ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

Τμ. Πληροφορικής

2011-2012

Τεχνολογία Βάσεων Δεδομένων
Εργασία εξαμήνου

Τριχόπουλος Ηλίας ΑΕΜ:1876
Ανδρονίδης Αναστάσιος ΑΕΜ:1891
Τριανταφυλλίδης Νικόλαος Πέτρος ΑΕΜ: 1843

Η παρακάτω εργασία υλοποιήθηκε στα πλαίσια του μαθήματος Τεχνολογία Βάσεων Δεδομένων. Σκοπός της εργασίας ήταν η υλοποίηση ενός Συστήματος Διαχείρισης Βάσεων Δεδομένων. Ο κώδικας υλοποίθηκε σε γλώσσα C++ και είναι προσανατολισμένο για χρήση σε περιβάλλον Windows. Το σύστημα αποτελείται από συγκεκριμένα τμήματα (modules) που πατάνε πάνω στο δοσμένο interface του συστήματος διαχείρισης αρχείων STORM.

Τα τμήματα αυτά είναι

1. Τμήμα χειρισμού εγγραφών (REM)
2. Τμήμα χειρισμού καταλόγων (INXM)
3. Τμήμα χειρισμού ερωτηματών (SSQLM)
4. Τμήμα χειρισμού συστήματος (SYSM)
5. Τμήμα διεπαφής (UIM)

Παρακάτω περιγράφεται ως ένα βαθμό η υλοποίηση των τμημάτων που έχουν υλοποιηθεί. Σημειώνεται πως το τμήμα UIM δεν είναι υλοποιημένο.

Ως επέκταση του συστήματος η λειτουργία της σύνδεσης πινάκων (Join) είναι υλοποιημένη για να δέχεται απεριόριστο αριθμό από πίνακες για σύνδεση. Συνεπώς υπάρχει μια έκδοση του συστήματος μαζί με την επέκταση.

1. Record Module (REM)

Το τμήμα REM έχει ως αντικείμενο την διαχείρηση των εγγραφών (records) της Βάσης Δεδομένων. Κάθε πίνακας στη Βάση Δεδομένων αποτελεί ένα αρχείο στο δίσκο. Κάθε αρχείο αποτελείται από pages οι οποίες χωρίζονται σε slots που έχουν το μέγεθος μιας εγγραφής. Όταν δημιουργείται ένας νέος πίνακας στη βάση δημιουργείται και ένα νέο αρχείο στο δίσκο. Η πρώτη σελίδα του αρχείου δεσμεύεται για τον File Header και όλα τα δεδομένα των εγγραφών τοποθετούνται από τη δεύτερη σελίδα και μετά. Κάθε αλλαγή στις εγγραφές του αρχείου απαιτεί το άνοιγμα του αρχείου μέσω της κατάλληλης μεθόδου. Κάθε νέα εγγραφή τοποθετείται στο τέλος του αρχείου. Η ανάγνωση μιας εγγραφής τοποθετεί το περιεχόμενό της στη μνήμη και κάθε επεξεργασία στα δεδομένα γίνεται στην κύρια μνήμη. Όταν διαγράφεται μια εγγραφή στη θέση της τοποθετείται η τελευταία εγγραφή στο αρχείο για να αποφεύγονται με αυτόν τον τρόπο τα κενά μέσα στο αρχείο.

Στη συγκεκριμένη υλοποίηση αποτελείται από τις παρακάτω κλάσεις:

REM_RecordFileManager.h
REM_RecordID.h
REM_RecordFileHandle.h
REM_RecordFileScan.h
REM_RecordHandle.h

και το header αρχείο REM_Components.h.

REM_RecordFileManager:

Η κλάση καλεί τις αντίστοιχες μεθόδους από το τμήμα STORM που βρίσκεται κάτω από το τμήμα REM για τη δημιουργία και καταστροφή καθώς και το άνοιγμα και το κλείσιμο των αρχείων όπου περιέχονται οι εγγραφές κάθε πίνακα. Περιέχει τις παρακάτω μεθόδους:

- REM_RecordFileManager (STORM_StorageManager *sm):

Ο κατασκευαστής της κλάσης.

- ~REM_RecordFileManager ():

Ο καταστροφέας της κλάσης.

- t_rc CreateRecordFile (const char *fname, int rs):

Η συγκεκριμένη μέθοδος παίρνει ως όρισμα το όνομα του αρχείου και το μέγεθος της κάθε εγγραφής μέσα στο αρχείο και καλεί την μέθοδο CreateFile του STORM_StorageManager για να δημιουργήσει ένα νέο αρχείο εγγραφών. Στη συνέχεια ανοίγει το αρχείο και δεσμεύει την πρώτη σελίδα μέσα του για να δημιουργήσει το header του αρχείου και να γράψει μέσα του τα απαιτούμενα δεδομένα.

- t_rc DestroyRecordFile (const char *fname):

Η μέθοδος αυτή παίρνει ως όρισμα το όνομα του αρχείου και καλεί τη μέθοδο DestroyFile του STORM_StorageManager για να διαγράψει το αρχείο με αυτό το όνομα από το δίσκο.

- t_rc OpenRecordFile (const char *fname, REM_RecordFileHandle &rfh):

Η μέθοδος είναι υπεύθυνη για το άνοιγμα των αρχείων και την αντιγραφή την αντιγραφή του περιεχομένου τους στη μνήμη.

- t_rc CloseRecordFile (REM_RecordFileHandle &rfh):

Η μέθοδος είναι υπεύθυνη για το κλεισιμο των αρχείων και την απομάκρυνση των σελίδων τους από τη μνήμη.

REM_RecordID:

Η κλάση είναι υπεύθυνη για την σύνθεση των Record IDs που αποτελούνται από ένα pageID τον αριθμό του slot όπου βρίσκεται το record. Περιέχει τις παρακάτω μεθόδους:

- REM_RecordID():

Κατασκευαστής RID για άγνωστα pageID και slot numbers.

- REM_RecordID(int pageID, int slot):

Κατασκευαστής RID για γνωστά pageID και slot numbers.

- ~REM_RecordID():

Ο καταστροφέας της κλάσης.

- t_rc GetPageID(int &pageID) const:

Η μέθοδος επιστρέφει το pageID του RID.

- t_rc GetSlot(int &slot) const:

Η μέθοδος επιστρέφει το slot number του RID.

- t_rc SetPageID (int pageID);

Η μέθοδος γράφει το αντίστοιχο pageID στο RID που έχει δημιουργηθεί.

- t_rc SetSlot (int slot):

Η μέθοδος γράφει το αντίστοιχο slot number στο RID που έχει δημιουργηθεί.

REM_RecordFileHandle:

Η κλάση χειρίζεται τα αρχεία εγγραφών που έχουν δημιουργηθεί. Εκτελεί λειτουργίες ανάγνωσης, εγγραφής, διαγραφής και ενημέρωσης των records χρησιμοποιώντας τις κατάλληλες μεθόδους από

το τμήμα STORM. Περιέχει τις παρακάτω μεθόδους:

- REM_RecordFileHandle():

Ο κατασκευαστής της κλάσης.

- ~REM_RecordFileHandle():

Ο καταστροφέας της κλάσης.

- t_rc ReadRecord (const REM_RecordID &rid, REM_RecordHandle &rh):

Η μέθοδος παίρνει ως όρισμα το ID του record και τον χειριστή του αρχείου που το περιέχει και αντιγράφει τα περιεχόμενα του record στη μνήμη.

- t_rc InsertRecord (const char *pData, REM_RecordID &rid):

Η μέθοδος δέχεται ως όρισμα ένα string που περιέχει αυτά τα δεδομένα ενός record και εισάγει στο τέλος του αρχείου μια καινούργια εγγραφή που περιέχει αυτά τα δεδομένα.

- t_rc DeleteRecord (const REM_RecordID &rid):

Η μέθοδος δέχεται ως όρισμα ένα RID και διαγράφει το αντίστοιχο record από το αρχείο. Στη συνέχεια για εξοικονόμηση χώρου τοποθετεί το τελευταίο record που υπάρχει στο αρχείο στην θέση του record που μόλις διαγράφηκε.

- t_rc UpdateRecord (const REM_RecordHandle &rh):

Η μέθοδος παίρνει ως όρισμα ένα RID και αντιγράφει από τη μνήμη στο αρχείο την ανανεωμένη έκδοση αυτού του record.

- t_rc FlushPages () const:

Η συνάρτηση καλεί την μέθοδο FlushAllPages από το STORM_FileHandle για να απομακρύνει όλες τις σελίδες του αρχείου από τη μνήμη.

REM_RecordFileScan:

Η κλάση είναι υπεύθυνη για την αναζήτηση records μέσα στο αρχείο με βάση κάποια συνθήκη που πρέπει να ικανοποιεί κάποιο χαρακτηριστικό τους. Περιέχει τις εξής μεθόδους:

- REM_RecordFileScan():

Ο κατασκευαστής της κλάσης.

- ~REM_RecordFileScan():

Ο καταστροφέας της κλάσης.

- t_rc OpenRecordScan (REM_RecordFileHandle &rfh,

t_attrType attrType,
int attrLength,
int attrOffset,
t_compOp compOp,
void *value):

Η συνάρτηση δέχεται ως ορίσματα τις παραμέτρους αναζήτησης και τον χειριστή του αρχείου και ξεκινάει τη σειριακή αναζήτηση μέσα στο αρχείο.

- t_rc GetNextRecord(REM_RecordHandle &rh):

Η μέθοδος εκτελεί της απαραίτητες συγκρίσεις για να επιστρέψει το επόμενο record που ικανοποιεί τη συνθήκη αναζήτησης.

- t_rc CloseRecordScan():

Με αυτή τη μέθοδο τερματίζεται η σειριακή αναζήτηση.

Επίσης έχει υλοποιηθεί εσωτερικά η συνάρτηση bool SatisfiesConditions(const char *pData) η οποία περιέχει τους τελεστές σύγκρισης που χρησιμοποιούνται για την αναζήτηση των records μέσα στο αρχείο που ικανοποιούν την συνθήκη που έχει επιλεγεί.

REM_RecordHandle:

Η κλάση είναι υπεύθυνη για το χειρισμό της κάθε εγγραφής. Η κλάση δουλεύει με αντίγραφο της κάθε εγγραφής στη μνήμη και οι αλλαγές που ενδεχομένως να γίνουν ενημερώνονται στο δίσκο αργότερα. Περιέχει τις παρακάτω μεθόδους:

- REM_RecordHandle():

Ο κατασκευαστής της κλάσης.

- ~REM_RecordHandle():

Ο καταστροφέας της κλάσης.

- t_rc GetData(char *&pData) const:

Η συνάρτηση επιστρέφει τα δεδομένα που υπάρχουν μέσα στο record.

- t_rc GetRecordID(REM_RecordID &rid) const:

Η συνάρτηση επιστρέφει τα pageID και slot number που συνθέτουν το record ID.

2. INDEXING MODULE (INXM)

Το τμήμα INXM είναι υπεύθυνο για τη δημιουργία και χειρισμό των καταλόγων που υπάρχουν στη Βάση Δεδομένων. Κάθε κατάλογος χτίζεται γύρω από ένα συγκεκριμένο πεδίο ενός πίνακα της βάσης. Η δομή δεδομένων στην οποία στηρίζονται οι κατάλογοι είναι ένα B+ tree το οποίο στα φύλλα του έχει ζευγάρια (key,value) όπου το key είναι το record ID και value η τιμή του χαρακτηριστικού στο οποίο αναφέρεται ο κατάλογος για το συγκεκριμένο record. Κάθε κόμβος είναι μια page του τμήματος STORM. Τα φύλλα αποτελούν μια συνδεδεμένη λίστα τεχνική που χρησιμοποιείται για την αποφυγή των διπλοεγγραφών. Ο κατάλογος μπορεί να ξεχωρίζει τα φύλλα από τους ενδιάμεσους κόμβους μέσω της πληροφορίας που υπάρχει στο file header.

Αποτελείται από τις παρακάτω κλάσεις:

INXM_IndexManager.h
INXM_IndexHandle.h
INXM_IndexScan.h

καθώς και το αρχείο INXM_Components.h

INXM_IndexManager:

Η κλάση καλεί τις κατάλληλες μεθόδους από το τμήμα STORM που βρίσκεται κάτω από το τμήμα INXM για τη δημιουργία και καταστροφή καθώς και το άνοιγμα και το κλείσιμο των αρχείων όπου περιέχονται οι κατάλογοι που χρησιμοποιεί η βάση δεδομένων. Περιέχει τις παρακάτω μεθόδους:

- INXM_IndexManager (STORM_StorageManager *sm):

Ο κατασκευαστής της κλάσης.

- ~INXM_IndexManager():

Ο καταστροφέας της κλάσης.

- t_rc CreateIndex (const char *fname,
  int indexNo,
  t_attrType attrType,
  int attrLength):

Η μέθοδος δέχεται ως όρισμα το όνομα του αρχείου που περιέχει τον κατάλογο, τον αριθμό του καταλόγου καθώς και πληροφορίες για το χαρακτηριστικό γύρω από το οποίο θα χτίσει τον κατάλογο και καλεί την μέθοδο CreateFile του STORM_StorageManager για να δημιουργήσει ένα νέο αρχείο καταλόγου. Η πρώτη σελίδα δεσμεύεται για την εισαγωγή της πληροφορίας του File Header.

- t_rc DestroyIndex (const char *fname, int indexNo):

Η μέθοδος αυτή παίρνει ως όρισμα το όνομα του αρχείου και καλεί τη μέθοδο DestroyFile του STORM_StorageManager για να διαγράψει το αρχείο με αυτό το όνομα από το δίσκο.

- t_rc OpenIndex (const char *fname, int indexNo, INXM_IndexHandle &ih):

Η συνάρτηση ανοίγει τα αρχεία των καταλόγων και αντιγράφει τα περιεχόμενά τους στην κύρια μνήμη.

- t_rc CloseIndex (INXM_IndexHandle &ih):

Η μέθοδος είναι υπεύθυνη για το κλεισιμο των αρχείων των καταλόγων και την απομάκρυνση των σελίδων τους από τη μνήμη.

INXM_IndexHandle.h:

Η κλάση αυτή υλοποιεί τον χειριστή των καταλόγων της Βάσης Δεδομένων. Εκτελεί λειτουργίες εισαγωγής και διαγραφής εγγραφών μέσα στον κατάλογο τον οποίο χειρίζεται. Περιέχει τις εξής συναρτήσεις:

- INXM_IndexHandle():

Ο κατασκευαστής της κλάσης.

- ~INXM_IndexHandle():

Ο καταστροφέας της κλάσης.

- t_rc InsertEntry(void *pData, const REM_RecordID &rid):

Η συνάρτηση δέχεται ένα σύνολο από δεδομένα και ένα Record ID και εισάγει στα φύλλα του B+ Δένδρου μια νέα εγγραφή key/value pair.

- t_rc DeleteEntry(void *pData, const REM_RecordID &rid):

Η συνάρτηση διαγράφει μια έγγραφη από το B+ δένδρο.

- t_rc FlushPages():

Η συνάρτηση απομακρύνει τις σελίδες του καταλόγου από τη μνήμη.

Επίσης εσωτερικά υλοποιείται ένα σύνολο από συναρτήσεις οι οποίες επιτελούν δομικές λειτουργίες στο B+ Δένδρο. Αυτές είναι οι:

- bool LeafHasRoom(STORM_PageHandle pageHandle);
- int Cut(int length);
- int KeyCmp(void *key,void *key2);
- t_rc LoadInitHeaders(STORM_PageHandle pageHandle, INXM_InitPageHeader &initPageHeader);
- t_rc LoadNodeHeaders(int pageID, INXM_InitPageHeader &initPageHeader, INXM_NodePageHeader &nodePageHeader);

- t_rc LoadNodeHeaders(STORM_PageHandle pageHandle, INXM_InitPageHeader &initPageHeader, INXM_NodePageHeader &nodePageHeader);
- t_rc UpdateNodeHeaders(STORM_PageHandle pageHandle, INXM_InitPageHeader initPageHeader, INXM_NodePageHeader nodePageHeader);
- t_rc ReadNode(int page, int slot, INXM_Node &node);
- t_rc ReadNode(STORM_PageHandle pageHandle, int slot, INXM_Node &node);
- t_rc ReadData(STORM_PageHandle pageHandle, int slot, INXM_Data &data);
- t_rc FindLeaf(int rootPageID,void *pData, STORM_PageHandle &leafPageHandle);
- t_rc FindAndAppend(int rootPageID, void *key, const REM_RecordID &rid);
- t_rc EditData(STORM_PageHandle pageHandle, int slot, INXM_Data data);
- t_rc EditNode(STORM_PageHandle pageHandle, int slot, INXM_Node node);
- t_rc WriteNode(STORM_PageHandle pageHandle, int insertPoint, void *key, int left, int slot);
- t_rc WriteNode(STORM_PageHandle pageHandle, void *key, int left, int slot);
- t_rc WriteData(STORM_PageHandle pageHandle, const REM_RecordID &rid, int &slot);
- t_rc CreateNodePage(STORM_PageHandle &pageHandle, int &pageID, int parent, int next, int previous, bool isLeaf);
- t_rc CreateLastDataPage(STORM_PageHandle &pageHandle);
- t_rc InsertIntoParent(int rootID, STORM_PageHandle leftPage, INXM_Node &keyNode, STORM_PageHandle rightPage);
- t_rc InsertIntoLeaf(STORM_PageHandle leafPageHandle, void *key, const REM_RecordID &rid);
- t_rc InsertIntoNoLeaf(STORM_PageHandle parentPage, int left_index, void* key, int rightPageID);
- t_rc StartNewTree(void *pData, const REM_RecordID &rid);
- t_rc InsertIntoLeafAfterSplitting(int rootID, STORM_PageHandle leafPageHandle, void *key, const REM_RecordID &rid);

- INXM_IndexScan:

Η κλάση είναι υπεύθυνη για την ανάκτηση του περιεχομένου των εγγραφών του καταλόγου. Περιέχει τις παρακάτω μεθόδους:

- INXM_IndexScan():

Ο κατασκευαστής της κλάσης.

- ~INXM_IndexScan():

Ο καταστροφέας της κλάσης.

- t_rc OpenIndexScan(const INXM_IndexHandle &ih,
  t_compOp compOp,
  void *value):

Η συνάρτηση δέχεται ως όρισμα τον χειριστή του καταλόγου μια παράμετρο αναζήτησης και ένα τελεστή σύγκρισης και ξεκινάει την προσπέλαση του B+ δένδρου για την εύρεση των records που ικανοποιούν την αναζήτηση.

- t_rc GetNextEntry(REM_RecordID &rid):

Κατά την προσπέλαση του B+ δένδρου η συνάρτηση επιστρέφει την επόμενη εγγραφή που ικανοποιεί τη συνθήκη αναζήτησης.

- t_rc CloseIndexScan():

Η συνάρτηση τερματίζει την προσπέλαση του καταλόγου.

Εσωτερικά έχουν υλοποιηθεί οι παρακάτω συναρτήσεις για την αναζήτηση μέσα στο B+ δένδρου:

- int KeyCmp(void *key,void *key2);
- t_rc ReadData(int page, int slot, INXM_Data &node);
- t_rc ReadData(STORM_PageHandle pageHandle, int slot, INXM_Data &data);
- t_rc ReadNode(int page, int slot, INXM_Node &node);
- t_rc ReadNode(STORM_PageHandle pageHandle, int slot, INXM_Node &node);
- t_rc LoadNodeHeaders(int pageID, INXM_InitPageHeader &initPageHeader, INXM_NodePageHeader &nodePageHeader);
- t_rc LoadNodeHeaders(STORM_PageHandle pageHandle, INXM_InitPageHeader &initPageHeader, INXM_NodePageHeader &nodePageHeader);
- t_rc FindLeaf(STORM_PageHandle rootPage,void *pData, STORM_PageHandle &leafPageHandle);
- t_rc FollowList(REM_RecordID &rid);

3. SIREN SQL MODULE (SSQLM)

Το τμήμα SSQLM είναι υπεύθυνο για την επεξεργασία των SQL ερωτημάτων που στέλνονται στη βάση. Αντικείμενο του είναι ουσιαστικά να στέλνει στα παρακάτω τμήματα του REM και του INXM τις κατάλληλες παραμέτρους για την εκτέλεση των λειτουργιών τους. Αποτελείται από τις παρακάτω κλάσεις:

SSQLM_DDL_Manager.h
SSQLM_DML_Manager.h

SSQLM_DDL_Manager:

Η κλάση είναι υπεύθυνη για την επεξεργασία των ερωτημάτων που υπάγονται στην υπογλώσσα ορισμού δεδομένων DDL. Περιέχει τις εξής μεθόδους:

- SSQLM_DDL_Manager(REM_RecordFileManager *rfm, INXM_IndexManager *im, char *dbName):

Ο κατασκευαστής της κλάσης.

- ~SSQLM_DDL_Manager():

Ο καταστροφέας της κλάσης.

- t_rc CreateTable(const char *tname, const char *attributes):

Η συνάρτηση δέχεται ως όρισμα το όνομα ενός πίνακα και ένα string με όλες της στήλες αυτού του πίνακα. Γράφει τα κατάλληλα μεταδεδομένα στο αρχείο attr.met και στη συνέχεια καλεί τη μέθοδο CreateRecordFile του REM_FileManager για να δημιουργήσει ένα αρχείο εγγραφών με όνομα, το

όνομα του πίνακα.

- t_rc DropTable(char *tname):

Η μέθοδος καλεί την συνάρτηση DestroyRecordFile του REM_FileManager για να διαγράψει το αρχείο του πίνακα από το δίσκο. Στη συνέχεια αφαιρεί τα μεταδεδομένα του από το αρχείο attr.met.

- t_rc CreateIndex(char *tname, const char *attrName):

Η μέθοδος δέχεται ως όρισμα το όνομα ενός πίνακα και το όνομα μιας στήλης του. Γράφει τα κατάλληλα μεταδεδομένα στο αρχείο attr.met και στη συνέχεια καλεί τη μέθοδο CreateIndex του INXM_IndexManager για να δημιουργήσει έναν κατάλογο για τη συγκεκριμένη στήλη του πίνακα.

- t_rc DropIndex(char *tname, const char *attrName, int indexNo):

Η μέθοδος δέχεται ως όρισμα τον αριθμό του καταλόγου του χαρακτηριστικού και καλεί τη συνάρτηση DestroyIndex του INXM_IndexManager για να διαγράψει το αρχείο του καταλόγου από το δίσκο. Στη συνέχεια ενημερώνει τα μεταδεδομένα του πίνακα σχετικά με την καταστροφή του καταλόγου.

Εσωτερικά στην κλάση έχουν δημιουργηθεί οι κάτωθι συναρτήσεις που επεμβαίνουν στα μεταδεδομένα της Βάσης Δεδομένων για την εκτέλεση των απαραίτητων ενημερώσεων:

- t_rc OpenRelmet(char *dbName);
- t_rc OpenAttrmet(char *dbName);
- t_rc EditAttrMet(char *str, const char *tName, int &recSize);
- t_rc EditRelMet(const char *tName, int recSize, int numOfColumns);
- t_rc DeleteTableMeta(const char *tName);
- t_rc FindRecordInRelMet(const char *tName, REM_RecordHandle &rh);
- t_rc FindRecordInAttrMet(const char *tName, REM_RecordHandle &rh);
- t_rc FindRecordInAttrMet(char *tName, const char *attrName, REM_RecordHandle &rh);
- t_rc UpdateRelmetIndexes(const char *tName, REM_RecordHandle &rh, int &indexNo, bool increase);
- t_rc UpdateAttrmetIndexNo(char *tName, const char *attrName, REM_RecordHandle &rh, t_attrType &attrType, int &attrLength, int indexNo);
- t_rc GetIndexNo(char *pData, int &indexNo);

SSQLM_DML_Manager:

Η κλάση είναι υπεύθυνη για την επεξεργασία των ερωτημάτων που υπάγονται στην υπογλώσσα επεξεργασίας δεδομένων DML. Περιέχει τις εξής μεθόδους:

- SSQLM_DML_Manager(REM_RecordFileManager *rfm, INXM_IndexManager *im, char *dbName):

Ο κατασκευαστής της κλάσης.

- ~SSQLM_DML_Manager():

Ο καταστροφέας της κλάσης.

- t_rc Select(const char *tName, vector<char *> columns, vector<char *> recordsFromWhereFunction, vector<char *> *finalResults):

Η συνάρτηση δέχεται το όνομα του πίνακα και ένα σύνολο από διανύσματα που αφορούν τις στήλες του πίνακα και τα αποτελέσματα της συνθήκης where για να επιστρέψει τις στήλες του πίνακα που ζητήθηκαν με την εντολή select.

- t_rc Join(char *table1, char* table2, char *connectionAttribute):

Η συνάρτηση δέχεται ως όρισμα τα ονόματα 2 πινάκων και το χαρακτηριστικό πάνω στο οποίο ζητήθηκε να γίνει η σύνδεση και επιστρέφει τα αποτελέσματα της σύνδεσης των 2 πινάκων. Η συγκεκριμένη λειτουργία έχει υλοποιηθεί για να δέχεται απεριόριστο αριθμό από πίνακες και όχι μόνο για 3 πίνακες όπως ζητούσε η εκφώνηση της εργασίας.

- t_rc Where(const char *tName, char *conditions, vector<char *> *finalResultRecords, vector<REM_RecordID> *finalResultsRIDs):

Η μέθοδος δέχεται ως όρισμα το όνομα ενός πίνακα και μια συνθήκη. Επιστρέφει όλες τις εγγραφές που ικανοποιούν την συνθήκη που έχει δοθεί.

- t_rc Insert(const char *tName, const char *record):

Η μέθοδος δέχεται ως όρισμα το όνομα ενός πίνακα και ένα string που περιέχει ένα record. Τοποθετεί το record αυτό στο αρχείο του πίνακα. Στη συνέχεια ενημερώνει κατάλληλα τα μεταδεδομένα του πίνακα.

- t_rc Delete(const char *tName, REM_RecordID ridsToDelete, char *recordsToDelete):

Η συνάρτηση δέχεται το όνομα του πίνακα και το ID του record προς διαγραφή και αφαιρεί το αντίστοιχο record από το αρχείο του πίνακα. Στη συνέχεια ενημερώνει κατάλληλα τα μεταδεδομένα του πίνακα.

- t_rc Update(const char *tName, vector<REM_RecordID> ridsFromWhere, char *setAttributes):

Η συνάρτηση δέχεται το όνομα του πίνακα και ένα σύνολο από IDs εγγραφών καθώς και ένα σύνολο από νέες τιμές. Ενημερώνει τις αντίστοιχες εγγραφές με τις νέες τιμές που δόθηκαν. Στη συνέχεια ενημερώνει κατάλληλα τα μεταδεδομένα του πίνακα.

Έχουν υλοποιηθεί εσωτερικά στην κλάση οι παρακάτω συναρτήσεις για την ενημέρωση των μεταδεδομένων και την εκτέλεση των απαραίτητων διαδικασιών για την επεξεργασία των δεδομένων:

- t_rc OpenRelmet(char *dbName);
- t_rc OpenAttrmet(char *dbName);
- t_rc GetAttrInfo(char *rec, int &offset, char *&type, int &size, int &indexID);
- t_rc GetAttrInfoJoin(char *rec, char *&attrName, int &offset, char *&type, char *&size, int &indexID);
- t_rc GetConditionInfo(char *condition, char *&conditionAttribute, t_compOp &comp, char *&conditionValue);
- t_rc FindAttributeInAttrmet(const char *tName, char *attributeName, int &offset, char

*&type, int &size, int &indexID);
- t_rc CheckIfTableHasIndexes(const char *tName,bool &hasIndexes);
- t_rc GetTableRecordSize(char *tName, int &recordSize);
- t_rc ConCatRecords(REM_RecordID rid, REM_RecordFileHandle *rfh, REM_RecordHandle *rh, int offset, int size, int recordSize, char *firstRecord, char *&newRecord);
- t_rc RebuildTableDefinition(char *tName, char *&tableDefinition);
- t_rc RebuildTableDefinitionChopped(char *tName, char *&tableDefinition, char *connectionAttribute);

5. SYSTEM MODULE (SYSM)

Το τμήμα SYSM είναι υπεύθυνο για τον χειρισμό πολλαπλών Βάσεων Δεδομένων. Εκτελεί λειτουργίες δημιουργίας και καταστροφής Βάσεων Δεδομένων καθώς και ανοίγματος και κλεισίματος μιας ΒΔ καθώς κάθε φορά μόνο μια ΒΔ είναι ενεργή. Κάθε ΒΔ είναι ένας ξεχωριστός φάκελος στο δίσκο που περιλαμβάνει εκτός από τα πραγματικά δεδομένα 2 αρχεία μεταδεδομένων της ΒΔ που περιέχουν πληροφορίες για τους πίνακες και τα χαρακτηριστικά κάθε πίνακα μέσα στη ΒΔ. Αποτελείται από τις παρακάτω κλάσεις:

SYSM_MetadataManager.h
SYSM_DatabaseManager.h

και το header αρχείο SYSM_Metadata.h.

SYSM_MetadataManager:

Η κλάση είναι υπεύθυνη για τη δημιουργία και τη διαχείριση των μεταδεδομένων της ΒΔ. Περιέχει τις μεθόδους:

- SYSM_MetadataManager(SYSM_DatabaseManager *dbm):

Ο κατασκευαστής της κλάσης.

- ~SYSM_MetadataManager():

Ο καταστροφέας της κλάσης.

- t_rc RecordRelationMeta(SYSM_relmet *relation):

Η συνάρτηση καταγράφει στο αρχείο rel.met τα αρχικά μεταδεδομένα για μια ΒΔ που έχει δημιουργηθεί.

- t_rc RecordAttributeMeta(SYSM_attrmet *attribute):

Η συνάρτηση καταγράφει στο αρχείο attr.met τα αρχικά μεταδεδομένα για τους πίνακες σε μια ΒΔ που έχει δημιουργηθεί.

SYSM_DatabaseManager:

Η κλάση είναι υπεύθυνη για τις εργασίες δημιουργίας και καταστροφής Βάσεων Δεδομένων καθώς και άνοιγμα και κλείσιμο της μιας ενεργής ΒΔ. Περιέχει τις μεθόδους:

- SYSM_DatabaseManager(REM_RecordFileManager *rfm):

Ο κατασκευαστής της κλάσης.

- ~SYSM_DatabaseManager();

Ο καταστροφέας της κλάσης.

- t_rc CreateDatabase(const char *dbName):

Η συνάρτηση δέχεται το όνομα μιας νέας ΒΔ και δημιουργεί έναν φάκελο στο δίσκο που περιέχει τα δεδομένα της ΒΔ. Στη συνέχεια δημιουργεί τα αρχεία rel.met και attr.met που περιέχουν τα μεταδεδομένα της ΒΔ.

- t_rc DropDatabase (const char *dbName):

Η μέθοδος δέχεται το όνομα μιας ΒΔ και διαγράφει από το δίσκο το φάκελο της ΒΔ μαζί με όλα τα αρχεία των δεδομένων της.

- t_rc OpenDatabase (const char *dbName):

Η μέθοδος δέχεται ως όρισμα το όνομα μιας ΒΔ και κάνει αυτή τη ΒΔ ενεργή στο σύστημα.

- t_rc CloseDatabase ();

Η μέθοδος κλείνει την ενεργή ΒΔ.

- const char *getdbName():

Η μέθοδος επιστρέφει το όνομα της ενεργής ΒΔ.

- bool isOpen():

Η μέθοδος ελέγχει αν υπάρχουν ενεργές ΒΔ στο σύστημα.