορούν κυρίως ελέγχους για τη θέση (offset) του πεδίου μέσα στις εγγραφές και τη δέσμευση μνήμης.

Να σημειωθεί ότι κατά την παρουσίαση εγγράφων των παραπάνω τύπων κάποια από τα αποτελέσματα εμφανίζονται στρογγυλοποιημένα.

Υποστήριξη χρηστών και δικαιωμάτων

Η δεύτερη επέκταση είναι η υποστήριξη χρηστών και δικαιωμάτων σε επίπεδο βάσεων δεδομένων. Ο χρήστης μπορεί να έχει ή όχι το δικαίωμα να κάνει οποιαδήποτε πράξη πάνω στους πίνακες μίας βάσης.

Κατά την εκκίνηση της εφαρμογής ο χρήστης πρέπει να κάνει είσοδο στο σύστημα. Για να το κάνει αυτό πρέπει να γράψει “-u [username] -p [password]”.

Το σύστημα επεξεργάζεται την είσοδο. Αν ο χρήστης δεν έχει δώσει έγκυρα στοιχεία, τον προτρέπει να ξανά δώσει τα στοιχεία.

Μόλις γίνει επιτυχώς το login, ο χρήστης μπορεί να συνεχίσει για να ανοίξει κάποια βάση.

Για να κάνει logout, ο χρήστης πρέπει να δώσει την εντολή εξόδου (quit) από την εφαρμογή.

Όταν πάει ο χρήστης να ανοίξει μία βάση, γίνεται έλεγχος αν έχει το δικαίωμα για τη συγκεκριμένη βάση. Αν δεν έχει, εμφανίζεται κατάλληλο μήνυμα και δεν ανοίγει η βάση.

Τα στοιχεία των χρηστών είναι αποθηκευμένα σε έναν πίνακα και τα δικαιώματα τους σε ένα άλλο.

Για τον χρήστη root, παρέχονται οι παρακάτω δυνατότητες, κατά τις οποίες ουσιαστικά τροποποιούμε τους προαναφερθέντες πίνακες:

- Εισαγωγή χρήστη με την εντολή
  "create user [username] with password="[password]";"

- Διαγραφή χρήστη με την εντολή
  "delete user [username] with password="[password]";"

- Σημείωση! Το password πρέπει να περικλείεται σε εισαγωγικά.

- Παραχώρηση δικαιωμάτων για κάποια βάση ή όλες τις βάσεις αντίστοιχα με τις εντολές "grant privilege on [database] to [username]" και "grant all privileges to [username]".

- Αφαίρεση δικαιωμάτων για κάποια βάση κατά τη διαγραφή χρήστη.

- Αφαίρεση δικαιωμάτων για όλες τις βάσεις κατά τη διαγραφή χρήστη.

Σημεία Προσοχής

Τα παραδοτέα αρχεία συνοδεύονται από ένα φάκελο, ο οποίος πρέπει να αποθηκευτεί στο δίσκο C του υπολογιστή. Ο φάκελος αυτός ονομάζεται "sirenbase" και περιέχει δύο άλλους φακέλους: τον "bin" και τον "data". Ο φάκελος "bin" περιέχει τα δύο εκτελέσιμα αρχεία (SirenBasic.exe και SirenExtension.exe). Ο φάκελος "data" περιέχει κάποιες βάσεις δεδομένων:

- Η βάση "sirenbase" δεν πρέπει να σβηστεί διότι περιέχει πληροφορίες για τους χρήστες του συστήματος, οι οποίες είναι χρήσιμες για το κομμάτι της επέκτασης. Χωρίς αυτή τη βάση το κομμάτι της επέκτασης δεν μπορεί να λειτουργήσει ουσιαστικά.

- Οι υπόλοιπες βάσεις είναι ενδεικτικές και δεν θα υπάρξει κάποιο πρόβλημα εάν σβηστούν. Να σημειωθεί ότι για τον έλεγχο της σωστής λειτουργίας των δύο προγραμμάτων, χρησιμοποιήθηκαν αυτές οι βάσεις δεδομένων.